UNIL | Université de Lausanne

Unicentre
CH-1015 Lausanne
http://serval.unil.ch

Year : 2023

Exploring Three-Dimensional Geodynamic Processes: Insights From Numerical Modelling on Diapirism and Mountain Stability

Macherel Emilie

Macherel Emilie, 2023, Exploring Three-Dimensional Geodynamic Processes: Insights From Numerical Modelling on Diapirism and Mountain Stability

Originally published at : Thesis, University of Lausanne

Posted at the University of Lausanne Open Archive http://serval.unil.ch
Document URN : urn:nbn:ch:serval-BIB_44680E85064B0

Droits d’auteur
L'Université de Lausanne attire expressément l'attention des utilisateurs sur le fait que tous les documents publiés dans l'Archive SERVAL sont protégés par le droit d'auteur, conformément à la loi fédérale sur le droit d'auteur et les droits voisins (LDA). A ce titre, il est indispensable d'obtenir le consentement préalable de l'auteur et/ou de l’éditeur avant toute utilisation d'une oeuvre ou d'une partie d'une oeuvre ne relevant pas d'une utilisation à des fins personnelles au sens de la LDA (art. 19, al. 1 lettre a). A défaut, tout contrevenant s'expose aux sanctions prévues par cette loi. Nous déclinons toute responsabilité en la matière.

Copyright
The University of Lausanne expressly draws the attention of users to the fact that all documents published in the SERVAL Archive are protected by copyright in accordance with federal law on copyright and similar rights (LDA). Accordingly it is indispensable to obtain prior consent from the author and/or publisher before any use of a work or part of a work for purposes other than personal use within the meaning of LDA (art. 19, para. 1 letter a). Failure to do so will expose offenders to the sanctions laid down by this law. We accept no liability in this respect.

UNIL | Université de Lausanne
Institut des sciences
de la Terre

Faculté des géosciences et de l'environnement

Institut des sciences de la Terre

EXPLORING THREE-DIMENSIONAL GEODYNAMIC PROCESSES: INSIGHTS FROM NUMERICAL MODELLING ON DIAPIRISM AND MOUNTAIN STABILITY

THÈSE DE DOCTORAT EN SCIENCES DE LA TERRE

présentée à la Faculté des géosciences et de l'environnement de l'Université de Lausanne

par

EMILIE MACHEREL

Maîtrise ès Sciences en géosciences de l'environnement de l'Université de Lausanne

Jury

Prof. Dr. Stefan M. Schmalholz	Université de Lausanne	Directeur de thèse
Prof. Dr. Yury Y. Podladchikov	Université de Lausanne	Expert interne
Prof. Dr. Laetitia Le Pourhiet	Sorbonne Université	Experte externe
Prof. Dr. Marie-Elodie Perga	Université de Lausanne	Président du jury

Lausanne, 2023

Unil
UNIL | Université de Lausanne
Faculté des géosciences et de l'environnement
bâtiment Géopolis bureau 4631

IMPRIMATUR

Vu le rapport présenté par le jury d'examen, composé de

Présidente de la séance publique : Mme la Professeure Marie-Elodie Perga
Présidente du colloque : Mme la Professeure Marie-Elodie Perga
Directeur de thèse : M. le Professeur Stefan Schmalholz
Expert interne : M. le Professeur Yury Y.Podladchikov
Experte externe : Mme la Professeure Laetitia Le Pourhiet

Le Doyen de la Faculté des géosciences et de l'environnement autorise l'impression de la thèse de

Madame Emilie MACHEREL

Titulaire d'un
Master Géologie
de l'Université de Lausanne

intitulée

EXPLORING THREE-DIMENSIONAL GEODYNAMIC PROCESSES: INSIGHTS FROM NUMERICAL MODELLING ON DIAPIRISM AND MOUNTAIN STABILITY

Lausanne, le 22 novembre 2023

Pour le Doyen de la Faculté des géosciences et de l'environnement

Marie Elodie Pe

Professeure Marie-Elodie Perga

Contents

1 General Introduction 1

1.1 Outline of the Thesis 2

1.2 Motivation 3

1.3 Objectives and Geological Contexts 6

1.3.1 Chapter 2: Stress State Around a Diapir 7

1.3.2 Chapter 3: Stress State Around a Continental Plateau 9

1.4 Approach and Method 13

1.4.1 Mathematical Model 13

1.4.2 Viscous Deformation Behaviors 14

1.4.3 Numerical Method 16

2 Quantifying Diapir Ascent Velocities in Power-Law Viscous Rock Under Far-Field Stress: Integrating Analytical Estimates, 3D Numerical Calculations and Geodynamic Applications 25

2.1 Introduction 28

2.2 Model 32

2.2.1 Flow Law and Effective Viscosity 32

2.2.2 Analytical Estimates for Diapir Ascent Velocity in Deforming Power-Law Viscous Medium 34

2.2.3 3D Mathematical Model 37

2.2.4 Numerical Method 38

2.2.5 Model Configuration 40

2.3 Results 42

2.3.1 Distribution of Stress, Pressure and Effective Viscosity 42

2.3.2 Stress Decomposition and Magnitudes 44

2.3.3 Comparison of Analytical and Numerical Ascent Velocities 46

2.4 Discussion 50

2.4.1 Characteristic Stresses for Experimentally Derived Flow Laws . . . 50

2.4.2 Increase of Diapir Ascent Velocity by Two Types of Stress Weakening 52

2.4.3 Applications to Sediment Diapirs, Mantle Plumes, (U)HP Terranes and Plutons 55

2.4.4 Simplifications 61

2.5 Conclusions 62

2.6 Acknowledgements 64

2.7 Open Research 64

Appendix A. Cylindrical Coordinates System 73

Appendix B. Spherical Coordinates System 75

Appendix C. Comparison of Cartesian, Cylindrical and Spherical Coordinates Systems 77

Appendix D. Numerical Performances 80

Appendix E. Parameters Used in Figure 2.9 82

3 3D Stresses and Gravitational Spreading Velocities in Continental Plateaus: Analytical Estimates, Numerical Calculations With Earth's Curvature and Application to the Tibetan Plateau 83

3.1 Introduction 86

3.2 Model 92

3.2.1 Analytical Estimates for Stress and Spreading Velocity 92

3.2.2 Mathematical Model 95

3.2.3 Numerical Method 96

3.2.4 Model Configuration 97

3.3 Results 100

3.3.1 Overview 100
3.3.2 General Stress State Around a Continental Plateau 100
3.3.3 Impact of the Curvature 102
3.3.4 Impact of the Corner 107
3.3.5 Impact of the Crust and Mantle Strength 112
3.3.6 Impact of a Stress-Dependent Power-Law Flow Law 115
3.3.7 Estimates of Stress and Spreading Velocity 118
3.4 Discussion 121
3.4.1 Viscous Deformation of the Lithosphere 121
3.4.2 Curvature 122
3.4.3 Crustal and Lithospheric Mantle Viscosities 123
3.4.4 Shear Stresses 124
3.4.5 Stress Versus Strength Relationship 126
3.4.6 Simplifications 126
3.4.7 Dimensionless Stress Ratio and Analytical Estimate for Spreading Velocity 128
3.4.8 Applications to the Tibetan Plateau 129
3.5 Conclusions 131
3.6 Acknowledgements 133
3.7 Open Research 133
Appendix A. Method 145

4 General Conclusion and Perspectives 149

A Numerical Grids 157

B Code for the Diapir in Cartesian Coordinates 159

C Code for the Diapir in Cylindrical Coordinates 167

D Code for the Diapir in Spherical Coordinates 175

E Code for the Continental Plateau in Spherical Coordinates 185

Abstract

Lithospheric stresses play a crucial role in geological processes at various spatial and temporal scales. However, the distribution and magnitudes of these stresses are poorly constrained in diverse geodynamic settings.

The objective of this thesis is to enhance our understanding of stress distribution and magnitudes surrounding two distinct geological features and their implications. The first part of the thesis focuses on diapirism, a process where a less dense body rises into a denser material, playing a major role in the Earth's heat and mass transfer. Quantifying the upward velocity of buoyant bodies is essential, as it determines the significance of diapirism in different geological contexts. To achieve this, we developed a 3-D numerical algorithm in spherical coordinates using the Julia programming language. The algorithm solves the Stokes equations under the influence of gravity, with rheology incorporating a combination of linear and power-law viscous flow laws. Employing finite difference and pseudo-transient methods, the algorithm calculates instantaneous stress and velocity fields.

In the first study, the model consists of a buoyant sphere ascending in a denser medium under strike-slip shearing. Results indicate that the diapir's ascent velocity depends on two stress ratios: (a) regional stress to characteristic stress, and (b) buoyancy stress to characteristic stress. Regional stress arises from far-field deformation, while buoyancy stress originates from the diapir itself as it attempts to move upwards. The characteristic stress is a material property, marking the stress level at which deformation behavior transitions from diffusion creep (linear flow law) to dislocation creep (power-law flow law). Indeed the stress ratios generate stress weakening which significantly impacts the rising velocity of the diapir. Furthermore, we compared our results with existing analytical estimates, demonstrating their accuracy and applicability across a wide range of geodynamic settings. Additionally, we were able to improve these estimates by introducing a shape factor and an extra term accounting for the impact of the regional stresses.

The second part of the thesis focuses on continental plateaus. It is well-established that stress states around significant topography cannot be lithostatic to maintain their geometry over extended periods of time. Depth-averaged stress and strength estimations exist, but the spatial distribution of these quantities throughout the lithosphere remains unclear. The goal of the second part of the thesis is to provide constraints on their spatial distribution. To accomplish this, we employed the same numerical algorithm as in the first part of the thesis, with a predefined geometry of a continental plateau. We calculated stress and velocity fields generated only by gravity and tested the impact of various parameters and characteristics on these fields. Results reveal that curvature has a minor impact on the stress field, while the corner region significantly influences stress distribution, particularly shear stresses. Varying viscosities of the crust and lithospheric mantle strongly affect stress distribution and plateau spreading velocities. Our results show that lithospheric strength is mainly located in the crust rather than the lithospheric mantle. Also, our findings indicate that stress cannot serve as a proxy for strength. Additionally, we derive simple analytical estimates for horizontal deviatoric stress and spreading velocities of a continental plateau.

Résumé

Les contraintes lithosphériques jouent un rôle crucial dans les processus géologiques à différentes échelles spatiales et temporelles. Cependant, la distribution et l'amplitude de ces contraintes sont mal connues dans divers contextes géodynamiques. L'objectif de cette thèse est ainsi d'améliorer notre compréhension des contraintes et de leur implication au sein de deux caractéristiques géologiques distinctes.

La première partie de cette thèse est dédiée à l'étude du diapirisme : ce processus consiste en l'élévation d'un corps moins dense au sein d'un milieu plus dense. Il joue un rôle majeur quant au transfert de chaleur et de masse dans la Terre. C'est pourquoi quantifier la vitesse d'ascension de ces corps est essentiel. Une telle chose permettrait de déterminer l'importance du diapirisme dans divers contextes géologiques. Pour ce faire, nous avons développé un algorithme numérique 3D en coordonnées sphériques ; nous y avons utilisé le langage de programmation Julia. Cet algorithme résout les équations de Stokes sous l'influence de la gravité, avec une rhéologie combinant des lois d'écoulement visqueux linéaire et non-linéaire. En utilisant les méthodes des différences finies et pseudo-transiente, il calcule les champs de contraintes et de vitesses instantanées. L'étude du diapirisme est menée au moyen d'une modélisation numérique. Ce modèle consiste en une sphère s'élevant dans un milieu plus dense, le tout dans une zone de cisaillement. Les résultats indiquent que la vitesse d'ascension du diapir dépend de deux ratios de contraintes : (a) la contrainte régionale par rapport à une contrainte caractéristique, et (b) la contrainte de flottabilité par rapport à la même contrainte caractéristique. La contrainte régionale provient de la déformation à grande échelle, tandis que la contrainte de flottabilité prend son origine du diapir lui-même alors qu'il tente de s'élever. La contrainte caractéristique

est une propriété du matériau, marquant le niveau de contrainte auquel la déformation passe de diffusion (loi d'écoulement linéaire) à dislocation (loi d'écoulement non-linéaire). En effet, les ratios de contraintes génèrent une modification de la viscosité, ce qui impacte grandement la vitesse de remontée du diapir. De plus, nous avons comparé nos résultats avec des estimations analytiques existantes, démontrant leur précision et leur applicabilité dans beaucoup de contextes géodynamiques. De plus, nous avons comparé nos résultats avec des estimations analytiques existantes, ce qui nous a permis de les améliorer en y ajoutant un terme prenant en compte l'effet des contraintes régionales. Les estimations analytiques démontrent une grande précision et une bonne applicabilité dans beaucoup de contextes géodynamiques.

La deuxième partie de la thèse se concentre sur les plateaux continentaux. Il est bien établi que les états de contrainte autour d'une topographie significative ne peuvent pas être lithostatiques pour maintenir la géométrie sur de longues périodes. Des estimations de contraintes et de résistance moyennées sur la profondeur existent, mais la distribution spatiale de ces quantités à travers la lithosphère demeure floue. L'objectif de la deuxième partie de la thèse est, ainsi, de mieux définir leur distribution spatiale. Pour ce faire, nous avons utilisé le même algorithme numérique que dans la première partie de la thèse, avec une géométrie prédéfinie de plateau continental. Nous avons alors calculé les champs de contraintes et de vitesses générés uniquement par la gravité et testé l'impact de divers paramètres et caractéristiques sur ces champs. Les résultats révèlent que la courbure a un impact mineur sur le champ des contraintes, tandis que les variations latérales de l'épaisseur de la croûte influent significativement la distribution des contraintes, en particulier les contraintes de cisaillement. La variation des viscosités de la croûte et du manteau lithosphérique montre une forte influence sur la distribution des contraintes et des vitesses de propagation du plateau. Nos résultats montrent également que la résistance lithosphérique se trouve principalement dans la croûte plutôt que dans le

manteau lithosphérique. De plus, nos conclusions indiquent que les contraintes ne peuvent pas être utilisées comme indicateur de la résistance. Finallement, nous avons développé des estimations analytiques simples afin d'évaluer les contraintes horizontales ainsi que la vitesse d'écoulement d'un plateau continental.

CHAPTER 1

General Introduction

1.1 Outline of the Thesis

This thesis is structured in four chapters and an appendix section. Chapters 2 and 3 are written in the form of scientific articles supported by a general introduction (this chapter). It presents the motivation of this thesis, followed by the geological contexts of both studies, and finally the approach and method employed. Chapter 4 is a general conclusion that summarizes the main results of this thesis and provides future perspectives for this study.

- *Chapter 2: Quantifying Diapir Ascent Velocities in Power-Law Viscous Rock Under Far-Field Stress: Integrating Analytical Estimates, 3D Numerical Calculations and Geodynamic Applications.*

  This chapter is a scientific article accepted for publication in the journal Geochemistry, Geophysics, Geosystems. The co-authors are Yury Podladchikov, Ludovic Räss and Stefan M. Schmalholz.

  In this part of the thesis, we present a 3D numerical model that calculates the stress and velocity fields around a diapir rising in a strike-slip environment. We investigate the effects of two stress ratios on the rising velocity of the diapir. This velocity of ascent is essential to measure the importance of diapirism in different geodynamic settings. Furthermore, we compare our results with existing analytical estimates.

- *Chapter 3: 3D Stresses and Gravitational Spreading Velocities in Continental Plateaus: Analytical Estimates, Numerical Calculations With Earth's Curvature and Application to the Tibetan Plateau.*

  This chapter is a scientific article submitted in the journal Geochemistry, Geophysics, Geosystems. The co-authors are Ludovic Räss and Stefan M. Schmalholz.

  This chapter aims to better constrain the stress distribution and magnitudes in and around continental plateaus. We use the same numerical model as Chapter 2 and we test the impact of different parameters on the resulting stress field. This study allows us to better understand the relative contribution of the continental crust and of the lithospheric mantle in holding the stresses supporting the plateau over long time scales. Also, we derive simple analytical estimates for the crustal horizontal deviatoric stress and spreading velocities of continental plateaus.

1.2 Motivation

The motivation for this PhD thesis is to provide a better understanding of the stress distribution and magnitudes in the case of two full 3D geological applications, namely diapirism and continental plateaus. The focus is on the lithospheric stresses because they control many major processes, such as seismicity, volcanism, mountain formation, etc. (e.g., Turcotte and Schubert, 2021). The following paragraphs present why stresses are crucial to quantify the different types of lithospheric stresses and their impact, followed by a general context of diapirism and continental plateaus in Section 1.3. Finally, the approach and method employed in this thesis are introduced in Section 1.4.

Stresses are a force per unit surface, with the unit Pascal (Pa) (Turcotte and Schubert, 2021). They act on each surface of an object at the same time but with various intensities (Figure 1.1a). They are essential to evaluate how materials react to external or internal forces, which is fundamental to understand rock mechanics. Stresses can be generated by different geological processes, such as plate tectonic movements, temperature variations, or metamorphic reactions. There exist three main types of stress (Turcotte and Schubert, 2021). The first type is normal stresses. They act perpendicular to the surface of a material and can be compressive or extensive (Figure 1.1b and c). The second type is shearing stresses. They occur when the force is applied parallel to the surface of a material, leading to relative sliding (Figure 1.1d). Finally, the third type is hydrostatic (or lithostatic) stresses (Figure 1.1e). In this case, stresses are identical in all directions, and their magnitude results from the pressure exerted by the rock column above a given position. These stresses increase with depth. Deformation arises when the stress state differs from the lithostatic in one or more directions. Indeed, stresses are a 3D feature that can trigger complex deformations. Hence, understanding their three-dimensional distribution is a challenge and is necessary to better interpret and apprehend geological processes.

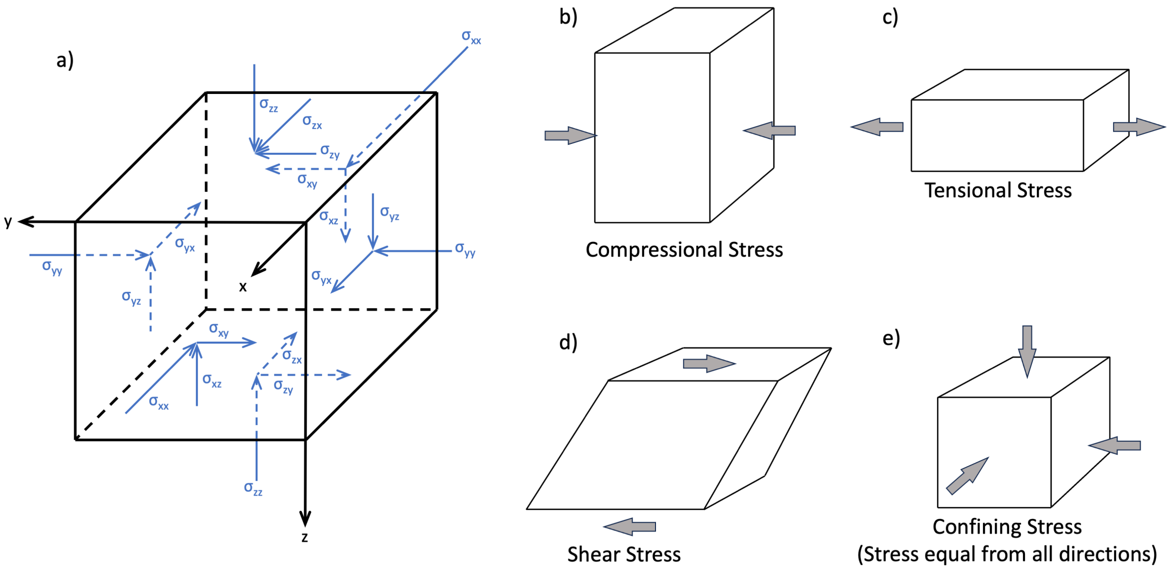

Figure 1.1: (a) Stresses acting on a cube in 3D (Turcotte and Schubert, 2021). The other panels display the different types of stresses acting on a volume (Nelson, 2015): (b) compressive stress, (c) extensive stress, (d) shear stress, and (e) confining stress (when in a lithostatic stress state).

Stresses can be represented in the Mohr circle (Figure 1.2). Mean stress is defined at the centre of the circle. Maximum and minimum normal stresses, σ_1 and σ_3 respectively, are indicated where the circle crosses the x-axis. Hence, deviatoric stresses are defined by the radius of the circle, and differential stress ($\sigma_1 - \sigma_3$) is represented by the diameter of the circle.

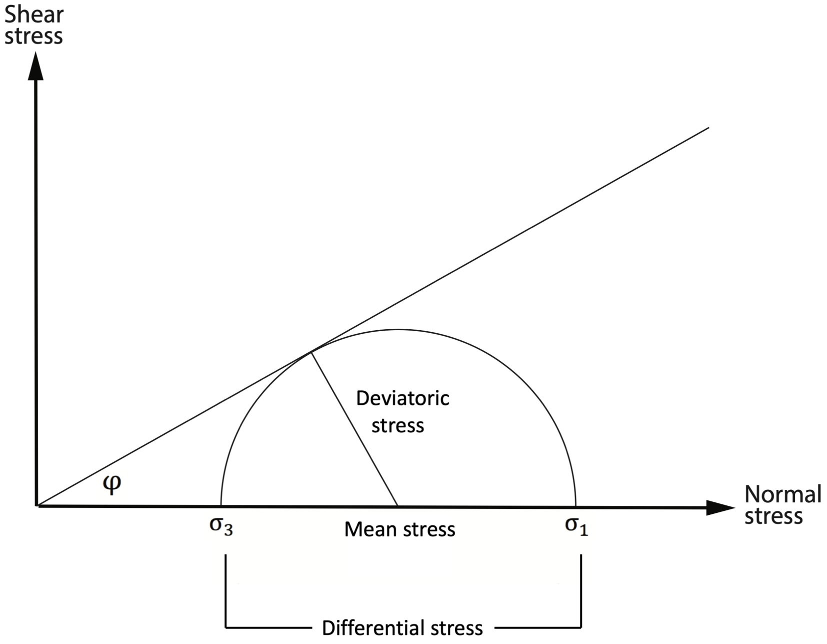

Figure 1.2: Schematic representation of Mohr's circle.

Stresses induce deformations at all scales. At the microscopic scale, they act on individual grains, leading to folding, stretching, or even breaking of minerals (Figure 1.3a) (Plümper et al., 2022). At a local scale, they are responsible for different geological structures such as faults, shear zones, or folds (Figure 1.3b) (Gay and Weiss, 1974; Rutter, 1999; Gudmundsson et al., 2010). At a regional scale, stresses have a major impact on the geography, geomorphology and the geology of large areas as they control the formation of, for example, mountain belts, sedimentary basins, and major faults (Figure 1.3c) (Turcotte and Schubert, 2021).

Over geological time scales, deformations, and the accompanying stresses are inherently linked with motion. Motion implies velocity, and these three elements interact to form a complex system. In this thesis, we try to understand it better through two different applications occurring at different time and spatial scales, diapirism and continental plateaus. Furthermore, these two applications represent full 3D geological features that cannot be

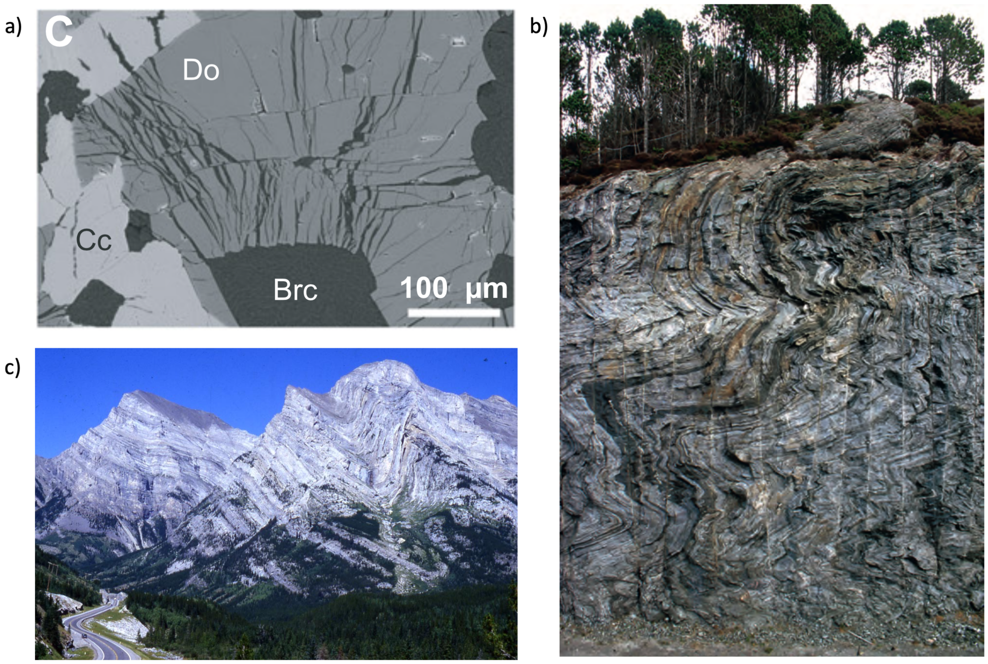

Figure 1.3: Examples of stress-induced deformations at (a) microscopic scale (Plümper et al., 2022), (b) local scale (Fossen, 2016), and (c) regional scale (Website of Eastern Connecticut University).

simplified in 2D. Hence, we perform full 3D numerical calculations to ensure a better understanding of these phenomenons.

1.3 Objectives and Geological Contexts

In the previous section, we have seen that lithospheric stresses are essential to understand the dynamics of many geological processes. However, in the case of the two geological features studied in this thesis, diapirism and continental plateaus, stresses are poorly understood and constrained. The following sections present their geological contexts and the objectives of each study.

1.3.1 Chapter 2: Stress State Around a Diapir

Diapirism is an important mechanism of mass and heat transportation in Earth. It is the mechanism by which less dense material moves upwards through the denser surrounding rocks (Figure 1.4) (Turcotte and Schubert, 2021). The rising material can be, for exemple, magma (e.g., Marsh, 1982; Weinberg and Podladchikov, 1994; Michail et al., 2021), rock salt (e.g., Jackson et al., 1990; Poliakov et al., 1993), mud (e.g., Mazzini et al., 2009) and sediments buried at subduction zones (e.g., Behn et al., 2011; Klein and Behn, 2021; Smye and England, 2023). Hence, diapirism can occur at various spatial and temporal scales.

One of the main questions around diapirism is the velocity of ascent, which is essential to understand the importance of this process in various geological settings. However, a major difficulty in estimating the velocity of ascent lies in the fact that diapirs form and rise at large depths, where it is impossible to directly measure the velocity. Hence, much research is performed based on mineral dating, studies of the surrounding rocks, observation of current volcanic activity (e.g., Skinner and Tanaka, 2006; Fossen, 2016; Burton-Johnson et al., 2022; Copley et al., 2023) and numerical modeling (e.g., Polyansky et al., 2016; Louis-Napoléon et al., 2020).

Simplest estimates of rising velocities of a diapir concern the movement of a rigid sphere in a weaker linear viscous surrounding medium. The displacement is triggered by the density difference between the sphere and the medium. This estimate is called the Stokes law (Stokes, 1850). However, in geology, diapirs are generally weaker than surrounding rocks. In early 20$^{\text{th}}$ century, Hadamard (1911) and Rybczynski (1911) presented an analytical estimation to predict the velocity of ascent of a diapir, knowing only its size, the density difference between the diapir and the surrounding rocks and their respective viscosities. Their analytical estimation is based on many simplifications.

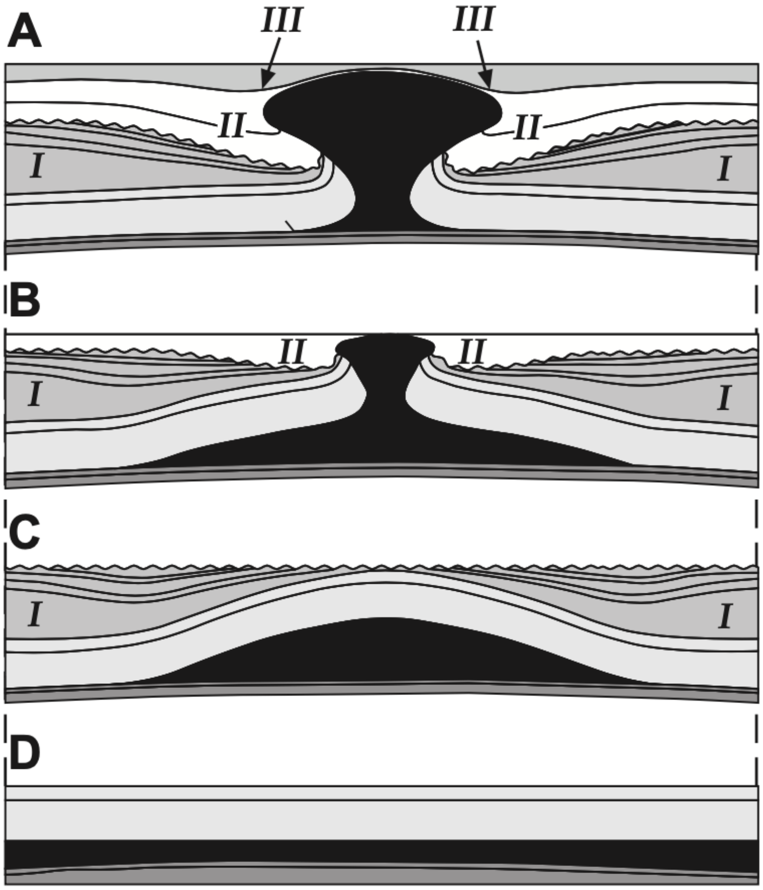

Figure 1.4: Sketches of diapirism process (Vendeville, 2002) from the present-day post-diapir stage (a) to initial situation (d). The black layer is less dense than the upper layers, hence it wants to move upwards.

First, they suppose that rocks behave like a Newtonian fluid. This means that the viscosity of rocks is constant, it does not depend on external parameters such as stresses, temperature, or chemical composition variations. In other words, it is linear viscous. Also, they suppose that diapir material properties, such as density, viscosity, and velocity are constant in time. They also consider an idealized geometry of the diapir, a sphere or a cylinder. Furthermore, they neglect the thermal effects and the complex interactions between the diapir and the surrounding rocks.

This analytical estimation has been further developed by many researchers (e.g., Dazhi and Tanner, 1985; Kawase and Moo-Young, 1986) to obtain better evaluation of the ascent velocity of diapirs. One in particular was made by Weinberg and Podladchikov (1994). They introduced a viscosity dependence on temperature and stresses and studied the ascent mechanism in a porous medium. Their solution is still very simplified and only provides the rising velocity of a buoyant body, without knowing the stress distribution around the diapir. Their solution is used in current studies (e.g., Vynnycky and O'Brien, 2013; Klein and Behn, 2021; Copley et al., 2023), but it was never tested with a full 3D numerical algorithm.

The aims of this study are (a) to quantify the velocity of ascent of a weak diapir, represented by a sphere, in a medium under strike-slip shearing. This allows to better understand the impact of buoyancy and regional stresses on the rising velocity by systematically varying the density difference between the diapir and the surrounding medium, $\Delta\rho$, and the characteristic stress, τ_C, marking the stress at which the deformation behavior changes from linear to power-law viscous (see Section 1.4.2). (b) To compare our numerical results with the analytical estimates of Weinberg and Podladchikov (1994).

1.3.2 Chapter 3: Stress State Around a Continental Plateau

The second part of this thesis is focusing on continental plateaus. Examples of such geological features are the Tibetan plateau (Figure 1.5a), located at the limit between the Indian and the Eurasian plates, and the Altiplano (Figure 1.5b) associated with the subduction of the Nazca plate beneath the South American plate. They are characterized by large areas located high above sea level (average of $5'000$ m altitude for the Tibetan plateau) and a thick crust that may be in isostatic equilibrium. Also, they are mostly associated with convergent plate boundaries and mountain formation.

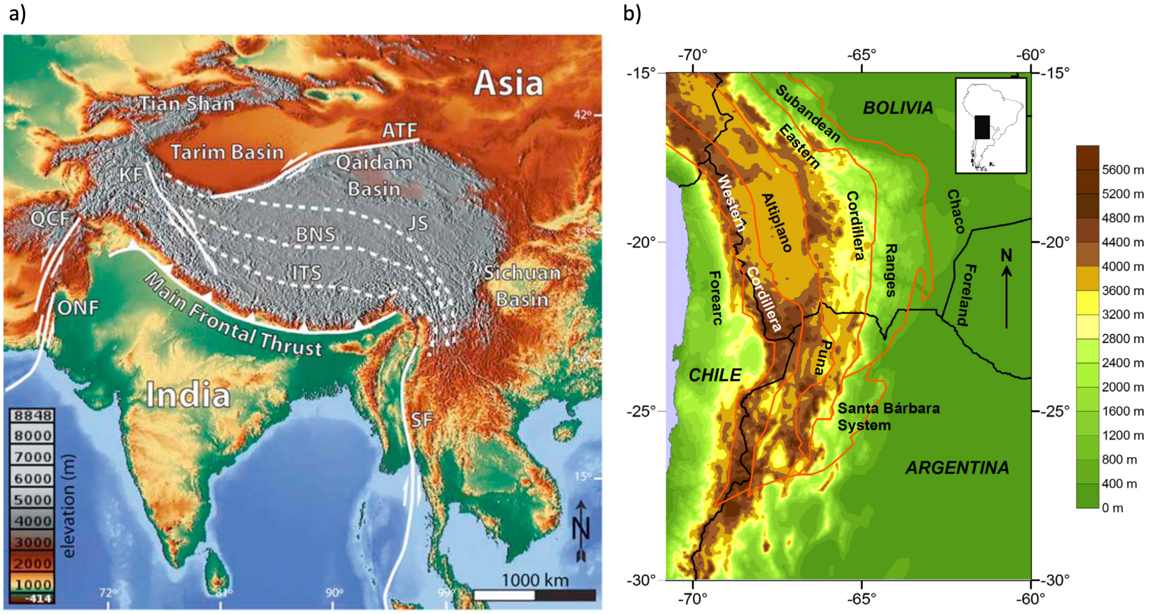

Figure 1.5: Examples of continental plateaus: (a) Tibetan plateau (Lechmann et al., 2014), and (b) Altiplano (Prezzi et al., 2014)

It is well known that the presence of large topography and crustal thickness variations implies that the stress state within the lithosphere is not lithostatic (Darwin, 1882; Jeffreys, 1932; Turcotte and Schubert, 2021). Otherwise, such large features would flow apart instantaneously, like honey on a plate (Molnar, 2015).

There exist different models that allow to estimate the depth-integrated, or averaged, lithospheric strength and stresses, such as thin viscous sheet models (England and McKenzie, 1982; England and Houseman, 1986, 1988; England and Molnar, 1997; Medvedev and Podladchikov, 1999; Flesch et al., 2001) and evaluations based on lateral variations of gravitational potential energy (GPE) (Parsons and Richter, 1980; Molnar and Lyon-Caen, 1988; Molnar et al., 1993; Schmalholz et al., 2014), or more generally on lateral variations of topography and crustal thickness (Artyushkov, 1973).

Strength, together with stresses, control lithospheric deformations (Schmalholz et al., 2019). Yet, the relative contributions of the crust and the lithospheric mantle remain debated. Some studies suggest that the crust is weak, but the plateau can hold because the lithospheric mantle is strong (e.g., England and Molnar, 2015), others, on the opposite, consider that the crust has to be strong in order to maintain large topographic features on Earth through geological time periods (e.g., Flesch et al., 2001).

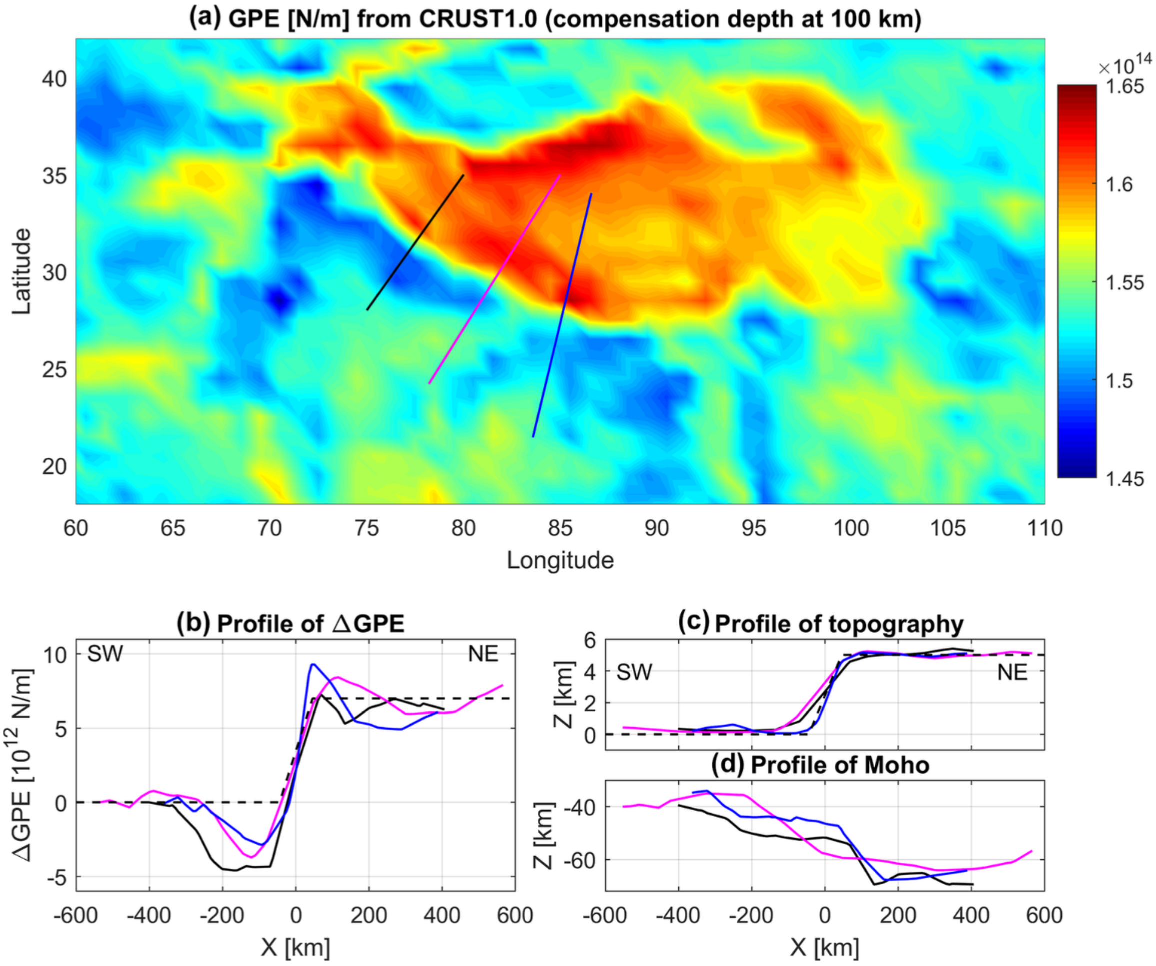

Figure 1.6: Map view of GPE values around continental plateau (Schmalholz et al., 2019). Other panels represent 3 horizontal profiles for (a) GPE values, (b) topography, and (c) depth of the Moho.

Gravitational potential energy (GPE) is defined by the depth-integrated lithostatic pressure. This calculation is reliable because it requires knowledge of topography, crustal thickness, and crustal and lithospheric mantle densities that are well-constrained. As a result, GPE is larger where crustal thickness and topography are important. Figure 1.6 displays the lateral variations of GPE around the Tibetan plateau. These lateral variations of GPE characterize an instability of the crust (England and Molnar, 2022). As a consequence, stresses are expected to show horizontal extension and vertical compression of the plateau, triggering a horizontal compression of the surrounding lowland. This gravity-driven flow is called gravitational collapse (Rey et al., 2001). Also, the fact that features like the Tibetan plateau exist and are maintained over geologic time scales implies that spreading velocities are small, hence rocks inside, around, or under the plateau need to be strong enough to support the stresses. However, the strength and stress magnitude and distribution within the lithosphere remain unclear.

This leads to the first aim of this study, that is to quantify the stresses inside and around a continental plateau exhibiting a corner region representing a lateral geometry variation. Second, continental plateaus cover very large areas (thousands to millions of km^2) and previous studies have shown that a small bending can significantly impact the mechanical stability of a thin sheet (Pini et al., 2016). Hence, we investigate the effect of the Earth's double curvature on the stress field by increasing the curvature of the model domain. Third, the relative contribution of the crust and of the lithospheric mantle to the strength of the lithosphere is still debated. In this study, we vary their respective viscosity to quantify their relative impact on the lithospheric strength. Furthermore, we test the impact of a non-linear rheology (power-law viscous flow law) inducing stress-weakening on the stress field and horizontal velocities.

1.4 Approach and Method

1.4.1 Mathematical Model

In this thesis, we use continuum mechanics to describe deformations and stress occurring around rising diapirs and continental plateaus. This is feasible because at macro-scale material appears to be continuous. In other words, material properties, such as density or temperature for example, vary gradually in space. To apply continuum mechanics, the studied closed system has to conserve mass, momentum and energy (Burg and Schmalholz, 2015). In this study, we focus on the mechanics, ignoring energy variations to understand the first order mechanical processes. Furthermore, we consider that material is incompressible.

These conditions lead to a system of equations including conservation of mass:

$$0 = \frac{\partial V_i}{\partial x_i} \tag{1.1}$$

and conservation of momentum (force balance equation):

$$0 = \frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial P}{\partial x_i} + \rho g_i \tag{1.2}$$

where V_i is the component of the velocity vector in direction x_i, τ_{ij} is the $(i, j)^{th}$ component of the deviatoric stress tensor, P is the pressure, ρ the density and g_i the gravitational acceleration in direction x_i. The total stress tensor is decomposed into a mean stress (pressure) and a deviatoric stress tensor: $\sigma_{ij} = -\delta_{ij} P + \tau_{ij}$, δ_{ij} being the Kronecker delta.

To this system are added constitutive equations, that are material dependant:

$$\tau_{ij} = 2\eta_{\mathrm{E}} \dot{\epsilon_{ij}} = 2\eta_{\mathrm{E}} \left(\frac{1}{2} \left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i} \right) \right) \tag{1.3}$$

where η_{E} is the effective viscosity, depending on the rheology, and $\dot{\epsilon_{ij}}$ is the $(i, j)^{th}$ component of the strain rate tensor. In the end, this system is a closed system of equations, composed by 10 equations and 10 unknowns (P, V_1, V_2, V_3, τ_{11}, τ_{22}, τ_{33}, τ_{12}, τ_{13}, τ_{23}, the stress tensor being symetric).

1.4.2 Viscous Deformation Behaviors

A reasonable assumption is that rocks deform like a fluid over geological time scales. In other words, the deformation behavior is viscous, that is when a material acts like a fluid when exposed to stresses. Material has a certain viscosity, which means that it resists the deformation by opposing the movement of inner particles. Also, deformation occurs gradually in time. This time dependence implies that if the deformation is slow, stresses will be small, and if the deformation is fast, stresses will be large. In a viscous material, the relation between stress and strain can be linear or non-linear. This leads to different types of viscous deformations (i.e., creep).

Diffusion creep (represented by a linear flow law) is a mechanism of viscous deformation based on atomic or vacancy movement inside of a solid material (Kohlstedt, 2007). When a stress is applied, atoms or vacancies start to move in the direction of the stress. If material is compressed (Figure 1.7), atoms move in the direction of the least compressive stress, which contributes to the thinning of the material in the direction of the largest stress. This mechanism is slow and can happen under relatively small stresses. Another viscous deformation mechanism is dislocation creep (represented by a power-law flow law). In this type of deformation, dislocations, that is a group or a line of atoms, are moving together inside of a crystal in response to stress (Figure 1.8) (Kohlstedt, 2007). This deformation is faster than diffusion creep and needs larger stresses to occur.

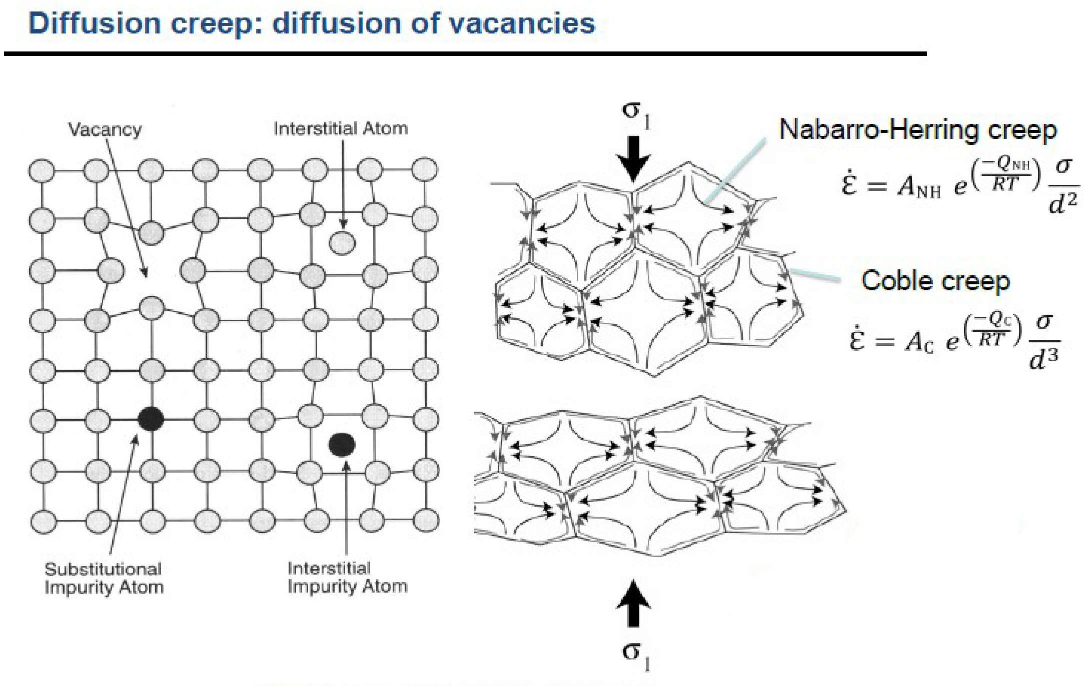

Figure 1.7: Diffusion creep : schematic view (Passchier and Trouw, 2005)

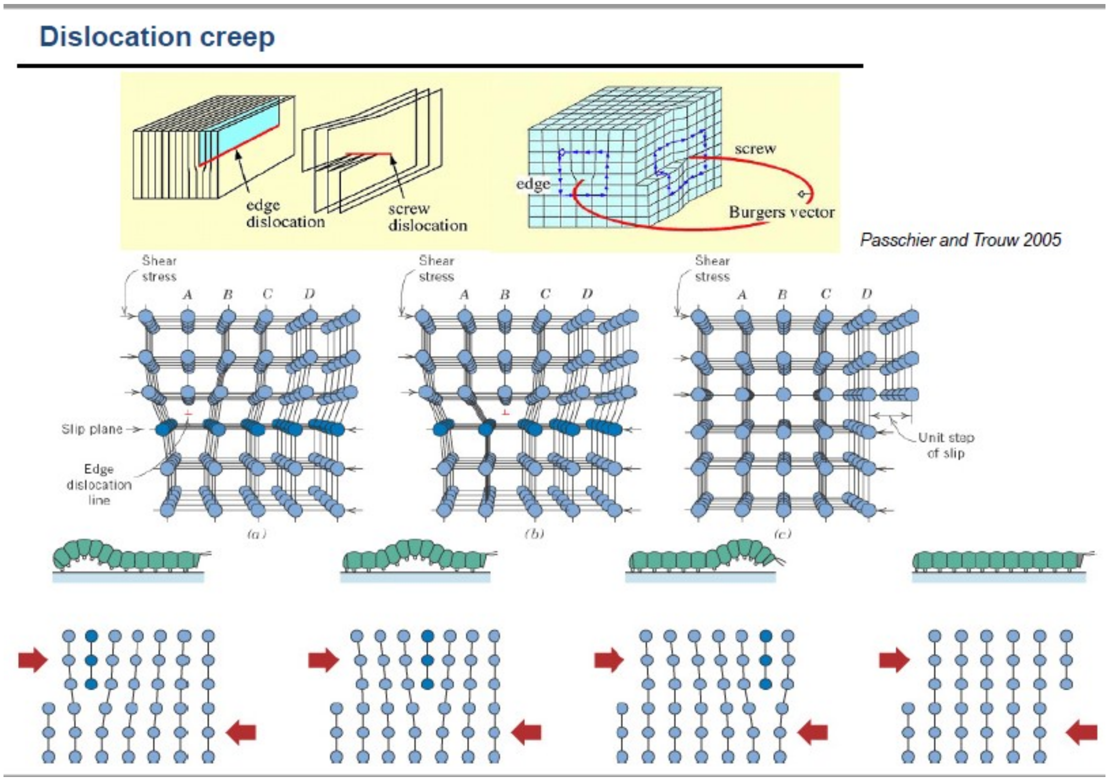

Figure 1.8: Dislocation creep : Schematic view (Passchier and Trouw, 2005)

In nature, different types of deformation mechanisms can occur simultaneously. In this thesis, we focus on viscous deformation because we want to understand the impact of diffusion creep and dislocation creep on the geological processes occuring around rising diapirs and continental plateaus. Hence, we use a combined flow law to define the rheology of the material (more explanations in Section 2.2.1). Practically, the effect of that combined flow law is that in regions where stresses are large (above a predefined stress threshold named characteristic stress), the effective viscosity decrease locally representing dislocation creep. On the opposite, in regions where stresses are small, the material behaves linearly and diffusion creep is dominant.

1.4.3 Numerical Method

Continuum mechanics provides a closed system of equations (same number of unknowns and equations), but no analytical solutions of the partial differential equations exist for complex geometries. Hence, we built a numerical algorithm capable of solving this system of equations.

To achieve this, one has to first discretize the equations on a numerical grid, here we use the finite difference method on a staggered grid with constant spacing (Gerya, 2019; Räss et al., 2022). Second, one has to decide on a numerical method to solve the equations. We use the pseudo-transient method to obtain a steady-state solution of the governing equations. This method is iterative and allows to solve the equations in a matrix-free way (Räss et al., 2022). The concept of the method is to add a pseudo-time derivative to the governing equations and give the model an initial guess of the stress and velocity fields. Because the initial guess does not satisfy the steady-state equations, the pseudo-transient method iterates until the pseudo-time derivatives become smaller than a predefined tolerance value. When this happens, a steady-state solution is found.

In this thesis, we built three different numerical algorithms solving the same system of equations in different coordinates systems (Cartesian, cylindrical and spherical coordinates), using the Julia language. The developped numerical codes are displayed in the Appendix section.

Bibliography

Artyushkov, E., 1973. Stresses in the lithosphere caused by crustal thickness inhomogeneities. Journal of Geophysical Research 78 (32), 7675–7708.

Behn, M. D., Kelemen, P. B., Hirth, G., Hacker, B. R., Massonne, H.-J., 2011. Diapirs as the source of the sediment signature in arc lavas. Nature Geoscience 4 (9), 641–646.

Burg, J.-P., Schmalholz, S. M., 2015. Basics of Continuum Mechanics. Lecture Notes.

Burton-Johnson, A., Riley, T., Harrison, R., Niocaill, C. M., Muraszko, J., Rowley, P., 2022. Does tectonic deformation control episodic continental arc magmatism? Evidence from granitic magnetic fabrics (AMS). Gondwana Research 112, 1–23.

Copley, A., Weller, O., Bain, H., 2023. Diapirs of crystal-rich slurry explain granite emplacement temperature and duration. Scientific Reports 13 (1).

Darwin, G. H., 1882. IV. on the stresses caused in the interior of the earth by the weight of continents and mountains. Philosophical Transactions of the Royal Society of London (173), 187–230.

Dazhi, G., Tanner, R., 1985. The drag on a sphere in a power-law fluid. Journal of Non-Newtonian Fluid Mechanics 17 (1), 1–12.

England, P., Houseman, G., 1986. Finite strain calculations of continental deformation: 2. comparison with the India-Asia collision zone. Journal of Geophysical Research: Solid Earth 91 (B3), 3664–3676.

England, P., Houseman, G., 1988. The mechanics of the Tibetan Plateau. Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences 326 (1589), 301–320.

England, P., McKenzie, D., 1982. A thin viscous sheet model for continental deformation. Geophysical Journal International 70 (2), 295–321.

England, P., Molnar, P., 1997. Active deformation of Asia: From kinematics to dynamics. Science 278 (5338), 647–650.

England, P., Molnar, P., 2015. Rheology of the lithosphere beneath the central and western Tien Shan. Journal of Geophysical Research: Solid Earth 120 (5), 3803–3823.

England, P., Molnar, P., 2022. Changes in plate motions caused by increases in gravitational potential energy of mountain belts. Geochemistry, Geophysics, Geosystems 23 (10).

Flesch, L. M., Haines, A. J., Holt, W. E., 2001. Dynamics of the India-Eurasia collision zone. Journal of Geophysical Research: Solid Earth 106 (B8), 16435–16460.

Fossen, H., 2016. Structural Geology. Cambridge University Press.

Gay, N., Weiss, L., 1974. The relationship between principal stress directions and the geometry of kinks in foliated rocks. Tectonophysics 21 (3), 287–300.

Gerya, T., 2019. Introduction to numerical geodynamic modelling. Cambridge University Press.

Gudmundsson, A., Simmenes, T. H., Larsen, B., Philipp, S. L., 2010. Effects of internal structure and local stresses on fracture propagation, deflection, and arrest in fault zones. Journal of Structural Geology 32 (11), 1643–1655.

Hadamard, J., 1911. Mouvement permanent lent d'une sphere liquide et visqueuse dans un liquide visqueux. Compt. Rend. Acad. Sci. 152, 1735–1738.

Jackson, M., Cornelius, R., Craig, C., Gansser, A., Stöcklin, J., Talbot, C., 1990. Salt diapirs of the Great Kavir, central Iran. Geological Society of America Memoirs 177.

Jeffreys, H., 1932. On the stresses in the Earth's crust required to support surface inequalities.(Second Paper.). Geophysical Journal International 3, 60–69.

Kawase, Y., Moo-Young, M., 1986. Approximate solutions for power-law fluid flow past a particle at low Reynolds numbers. Journal of Non-Newtonian Fluid Mechanics 21 (2), 167–177.

Klein, B. Z., Behn, M. D., 2021. On the evolution and fate of sediment diapirs in subduction zones. Geochemistry, Geophysics, Geosystems 22 (11).

Kohlstedt, D., 2007. Properties of rocks and minerals – constitutive equations, rheological behavior, and viscosity of rocks. In: Treatise on Geophysics. Elsevier, pp. 389–417.

Lechmann, S. M., Schmalholz, S. M., Hetényi, G., May, D. A., Kaus, B. J. P., 2014. Quantifying the impact of mechanical layering and underthrusting on the dynamics of the modern India-Asia collisional system with 3-D numerical models. Journal of Geophysical Research: Solid Earth 119 (1), 616–644.

Louis-Napoléon, A., Gerbault, M., Bonometti, T., Thieulot, C., Martin, R., Vanderhaeghe, O., 2020. 3-D numerical modelling of crustal polydiapirs with volume-of-fluid methods. Geophysical Journal International 222 (1), 474–506.

Marsh, B. D., 1982. On the mechanics of igneous diapirism, stoping, and zone melting. American Journal of Science 282 (6), 808–855.

Mazzini, A., Nermoen, A., Krotkiewski, M., Podladchikov, Y., Planke, S., Svensen, H., 2009. Strike-slip faulting as a trigger mechanism for overpressure release through piercement structures. Implications for the Lusi mud volcano, Indonesia. Marine and Petroleum Geology 26 (9), 1751–1765.

Medvedev, S. E., Podladchikov, Y. Y., 1999. New extended thin-sheet approximation for geodynamic applications—I. model formulation. Geophysical Journal International 136 (3), 567–585.

Michail, M., Rudolf, M., Rosenau, M., Riva, A., Gianolla, P., Coltorti, M., 2021. Shape of plutons in crustal shear zones: A tectono-magmatic guide based on analogue models. Journal of Structural Geology 150, 104417.

Molnar, P., 2015. Plate Tectonics: A Very Short Introduction. Oxford University Press.

Molnar, P., England, P., Martinod, J., 1993. Mantle dynamics, uplift of the Tibetan Plateau, and the Indian monsoon. Reviews of Geophysics 31 (4), 357–396.

Molnar, P., Lyon-Caen, H., 1988. Some simple physical aspects of the support, structure, and evolution of mountain belts. Processes in continental lithospheric deformation 218, 179–207.

Nelson, S., 2015. Deformation of rock. Lecture Notes.

Parsons, B., Richter, F. M., 1980. A relation between the driving force and geoid anomaly associated with mid-ocean ridges. Earth and Planetary Science Letters 51 (2), 445–450.

Passchier, C. W., Trouw, R. A. J., 2005. Microtectonics, 2nd Edition. Springer-Verlag, Berlin Heidelberg.

Pini, V., Ruz, J. J., Kosaka, P. M., Malvar, O., Calleja, M., Tamayo, J., 2016. How two-dimensional bending can extraordinarily stiffen thin sheets. Scientific Reports 6 (1).

Plümper, O., Wallis, D., Teuling, F., Moulas, E., Schmalholz, S. M., Amiri, H., Müller, T., 2022. High-magnitude stresses induced by mineral-hydration reactions. Geology 50 (12), 1351–1355.

Poliakov, A. N., Podladchikov, Y., Talbot, C., 1993. Initiation of salt diapirs with frictional overburdens: numerical experiments. Tectonophysics 228 (3-4), 199–210.

Polyansky, O., Reverdatto, V., Babichev, A., Sverdlova, V., 2016. The mechanism of magma ascent through the solid lithosphere and relation between mantle and crustal diapirism: numerical modeling and natural examples. Russian Geology and Geophysics 57 (6), 843–857.

Prezzi, C., Llanos, M. P. I., Götze, H.-J., Schmidt, S., 2014. Thermal and geodynamic contributions to the elevation of the Altiplano–Puna plateau. Physics of the Earth and Planetary Interiors 237, 51–64.

Räss, L., Utkin, I., Duretz, T., Omlin, S., Podladchikov, Y. Y., 2022. Assessing the robustness and scalability of the accelerated pseudo-transient method. Geoscientific Model Development 15 (14), 5757–5786.

Rey, P., Vanderhaeghe, O., Teyssier, C., 2001. Gravitational collapse of the continental crust: definition, regimes and modes. Tectonophysics 342 (3-4), 435–449, Partial Melting of Crust and Flow of Orogens.

Rutter, E., 1999. On the relationship between the formation of shear zones and the form of the flow law for rocks undergoing dynamic recrystallization. Tectonophysics 303 (1-4), 147–158.

Rybczynski, W., 1911. Über die fortschreitende Bewegung einer flüssigen Kugel in einen zähen Medium. Bull. Acad. Sci. Cracovie 1, 40–46.

Schmalholz, S. M., Duretz, T., Hetényi, G., Medvedev, S., 2019. Distribution and magnitude of stress due to lateral variation of gravitational potential energy between Indian lowland and Tibetan plateau. Geophysical Journal International 216 (2), 1313–1333.

Schmalholz, S. M., Medvedev, S., Lechmann, S. M., Podladchikov, Y., 2014. Relationship between tectonic overpressure, deviatoric stress, driving force, isostasy and gravitational potential energy. Geophysical Journal International 197 (2), 680–696.

Skinner, J. A., Tanaka, K. L., 2006. Evidence for and implications of sedimentary diapirism and mud volcanism in the southern Utopia highland–lowland boundary plain, Mars. Icarus 186 (1), 41–59.

Smye, A. J., England, P. C., 2023. Metamorphism and deformation on subduction interfaces: 2. Petrological and tectonic implications. Geochemistry, Geophysics, Geosystems 24 (1), e2022GC010645.

Stokes, G., 1850. On the effect of internal friction of fluids on the motion of pendulums. Trans. Camb. phi1. S0c 9 (8), 106.

Turcotte, D., Schubert, G., 2021. Geodynamics. Cambridge University Press.

Vendeville, B., 2002. A New Interpretation of Trusheim's Classic Model of Salt-Diapir Growth. Vol. 52. Gulf Coast Association of Geological Societies Transactions.

Vynnycky, M., O'Brien, M. A., 2013. A quasi-asymptotic model for the ascent of magmatic diapirs. Geophysical and Astrophysical Fluid Dynamics 108 (1), 20–43.

Weinberg, R. F., Podladchikov, Y., 1994. Diapiric ascent of magmas through power law crust and mantle. Journal of Geophysical Research: Solid Earth 99 (B5), 9543–9559.

CHAPTER 2

Quantifying Diapir Ascent Velocities in Power-Law Viscous Rock Under Far-Field Stress: Integrating Analytical Estimates, 3D Numerical Calculations and Geodynamic Applications

Emilie Macherel[1], **Yuri Podladchikov**[1], **Ludovic Räss**[1,2,3], **and Stefan M. Schmalholz**[1]

[1]University of Lausanne, ISTE, Lausanne, Switzerland

[2]Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Switzerland

[3]Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland

Abstract

Diapirism is crucial for heat and mass transfer in many geodynamic processes. Understanding diapir ascent velocity is vital for assessing its significance in various geodynamic settings. Although analytical estimates exist for ascent velocities of diapirs in power-law viscous, stress weakening fluids, they lack validation through 3D numerical calculations. Here, we improve these estimates by incorporating combined linear and power-law viscous flow and validate them using 3D numerical calculations. We focus on a weak, buoyant sphere in a stress weakening fluid subjected to far-field horizontal simple shear. The ascent velocity depends on two stress ratios: (a) the ratio of buoyancy stress to characteristic stress, controlling the transition from linear to power-law viscous flow, and (b) the ratio of regional stress associated with far-field shearing to characteristic stress. Comparing analytical estimates with numerical calculations, we find analytical estimates are accurate within a factor of two. However, discrepancies arise due to the analytical assumption that deviatoric stresses around the diapir are comparable to buoyancy stresses. Numerical results reveal significantly smaller deviatoric stresses. As deviatoric stresses govern stress-dependent, power-law viscosity, analytical estimates tend to overestimate stress weakening. We introduce a shape factor to improve accuracy. Additionally, we determine characteristic stresses for representative mantle and lower crustal flow laws and discuss practical implications in natural diapirism, such as sediment diapirs in subduction zones, magmatic plutons or exhumation of ultra-high-pressure rocks. Our study enhances understanding of diapir ascent velocities and associated stress conditions, contributing to a thorough comprehension of diapiric processes in geology.

Plain Language Summary

A diapir is a volume of rock that rises within a larger, denser rock mass due to its lower density and the force of gravity. Understanding the speed at which diapirs ascend is crucial for determining their significance in specific geologic settings, such as subduction zones. In this study, we use advanced computer simulations to calculate the ascent velocity of a spherical diapir within a denser surrounding material. The surrounding material is subjected to horizontal shearing, and its behavior resembles that of a nonlinear fluid, where its resistance to shear, known as viscosity, depends on the applied stress. By conducting three-dimensional computer simulations, we not only test the accuracy of existing mathematical equations commonly used to estimate diapir velocity but also make improvements to enhance their precision. These equations help us estimate how quickly diapirs rise in different geodynamic environments. By advancing our understanding of diapir ascent velocities, we gain valuable insights into the processes that shape our planet's geological features.

Keypoints

- 3D GPU-based numerical calculations of diapir velocities in power-law viscous fluid under far-field stress

- New analytical velocity estimates are controlled by two stress ratios and agree with numerical results

- Stress weakening in tectonically active regions can increase diapir velocity by several orders of magnitudes

2.1 Introduction

Diapirism is an important mechanism of heat and mass transport in the Earth (e.g., Ramberg, 1968; Whitehead et al., 1975; Schubert et al., 2001; Turcotte and Schubert, 2021). It mainly occurs in viscously deforming rock (e.g., Turcotte and Schubert, 2021), but can also be initiated in settings with frictional overburden (e.g., Poliakov et al., 1993, 1996). Diapirism can occur on various temporal and spatial scales and is a mechanism for the ascent of, for example, magma (e.g., Marsh, 1982; Rabinowicz et al., 1987; Cruden, 1988; Weinberg and Podladchikov, 1994, 1995; Miller and Paterson, 1999; Burov et al., 2003; Cruden and Weinberg, 2018; Michail et al., 2021), rock salt (e.g., Jackson et al., 1990; Poliakov et al., 1993; Schultz-Ela et al., 1993; Jackson and Vendeville, 1994), mud (e.g., Mazzini et al., 2009) or sediments buried at subduction zones (e.g., Gerya and Yuen, 2003; Behn et al., 2011; Marschall and Schumacher, 2012; Klein and Behn, 2021; Smye and England, 2023). Magma ascent by diapirism is, for example, an important mechanism contributing to the volcanic and igneous plumbing systems (e.g., Cruden and Weinberg, 2018). At subduction zones, for example, sediment diapirs, which detach from subducting slabs and rise into the above, hotter mantle wedge, are presumably the reason for the so-called sediment melt signature in arc lavas (e.g., Plank and Langmuir, 1993; Behn et al., 2011). Furthermore, diapirism was suggested as potential mechanism for the exhumation of some high- and ultra-high-pressure, (U)HP, terranes, for which very fast, > 1 cm/yr, exhumation velocities have been estimated (e.g., Burov et al., 2001, 2014; Little et al., 2011; Schmalholz and Schenker, 2016; Schwarzenbach et al., 2021). For all the various forms of diapirism, the ascent velocity of the diapir is the essential quantity to assess the importance of diapirism for specific geodynamic settings.

The simplest estimate for the ascent velocity of a diapir is given by the so-called Stokes law which is applicable for the ascent, or fall, of a rigid sphere in a denser, or lighter, linear viscous fluid (Stokes, 1850). However, diapirs in geodynamics are mostly not rigid and are commonly mechanically weaker than the surrounding rocks (e.g., Weinberg and Podladchikov, 1994). Furthermore, viscous deformation of natural rock surrounding a rising diapir can occur by dislocation creep, which is described by a non-linear, power-law viscous flow flaw (e.g., Weinberg and Podladchikov, 1994). In a power-law viscous fluid, the effective viscosity depends on the stress, or alternatively the strain rate, in the fluid (e.g., Fletcher, 1974; Schmalholz and Fletcher, 2011; Turcotte and Schubert, 2021). For rocks, higher stresses cause smaller effective viscosities (e.g., Hirth and Kohlstedt, 2003; Karato, 2008). Here, we refer to the decrease of the effective viscosity caused by an increase in stress as stress weakening (e.g., Christensen, 1983). For diapirism, there are two fundamental mechanisms by which the stress, and hence the effective viscosity, in rocks surrounding a diapir can change (Figure 2.1): (a) The rock unit in which the diapir is rising undergoes a far-field deformation, for example, due to horizontal simple shear in a strike-slip environment (e.g., Michail et al., 2021; Nahas et al., 2023) or corner flow in a mantle wedge (e.g., Klein and Behn, 2021). The far-field, or regional, stresses associated with the regional deformation can modify the effective viscosity of the rocks surrounding the diaper. (b) The deformation in the surrounding rocks, caused by the rising diapir, generates stress variations around the diapir (e.g., Weinberg and Podladchikov, 1994). Such local stress variations around the diapir are related to the diapir's buoyancy stress and cause variations in the effective viscosity of the surrounding rock.

Analytical estimates of the ascent velocity of a diapir in a power-law viscous fluid have been presented by Weinberg and Podladchikov (1994). Weinberg and Podladchikov (1994) show that the reduction of the effective viscosity due to local stress weakening is essential for magma diapirs to be able to ascent with velocities of 10 to 100 m/yr. Such high velocities are needed so that magma diapirs can reach the upper crust before solidification. Similar high velocities have been suggested for the rise of sediment diapirs across the mantle wedge,

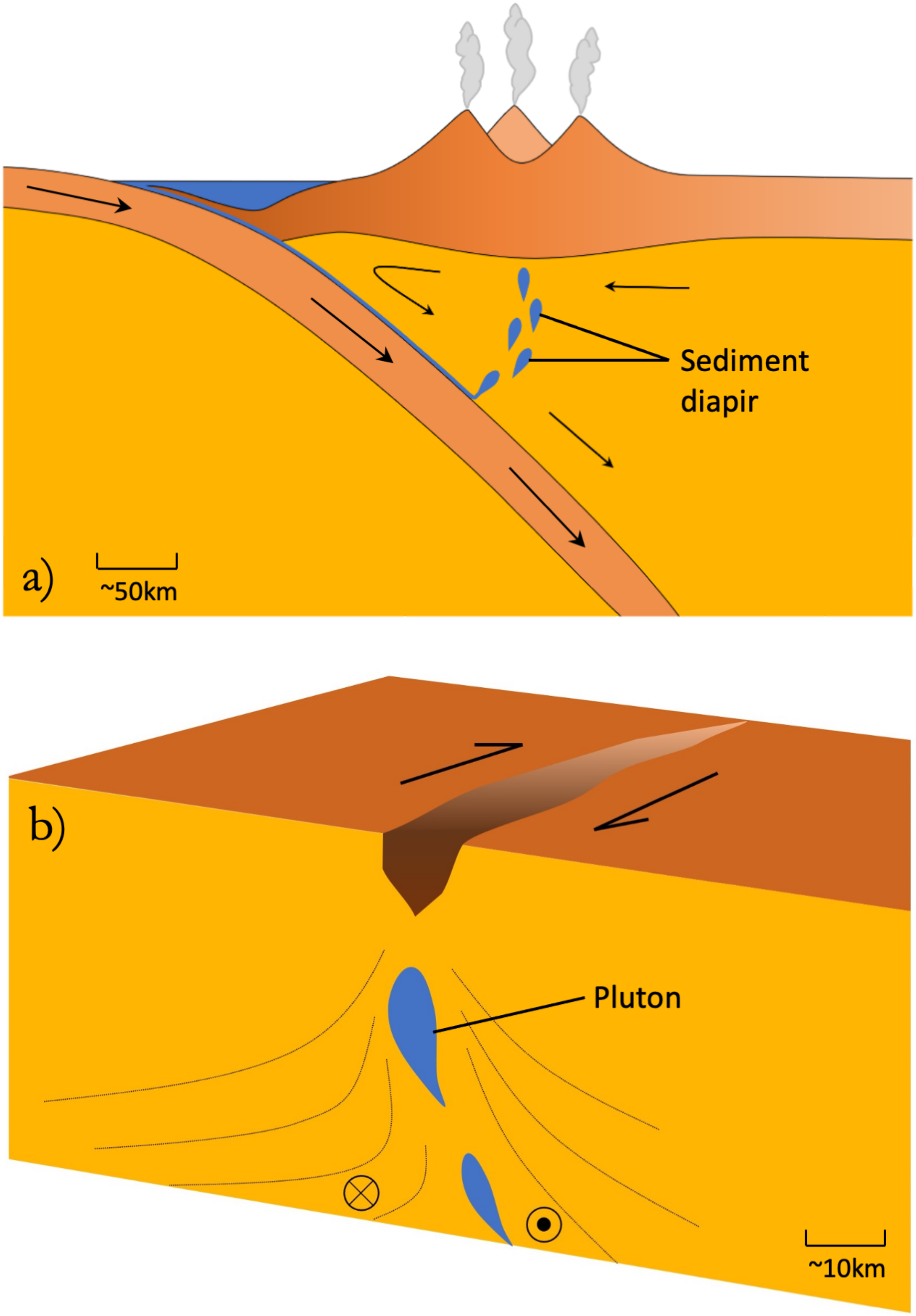

Figure 2.1: Sketch of two geodynamic settings in which diapirism can occur in deforming and stressed rock: (a) Sediment diapirs rising in a mantle wedge (after Klein and Behn, 2021). (b) Pluton rising in a crustal strike-slip zone (after Michail et al., 2021)

also enabled by stress weakening in power-law viscous mantle rocks (Klein and Behn, 2021). High velocities due to stress weakening in power-law viscous material are supported by two-dimensional (2D) numerical simulations of mantle convection (e.g., Larsen and Yeun, 1997). However, the analytical estimates derived by Weinberg and Podladchikov (1994) have never been tested and compared to results of full 3D numerical calculations.

Here, we perform full 3D numerical calculations to quantify the ascent velocity of a weak diapir in a stronger and deforming fluid. The flow law of the surrounding fluid is a combination of linear and power-law viscous flow. Such combined flow law can describe rock deformation by a combination of diffusion and dislocation creep (e.g., Karato, 2008). Our numerical algorithm is based on the staggered finite difference method and employs an iterative solution strategy. We programmed the algorithm in the Julia language and use GPUs for the numerical solution. In the numerical calculations, we consider effective viscosity variations in the surrounding fluid due to both regional stresses and local stress variations around the diaper. The regional stresses are caused by strike-slip shearing and the local stresses are caused by the upward movement of the diaper. We further elaborate the analytical estimates of Weinberg and Podladchikov (1994) by (a) implementing a combined linear and power-law viscous flow law, and (b) considering both regional tectonic stress and local buoyancy stress. We then compare the analytical estimates with the 3D numerical calculations.

The aims of our study are to (a) elaborate, test and improve analytical estimates for diapiric ascent velocities in a deforming power-law viscous fluid, (b) make a systematic quantification of the ascent velocity based on two dimensionless stress ratios, and (c) discuss the applicability of the results to typical crustal and mantle flow laws as well as to various diapir scenarios.

2.2 Model

2.2.1 Flow Law and Effective Viscosity

We consider a non-linear, power-law viscous flow law of the general form (Fletcher, 1974; Karato, 2008):

$$\dot{\epsilon} = \frac{1}{2}B\tau^n \ , \tag{2.1}$$

with $\dot{\epsilon}$ being the deviatoric strain rate, τ being the deviatoric stress, n being the power-law stress exponent and B being a material parameter. All symbols used in the text are listed in Table 2.1.

We reformulate Equation 2.1 to:

$$\tau \ = \ 2B^{-1}\tau^{(1-n)}\dot{\epsilon} \ . \tag{2.2}$$

Next, we multiply the right-hand side of Equation 2.2 by $\tau_{\mathrm{C}}^{(1-n)}/\tau_{\mathrm{C}}^{(1-n)}$, with τ_{C} being a characteristic stress magnitude that marks the stress at which the deformation behavior changes from diffusion to dislocation creep, and rearrange Equation 2.2 to:

$$\tau = 2\eta\left(\frac{\tau}{\tau_{\mathrm{C}}}\right)^{(1-n)}\dot{\epsilon} \ , \tag{2.3}$$

where $\eta = B^{-1}\tau_{\mathrm{C}}^{(1-n)}$. Introducing the characteristic stress τ_{C} has two benefits: (a) The parameter η has units of a viscosity, i.e. Pa·s, and (b) the impact of τ on the flow law is normalized by the magnitude of τ_{C}. The additional usefulness of introducing τ_{C} is presented further below. Equation 2.3 reduces to a linear viscous flow law for $n = 1$. A linear flow law typically describes diffusion creep (e.g., Karato, 2008; Turcotte and Schubert, 2021). A power-law viscous flow law typically describes dislocation creep (e.g., Karato,

Table 2.1: Mathematical Symbols Used in the Text.

Symbol	Name or Definition	Unit
L	Width, height and length of the model domain	[m]
R	Radius of the spherical diapir	[m]
d	Distance of ascent	[m]
η, η_{m}	Linear viscosity, Viscosity of the surrounding medium	[Pa·s]
$\eta_{\mathrm{E}}, \eta_{\mathrm{EL}}$	Effective viscosity, Effective viscosity for local stress	[Pa·s]
τ_{PT}	Pseudo-time	[s]
$t_{\mathrm{a}}, t_{\mathrm{c}}$	Magmatic diapir: time of ascent, cooling time	[s]
$\dot{\epsilon}$	Deviatoric strain rate	[1/s]
g	Gravity acceleration	[m/s^2]
ρ	Density	[kg/m^3]
$\Delta\rho$	Density difference ($\rho_{\mathrm{medium}} - \rho_{\mathrm{sphere}}$)	[kg/m^3]
$\tilde{\rho}$	Pseudo-density	[kg/m^3]
V_{S}	Far-field shearing velocity	[m/s]
V, V_{vert}	Velocity of ascent	[m/s]
V_0	Reference velocity	[m/s]
$V_{\mathrm{R}}, V_{\mathrm{L}}, V_{\mathrm{D}}$	Velocity estimates considering regional, local, and combined stresses	[m/s]
V_{C}	Critical velocity	[m/s]
P	Pressure (mean stress)	[Pa]
$\tau, \tau_{\mathrm{vert}}$	Deviatoric stress, Vertical deviatoric stress	[Pa]
τ_{II}	Square root of second invariant of deviatoric stress tensor	[Pa]
τ_{C}	Characteristic stress	[Pa]
$\tau_{\mathrm{R}}, \tau_{\mathrm{L}}$	Regional stress, Local stress	[Pa]
$\sigma, \sigma_{\mathrm{vert}}$	Total stress, Vertical total stress	[Pa]
$\tilde{K}, \tilde{G}$	Pseudo-bulk and pseudo-shear modulus	[Pa]
Ω	Viscosity ratio ($\eta_{\mathrm{medium}}/\eta_{\mathrm{sphere}}$)	[$-$]
n	Power-law stress exponent	[$-$]
m	$1/n$, inverse power-law stress exponent	[$-$]
B	Material parameter	[$-$]
$C_{\mathrm{R}}, C_{\mathrm{L}}$	Model parameters	[$-$]
G, M, X_{sol}	Model parameters depending on the parameter m	[$-$]
S	Shape factor	[$-$]
δ_{ij}	Kronecker delta	[$-$]
$\Delta\rho g R/\tau_{\mathrm{R}}$	Argand number	[$-$]

2008; Turcotte and Schubert, 2021) but can also effectively describe exponential flow laws describing, for example, low temperature plasticity (e.g., Schmalholz and Fletcher, 2011). In nature, both creep mechanisms can occur simultaneously and, hence, a combination of a linear and a power-law viscous flow law is often applied in geodynamic applications (e.g., Karato, 2008). The effective viscosity, η_{E}, for such combined flow law is represented by the pseudo-harmonic mean of the linear (Equation 2.3 with $n = 1$) and power-law (Equation 2.3 with $n > 1$) viscosities and is given by (e.g., Schmalholz and Podladchikov, 2013; Gerya, 2019):

$$\eta_{\mathrm{E}} = \frac{\eta}{1 + \left(\frac{\tau}{\tau_{\mathrm{C}}}\right)^{(n-1)}} \; . \tag{2.4}$$

The general flow law we use in this study reads (e.g., Schmalholz and Podladchikov, 2013; Gerya, 2019):

$$\tau = 2\eta_{\mathrm{E}}\dot{\epsilon} \; . \tag{2.5}$$

In the combined linear and power-law viscous flow law, the magnitude of τ_{C} determines the transition from a linear viscous flow to a power-law viscous flow. Examples of magnitudes of τ_{C} for crustal and mantle flow laws, determined by rock deformation experiments, are presented in the Discussion (Section 2.4).

2.2.2 Analytical Estimates for Diapir Ascent Velocity in Deforming Power-Law Viscous Medium

The ascent velocity of a diapir is controlled mostly by the effective viscosity of the surrounding medium and not by the effective viscosity of the material forming the diapir (e.g., Weinberg and Podladchikov, 1994). We assume that the effective viscosity of the surrounding medium, η_{E}, is given by Equation 2.4. We also assume that the effective viscosity of the diapir is smaller than the effective viscosity of the surrounding medium by a factor Ω, which is termed the viscosity ratio. For a spherical diapir with an effective viscosity that is smaller than the effective viscosity of the surrounding medium, the velocity of ascent, V, is given by (e.g., Hadamard, 1911; Rybczynski, 1911; Weinberg and Podladchikov, 1994):

$$V = \frac{1}{3}\frac{\Delta\rho g R^2}{\eta_\mathrm{E}} C_\mathrm{R} \ , \tag{2.6}$$

where $\Delta\rho$ is the density difference between the surrounding medium and the rising diapir, g is the gravitational acceleration, R is the radius of the sphere and the constant C_R is defined as (e.g., Weinberg and Podladchikov, 1994)

$$C_\mathrm{R} = \frac{\eta_\mathrm{E} + \eta_\mathrm{E}/\Omega}{\eta_\mathrm{E} + \frac{3}{2}\eta_\mathrm{E}/\Omega} = \frac{1 + 1/\Omega}{1 + 3/(2\Omega)} \ . \tag{2.7}$$

If $\tau/\tau_\mathrm{C} = 0$, then $\eta_\mathrm{E} = \eta$ (see Equation 2.4) and the velocity V corresponds to the ascent velocity of a linear viscous diapir rising in a linear viscous medium. We will use further below this velocity for linear viscous flow as reference velocity, V_0, to normalize the ascent velocities for power-law viscous flow. The reference velocity is

$$V_0 = \frac{1}{3}\frac{\Delta\rho g R^2}{\eta} C_\mathrm{R} \ . \tag{2.8}$$

Since for a power-law viscous flow law η_E depends on τ, the value of τ has to be estimated to calculate V. We consider two scenarios to estimate V: (a) There is a homogeneous regional deformation in the surrounding medium, for example, a shear deformation in a strike-slip environment, which generates a regional stress τ_R. This value of τ_R is used to calculate the effective viscosity of the surrounding medium, $\eta_\mathrm{E}(\tau = \tau_\mathrm{R})$, and to calculate the rising velocity under a regional stress field, V_R, with Equations 2.6 and 2.7, so that

$$V_\mathrm{R} = V(\tau = \tau_\mathrm{R}) \ . \tag{2.9}$$

(b) Local stress variations around the diapir are caused by the diapir rising in a deformable medium. We assume that these local stress magnitudes, τ_{L}, have the same magnitude as the buoyancy stress of the diapir, $\Delta\rho g R$ (Weinberg and Podladchikov, 1994). The rising velocity for which the impact of local stress variations in the surrounding medium are considered, V_{L}, has been derived by Weinberg and Podladchikov (1994) for a power-law viscous flow law and is given by:

$$V_{\mathrm{L}} = \frac{1}{3}\frac{\Delta\rho g R^2}{\eta_{\mathrm{EL}}} C_{\mathrm{L}} \ , \tag{2.10}$$

where

$$C_{\mathrm{L}} = \left(\frac{G + 1/\Omega}{X_{\mathrm{sol}}(GM + 3/(2\Omega))}\right)^{n} \ , \tag{2.11}$$

with

$$\begin{aligned} G &= 2.39 - 5.15m + 3.77m^2 \\ M &= 0.76 + 0.24m \\ X_{\mathrm{sol}} &= 1.3(1 - m^2) + m \ , \end{aligned} \tag{2.12}$$

where $m = 1/n$. The parameter C_{L} is only a function of the two dimensionless parameters n and Ω. The effective viscosity η_{EL} for local stress variations is:

$$\eta_{\mathrm{EL}} = 2S\eta \left(\frac{6\tau_{\mathrm{C}}}{\Delta\rho g R}\right)^{(n-1)} \ , \tag{2.13}$$

where η is the viscosity parameter inside the effective viscosity (Equation 2.4) of the surrounding medium and S is a shape factor. The shape factor S is a fitting parameter that can be adapted to better fit the numerical results. The value of S will be discussed in Section 2.3.3. Finally, the velocity estimate for a weak diapir rising in a deforming medium under a regional stress with a flow law combining diffusion and dislocation creep is:

$$V_{\mathrm{D}} = V_{\mathrm{R}} + V_{\mathrm{L}}\ . \tag{2.14}$$

We normalize V_{D} by V_0 which yields

$$\frac{V_{\mathrm{D}}}{V_0} = \frac{V_{\mathrm{R}}}{V_0} + \frac{V_{\mathrm{L}}}{V_0} = 1 + \left(\frac{\tau_{\mathrm{R}}}{\tau_{\mathrm{C}}}\right)^{(n-1)} + \frac{3}{6^n S}\frac{C_{\mathrm{L}}}{C_{\mathrm{R}}}\left(\frac{\Delta\rho g R}{\tau_{\mathrm{C}}}\right)^{(n-1)}\ . \tag{2.15}$$

We will test the analytical estimate for V_{D} with 3D numerical calculations which are described below.

2.2.3 3D Mathematical Model

We assume incompressible flow under gravity. The components of the total stress tensor, σ_{ij}, are decomposed into a pressure (mean stress), P, and deviatoric stress tensor components, τ_{ij}, so that $\sigma_{ij} = -\delta_{ij}P + \tau_{ij}$, whereby indexes i and j run from 1 to 3 and indicate the three spatial directions, and δ_{ij} is the Kronecker delta (Turcotte and Schubert, 2021). The equations for the conservation of mass for an incompressible fluid and for the conservation of linear momentum are:

$$0 = \frac{\partial V_i}{\partial x_i} \tag{2.16}$$

$$0 = \frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial P}{\partial x_i} + \rho g_i\ , \tag{2.17}$$

where V_i is the component of the velocity vector in direction x_i, ρ the density and g_i the gravity vector component. Components of the deviatoric stress tensor are defined as:

$$\tau_{ij} = 2\eta_{\mathrm{E}}\dot{\epsilon}_{ij} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right)\right) \ , \tag{2.18}$$

where $\dot{\epsilon}_{ij}$ are the components of the deviatoric strain rate tensor and η_{E} is defined in Equation 2.4. For the studied 3D flow, the value of τ used in Equation 2.4 is quantified by the square root of the second stress invariant

$$\tau_{\mathrm{II}} = \sqrt{1/2 * (\tau_{xx}^2 + \tau_{yy}^2 + \tau_{zz}^2) + \tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2} \ , \tag{2.19}$$

which is independent of the coordinates system.

2.2.4 Numerical Method

To numerically solve the system of governing equations (Equations 2.16 and 2.17) we discretize the differential equations using the finite difference method on a staggered grid with constant spacing (e.g., Gerya, 2019; Räss et al., 2022). We apply the pseudo-transient (PT) method to solve the discretized, non-linear equations in a matrix free fashion (e.g., Räss et al., 2022; Wang et al., 2022). The PT method is one of many iterative methods that exist since the 1950's (Frankel, 1950) and is used to solve stationary problems. The concept of the PT method is to add a pseudo-time derivative to the steady-state governing equations (e.g., Räss et al., 2022):

$$\begin{aligned} \frac{1}{\tilde{K}}\frac{\partial P}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial V_i}{\partial x_i} \\ \tilde{\rho}\frac{\partial V_i}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial P}{\partial x_i} + \rho g_i \end{aligned} \tag{2.20}$$

$$\frac{1}{2\tilde{G}}\frac{\partial \tau_{ij}}{\partial \tau_{\mathrm{PT}}} \quad = \quad -\frac{\tau_{ij}}{2\eta_{\mathrm{E}}} + \frac{1}{2}(\nabla_i V_j + \nabla_j V_i) \ ,$$

where $\tilde{K}$, $\tilde{\rho}$ and $\tilde{G}$ are numerical parameters and τ_{PT} is a pseudo-time. $\tilde{K}$ and $\tilde{G}$ can be considered as pseudo-bulk and pseudo-shear modulus respectively, and $\tilde{\rho}$ is a pseudo-density. With the pseudo-time derivatives, Equations 2.20 can be considered as pseudo-acoustic and inertial approximations of the mass and momentum balance equations, respectively. The initial guess of the pressure and velocity fields do not satisfy the steady state equations, hence the PT method consists in iterating until the imbalance is sufficiently small, that is when the PT time derivatives (Equations 2.20) are sufficiently small and have all reached a specific tolerance value. A detailed description of the applied PT method with examples of 3D calculations is given in Räss et al. (2022). For completeness, we present a numerical resolution and tolerance test in Appendix D. For the presented results, we used a numerical resolution of 207×207×207 and a tolerance for the iterative solver of 5×10^{-7}. The results of the resolution and tolerance test show that these values provide velocities which do not change significantly anymore for higher resolution or smaller tolerance.

We have also numerical algorithms for the studied 3D power-law viscous flow which are based on the governing equations formulated in cylindrical and spherical coordinates. These equations are given in Appendices A and B. To test our numerical implementation, we will perform numerical calculations for the same model configuration based on the governing equations in Cartesian, cylindrical and spherical coordinates. In the limit of negligible curvature and for the same boundary and initial conditions the numerical results based on cylindrical and spherical coordinates must be identical to the results based on Cartesian coordinates. The model configuration for cylindrical and spherical coordinates is described in Appendix C.

2.2.5 Model Configuration

The model configuration is a cube of dimension $[-L/2, L/2] \times [-L/2, L/2] \times [-L/2, L/2]$ containing a sphere of diameter $L/3$ at its center, with L indicating the model width, length and height (Figure 2.2). The viscosity parameter, η, of the sphere is always 100 times smaller than the one of the surrounding fluid ($\Omega = 100$). The applied flow law is the combined flow law given in Equation 2.4 and the power-law exponent is always 5. The sphere is always less dense than the surrounding fluid and we vary $\Delta\rho$ for different calculations.

We apply horizontal far-field simple shearing parallel to the horizontal x-direction (Figure 2.2). The boundary conditions are (a) free slip on the top and bottom faces of the cube, (b) on the lateral sides parallel to the shearing the velocities in y- and z-direction are zero and in the x-direction they correspond to the applied far-field shearing velocity V_s ($V_x = -V_s$ for $y = -L/2$ and $V_x = V_s$ for $y = L/2$), and (c) on the lateral sides orthogonal to the shearing the velocities in y- and z-direction are zero and the velocities in the x-direction vary linearly in the y-direction from $-V_s$ to V_s.

The model is configured in dimensionless form and also results will be displayed in dimensionless form. For the non-dimensionalization, we use three characteristic scales: one scale for length, which is the radius of the sphere R; one scale for stress, which is the buoyancy stress of the sphere $\Delta\rho g R$; and one scale for viscosity, which is the applied value of η in the surrounding medium, termed η_{m}. To describe the results, we will further use two dimensionless ratios, namely the ratio of the applied regional stress to characteristic stress, $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$, and the ratio of buoyancy stress to characteristic stress, $\Delta\rho g R/\tau_{\mathrm{C}}$. τ_{R} is the magnitude of the homogeneous shear stress in the model when the sphere has the same material properties as the surrounding material. Hence, τ_{R} represents the far-field stress which is not affected by the weak sphere.

The aims of the simulations are (a) to compare magnitudes of buoyancy stress and deviatoric stress around the sphere, (b) to perform systematic simulations to quantify the ascent velocity of the sphere in a strike-slip environment, by varying $\Delta\rho$ and τ_C (Equation 2.15), and (c) to compare the numerically calculated velocities with the analytical estimates from Equation 2.15 and to improve these estimates if possible.

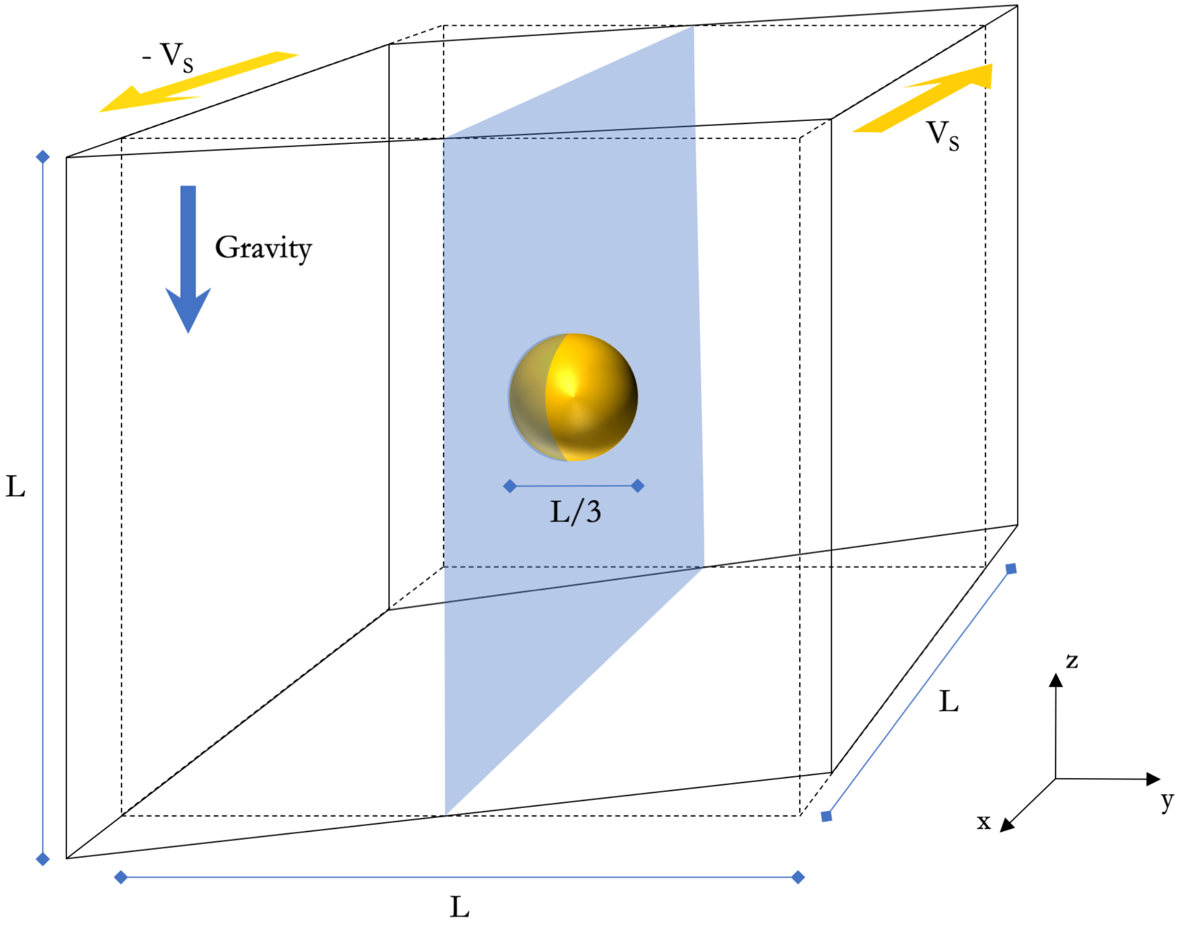

Figure 2.2: Model configuration: cube of size $[-L/2; L/2] \times [-L/2; L/2] \times [-L/2; L/2]$, with a less dense and weaker spherical inclusion of diameter $L/3$ at the model center. The entire model cube is sheared horizontally, parallel to the x-direction, and gravity acts in the vertical, z-direction.

2.3 Results

2.3.1 Distribution of Stress, Pressure and Effective Viscosity

For each presented simulation, we have calculated one time step to obtain the full 3D velocity and stress field. First, we show the distribution of the resulting effective viscosity, η_{E}, the second stress invariant, τ_{II}, and the pressure, P, for a representative simulation (Figure 2.3). In Figure 2.3, 1/8th of the cubic model domain is presented. The sphere is less dense than its surrounding and, hence, moves upwards as indicated by the velocity arrows in Figure 2.3b. In the following, we refer to the sphere as diapir. The applied simple shear is visible on the horizontal slice through the model domain (Figure 2.3b). The effective viscosity shows a decrease of about one order of magnitude directly above the diapir (Figure 2.3b). There are two regions on the sides of the diapir where the effective viscosity is even larger than the ambient viscosity. The variations in η_{E} can be explained by the distribution of τ_{II} (Figure 2.3c). Values of η_{E} are directly linked to τ_{II} (Equation 2.4): where the stresses are large, such as above the diapir, the effective viscosity decreases and where stresses are smaller, the effective viscosity does not change or even increases. The large stresses above the diapir are due to its upwards movement.

Figure 2.3d depicts the pressure field. We only consider the dynamic part of the pressure, which means that we subtract the lithostatic pressure, because only deviations from the static pressure field can cause movement. An interesting feature is the strong pressure gradient inside the diapir, because the pressure is decreasing with depth which is opposite to the lithostatic pressure. Similar to the deviatoric stress, the pressure in the surrounding medium is largest directly above the diapir.

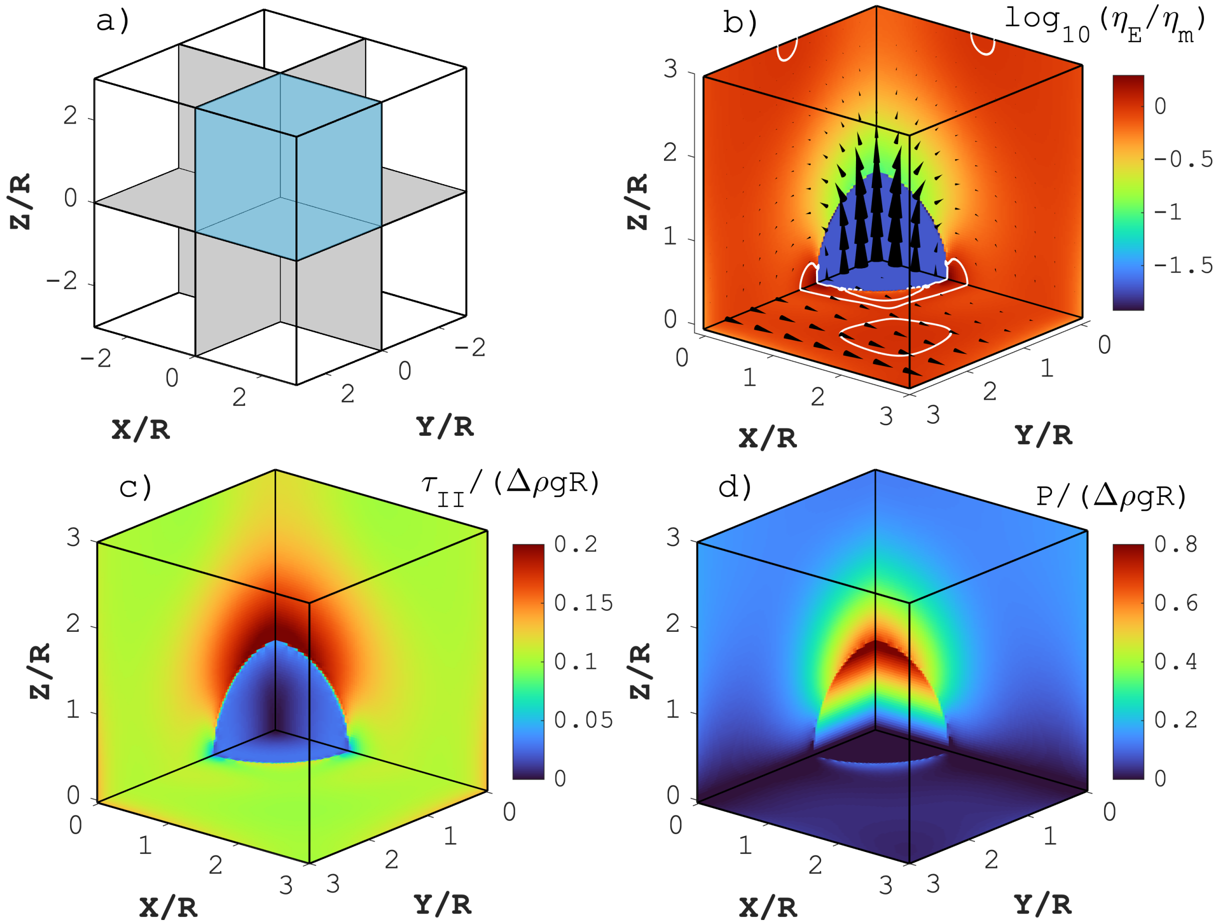

Figure 2.3: Representative numerical results for $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} = 1$ and $\Delta\rho g R/\tau_{\mathrm{C}} = 10$: (a) Location of $1/8^{th}$ of the model shown in panels (b-d). (b) Effective viscosity η_{E} normalized by η_{m}, the linear viscosity of the surrounding medium. Arrows indicate the velocity field and white contours highlight $log_{10}(\eta_{\mathrm{E}}/\eta_{\mathrm{m}}) = 0$. (c) Second invariant of deviatoric stress, τ_{II}, normalized by the buoyancy stress $\Delta\rho g R$. (d) Pressure, P, normalized by buoyancy stress $\Delta\rho g R$.

2.3.2 Stress Decomposition and Magnitudes

The total vertical stress is decomposed into the pressure and the vertical deviatoric stress, $\sigma_{\mathrm{vert}} = -P + \tau_{\mathrm{vert}}$. We quantify σ_{vert}, P, and τ_{vert} and compare the magnitudes with the buoyancy stress (Figure 2.4). This quantification is important because the analytical estimates for the diapir velocity use the buoyancy stress as proxy for the deviatoric stress which is used in the power-law flow law.

The vertical continuity of σ_{vert} across the diapir boundary in the horizontal middle of the model (at $Y = 0$, Figure 2.4d) results from the requirement of the vertical force balance. In contrast, both P and τ_{vert} can be discontinuous across the diapir boundary (Figure 2.4e and f). Indeed, P and τ_{vert} show a discrete jump across the boundary of the diapir. The absolute maximal values of σ_{vert} are close to the value of $\Delta\rho g R$, since the maximal value of their ratio is approximately one (Figure 2.4a). τ_{vert} is essentially zero inside the diapir since the effective viscosity inside the diapir is 100 times smaller than the one of the surrounding medium. Consequently, the absolute magnitudes of P are high inside the diapir at the top and bottom, in order to generate a continuous σ_{vert} required by the vertical force balance. Maximal values of P inside the diapir are, hence, close to the magnitudes of $\Delta\rho g R$.

Outside the diapir, directly above and below, maximal magnitudes of τ_{vert} are significantly smaller than magnitudes of σ_{vert} at the same positions. The effective viscosity in the analytical estimate is calculated with the magnitude of $\Delta\rho g R$ while in the numerical simulation it is controlled by the correct magnitude of τ_{II}. The magnitude of τ_{II} is smaller than $\Delta\rho g R$ (Figure 2.3c) and τ_{II} is also strongly variable around the diaper. We, therefore, expect that the analytical estimates for the diapir velocity will be different to the numerically calculated ones, because the stress magnitude which controls the effective viscosity is different in the analytical estimates compared to the numerical simulation.

The results presented in Figure 2.4 are reproduced by the numerical calculations based on cylindrical and spherical coordinates and are presented in Appendix C. The agreement between results calculated by three different numerical algorithms indicates the correct numerical implementation of the governing equations.

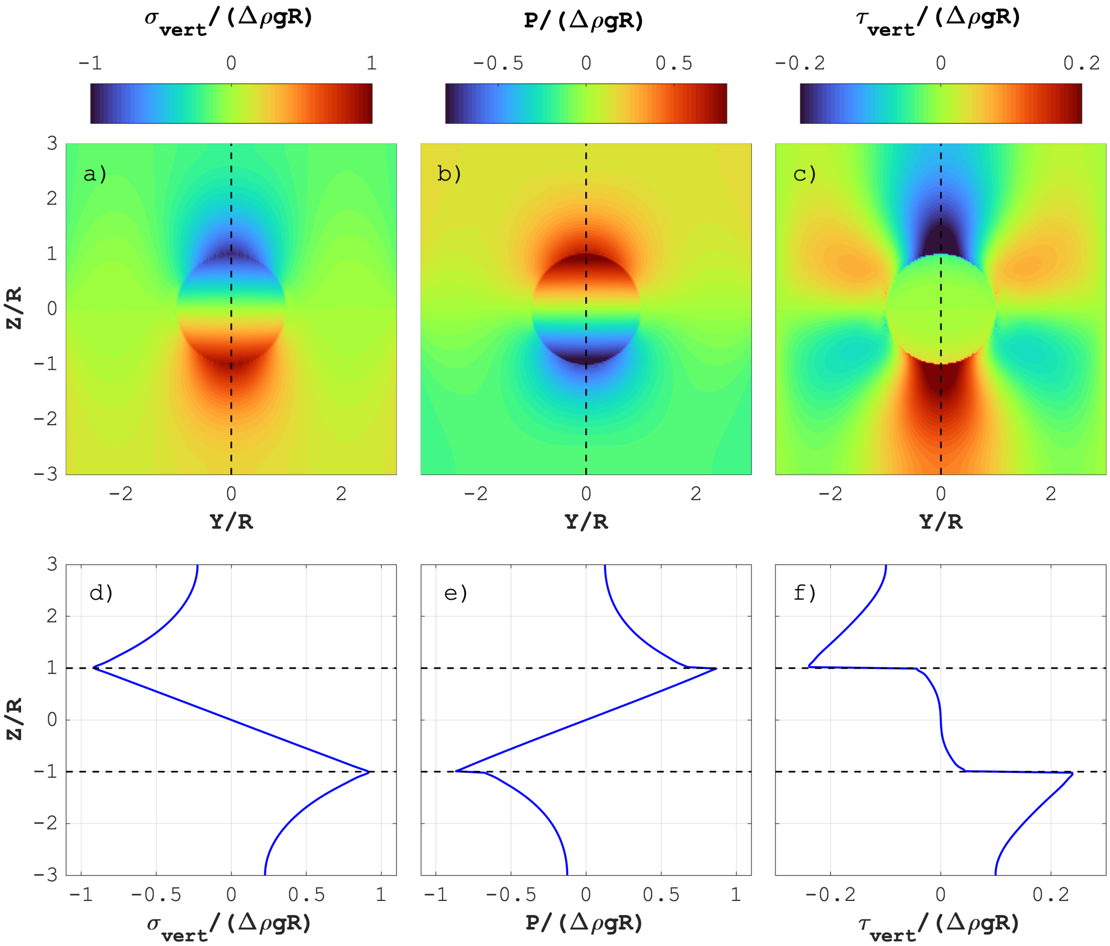

Figure 2.4: Numerical results for $\tau_R/\tau_C = 1$ and $\Delta\rho gR/\tau_C = 10$. (a-c) Vertical cross sections at position $X/R = 0$ (see Figure 2.3a), and (d-f) vertical profiles at position $X/R = 0$ and $Y/R = 0$. (a, d) display vertical total stress, (b, e) pressure, and (c, f) vertical deviatoric stress. The dashed line in panels (a-c) marks the position of profiles (d-f). All stresses are normalized by the buoyancy stress $\Delta\rho gR$.

2.3.3 Comparison of Analytical and Numerical Ascent Velocities

We compare the analytical estimates for the ascent velocity of a weak and less dense sphere, Equation 2.15, with our numerical results. The analytical estimates (details in Section 2.2.2) only provide the vertical velocity of the raising sphere and do not provide the spatial distribution of stresses. Hence, for each numerical simulation, we select the maximum vertical velocity obtained for the diapir and consider this velocity as the ascent velocity of the diaper. We normalize the vertical velocities by the corresponding values of V_0 which is the velocity of a linear viscous diapir rising in a linear viscous medium (see Equation 2.8).

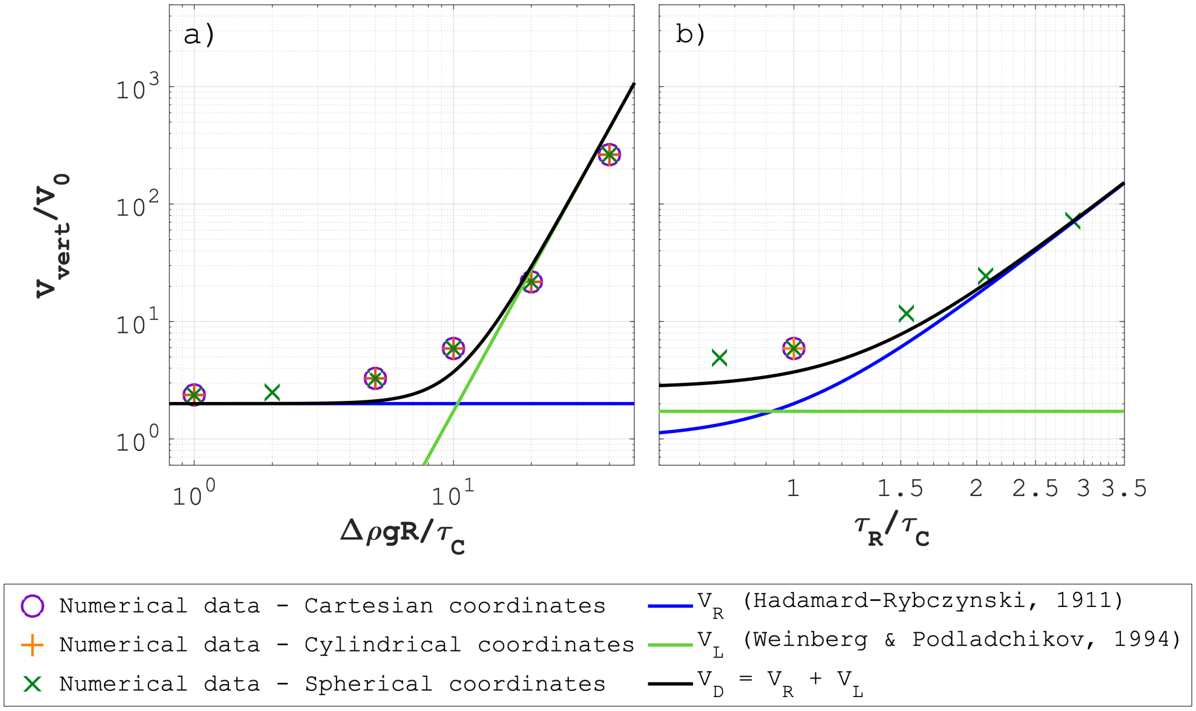

Figure 2.5: Comparison of numerical results (symbols, see legend) with analytical estimates from Equation 2.15 (lines, see legend). Analytical estimates are presented in Section 2.2.2. Vertical axis is the ascent velocity normalized by V_0 (see Equation 2.8). Horizontal axis is in (a) $\Delta\rho gR/\tau_{\mathrm{C}}$ for a value of $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} = 1$, and in (b) $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ for $\Delta\rho gR/\tau_{\mathrm{C}} = 10$.

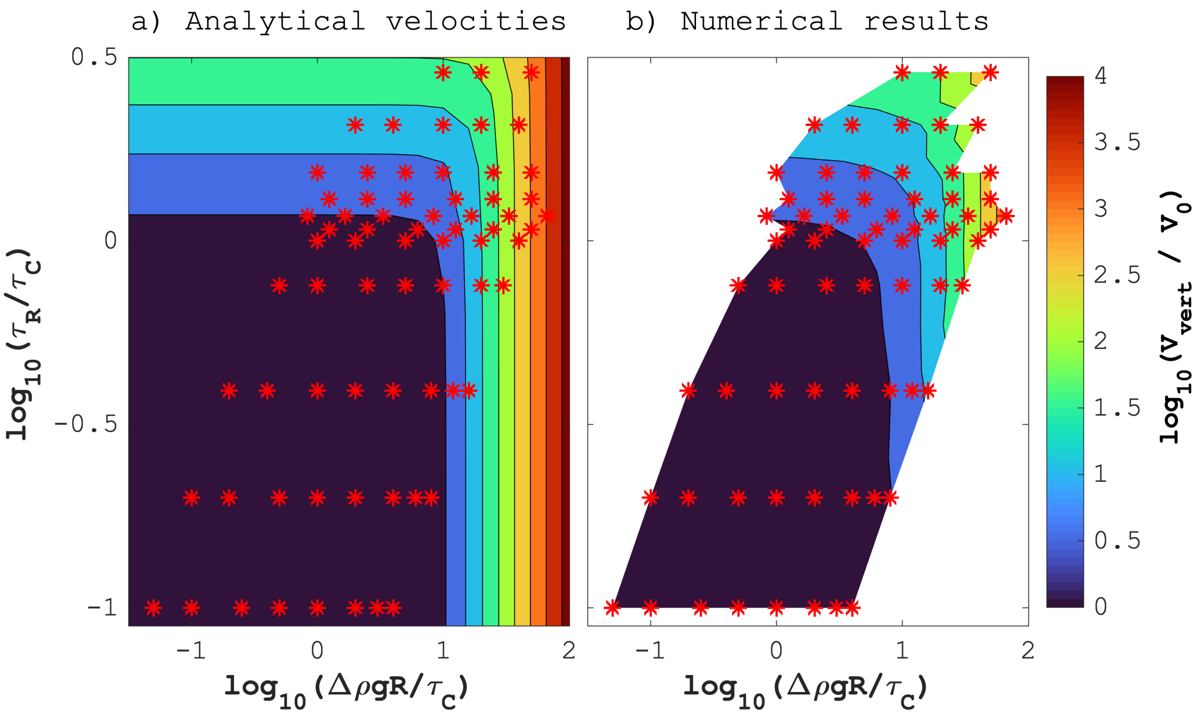

Figure 2.6: Analytical and numerical ascent velocities for a systematic variation of $\Delta\rho gR/\tau_\mathrm{C}$ and $\tau_\mathrm{R}/\tau_\mathrm{C}$. Ascent velocities are normalized by V_0 (see Equation 2.8). (a) Analytical estimates and (b) numerically calculated velocities. The stars represent the values of $\Delta\rho gR/\tau_\mathrm{C}$ and $\tau_\mathrm{R}/\tau_\mathrm{C}$ for which numerical calculations were performed.

The ascent velocity depends on the two stress ratios $\tau_\mathrm{R}/\tau_\mathrm{C}$ and $\Delta\rho gR/\tau_\mathrm{C}$ (Equation 2.15). Figure 2.5 presents the comparison between analytical estimates and the numerical results. Figure 2.5a displays vertical velocities of the diapir for various values of $\Delta\rho gR/\tau_\mathrm{C}$ and a fixed value $\tau_\mathrm{R}/\tau_\mathrm{C} = 1$. For $\Delta\rho gR/\tau_\mathrm{C} < \sim 10$ the velocity is controlled by V_R, for which the regional stress controls the effective viscosity, while for $\Delta\rho gR/\tau_\mathrm{C} > \sim 10$ it is controlled by V_L, for which the buoyancy stress controls the effective viscosity (see Section 2.2.2). The velocity is constant in the domain dominated by the regional stress and increases significantly in the buoyancy dominated domain. The numerical results agree with V_R and they capture the change in slope of the velocity with increasing $\Delta\rho gR/\tau_\mathrm{C}$. However, for $\Delta\rho gR/\tau_\mathrm{C} > \sim 10$ the numerical velocities are smaller than the analytically estimated ones.

For $\Delta\rho g R/\tau_{\mathrm{C}} > \sim 10$, the ascent velocities vary by approximately two orders of magnitude while applied values of $\Delta\rho g R/\tau_{\mathrm{C}}$ vary by a factor of approximately 4 only.

Figure 2.5b displays the vertical velocity for various values of $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ and a fixed value of $\Delta\rho g R/\tau_{\mathrm{C}} = 10$. For $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} < \sim 1$ the velocity is controlled by V_{L} while for $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} > \sim 1$ it is controlled by V_{R}. For $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} > \sim 1$ the velocities strongly increase with increasing $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$.

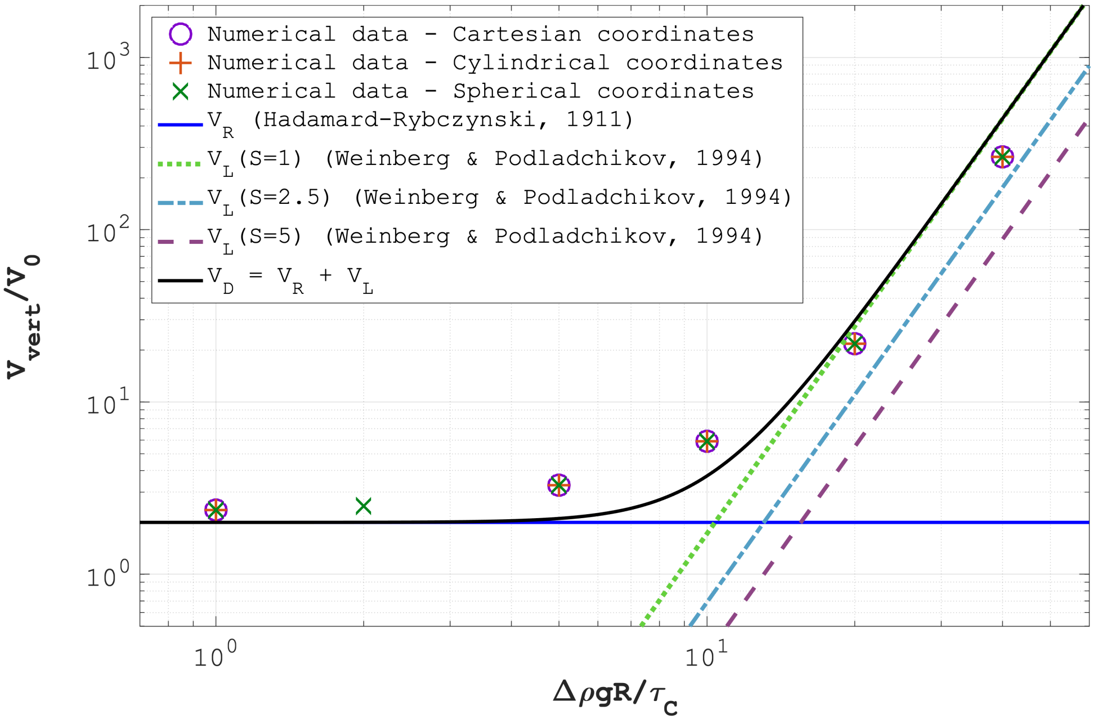

Figure 2.7: Comparison of numerically and analytically calculated ascent velocities for different shape factors S (see Equation 2.13 and legend). X-axis displays $\Delta\rho g R/\tau_{\mathrm{C}}$ and the vertical axis the ascent velocity normalized by V_0 (see Equation 2.8). Results are obtained for $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} = 1$. Only V_{L} depends on the shape factor.

We also performed a systematic comparison between the analytically estimated and the numerically calculated velocities by varying $\Delta\rho g R/\tau_{\mathrm{C}}$ and $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ (Figure 2.6). Figure 2.6a and b display the vertical velocities of the diapir obtained with the analytical estimates and the numerical simulations, respectively. The numerical results show the same trend of the velocity with varying values of $\Delta\rho g R/\tau_{\mathrm{C}}$ and $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ as the analytical estimates.

For normalized velocities $> \sim 10^4$, the numerical algorithm did not converge anymore due to the significant nonlinearities and the associated significant variations of the effective viscosity around the diapir.

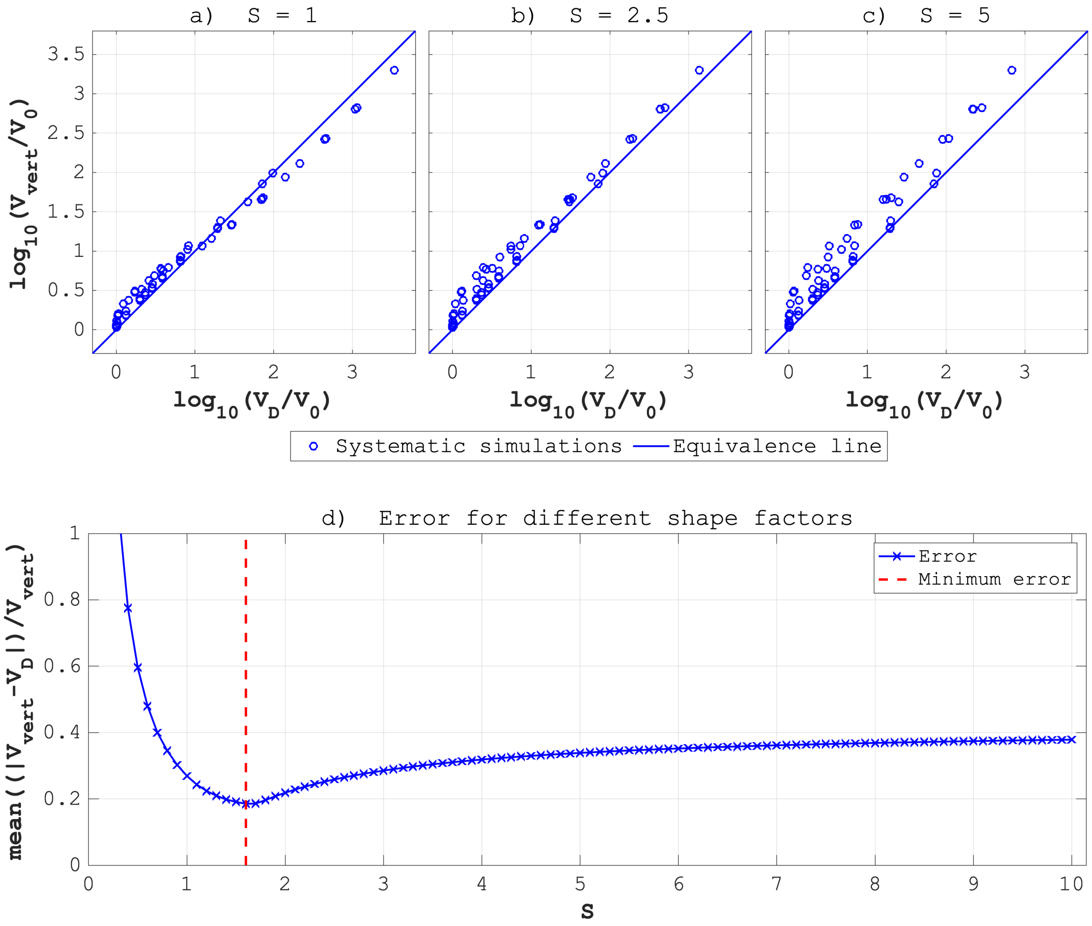

Figure 2.8: Comparison of analytically estimated velocities on the horizontal axis and the numerically calculated velocities on the vertical axis for different shape factors. (a) $S = 1$, (b) $S = 2.5$, and (c) $S = 5$. The solid line represent the equivalence between analytical and numerical results. (d) The average relative error of the analytical estimates compared to the numerical results for values of S between 0.25 and 10. The vertical red dashed line indicates the minimum relative error of $\approx$18 % for $S = 1.6$.

Figure 2.7 is similar to Figure 2.5a, but shows analytical estimates for different shape factors, S (see Equation 2.13). The value $S = 1$ was used in the original derivation of Weinberg and Podladchikov (1994). Increasing S allows to better predict the ascent velocity in the buoyancy dominated deformation regime, that is for $\Delta\rho g R/\tau_{\mathrm{C}} > 10$. However, too large values of S lead to an underestimation of the velocities. For three values of S we present the correspondence between the numerical and the analytical results. For $S = 1$, analytical estimates tend to overestimate the large velocities, for $S = 2.5$ the estimations fit better and for $S = 5$ the analytical estimates generally underestimate the ascent velocity.

A plot of all the numerically calculated velocities versus the corresponding analytical estimates, for the same parameters, shows that the analytical estimates capture well the first order trend of the numerical results (Figure 2.8a to c). The maximal relative error between the analytical estimate and an individual numerical result is only 72% for $S = 1$. Hence, all analytical estimates deviate by less than a factor of 2 from the numerical results. The relative error is calculated by $(|V_{num} - V_{ana}|)/V_{num} * 100$, hence an error of 100% means a deviation by a factor of 2. We varied S between 0.25 and 10 in the analytical estimate and calculated for each value of S the average relative error between the estimates and the numerical results (Figure 2.8d). The smallest average error occurs for $S = 1.6$ and is 18%.

2.4 Discussion

2.4.1 Characteristic Stresses for Experimentally Derived Flow Laws

The characteristic stress, τ_{C}, is the stress at which the deformation behavior changes from linear viscous flow, such as diffusion creep, to power-law viscous flow, such as dislocation creep. Hence, τ_{C} has a significant impact on the ascent velocity of a diapir.

To estimate values of τ_{C} in the mantle, we use the flow laws of olivine from Hirth and Kohlstedt (2003), their Table 1, for diffusion creep (wet olivine with constant C_{OH} and

10 mm grain size) and dislocation creep (wet olivine with constant C_{OH}) (Figure 2.9a). The values we used are listed in Appendix E. We vary systematically pressure, P, and temperature, T, and determine the stress for which the effective viscosities for diffusion and dislocation creep are equal. For P between 1 and 10 GPa and T between 500 and 1650 °C, values of τ_{C} are approximately between 0.1 and 100 MPa (Figure 2.9a).

To estimate values of τ_{C} in the lower crust, we use the flow laws for diffusion and dislocation creep of wet anorthite from Rybacki and Dresen (2000), their Table 2 (Figure 2.9b), the used values are listed in Appendix E. These flow laws are insensitive to P so we vary systematically T and the grain size to determine τ_{C}. For T between 500 and 1000 °C and grain size between 10 μm and 10 mm, values of τ_{C} are between 1 and 500 MPa.

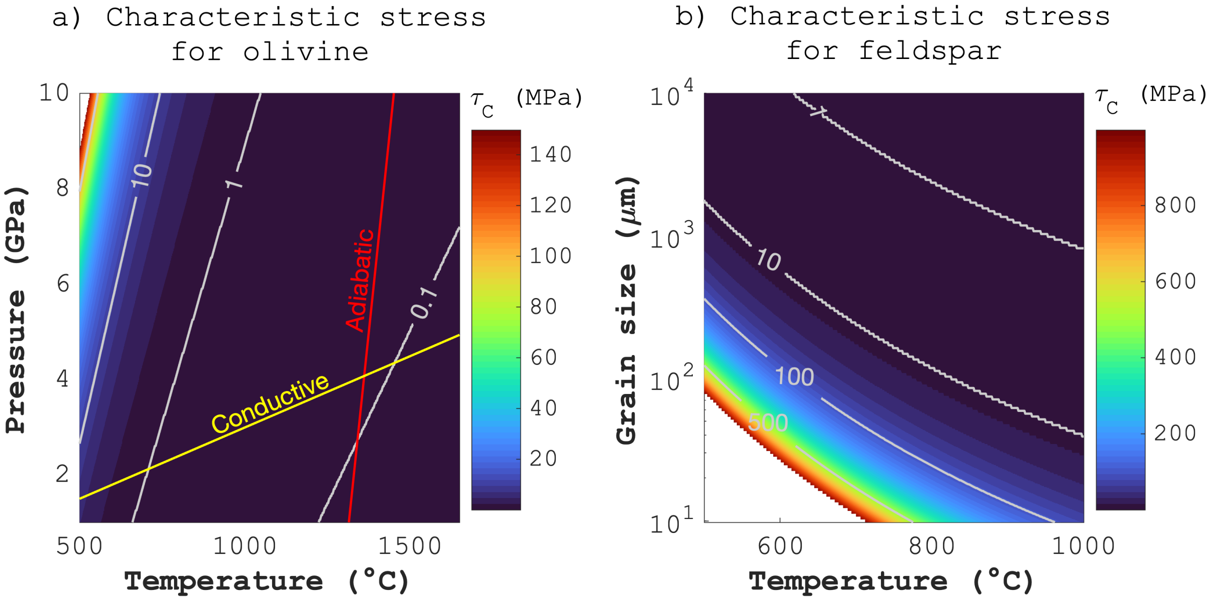

Figure 2.9: Color plots of characteristic stress, τ_{C} in Equation 2.4. (a) τ_{C} as function of pressure and temperature for upper mantle flow laws. The flow laws for diffusion (wet with constant C_{OH} and 10 mm grainsize) and dislocation (wet with constant C_{OH}) creep of olivine from Hirth and Kohlstedt (2003) (their Table 1) were used. (b) τ_{C} as function of grain size and temperature for lower crustal flow laws. The flow laws for diffusion and dislocation creep of wet anorthite from Rybacki and Dresen (2000) (their Table 2) were used.

Quartz is a representative mineral to estimate the effective flow law for upper crustal rocks. Many studies indicate that a power-law viscous flow law describes well the deformation of quartz under upper crustal conditions (e.g., Hirth et al., 2001). For extremely small grain sizes ($\approx$20 μm), such as observed in ultramylonites, quartz can also deform by diffusion creep (Kilian et al., 2011). We did here not estimate τ_{C} for quartz since most studies suggest a power-law viscous flow law for quartz.

Assuming that the flow laws considered above are representative for the mantle lithosphere and the lower crust and assuming that typical regional flow stresses, representing τ_{R}, in the mantle are between 0.1 and 10 MPa (Hirth and Kohlstedt, 2003; Karato, 2008) and in the lower crust between 1 and 100 MPa (Bürgmann and Dresen, 2008), ratios of $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ between 0.1 and 100 seem feasible.

Furthermore, assuming that typical values of $\Delta\rho$ for diapirs vary between 20 and 200 kg/m^3 and values of R between 1 and 100 km (see next Section), provides values of $\Delta\rho g R$ between 0.2 and 200 MPa. Therefore, stress ratios of $\Delta\rho g R/\tau_{\mathrm{C}}$ between 0.05 and 50 seem also feasible.

2.4.2 Increase of Diapir Ascent Velocity by Two Types of Stress Weakening

In our model, the nonlinear fluid surrounding the diapir is a stress weakening fluid for $n > 1$ because the effective viscosity decreases when the stress magnitude, quantified by τ_{II}, in the fluid increases. The applied, combined linear and power-law viscous flow law can describe diffusion and dislocation creep in crustal and mantle rocks (e.g., Karato, 2008; Kohlstedt and Hansen, 2015). Furthermore, the applied power-law viscous flow law can also describe low temperature plasticity for which apparent stress exponents can be much larger than 3 (e.g., Dayem et al., 2009; Schmalholz and Fletcher, 2011). Hence, the applied combined flow law is applicable to a wide range of rocks and deformation mechanisms.

In the analytical estimate for the ascent velocity we consider the impact of two types of stresses: regional stresses, τ_{R}, associated with far-field tectonic deformation in the rocks surrounding the diapir and buoyancy stresses, $\Delta\rho g R$, causing deformation locally around the rising diapir. Both stresses can cause stress weakening. If $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} > 1$ and/or $\Delta\rho g R/\tau_{\mathrm{C}} > 1$ both stresses can increase the ascent velocity significantly (Figure 2.10a). For values of $n = 3$ and $\Omega = 100$, values of $\Delta\rho g R/\tau_{\mathrm{C}} > {\sim}10$ are required to generate values of $V_{\mathrm{D}}/V_0 > 1$ and, hence, an increase in ascent velocity with respect to the velocity for linear viscous flow. The reason is the pre-factor of $3C_{\mathrm{L}}/6^n/S/C_{\mathrm{R}}$ in front of the term $(\Delta\rho g R/\tau_{\mathrm{C}})^{n-1}$ in the analytical velocity estimate (Equation 2.9). This pre-factor is 0.007 for $n = 3$ and $\Omega = 100$ and, hence, reduces the impact of the factor $\Delta\rho g R/\tau_{\mathrm{C}}$ on the velocity increase (Figure 2.10a). Since there is no pre-factor in front of $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$, values of $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} > 1$ cause values of $V_{\mathrm{D}}/V_0 > 1$ (Figure 2.10a).

In the velocity estimate of Weinberg and Podladchikov (1994) only the impact of $\Delta\rho g R$ is considered. Hence, diapirs with small R or small $\Delta\rho$ can have values of small $\Delta\rho g R$ which might not cause a significant velocity increase. Our solution shows that also diapirs associated with small values of $\Delta\rho g R$ can have fast ascent velocities if they rise in a tectonically active region with regional stresses $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} > 1$. Hence, the onset of tectonic deformation, such as strike-slip shearing, transpression or transtension can trigger a faster ascent of diapirs which had insignificant ascent velocities before the onset of tectonic activity and associated stresses. Indeed, for example, many plutons have been emplaced in tectonically active regions suggesting a potential causal link between pluton ascent and tectonic stress (e.g., Hutton and Reavy, 1992; Berger et al., 1996; Berdiel et al., 1997; Brown and Solar, 1999; Michail et al., 2021). We discuss the potential application of our velocity estimate to the ascent of plutons in the next Section.

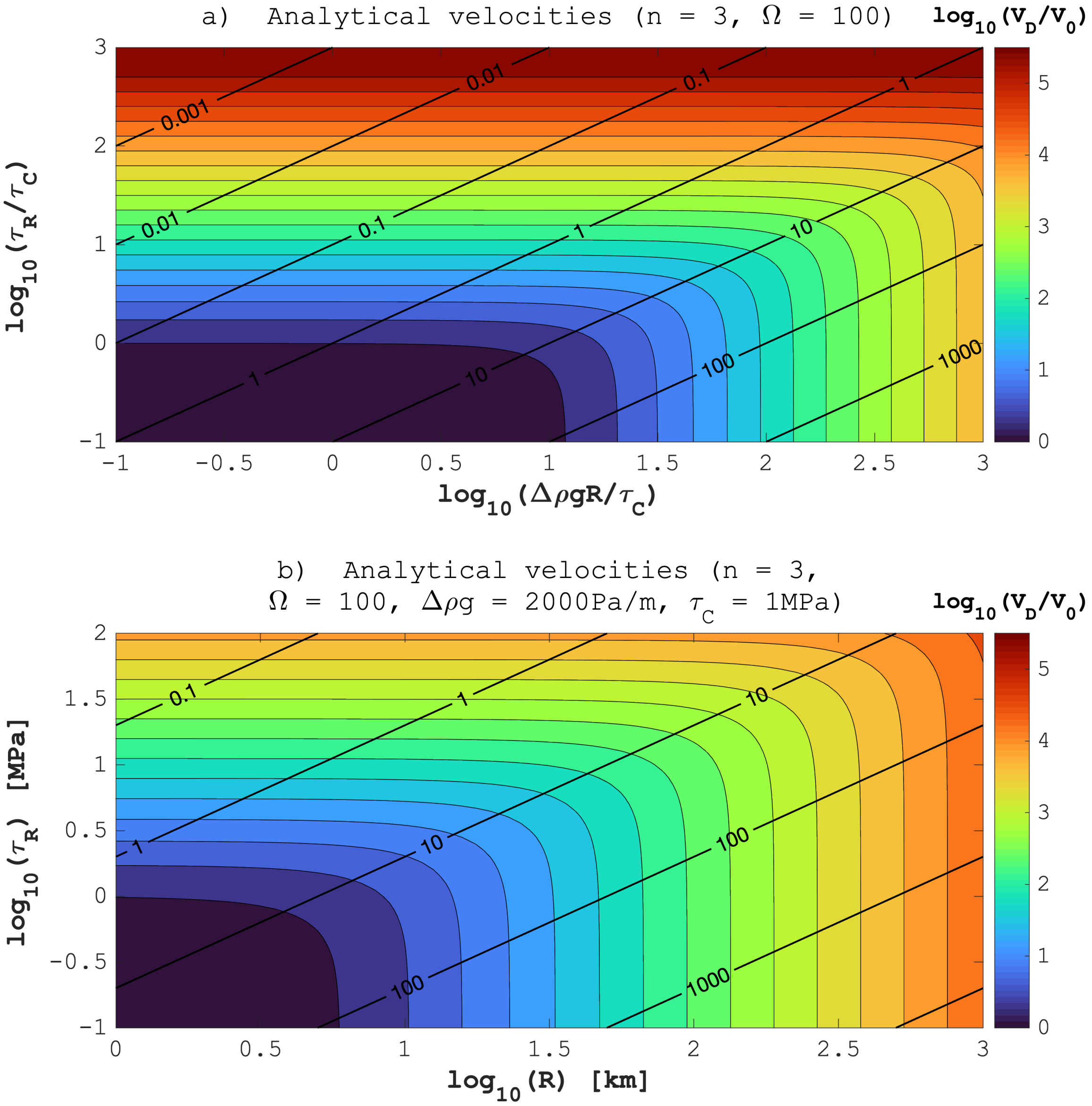

Figure 2.10: Color plots of analytical ascent velocities. (a) Velocities (normalized by V_0) as function of $\Delta\rho g R/\tau_{\mathrm{C}}$ and $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ for $n = 3$ and $\Omega = 100$. (b) Velocities (normalized by V_0) as function of R and τ_{R} for $n = 3$, $\Omega = 100$, $\Delta\rho g = 2000$ Pa/m and $\tau_{\mathrm{C}} = 1$ MPa. Black contour lines in both subplots indicate the corresponding values of $\Delta\rho g R/\tau_{\mathrm{R}}$.

For the numerical calculations, we consider a scenario with horizontal far-field simple shear. We do not model finite deformations but calculate the instantaneous velocity field. Hence, for our calculations mainly the magnitude of τ_{R} is important and not the orientation of the stress field. Therefore, our instantaneous solution for the far-field horizontal simple shear is approximately applicable to any scenario for which the far-field deformation causes deviatoric stresses in rocks surrounding a diapir. For example, for the ascent of diapirs within a deforming mantle wedge (e.g., Klein and Behn, 2021).

To illustrate the results with dimensional numbers, we further assume $\Delta\rho g = 2000$ Pa/m and $\tau_{\mathrm{C}} = 1$ MPa (Figure 2.10b). For τ_{R} increasing above 1 MPa, the diapir velocity, V_{D}, increases with respect to the velocity for linear viscous flow, V_0. Concerning buoyancy stresses, values of $R > \sim 5$ km are required to obtain a velocity increase (Figure 2.10b). For $\tau_{\mathrm{R}} = 100$ MPa the velocity would increase by four orders of magnitude and for $R \approx 15$ km the velocity would increase by one order of magnitude.

2.4.3 Applications to Sediment Diapirs, Mantle Plumes, (U)HP Terranes and Plutons

We discuss next some applications of our velocity estimate to different geodynamic settings involving diapirism. A dimensionless stress ratio which is frequently used in applications of analytical solutions to geodynamic processes is the so-called Argand number (e.g., England and McKenzie, 1982; Schmalholz et al., 2002). The Argand number is the ratio of gravity stress to stress caused by tectonic deformation (e.g., England and McKenzie, 1982; Schmalholz et al., 2002). For the considered scenario of diapirism in tectonically active regions the Argand number corresponds to the ratio $\Delta\rho g R/\tau_{\mathrm{R}}$ (black contours in Figure 2.10).

The analytical estimate of Equation 2.15 can be modified so that the velocity becomes an explicit function of $\Delta\rho g R/\tau_{\mathrm{R}}$:

$$\frac{V_{\mathrm{D}}}{V_0} = 1 + \left(\frac{\tau_{\mathrm{R}}}{\tau_{\mathrm{C}}}\right)^{(n-1)} \left(1 + \frac{3}{6^n S}\frac{C_{\mathrm{L}}}{C_{\mathrm{R}}}\left(\frac{\Delta\rho g R}{\tau_{\mathrm{R}}}\right)^{(n-1)}\right) . \tag{2.21}$$

Values of $\Delta\rho g R/\tau_{\mathrm{R}}$ for specific geodynamic settings may be more reliably estimated than values of $\Delta\rho g R/\tau_{\mathrm{C}}$ because they do not require knowledge of the rheology.

We apply the formula for the ascent velocity, Equation 2.21, to sediment diapirs in subduction zones (e.g., Klein and Behn, 2021), mantle plumes (e.g., Schubert et al., 2001), exhumed (U)HP units (e.g., Burov et al., 2014) and magmatic plutons associated with crustal deformation (e.g., Michail et al., 2021) (Figure 2.11). For all scenarios, the values of the required parameters, such as $\Delta\rho$, R or $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$, are uncertain and we chose representative values to illustrate particular applications of Equation 2.21. We plot V_{D} versus R and versus the corresponding value of the Argand number for different values of the linear viscosity of the rocks surrounding the diapir, η_{m} (Figure 2.11). For the presented velocity calculations, we assume $\tau_{\mathrm{R}} = 10$ MPa, $n = 3$, $\tau_{\mathrm{C}} = 1$ or 10 MPa, and $\Delta\rho = 20$ or 200 kg/m^3 (Figure 2.11).

For sediment diapirs in subduction zones, representative values of R range between 1 and 4 km and we assume $\Delta\rho = 200$ kg/m^3 as feasible value (example 4.1.1 in Klein and Behn, 2021). Klein and Behn (2021) combined the solution of Weinberg and Podladchikov (1994) with heat transfer calculations and a melting thermodynamic model. They show that their calculated velocities for rising sediment diapirs, or relamination, can be between 10 and 100 m/yr (Figure 2.11a and c). To obtain such velocities, values of η_{m} must be significantly smaller than 10^{17} Pa·s, if $\tau_{\mathrm{C}} = 10$ MPa and, hence, $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} = 1$ (Figure 2.11a). However, if $\tau_{\mathrm{C}} = 1$ MPa, values of η_{m} can be in the order of 10^{18} Pa·s to obtain the same velocities (Figure 2.11c). The plots in Figure 2.11 show that for a specific velocity a decrease of τ_{C}

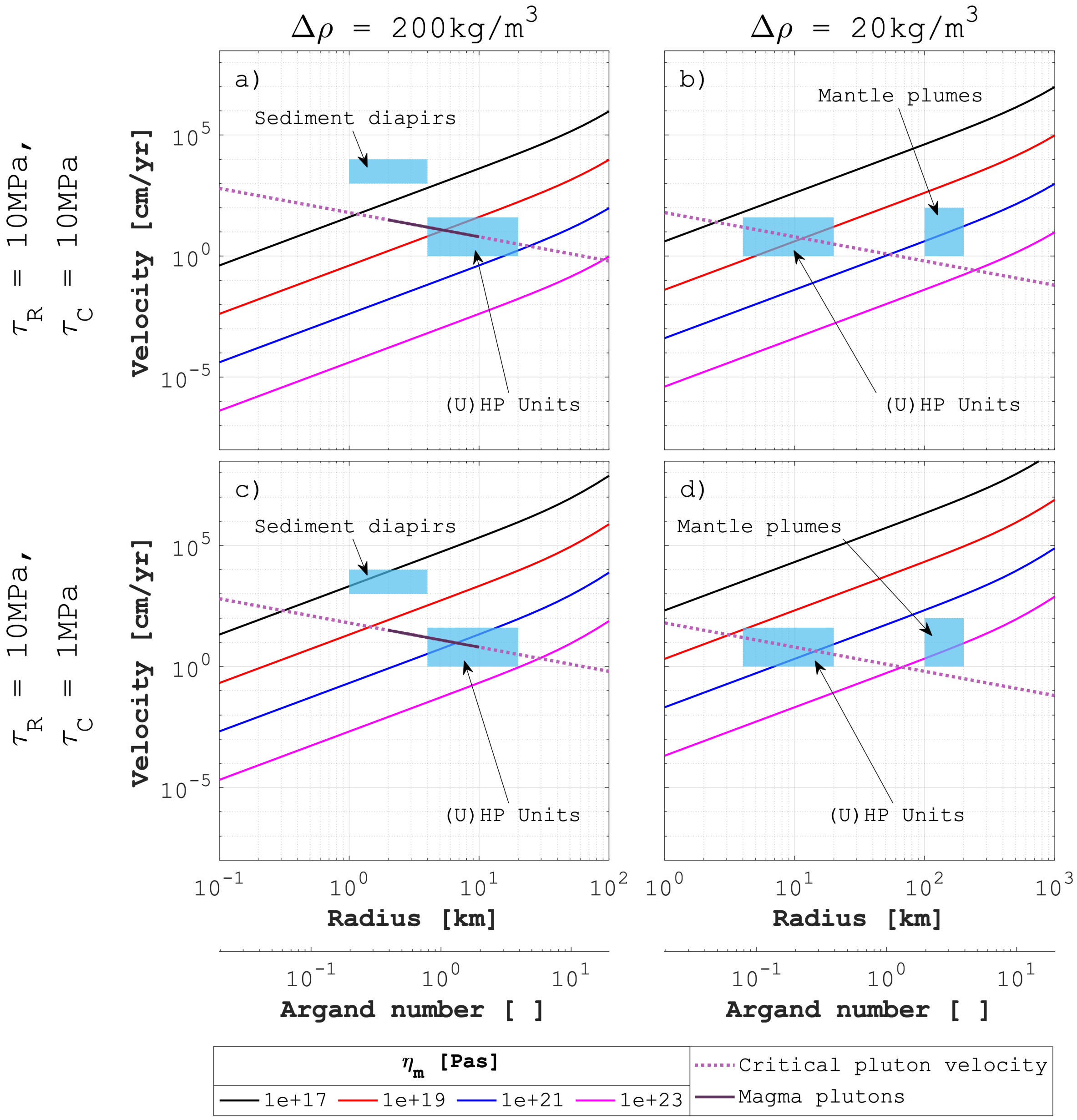

Figure 2.11: Analytical ascent velocities versus diapir radius and corresponding Argand number ($\Delta\rho g R/\tau_{\mathrm{R}}$) for different values of the linear viscosity, η_{m}, of the fluid surrounding the diapir. Applied values of τ_{R}, τ_{C} and $\Delta\rho$ are indicated in the figure. For all calculations n=3 and Ω=100 was used. Rectangles indicate range of data reported in literature for different geodynamic settings (see Section 2.4.3). The dotted line indicates the critical ascent velocity of plutons (Equation 2.22) for which the diapir rises as fast as it cools. The black line segment indicates the range of typical radii estimated for plutons (see Section 2.4.3).

by one order of magnitude increases the corresponding values of η_m by approximately two orders of magnitude. In other words, for the same η_m, a decrease of τ_C by one order of magnitude decreases the η_E by two orders of magnitude and, hence, increases the velocity by two orders of magnitude. For $n > 3$ (e.g., Klein and Behn (2021) used a flow law with $n = 3.5$), the stress weakening and velocity increase would be larger. The above example can of course be done with smaller values of τ_R and τ_C. The results suggest that to achieve the high velocities for sediment diapirs, stress weakening in the surrounding rock is essential. Flow stresses in the mantle wedge, for example due to corner flow, likely contribute to the stress weakening.

For the application to mantle plumes, we assume R between 100 and 200 km and $\Delta\rho$ = 20 kg/m^3 Schubert et al. (2001) (Figure 2.11b and d). The ascent velocity of plumes may range between few cm/yr up to 1 m/yr (Schubert et al., 2001). To achieve such velocities, η_m needs to be between 10^{20} and 10^{21} Pa·s which agrees with viscosity estimates for the mantle (Table 11.3 in Schubert et al., 2001). Deviatoric stresses due to mantle convection may range between 0.1 and 1 MPa (e.g., Hirth and Kohlstedt, 2003) and τ_C for olivine ranges between 0.1 and 0.5 MPa for pressures between 4 and 10 GPa (Figure 2.9a). Therefore, values of $\tau_\mathrm{R}/\tau_\mathrm{C}$ could be >1 which would increase the corresponding ascent velocities. However, the velocities estimated for mantle plumes can be obtained without stress weakening so that for mantle plumes stress weakening seems not essential.

(U)HP crustal units, or terranes, have been exhumed in many places worldwide (e.g., Kylander-Clark et al., 2012; Burov et al., 2014). The mechanisms of exhumation are still disputed and may vary for different geodynamic settings (e.g., Hacker and Gerya, 2013; Warren, 2013). Exhumation by diapirism has been suggested as potential exhumation mechanism (e.g., Burov et al., 2001, 2014; Little et al., 2011; Schmalholz and Schenker, 2016; Schwarzenbach et al., 2021) because diapirism is able to explain the sometimes high estimates for ascent velocities of > 1 cm/yr (e.g., Hermann and Rubatto, 2014), sometimes even > 10 cm/yr (e.g., Schwarzenbach et al., 2021). Such high exhumation velocities are typically estimated for the deeper part of the exhumation path, where ambient rock pressures are $>\approx 1$ GPa. Estimates for $\Delta\rho$ for the exhumation of (U)HP units are commonly between 20 and 200 kg/m^3 and representative spherical radii, which would generate the same spherical volume as the observed (U)HP rock volume, are between 4 and 20 km (e.g., Kylander-Clark et al., 2012; Schwarzenbach et al., 2021). Stress weakening as mechanism to significantly increase ascent velocities has also been suggested as explanation for potentially fast exhumation velocities (e.g., Burov et al., 2014; Schmalholz and Schenker, 2016). For example, Schmalholz and Schenker (2016) proposed that oblique subduction and associated strike-slip shearing could have caused stress weakening along the subduction interface which might explain the high exhumation velocity, along the subduction interface, of a small UHP unit, namely the Brossasco-Isasca sub-unit in the Dora Maira massif, Western Alps.

The mechanisms of pluton ascent in the continental crust are still disputed and velocities of pluton ascent are less constrained than exhumation velocities of (U)HP units. Two commonly proposed mechanisms are diapirism and dike intrusion associated with fracture propagation (e.g., Clemens and Mawer, 1992; Rubin, 1993; Petford, 1996; Miller and Paterson, 1999). A main argument against diapirism is that estimated ascent velocities are so slow that the pluton would lose significant heat during ascent, consequently solidify and stop ascending (e.g., Marsh, 1982; Clemens and Mawer, 1992; Petford, 1996). However, Weinberg and Podladchikov (1994) suggested that stress weakening due to buoyancy stress can increase the ascent velocity sufficiently to avoid significant heat loss during ascent. Furthermore, many plutons ascended in tectonically active regions exhibiting some component of strike-slip, transpression or transtension (e.g., Hutton and Reavy, 1992; Berger et al., 1996; Berdiel et al., 1997; Brown and Solar, 1999; Michail et al., 2021). The regional stresses associated with these tectonic activities could have also contributed to stress weakening and velocity increase.

To evaluate whether stress weakening can enable a pluton to rise a significant distance without significant cooling, say 10 times its radius, we estimate the critical velocity, V_{C}, required for such rise, taking into account the heat loss during ascent. We perform here a very simple, back-of-the-envelope, calculation to estimate V_{C}. The time, or duration, of ascent of a diapir can be calculated by $t_{\mathrm{a}} = d/V_{\mathrm{D}}$ whereby d is the distance of ascent. To avoid thermal cooling during ascent, the diapir must essentially rise faster than it cools. Assuming first cooling by heat conduction only, the time of cooling of a diapir with radius R is $t_{\mathrm{c}} = R^2/\kappa$, whereby κ is the thermal diffusivity. Considering also enhancement of cooling by advection, t_{c} can be modified by using the Nusselt number, Nu (e.g., Marsh, 1982), to get $t_{\mathrm{c}} = R^2/\kappa/Nu$. Solving $t_{\mathrm{c}} = t_{\mathrm{a}}$ for the velocity provides a critical velocity, V_{C}, for which the pluton rises as fast as it cools:

$$V_{\mathrm{C}} = Nu \frac{d\kappa}{R^2} \ . \tag{2.22}$$

The pluton velocity, V_{D}, must be faster than V_{C} to avoid large heat loss during ascent. Assuming here that a pluton should be able to rise at least a distance of 10 times its radius, $d = 10R$, yields as condition for pluton ascent by diapirism:

$$V_{\mathrm{D}} > Nu \frac{10\kappa}{R} \ . \tag{2.23}$$

To plot also V_{C} versus R in Figure 2.11 we assume typical values $Nu = 2$ and $\kappa = 10^{-6}$ m^2/s (e.g., Marsh, 1982). We further assume R between 2 and 10 km and $\Delta\rho = 200$ kg/m^3 as feasible values for crustal plutons (e.g., Miller and Paterson, 1999; Michail et al., 2021). Based on the above calculation, the velocity for plutons must be larger than approximately

10 cm/yr (black segment on dotted lines in Figure 2.11). For linear viscous flow, η_m of the surrounding rocks must then be smaller than approximately 10^{19} Pa·s (Figure 2.11a). For stress weakening due to tectonic deformation with $\tau_\mathrm{R}/\tau_\mathrm{C} = 10$, η_m must be smaller than approximately 10^{21} Pa·s (Figure 2.11c). If η_m is 10^{21} Pa·s and $\tau_\mathrm{R}/\tau_\mathrm{C} = 100$, then V_D is approximately 100 times faster than V_C and pluton ascent by diapirism seems possible. Our simple calculations suggest that pluton ascent by diapirism is possible if τ_R is high, say between 10 and 100 MPa, and τ_C is low, say between 0.1 and 1 MPa. More generally, tectonic activity may cause regional stresses which are significantly larger than critical stresses so that stress weakening can significantly decrease the effective viscosity of the surrounding rock. This viscosity decrease can be large enough so that plutons can rise as diapirs considerably faster than they cool.

2.4.4 Simplifications

Our study focuses on creating and testing analytical estimates for the velocity at which diapirs rise in a ductile rock under far-field stress. To achieve this, we have simplified the geodynamic scenario and numerical model configuration compared to natural situations.

We calculate only one numerical time step to determine the instantaneous ascent velocity for a spherical diapir. As a diapir rises, it may change its geometry. For example, when it rises to areas with higher viscosity, it can flatten vertically and become elliptical, with its long axis being perpendicular to the direction of ascent (e.g., Weinberg and Podladchikov, 1994). The ascent velocity of a diapir with such an elliptical shape is smaller than the velocity for spherical geometry. For elliptical aspect ratios of 2 and power-law stress exponents of 3, the ascent velocity of an elliptical diapir is a factor of 2 slower compared to a spherical diapir, and for a power-law stress exponent of 5, it is a factor of 4 slower (e.g., Weinberg and Podladchikov, 1994). Given our focus on order-of-magnitude variations in ascent velocity resulting from stress weakening, the variation in ascent velocity due to diapir shape changes, falling within a range of two to fourfold, has a minor impact on our primary findings.

We employ a combination of linear and power-law viscous flow laws, that is applicable to a range of deformation mechanisms, including diffusion creep, dislocation creep, and exponential creep, commonly known as low-temperature plasticity (e.g., Karato, 2008). The power-law viscous flow law is also well-suited for describing diffusion creep involving grain size evolution, where grain size dynamically responds to flow stress (e.g., Montési and Zuber, 2002). Additionally, a power-law viscous flow law with a high stress exponent effectively captures pressure-insensitive plastic deformation, such as deformation governed by a von Mises yield stress (e.g., Fletcher and Hallet, 1983). Consequently, the chosen flow law encompasses a broad range of ductile deformation mechanisms. Potential enhancements to our analytical and numerical models could include (a) more sophisticated models for grain size evolution, such as the paleo-wattmeter model (e.g., Austin and Evans, 2007), (b) the incorporation of frictional-plastic deformation, which involves pressure-sensitive yield stress (e.g., Poliakov et al., 1993), or (c) the consideration of fluid and reaction-induced weakening linked to (de)hydration reactions and melting (e.g., White and Knipe, 1978; Jamieson et al., 2011; Schmalholz et al., 2020).

2.5 Conclusions

In this study, we investigated the ascent velocity of a weak and buoyant spherical inclusion within a nonlinear viscous fluid under far-field stress, which is relevant to a wide range of natural diapirism in tectonically active regions. By deriving analytical estimates for the diapir ascent velocity in dimensionless form, we scaled the velocity with the corresponding velocity for linear viscous flow. The ascent velocity is controlled by two stress ratios: (a) the ratio of the diapir's buoyancy stress, $\Delta\rho g R$, to the characteristic stress, τ_{C}, at the transition from linear to power-law viscous flow, and (b) the ratio of regional stress, τ_{R}, to τ_{C}, whereby τ_{R} is caused by the far-field tectonic deformation. The equation for the analytical estimates shows that both stress ratios can significantly increase the velocity because the stress ratios are added and both ratios exhibit the same power-law stress exponent of $(n-1)$. The stress ratios start to considerably increase the ascent velocity once they become larger than one. Hence, both local buoyancy and regional tectonic stresses can increase the ascent velocity because they can cause stress weakening in the rocks surrounding the diapir.

Comparing the analytical estimates with full 3D numerical calculations, we found that the analytical estimates are accurate within a factor of less than two, with a relative error smaller than 80%, across a wide range of stress ratios. This highlights the usefulness of the analytical estimates in assessing the importance and impact of diapirism in diverse geodynamic settings. However, the analytical estimates deviate the most from the numerical results when buoyancy stresses dominate the ascent velocity. This discrepancy arises because the analytical estimates use the diapir's buoyancy stress as a proxy for the deviatoric stress, which is required to calculate the effective, stress-dependent viscosity in the surrounding fluid. Numerical calculations demonstrate that deviatoric stresses around the diapir can be significantly smaller than buoyancy stresses, leading to less intense stress weakening in the surrounding fluid than predicted by the analytical estimates. Introducing a shape factor improves the accuracy of the analytical estimates. The numerical results further show that the pressure inside the weak diapir deviates from the lithostatic pressure and the deviation is on the order of $\Delta\rho g R$.

We calculated τ_{C} for typical mantle and lower crustal flow laws and estimated ranges of magnitudes for τ_{R} and $\Delta\rho g R$. Both ratios of $\tau_{\mathrm{R}}/\tau_{\mathrm{C}}$ and $\Delta\rho g R/\tau_{\mathrm{C}}$ could vary between 0.1 and 100 in nature. For the applied parameters, a significant increase of the ascent velocity caused by stress weakening starts for $\tau_\mathrm{R}/\tau_\mathrm{C} > 1$ and $\Delta\rho g R/\tau_\mathrm{C} > 10$. Our calculations show that stress weakening can cause ascent velocities of diapirs that are up to four orders of magnitude faster compared to ascent velocities calculated for linear viscous flow. Therefore, lithospheric and mantle stresses as well as temporal and spatial changes of these stresses can have a dramatic effect on diapir ascent velocities. Similarly, changes in rock rheology, due to for example fluid infiltration or grain size variation, can change magnitudes of τ_C and, consequently, strongly affect ascent velocities. The presented analytical estimates facilitate the quantification of such stress-induced changes in diapir ascent velocities.

2.6 Acknowledgements

This work was supported by SNSF Grant 200020 197218. LR acknowledges the Swiss University Conference and the Swiss Council of Federal Institutes of Technology for supporting this research through the Platform for Advanced Scientific Computing (PASC) program and the Swiss National Supercomputing Center (CSCS) through project ID c23. This work was supported by the University of Lausanne.

2.7 Open Research

Current and future versions of the SphericalStokes.jl software used in this study are publicly available on GitHub at `https://github.com/PTsolvers/SphericalStokes`. The exact version used in this study is archived on Zenodo and can be accessed at `https://zenodo.org/doi/10.5281/zenodo.10093648` Macherel et al. (2023).

Bibliography

Austin, N. J., Evans, B., 2007. Paleowattmeters: A scaling relation for dynamically recrystallized grain size. Geology 35 (4), 343–346.

Behn, M. D., Kelemen, P. B., Hirth, G., Hacker, B. R., Massonne, H.-J., 2011. Diapirs as the source of the sediment signature in arc lavas. Nature Geoscience 4 (9), 641–646.

Berdiel, T. R., Gapais, D., Brun, J.-P., 1997. Granite intrusion along strike-slip zones in experiment and nature. American Journal of Science 297 (6), 651–678.

Berger, A., Rosenberg, C., Schmid, S., 1996. Ascent, emplacement and exhumation of the Bergell pluton within the Southern Steep Belt of the Central Alps. Schweizerische mineralogische und petrographische Mitteilungen 76, 357–382.

Brown, M., Solar, G., 1999. The mechanism of ascent and emplacement of granite magma during transpression: a syntectonic granite paradigm. Tectonophysics 312 (1), 1–33.

Bürgmann, R., Dresen, G., 2008. Rheology of the lower crust and upper mantle: Evidence from rock mechanics, geodesy, and field observations. Annu. Rev. Earth Planet. Sci. 36, 531–567.

Burov, E., François, T., Agard, P., Le Pourhiet, L., Meyer, B., Tirel, C., Lebedev, S., Yamato, P., Brun, J.-P., 2014. Rheological and geodynamic controls on the mechanisms of subduction and HP/UHP exhumation of crustal rocks during continental collision: Insights from numerical models. Tectonophysics 631, 212–250.

Burov, E., Jaupart, C., Guillou-Frottier, L., 2003. Ascent and emplacement of buoyant magma bodies in brittle-ductile upper crust. Journal of Geophysical Research: Solid Earth 108 (B4).

Burov, E., Jolivet, L., Le Pourhiet, L., Poliakov, A., 2001. A thermomechanical model of exhumation of high pressure (HP) and ultra-high pressure (UHP) metamorphic rocks in Alpine-type collision belts. Tectonophysics 342 (1-2), 113–136.

Christensen, U., 1983. Convection in a variable-viscosity fluid: Newtonian versus power-law rheology. Earth and Planetary Science Letters 64 (1), 153–162.

Clemens, J., Mawer, C., 1992. Granitic magma transport by fracture propagation. Tectonophysics 204 (3-4), 339–360.

Cruden, A. R., 1988. Deformation around a rising diapir modeled by creeping flow past a sphere. Tectonics 7 (5), 1091–1101.

Cruden, A. R., Weinberg, R., 2018. Mechanisms of magma transport and storage in the lower and middle crust—magma segregation, ascent and emplacement. Volcanic and igneous plumbing systems, 13–53.

Dayem, K. E., Houseman, G. A., Molnar, P., 2009. Localization of shear along a lithospheric strength discontinuity: Application of a continuous deformation model to the boundary between Tibet and the Tarim Basin. Tectonics 28 (3).

England, P., McKenzie, D., 1982. A thin viscous sheet model for continental deformation. Geophysical Journal International 70 (2), 295–321.

Fletcher, R. C., 1974. Wavelength selection in the folding of a single layer with power-law rheology. American Journal of Science 274 (9), 1029–1043.

Fletcher, R. C., Hallet, B., 1983. Unstable extension of the lithosphere: A mechanical model for basin-and-range structure. Journal of Geophysical Research: Solid Earth 88 (B9), 7457–7466.

Frankel, S. P., 1950. Convergence rates of iterative treatments of partial differential equations. Mathematics of Computation 4, 65–75.

Gerya, T., 2019. Introduction to numerical geodynamic modelling. Cambridge University Press.

Gerya, T., Yuen, D. A., 2003. Rayleigh–Taylor instabilities from hydration and melting propel 'cold plumes' at subduction zones. Earth and Planetary Science Letters 212 (1-2), 47–62.

Hacker, B. R., Gerya, T. V., 2013. Paradigms, new and old, for ultrahigh-pressure tectonism. Tectonophysics 603, 79–88.

Hadamard, J., 1911. Mouvement permanent lent d'une sphere liquide et visqueuse dans un liquide visqueux. Compt. Rend. Acad. Sci. 152, 1735–1738.

Hermann, J., Rubatto, D., 2014. Subduction of continental crust to mantle depth: geochemistry of ultrahigh-pressure rocks. In: Treatise on Geochemistry, 2nd Edition. Elsevier.

Hirth, G., Kohlstedt, D., 2003. Rheology of the upper mantle and the mantle wedge: A view from the experimentalists. Geophysical monograph-american geophysical union 138, 83–106.

Hirth, G., Teyssier, C., Dunlap, J. W., 2001. An evaluation of quartzite flow laws based on comparisons between experimentally and naturally deformed rocks. International Journal of Earth Sciences 90, 77–87.

Hutton, D., Reavy, R., 1992. Strike-slip tectonics and granite petrogenesis. Tectonics 11 (5), 960–967.

Jackson, M., Cornelius, R., Craig, C., Gansser, A., Stöcklin, J., Talbot, C., 1990. Salt diapirs of the Great Kavir, central Iran. Geological Society of America Memoirs 177.

Jackson, M., Vendeville, B., 1994. Regional extension as a geologic trigger for diapirism. Geological society of America bulletin 106 (1), 57–73.

Jamieson, R. A., Unsworth, M. J., Harris, N. B., Rosenberg, C. L., Schulmann, K., 2011. Crustal melting and the flow of mountains. Elements 7 (4), 253–260.

Karato, S.-i., 2008. Deformation of Earth materials. An introduction to the rheology of Solid Earth 463.

Kilian, R., Heilbronner, R., Stünitz, H., 2011. Quartz grain size reduction in a granitoid rock and the transition from dislocation to diffusion creep. Journal of Structural Geology 33 (8), 1265–1284.

Klein, B. Z., Behn, M. D., 2021. On the evolution and fate of sediment diapirs in subduction zones. Geochemistry, Geophysics, Geosystems 22 (11).

Kohlstedt, D., Hansen, L., 2015. 2.18 - constitutive equations, rheological behavior, and viscosity of rocks. In: Treatise on Geophysics (Second Edition). Elsevier, Oxford, pp. 441–472.

Kylander-Clark, A. R., Hacker, B. R., Mattinson, C. G., 2012. Size and exhumation rate of ultrahigh-pressure terranes linked to orogenic stage. Earth and Planetary Science Letters 321, 115–120.

Larsen, T. B., Yeun, D. A., 1997. Fast plumeheads: Temperature-dependent versus non-Newtonian rheology. Geophysical Research Letters 24 (16), 1995–1998.

Little, T., Hacker, B., Gordon, S., Baldwin, S., Fitzgerald, P., Ellis, S., Korchinski, M., 2011. Diapiric exhumation of Earth's youngest (UHP) eclogites in the gneiss domes of the D'Entrecasteaux Islands, Papua New Guinea. Tectonophysics 510 (1-2), 39–68.

Macherel, E., Räss, L., Schmalholz, S. M., 2023. PTsolvers/SphericalStokes: SphericalStokes.jl 1.0.1.

Marschall, H. R., Schumacher, J. C., 2012. Arc magmas sourced from mélange diapirs in subduction zones. Nature Geoscience 5 (12), 862–867.

Marsh, B. D., 1982. On the mechanics of igneous diapirism, stoping, and zone melting. American Journal of Science 282 (6), 808–855.

Mazzini, A., Nermoen, A., Krotkiewski, M., Podladchikov, Y., Planke, S., Svensen, H., 2009. Strike-slip faulting as a trigger mechanism for overpressure release through piercement structures. Implications for the Lusi mud volcano, Indonesia. Marine and Petroleum Geology 26 (9), 1751–1765.

Michail, M., Rudolf, M., Rosenau, M., Riva, A., Gianolla, P., Coltorti, M., 2021. Shape of plutons in crustal shear zones: A tectono-magmatic guide based on analogue models. Journal of Structural Geology 150, 104417.

Miller, R., Paterson, S., 1999. In defense of magmatic diapirs. Journal of Structural Geology 21 (8), 1161–1173.

Montési, L. G., Zuber, M. T., 2002. A unified description of localization for application to large-scale tectonics. Journal of Geophysical Research: Solid Earth 107 (B3), ECV–1.

Nahas, I., Gonçalves, L., Gonçalves, C. C., Raposo, M. I. B., 2023. Unraveling the relationship between a tonalitic pluton and shear zones: Insights from magnetic fabrics and microstructures of the Alto Maranhão batholith, Mineiro belt, southern São Francisco craton. Journal of Structural Geology, 104887.

Petford, N., 1996. Dykes or diapirs? Earth and Environmental Science Transactions of the Royal Society of Edinburgh 87 (1-2), 105–114.

Plank, T., Langmuir, C. H., 1993. Tracing trace elements from sediment input to volcanic output at subduction zones. Nature 362 (6422), 739–743.

Poliakov, A. N., Podladchikov, Y., Talbot, C., 1993. Initiation of salt diapirs with frictional overburdens: numerical experiments. Tectonophysics 228 (3-4), 199–210.

Poliakov, A. N., Podladchikov, Y. Y., Dawson, E. C., Talbot, C. J., 1996. Salt diapirism with simultaneous brittle faulting and viscous flow. Geological Society, London, Special Publications 100 (1), 291–302.

Rabinowicz, M., Ceuleneer, G., Nicolas, A., 1987. Melt segregation and flow in mantle diapirs below spreading centers: Evidence from the Oman Ophiolite. Journal of Geophysical Research: Solid Earth 92 (B5), 3475–3486.

Ramberg, H., 1968. Fluid dynamics of layered systems in the field of gravity, a theoretical basis for certain global structures and isostatic adjustment. Physics of the Earth and Planetary Interiors 1 (2), 63–87.

Räss, L., Utkin, I., Duretz, T., Omlin, S., Podladchikov, Y. Y., 2022. Assessing the robustness and scalability of the accelerated pseudo-transient method. Geoscientific Model Development 15 (14), 5757–5786.

Rubin, A. M., 1993. Dikes vs. diapirs in viscoelastic rock. Earth and Planetary Science Letters 117 (3-4), 653–670.

Rybacki, E., Dresen, G., 2000. Dislocation and diffusion creep of synthetic anorthite aggregates. Journal of Geophysical Research: Solid Earth 105 (B11), 26017–26036.

Rybczynski, W., 1911. Über die fortschreitende Bewegung einer flüssigen Kugel in einen zähen Medium. Bull. Acad. Sci. Cracovie 1, 40–46.

Schmalholz, S. M., Fletcher, R. C., 2011. The exponential flow law applied to necking and folding of a ductile layer. Geophysical Journal International 184 (1), 83–89.

Schmalholz, S. M., Moulas, E., Plümper, O., Myasnikov, A. V., Podladchikov, Y. Y., 2020. 2D hydro-mechanical-chemical modeling of (de)hydration reactions in deforming heterogeneous rock: The periclase-brucite model reaction. Geochemistry, Geophysics, Geosystems 21 (11), e2020GC009351.

Schmalholz, S. M., Podladchikov, Y., Burg, J.-P., 2002. Control of folding by gravity and matrix thickness: Implications for large-scale folding. Journal of Geophysical Research: Solid Earth 107 (B1), ETG–1.

Schmalholz, S. M., Podladchikov, Y. Y., 2013. Tectonic overpressure in weak crustal-scale shear zones and implications for the exhumation of high-pressure rocks. Geophysical Research Letters 40 (10), 1984–1988.

Schmalholz, S. M., Schenker, F. L., 2016. Exhumation of the Dora Maira ultrahigh-pressure unit by buoyant uprise within a low-viscosity mantle oblique-slip shear zone. Terra Nova 28 (5), 348–355.

Schubert, G., Turcotte, D. L., Olson, P., 2001. Mantle convection in the Earth and planets. Cambridge University Press.

Schultz-Ela, D., Jackson, M., Vendeville, B., 1993. Mechanics of active salt diapirism. Tectonophysics 228 (3), 275–312.

Schwarzenbach, E. M., Zhong, X., Caddick, M. J., Schmalholz, S. M., Menneken, M., Hecht, L., John, T., 2021. On exhumation velocities of high-pressure units based on insights from chemical zoning in garnet (Tianshan, NW China). Earth and Planetary Science Letters 570, 117065.

Smye, A. J., England, P. C., 2023. Metamorphism and deformation on subduction interfaces: 2. Petrological and tectonic implications. Geochemistry, Geophysics, Geosystems 24 (1), e2022GC010645.

Stokes, G., 1850. On the effect of internal friction of fluids on the motion of pendulums. Trans. Camb. phi1. S0c 9 (8), 106.

Turcotte, D., Schubert, G., 2021. Geodynamics. Cambridge University Press.

Wang, L. H., Yarushina, V. M., Alkhimenkov, Y., Podladchikov, Y., 2022. Physics-inspired pseudo-transient method and its application in modelling focused fluid flow with geological complexity. Geophysical Journal International 229 (1), 1–20.

Warren, C., 2013. Exhumation of (ultra-)high-pressure terranes: concepts and mechanisms. Solid Earth 4 (1), 75–92.

Weinberg, R. F., Podladchikov, Y., 1994. Diapiric ascent of magmas through power law crust and mantle. Journal of Geophysical Research: Solid Earth 99 (B5), 9543–9559.

Weinberg, R. F., Podladchikov, Y. Y., 1995. The rise of solid-state diapirs. Journal of Structural Geology 17 (8), 1183–1195.

White, S. t., Knipe, R., 1978. Transformation- and reaction-enhanced ductility in rocks. Journal of the Geological Society 135 (5), 513–516.

Whitehead, J., John, A., Luther, Douglas, S., 1975. Dynamics of laboratory diapir and plume models. Journal of Geophysical Research (1896-1977) 80 (5), 705–717.

Appendix A. Cylindrical Coordinates System

The equations for conservation of mass and linear momentum for an incompressible fluid under gravity in cylindrical coordinates are:

$$\begin{aligned}
0 &= -\left(\frac{\partial V_r}{\partial r} + \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{\partial V_z}{\partial z} + \frac{V_r}{r}\right) \\
0 &= \frac{\partial(-P+\tau_{rr})}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{\partial \tau_{rz}}{\partial z} + \frac{(-P+\tau_{rr})}{r} - \frac{(-P+\tau_{\theta\theta})}{r} - \rho g \\
0 &= \frac{\partial \tau_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial(-P+\tau_{\theta\theta})}{\partial \theta} + \frac{\partial \tau_{\theta z}}{\partial z} + 2\frac{\tau_{r\theta}}{r} \\
0 &= \frac{\partial \tau_{rz}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{\theta z}}{\partial \theta} + \frac{\partial(-P+\tau_{zz})}{\partial z} + \frac{\tau_{rz}}{r} \qquad (2.24)
\end{aligned}$$

with V_r, V_θ, and V_z being the components of the velocity vector in direction r, θ, and z respectively. τ_{ij} are the $(i,j)^{th}$ components of the deviatoric stress tensor, P is the pressure, ρ is the density and g the gravitational acceleration. Components of the deviatoric stress tensor are defined as:

$$\begin{aligned}
\tau_{rr} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{rr} = 2\eta_{\mathrm{E}}\left(\frac{\partial V_r}{\partial r}\right) \\
\tau_{\theta\theta} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\theta\theta} = 2\eta_{\mathrm{E}}\left(\frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{V_r}{r}\right) \\
\tau_{zz} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{zz} = 2\eta_{\mathrm{E}}\left(\frac{\partial V_z}{\partial z}\right) \\
\tau_{r\theta} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{r\theta} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_\theta}{\partial r} + \frac{1}{r}\frac{\partial V_r}{\partial \theta} - \frac{V_\theta}{r}\right)\right) \\
\tau_{rz} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{rz} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_z}{\partial r} + \frac{\partial V_r}{\partial z}\right)\right) \\
\tau_{\theta z} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\theta z} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{1}{r}\frac{\partial V_z}{\partial \theta} + \frac{\partial V_\theta}{\partial z}\right)\right) \qquad (2.25)
\end{aligned}$$

where $\dot{\epsilon}_{ij}$ are the $(i,j)^{th}$ components of the strain rate tensor, and η_{E} is the effective viscosity used in the numerical calculations (see Section 2.2.3, Equation 2.4).

The numerical implementation used is the same as for the Cartesian coordinates (see Section 2.2.4):

$$\begin{aligned}
\frac{1}{\tilde{K}}\frac{\partial P}{\partial \tau_{\mathrm{PT}}} &= -\left(\frac{\partial V_r}{\partial r} + \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{\partial V_z}{\partial z} + \frac{V_r}{r}\right)\\
\tilde{\rho}\frac{\partial V_r}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial(-P+\tau_{rr})}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{\partial \tau_{rz}}{\partial z} + \frac{(-P+\tau_{rr})}{r} - \frac{(-P+\tau_{\theta\theta})}{r} - \rho g\\
\tilde{\rho}\frac{\partial V_\theta}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial(-P+\tau_{\theta\theta})}{\partial \theta} + \frac{\partial \tau_{\theta z}}{\partial z} + 2\frac{\tau_{r\theta}}{r}\\
\tilde{\rho}\frac{\partial V_z}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{rz}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{\theta z}}{\partial \theta} + \frac{\partial(-P+\tau_{zz})}{\partial z} + \frac{\tau_{rz}}{r}\\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{rr}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{rr}}{2\eta_{\mathrm{E}}} &= \frac{\partial V_r}{\partial r}\\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\theta\theta}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\theta\theta}}{2\eta_{\mathrm{E}}} &= \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{V_r}{r}\\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{zz}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{zz}}{2\eta_{\mathrm{E}}} &= \frac{\partial V_z}{\partial z}\\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{r\theta}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{r\theta}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{\partial V_\theta}{\partial r} + \frac{1}{r}\frac{\partial V_r}{\partial \theta} - \frac{V_\theta}{r}\right)\\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{rz}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{rz}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{\partial V_z}{\partial r} + \frac{\partial V_r}{\partial z}\right)\\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\theta z}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\theta z}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{1}{r}\frac{\partial V_z}{\partial \theta} + \frac{\partial V_\theta}{\partial z}\right)
\end{aligned} \tag{2.26}$$

where $\tilde{K}$, $\tilde{\rho}$, and $\tilde{G}$ are numerical parameters and τ_{PT} is a pseudo-time. $\tilde{K}$ and $\tilde{G}$ can be considered as pseudo-bulk and pseudo-shear modulus, respectively, and $\tilde{\rho}$ as a pseudo-density. With these parameters, Equation 2.26 can be considered as acoustic and inertial approximations of the mass and momentum balance equations respectively.

Appendix B. Spherical Coordinates System

The equations for conservation of mass and linear momentum for an incompressible fluid under gravity in spherical coordinates are:

$$\begin{aligned}
0 &= -\left(\frac{\partial V_r}{\partial r} + \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + 2\frac{V_r}{r} + \frac{V_\theta}{r}cot(\theta)\right) \\
0 &= \frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \tau_{r\varphi}}{\partial \varphi} + 2\frac{\sigma_{rr}}{r} - \frac{\sigma_{\theta\theta}}{r} - \frac{\sigma_{\varphi\varphi}}{r} + \frac{\tau_{r\theta}}{r}cot(\theta) - \rho g \\
0 &= \frac{\partial \tau_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \tau_{\theta\varphi}}{\partial \varphi} + 3\frac{\tau_{r\theta}}{r} + \frac{\sigma_{\theta\theta}}{r}cot(\theta) - \frac{\sigma_{\varphi\varphi}}{r}cot(\theta) \\
0 &= \frac{\partial \tau_{r\varphi}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{\theta\varphi}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \sigma_{\varphi\varphi}}{\partial \varphi} + 3\frac{\tau_{r\varphi}}{r} + 2\frac{\tau_{\theta\varphi}}{r}cot(\theta) \qquad (2.27)
\end{aligned}$$

with V_r, V_θ, and V_φ being the components of the velocity vector in direction r, θ, and φ respectively. τ_{ij} are the $(i,j)^{th}$ components of the deviatoric stress tensor, P is the pressure, $\sigma_{ij} = -P + \tau_{ij}$ is the total stress, ρ is the density and g the gravitational acceleration. Components of the deviatoric stress tensor are defined as:

$$\begin{aligned}
\tau_{rr} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{rr} = 2\eta_{\mathrm{E}}\left(\frac{\partial V_r}{\partial r}\right) \\
\tau_{\theta\theta} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\theta\theta} = 2\eta_{\mathrm{E}}\left(\frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{V_r}{r}\right) \\
\tau_{\varphi\varphi} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\varphi\varphi} = 2\eta_{\mathrm{E}}\left(\frac{1}{rsin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + \frac{V_r}{r} + \frac{V_\theta}{r}cot(\theta)\right) \\
\tau_{r\theta} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{r\theta} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_\theta}{\partial r} + \frac{1}{r}\frac{\partial V_r}{\partial \theta} - \frac{V_\theta}{r}\right)\right) \\
\tau_{r\varphi} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{r\varphi} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_\varphi}{\partial r} + \frac{1}{rsin(\theta)}\frac{\partial V_r}{\partial \varphi} - \frac{V_\varphi}{r}\right)\right) \\
\tau_{\theta\varphi} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\theta\varphi} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{1}{r}\frac{\partial V_\varphi}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial V_\theta}{\partial \varphi} - \frac{V_\varphi}{r}cot(\theta)\right)\right) \qquad (2.28)
\end{aligned}$$

where $\dot{\epsilon}_{ij}$ are the $(i,j)^{th}$ components of the strain rate tensor, and η_{E} is the effective viscosity used in the numerical calculations (see Section 2.2.3, Equation 2.4).

The numerical implementation used is the same as for the Cartesian coordinates (see Section 2.2.4):

$$\begin{aligned}
\frac{1}{\tilde{K}}\frac{\partial P}{\partial \tau_{\mathrm{PT}}} &= -\left(\frac{\partial V_r}{\partial r} + \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{1}{r sin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + 2\frac{V_r}{r} - \frac{V_\theta}{r}cot(\theta)\right) \\
\tilde{\rho}\frac{\partial V_r}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{1}{r sin(\theta)}\frac{\partial \tau_{r\varphi}}{\partial \varphi} + 2\frac{\sigma_{rr}}{r} - \frac{\sigma_{\theta\theta}}{r} - \frac{\sigma_{\varphi\varphi}}{r} + \frac{\tau_{r\theta}}{r}cot(\theta) - \rho g \\
\tilde{\rho}\frac{\partial V_\theta}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{1}{r sin(\theta)}\frac{\partial \tau_{\theta\varphi}}{\partial \varphi} + 3\frac{\tau_{r\theta}}{r} + \frac{\sigma_{\theta\theta}}{r}cot(\theta) - \frac{\sigma_{\varphi\varphi}}{r}cot(\theta) \\
\tilde{\rho}\frac{\partial V_\varphi}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{r\varphi}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{\theta\varphi}}{\partial \theta} + \frac{1}{r sin(\theta)}\frac{\partial \sigma_{\varphi\varphi}}{\partial \varphi} + 3\frac{\tau_{r\varphi}}{r} + 2\frac{\tau_{\theta\varphi}}{r}cot(\theta) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{rr}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{rr}}{2\eta_{\mathrm{E}}} &= \frac{\partial V_r}{\partial r} \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\theta\theta}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\theta\theta}}{2\eta_{\mathrm{E}}} &= \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{V_r}{r} \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\varphi\varphi}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\varphi\varphi}}{2\eta_{\mathrm{E}}} &= \frac{1}{r sin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + \frac{V_r}{r} + \frac{V_\theta}{r}cot(\theta) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{r\theta}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{r\theta}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{\partial V_\theta}{\partial r} + \frac{1}{r}\frac{\partial V_r}{\partial \theta} - \frac{V_\theta}{r}\right) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{r\varphi}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{r\varphi}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{\partial V_\varphi}{\partial r} + \frac{1}{r sin(\theta)}\frac{\partial V_r}{\partial \varphi} - \frac{V_\varphi}{r}\right) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\theta\varphi}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\theta\varphi}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{1}{r}\frac{\partial V_\varphi}{\partial \theta} + \frac{1}{r sin(\theta)}\frac{\partial V_\theta}{\partial \varphi} - \frac{V_\varphi}{r}cot(\theta)\right)
\end{aligned} \tag{2.29}$$

where $\tilde{K}$, $\tilde{\rho}$, and $\tilde{G}$ are numerical parameters, τ_{PT} is a pseudo-time. $\tilde{K}$ and $\tilde{G}$ can be considered as pseudo-bulk and pseudo-shear modulus respectively, and $\tilde{\rho}$ as a pseudo-density. With these parameters, Equation 2.29 can be considered as acoustic and inertial approximations of the mass and momentum balance equations respectively.

Appendix C. Comparison of Cartesian, Cylindrical, and Spherical Coordinates Systems

Model Configuration

The model configuration in Cartesian coordinates is displayed in Figure 2.2. In cylindrical coordinates, the r-axis is the radial component, the θ-axis is the angular coordinate ($\theta \in [0, 2\pi]$) and the z-axis is the height of the cylinder (Figure 2.12a). Gravity acts in the radial direction pointing towards the central axis of the cylinder and shearing occurs along direction Z, parallel to the cylinder axis. The model configuration is essentially the same as in Cartesian coordinates (Figure 2.2), and a pseudo-cube, representing the model domain, is taken at the rim of the cylinder (Figure 2.12a). This method allows to decrease the curvature of the model domain by increasing the radius of the cylinder. In spherical coordinates, r is the radial distance, $\theta \in [0, \pi]$ is the polar angle and $\varphi \in [0, 2\pi]$ is the azimuthal angle (Figure 2.12b). In the spherical model, gravity acts along the radial axis, pointing toward the center of the sphere. Shearing occurs along the azimuthal axis φ. The model configuration is again essentially the same as in Cartesian coordinates (Figure 2.2), and a pseudo-cube is taken at the surface of the sphere. As in cylindrical coordinates, this method allows to decrease the curvature of the model domain by increasing the radius of the sphere.

As a first step of comparison of the results of the three different coordinate systems, we consider a large radius defining the curvature in the cylindrical and spherical coordinate systems. Hence, the geometry of the employed model domain for the cylindrical and spherical coordinates is essentially the same cube as for the Cartesian coordinate system. Consequently, also the applied boundary conditions are essentially identical for the three coordinates systems. The aim of these simulations is to compare the results obtained for

Cartesian, cylindrical and spherical coordinates, which represents a test of three different numerical algorithms employing different system of equations. If the results from the three algorithms are equal, then the numerical implementation of the employed system of equations is correct.

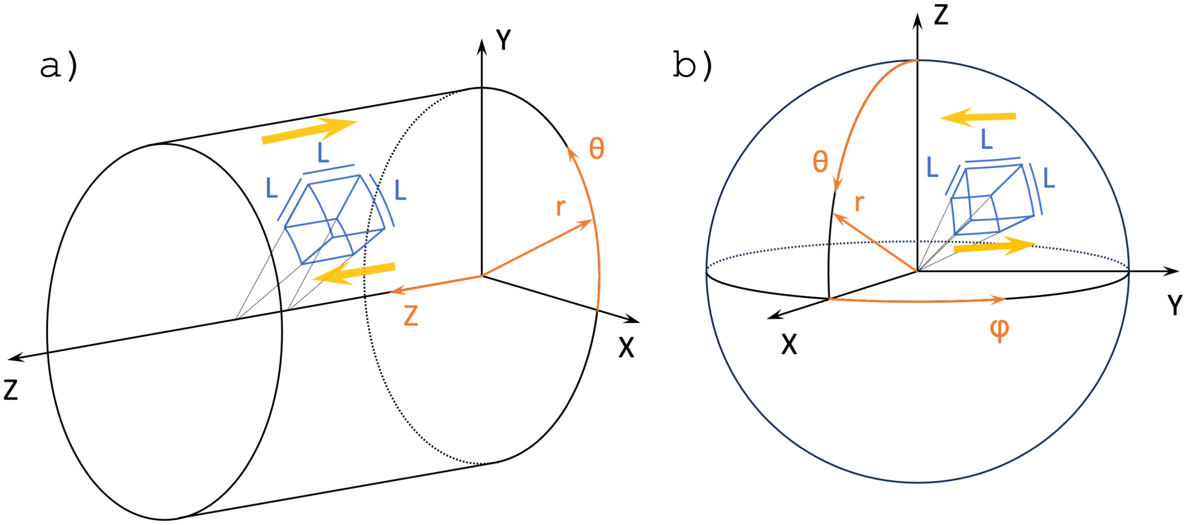

Figure 2.12: Model domain in (a) cylindrical and, (b) spherical coordinates systems. (a) Cylinder is rotated so the Z-axis becomes a horizontal coordinate and gravity acts in the radial direction. The cylindrical coordinates (r, θ, Z) are displayed in orange. Model domain of size $L \times L \times L$ is taken at the rim of the cylinder (blue area) and shearing occurs in direction Z (yellow arrows). (b) For the spherical coordinates, axis (r, θ, φ) are displayed in orange and gravity points toward the center of the sphere. The model domain of size $L \times L \times L$ is at the surface of the sphere (blue area) and the shearing occurs along the φ-direction (yellow arrows). In both coordinates systems, the diapir is of size $L/3$ and is located at the center of the domain (see Figure 2.2).

Results

We performed the stress quantification with three different numerical algorithms for different governing equations that are formulated for Cartesian, cylindrical and spherical coordinates (compare Section 2.2.3, Appendices A and B). Figure 2.13 displays the comparison between the three algorithms for the total vertical stress, pressure and vertical

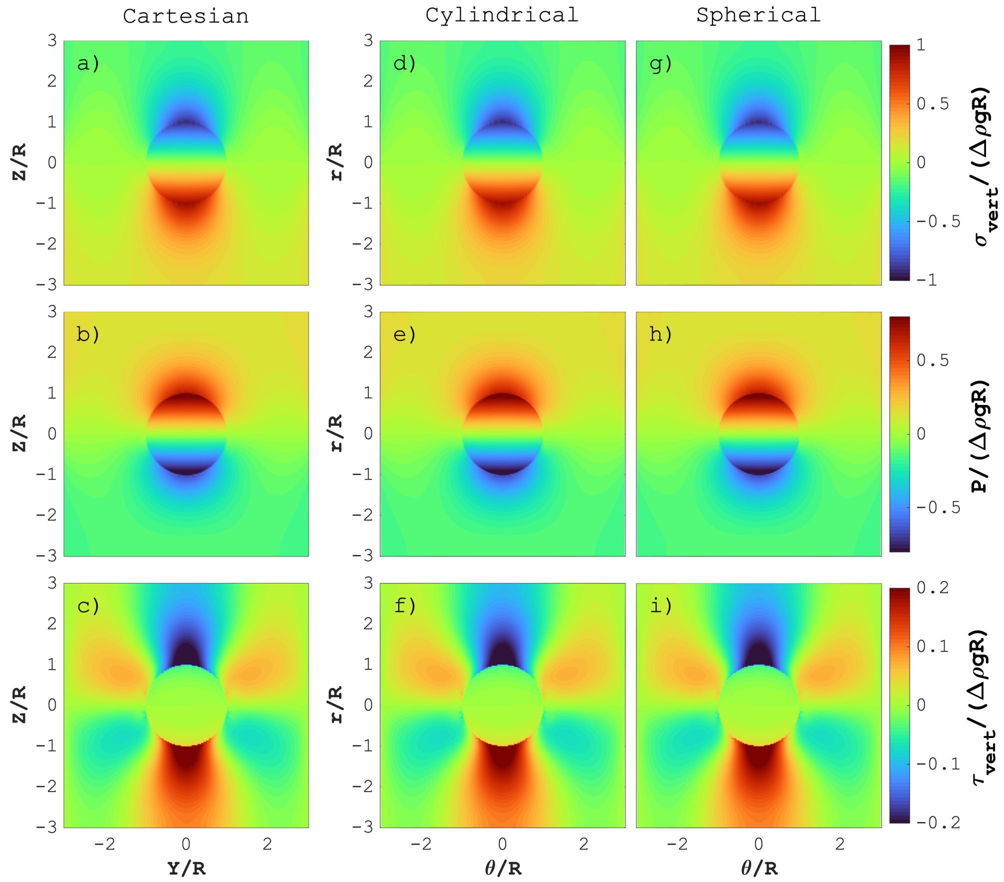

Figure 2.13: Numerical results for $\tau_{\mathrm{R}}/\tau_{\mathrm{C}} = 1$ and $\Delta\rho g R/\tau_{\mathrm{C}} = 10$. Vertical cross sections at position $X/R = 0$ and $\theta/R = 0$ (see Figure 2.3a) of vertical total stress (a, d, g), pressure (b, e, h) and vertical deviatoric stress (c, f, i) in Cartesian (a, b, c), cylindrical (d, e, f) and spherical (g, h, i) coordinates. All stresses are normalized by the buoyancy stress $\Delta\rho g R$. For comparison, the curvature used in the cylindrical and spherical coordinates is so small that it is not visible.

deviatoric stress (for a detailed explanation of these stresses and their relationship, see Section 2.3.2). The numerical results of the three algorithms are identical. This agreement suggests that the three algorithms and the three systems of equations are correct (Figure 2.13). This agreement is further confirmed by Figure 2.5, which shows that the maximum vertical velocities calculated by the three algorithms are equal.

Appendix D. Numerical Performances

Evaluating the performance of an algorithm is a challenging task, especially if one seeks at employing some absolute instead of relative metrics. In the present study, we are relying on iterative and matrix-free stencil-based solvers. In that particular configuration, we identify three criteria to evaluate performance, namely: (a) the effective memory throughput, (b) the scalability of iteration count with resolution, and (c) the parallel efficiency.

First, the effective memory throughput (Figure 2.14b) is used to evaluate the amount of non-redundant memory transfer that leads to saturating the memory bandwidth, which is the limiting factor in our configuration since we are memory-bound (further details in Räss et al., 2022). This means that in our implementation, data transfers between computer units and main GPU memory are the bottleneck, and not the arithmetic operations themselves. The effective memory throughput is defined as $T_{\mathrm{eff}} = A_{\mathrm{eff}}/t_{\mathrm{it}}$, where A_{eff} is the effective memory access in GB and t_{it} is the time per iteration in seconds. Evaluating T_{eff} as function of resolution, we reach a plateau for resolutions larger 255 (Figure 2.14b). This means that passed this resolution the memory bandwidth is saturated, hence the maximal performance is reached. Also, the peak value of T_{eff}, here about 200GB/s for an Nvidia A100 GPU, means that we are about 6x below memory copy only rates (if not doing any actual computations). Further optimisations such as using shared memory to reduce cache misses could lead to bridge most of this gap, especially for 3D computations.

Second, we assess the scalability of the iteration count as function of numerical resolution (Figure 2.14a). The iteration count per time step normalized by the numerical grid resolution in one of the spatial direction, iter/n_r, remains constant and even slightly drops while resolution increases, confirming the (super-)linear scaling of the accelerated pseudo-transient (PT) method.

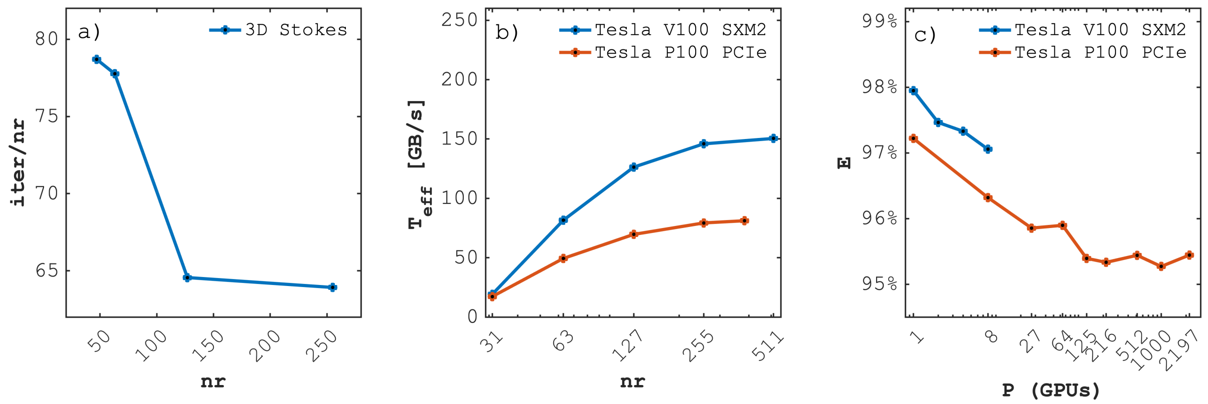

Figure 2.14: Performance evaluation. (a) Scaling of the normalized iteration count as function of the spatial resolution n_r, (b) effective memory throughput T_{eff} as function of numerical grid resolution n_r evaluated on two different GPUs (Nvidia Tesla P100 and Tesla V100), and (c) the parallel efficiency E evaluated on two different GPUs.

Third, we evaluate the parallel efficiency of our multi-GPU implementation (Figure 2.14c). Multi-GPU configuration is required if the problem we solve is larger than the optimal problem size we can fit onto a single GPU. In this case, we use a weak scaling approach to increase the computational resources proportionally to the global problem size. In this configuration, the parallel efficiency of the solver is important as no time should be lost in communication overhead given the distributed memory setup. Our results show that increasing the number of GPUs has almost no effect on the time per iteration. We achieve this ideal scaling by overlapping MPI communication behind the physics calculations. Our algorithm scales ideally up to 2197 GPUs, on the Piz Daint supercomputer at the Swiss National Supercomputing Center (CSCS). Hence it can be used to solve larger problems.

Finally, we also evaluate the sensitivity of the physical results on the numerical resolution and exit criteria (tolerance) for our iterative solver. The exit criteria represents the non-linear tolerance value we converge the residuals to using the PT scheme (Equation 2.20). Figure 2.15 shows that a spatial resolution of minimum 207 grid points in one of the spatial directions (total resolution is 207^3) and an exit criteria of maximum 10^{-6} are necessary to deliver accurate results.

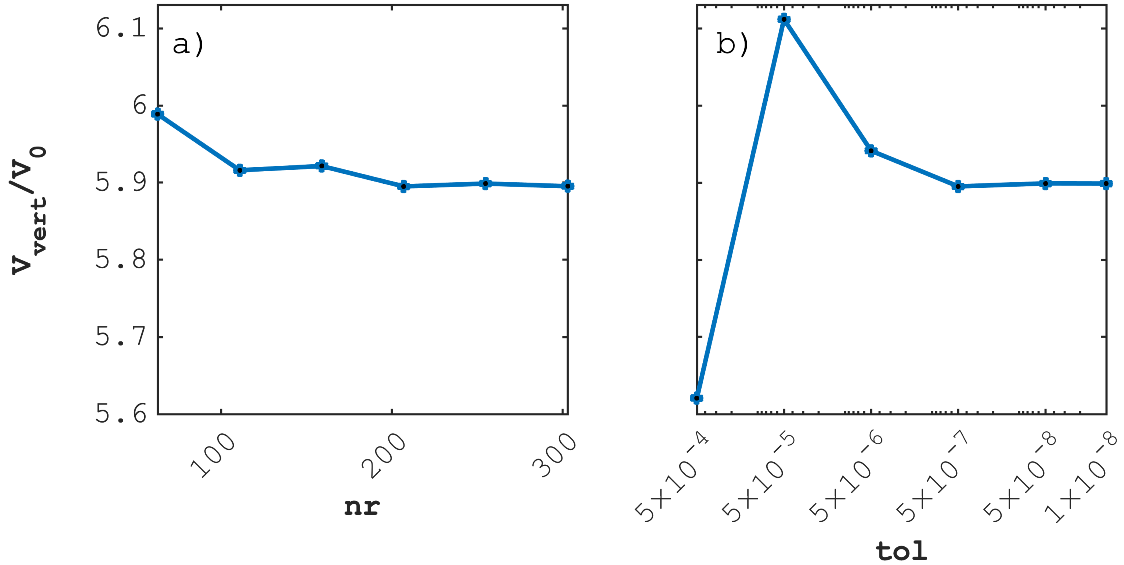

Figure 2.15: Numerical convergence test. (a) Numerical resolution, and (b) nonlinear tolerance required to obtain accurate results. Convergence is achieved when values of V_{vert}/V_0 reach a constant value.

Appendix E. Parameters Used in Figure 2.9

In this section, we provide the parameters used to calculate the flow laws in Figure 2.9 (Table 2.2).

Table 2.2: Parameters Used in Flow Laws.

Flow law for olivine (Hirth and Kohlstedt, 2003): $\dot{\epsilon} = A * \sigma^n * D^{-p} * C_{\text{OH}}^r * exp(-\frac{E+P*V}{R*T})$.
$D = 10$ mm and $C_{\text{OH}} = 1000\,\text{H}/10^6\,\text{Si}$.

Parameters	A $[\text{MPa}^{-\text{n}} \mu\text{m}^{\text{p}} \text{s}^{-1}]$	n	p	r	E [kJ/mol]	V $[10^{-6}\text{m}^3/\text{mol}]$
Wet diffusion	10^6	1	3	1	335	4
Wet dislocation	90	3.5	0	1.2	480	11

Flow law for anorthite (Rybacki and Dresen, 2000): $\dot{\epsilon} = A * \sigma^n * D^{-p} * exp(-\frac{E}{R*T})$.

Parameters	A $[\text{MPa}^{-\text{n}} \mu\text{m}^{\text{p}} \text{s}^{-1}]$	n	p	E [kJ/mol]
Wet diffusion	$10^{1.7}$	1	3	170
Wet dislocation	$10^{2.6}$	3	0	356

D is grain size and R is gas constant.

CHAPTER 3

3D Stresses and Gravitational Spreading Velocities in Continental Plateaus: Analytical Estimates, Numerical Calculations With Earth's Curvature and Application to the Tibetan Plateau

Emilie Macherel[1], **Ludovic Räss**[1,2,3], **Stefan M. Schmalholz**[1]

[1]University of Lausanne, ISTE, Lausanne, Switzerland

[2]Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Switzerland

[3]Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland

Abstract

Understanding stresses is crucial for geodynamics since they govern rock deformation and metamorphic reactions. However, the magnitudes and distribution of crustal stresses are still uncertain. Here, we use a 3D numerical model to investigate stress magnitudes and distributions around continental plateaus that result from lateral crustal thickness variations like those observed for the Tibetan plateau. We consider a scenario of gravitational collapse and do not consider any far-field deformation such as plate convergence. We assume viscous creep, a simplified plateau geometry, and simplified viscosity and density distributions to focus on the main features of the 3D stress field. Specifically, we study the impact of Earth's double curvature, the plateau's corner, the viscosity ratio between the crust and lithospheric mantle, and a stress-dependent power-law flow law, on the crustal stress field. Our results show that Earth's curvature has a minor impact on the stress distribution. Conversely, the corner strongly affects the stress distribution, particularly the shear stresses, which increase nonlinearly toward the corner. Two orders of magnitude variation in crustal and lithospheric mantle viscosities change the maximum crustal stress only by a factor of $\approx$2. We derive simple analytical estimates for the crustal deviatoric stress and spreading velocity. These estimates agree to first order with 3D numerical calculations. We apply these estimates to calculate the average crustal viscosity in the eastern Tibetan plateau as $\approx 10^{22}$ Pa·s. We further discuss potential implications of our results to strike-slip faulting and fast exhumation around the Tibetan plateau's syntaxes.

Plain Language Summary

This study focuses on understanding the stresses in the Earth's lithosphere, which is crucial for understanding how rocks deform and undergo metamorphic reactions. However, there

is still uncertainty about the exact magnitudes and distribution of these stresses, especially in three dimensions (3D). To estimate long-term stresses in the lithosphere, scientists often use observed variations in the thickness of the Earth's crust around continental plateaus, like the Tibetan plateau which has an average altitude of approximately 5 km. Here, a 3D numerical model is used to explore the magnitudes and distribution of stress around these plateaus. The study considers factors like the Earth's curvature, variations in crustal thickness, and the viscosity (a measure of a material's resistance to flow) of different parts of the lithosphere. The results show that Earth's curvature has a minor impact on the stress distribution. However, the corner regions of plateaus strongly influence the stress distribution. The plateaus we study will flow apart under gravity on geological time scales which is a process often termed gravitational collapse. We derive simple mathematical equations to estimate the crustal stress and the gravitational spreading velocity and we test these estimates with the results of the performed 3D numerical calculations.

Keypoints

- The double curvature of the Earth has a minor impact on the 3D stress distribution around the Tibetan plateau
- Shear stresses are largest around the plateau's corner region and increase nonlinearly toward the corner region
- Simple analytical estimates for crustal deviatoric stress and spreading velocity agree with 3D numerical results

3.1 Introduction

Stress is a crucial quantity for geodynamic processes since it governs rock deformation (e.g., Karato, 2008; Turcotte and Schubert, 2021) and metamorphic reactions (e.g., Philpotts and Ague, 2022). Many studies have focused on various aspects of stresses in the lithosphere, such as their sources, magnitude, distribution, or orientation (e.g., Turcotte and Oxburgh, 1976; McGarr and Gay, 1978; Bott and Kusznir, 1984; Zoback, 1992; Engelder, 2014). Differential stresses quantify the deviation from the hydrostatic, or lithostatic, state of stress for which all principal stresses are equal (e.g., Turcotte and Schubert, 2021). It has been known for more than a hundred years that the existence of dry land and any differences in height on the Earth's surface is decisive evidence that the stress in the Earth is not hydrostatic, or lithostatic (e.g., Darwin, 1882; Jeffreys, 1932). Therefore, mountain ranges, continental plateaus, and continental tectonic plates cannot be in a lithostatic state of stress and must exhibit differential stresses, even if they do not deform significantly (e.g., Jeffreys, 1932; Turcotte and Schubert, 2021).

The magnitudes of these differential stresses in the lithosphere, especially in the continental crust, remain still debated (e.g., Kanamori, 1980; Hardebeck and Okada, 2018). The median value of stress drop from earthquakes is approximately 4 MPa for all fault types and this value is approximately constant with depth (e.g., Allmann and Shearer, 2009). Some studies propose that such stress drops in earthquakes measure a quantity that is close to the total shear stress on the faults because of, for example, observations of the reversal of slip (e.g., England and Molnar, 2015) or the rotation of the principal stress axes following earthquakes (e.g., Hardebeck and Okada, 2018). Such studies based on stress drops imply that the continental crust is mechanically weak and that differential stresses in the crust are $\leq$20 MPa (e.g., Hardebeck and Okada, 2018). In contrast, stress estimates for naturally deformed rock associated with pseudotachylyte formation, representing paleoearthquakes, often range between 400 and 1000 MPa. (e.g., Andersen et al., 2008; Brückner and Trepmann, 2021; Campbell and Menegon, 2022), suggesting that shear stresses can be significantly larger than median stress drop magnitudes of 4 MPa. The assumption that low magnitudes of stress drops are close to total stress magnitudes in the crust also disagrees with stress estimates from boreholes which provide differential stresses of ≈150 MPa at a depth of 8 km, in agreement with a friction coefficient of ≈0.6 (Townend and Zoback, 2000). Additionally, significant stresses can also be generated by volumetric rock deformation. For example, deviations from lithostatic pressure can be up to 1.5 GPa for host-inclusion systems, for example, when coesite enclosed in garnet is exhumed and should transform back to quartz (e.g., Angel et al., 2015). Also, the hydration of periclase, enclosed in calcite, causes the reaction from periclase to brucite which generates a volumetric expansion (e.g., Plümper et al., 2022). This volumetric deformation can cause differential stresses of several hundreds of MPa in the calcite surrounding the brucite (Plümper et al., 2022).

The various stress estimates mentioned above are representative of spatial scales significantly smaller than the crustal thickness and/or of short-term, or transient, processes. Hence, the contribution of such local and transient stresses to the long-term, say >1 Myr, strength and state of stress of the entire crust and lithosphere remains unclear. A common method to estimate lithospheric stress magnitudes that are representative of its long-term stress and strength is based on vertical integrals of the two-dimensional (2D) force balance equations (e.g., Dalmayrac and Molnar, 1981; Fleitout and Froidevaux, 1982; Molnar and Lyon-Caen, 1988). With this method, the vertical integral of the horizontal differential stress across the lithosphere can be estimated from the lateral variations of crustal thickness and topography (e.g., Jeffreys, 1932; Artyushkov, 1973) or, more generally, from lateral variations of the gravitational potential energy per unit area, GPE (e.g., Parsons and Richter, 1980; Molnar and Lyon-Caen, 1988; Molnar et al., 1993; Schmalholz et al., 2014). These integrated stress estimates are robust because they are independent of constitutive equations, such as creep flow laws, and require only representative values for topography, crustal thickness, and densities of the crust and lithospheric mantle, which are well constrained.

The Tibetan plateau is the largest continental plateau on Earth, exhibiting an average topography of ≈5 km and a crustal thickness of ≈70 km (e.g., Kind et al., 2002; Royden et al., 2008; Nábělek et al., 2009)(Figure 3.1). Therefore, it has been the focus of many theoretical studies on lithospheric stress (e.g., Jeffreys, 1932; England and Houseman, 1988; Molnar and Lyon-Caen, 1988; Molnar et al., 1993; Liu and Yang, 2003; Ghosh et al., 2006; Warners-Ruckstuhl et al., 2013). For the Tibetan plateau, GPE-based estimates for the vertically integrated stress differences vary between 0.7×10^{13} and 10^{13} N/m (Figure 3.1c), and the corresponding vertically averaged stresses are between 70 and 100 MPa for a 100 km thick lithosphere (e.g., Molnar and Lyon-Caen, 1988; Molnar et al., 1993; Schmalholz et al., 2014, 2019). To study 3D aspects of the formation of the Tibetan plateau, so-called thin viscous sheet models have been applied that are based on vertical integrals of the 3D force balance equations (e.g., Bird and Piper, 1980; England and McKenzie, 1982; England and Houseman, 1986, 1988; Medvedev and Podladchikov, 1999). Based on the comparison between observations and model predictions, studies employing thin viscous sheet models could, for example, estimate that the vertically averaged effective lithospheric viscosity of the Tibetan plateau is in the order of 10^{22} Pa·s (e.g., England and Molnar, 1997; Flesch et al., 2001).

Models that are based on vertical integrals of the force balance equations, such as thin viscous sheet models, cannot estimate the vertical distribution of stress across the lithosphere. Therefore, the relative contributions of the continental crust and the lithospheric mantle to the long-term stress and strength of the lithosphere remain unclear. For example, some studies argue that a significant part of the lithospheric strength, required to support the Tibetan plateau, resides in the seismogenic upper crust (e.g., Flesch et al., 2001). In

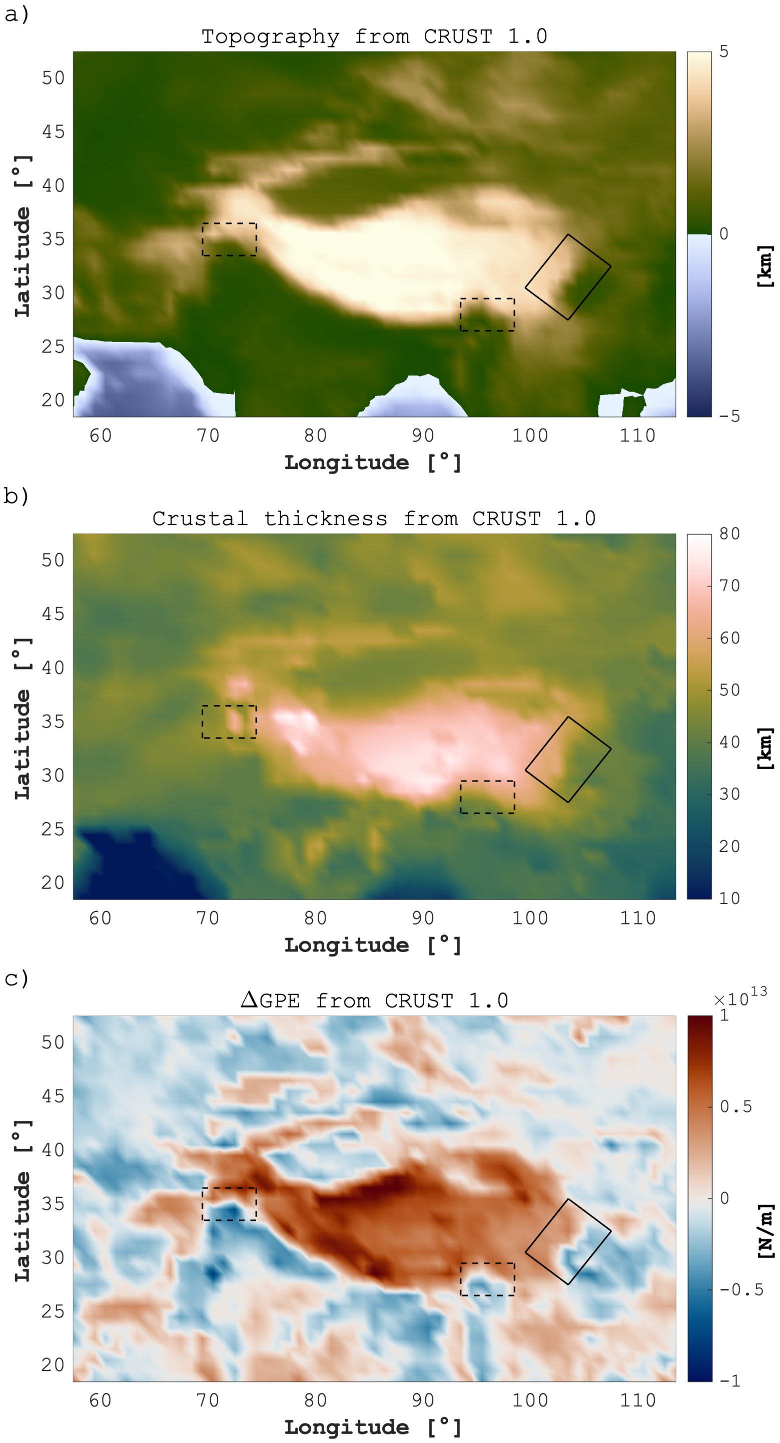

Figure 3.1: Map view of the (a) topography, (b) crustal thickness, and (c) lateral variations of gravitational potential energy per unit area (ΔGPE) of the Tibetan plateau and its surroundings using the CRUST 1.0 dataset (http://igppweb.ucsd.edu/∼gabi/rem.html; Laske et al., 2013). The black rectangle indicates the region in which the Longmen Shan is located and the two dashed rectangles indicate the regions in which the two syntaxes are located. (see Discussion 3.4.8).

contrast, other studies on the neighbouring Tien Shan propose that the major part (up to 90 %) of the lithospheric strength resides in the ductile lithospheric mantle (e.g., England and Molnar, 2015). One argument for a weak continental crust is that typical stress drop magnitudes are on the order of a few MPas (e.g., Allmann and Shearer, 2009) so that the vertically integrated crustal stress does not contribute significantly to the stress integrated across the lithosphere (e.g., England and Molnar, 2015). However, 2D numerical models considering vertical viscosity and stress variations show that the Tibetan plateau collapses within less than 1 Myr if shear stresses in the crust are limited to 5 MPa (Schmalholz et al., 2019).

2D models can consider the impact of vertical viscosity variations on the lithospheric stress distribution, but they cannot evaluate the impact of lateral topography and crustal thickness variations on the stresses, for example, around the corner of the plateau. To evaluate the impact of both vertical strength variations and lateral geometry variations on the lithospheric stress field, 3D numerical models are required (e.g., Lechmann et al., 2014; Pusok and Kaus, 2015; Bischoff and Flesch, 2019; Chen et al., 2020; Yang et al., 2020; Zhang et al., 2022). However, essentially all 3D numerical models focusing on the Tibetan plateau employ a rectangular model geometry and do not consider the potential impact of Earth's double curvature on the stress field around the Tibetan plateau (e.g., Lechmann et al., 2014; Chen et al., 2020; Yang et al., 2020; Zhang et al., 2022). The double curvature of the Earth's lithosphere could potentially have an impact on its mechanical strength and stress state because mechanical studies demonstrate that small amounts of bending of plates reinforce the mechanical stability of thin sheets (e.g., Pini et al., 2016). Several numerical studies investigated the impact of a locally curved and geometrically-stiffened plate on the India-Asia collision and related exhumation (e.g., Bendick and Ehlers, 2014; Koptev et al., 2019). Amongst others, these studies show that the geometrical stiffening effect might be important for localized regions with fast exhumation such as syntaxes.

Furthermore, several theoretical studies investigated the impact of a spherical geometry on free subduction. Chamolly and Ribe (2021) found that the Earth's sphericity has a modest impact on the sinking speed of a slab but a much larger impact on the stress field in the slab which is important for the along-strike buckling of slabs. Chen et al. (2022) compared results of free subduction simulations in 3D Cartesian and spherical shell domains. They found differences between the simulations that are related, for example, to the reduction of space with depth in spherical shells which enhances along-strike buckling and trench curvature. The differences between Cartesian and spherical models increase for wider slabs. Therefore, both the Earth's curvature and lateral geometry variations around corners likely have an impact on the magnitude and distribution of the lithospheric stress field around continental plateaus, but the magnitude of this impact remains unclear.

Here, we apply a 3D numerical model for viscous flow under gravity to investigate the stress field around continental plateaus. We employ typical values for topography, crustal thickness, and densities for the Tibetan plateau. We do not consider any far-field deformation, such as convergence, and, hence, we study a continental plateau that will undergo a gravitational collapse (e.g., Rey et al., 2001). In our study, we quantify (a) the impact of increasing curvature on the crustal stress field around continental plateaus, (b) the magnitudes of differential stresses in the crust required to support the Tibetan plateau, (c) the impact of corners on normal and shear stresses in the crust, (d) the impact of viscosity differences between the crust and lithospheric mantle and (e) the impact of a stress-dependent, power-law flow law on the crustal stress field and horizontal velocities resulting from gravitational spreading of the plateau. Furthermore, we derive simple analytical estimates for the horizontal deviatoric stress in the crust and the horizontal gravitational spreading velocity. We test these estimates with the results of the performed 3D numerical calculations.

3.2 Model

3.2.1 Analytical Estimates for Stress and Spreading Velocity

Stress magnitudes in the lithosphere around continental plateaus and mountains can be estimated from lateral variations, or differences, of the gravitational potential energy per unit area, GPE, (e.g., Jeffreys, 1932; Dalmayrac and Molnar, 1981; Molnar and Lyon-Caen, 1988). The GPE is the vertical integral of the lithostatic pressure (e.g., Molnar and Lyon-Caen, 1988). The difference in GPE, ΔGPE, between plateau and surrounding lowland is given by (e.g., Molnar and Lyon-Caen, 1988; Schmalholz et al., 2014)

$$\Delta\mathrm{GPE} = \rho_{\mathrm{c}} g h_{\mathrm{a}} \left(h_{\mathrm{c}} + \frac{\rho_{\mathrm{m}}}{\rho_{\mathrm{m}} - \rho_{\mathrm{c}}} \frac{h_{\mathrm{a}}}{2} \right) \tag{3.1}$$

where ρ_{c}, ρ_{m}, h_{c}, h_{a} and g are the crustal density, the mantle density, the crustal thickness of the lowland, the altitude of the plateau and the gravitational acceleration, respectively (Figure 3.2b). Schmalholz et al. (2014) provide an estimate for the magnitude of the horizontal deviatoric stress in the crust, τ_{e}, given by

$$\tau_{\mathrm{e}} \approx \frac{\Delta\mathrm{GPE}}{2\left(2h_{\mathrm{c}} + h_{\mathrm{a}} + h_{\mathrm{r}}\right)} \tag{3.2}$$

where h_{r} is the thickness of the crustal root under the plateau (Figure 3.2b). Using the isostatic relation

$$h_{\mathrm{r}} = \frac{\rho_{\mathrm{c}} h_{\mathrm{a}}}{\rho_{\mathrm{m}} - \rho_{\mathrm{c}}} \tag{3.3}$$

and the expression for ΔGPE from Equation 3.1 yields an expression for τ_{e} in the form of

$$\tau_{\mathrm{e}} \approx \frac{\rho_{\mathrm{c}} g h_{\mathrm{a}}}{4}. \tag{3.4}$$

This stress estimate is independent of rheological parameters such as viscosity.

The ΔGPE between plateau and lowland will cause a gravitational collapse of the continental plateau which was defined by Rey et al. (2001) as "the gravity-driven ductile flow that effectively reduces lateral contrasts in gravitational potential energy". We assume that the crust behaves during gravitational collapse like a linear viscous fluid. The deviatoric stress, τ_e, in the crust generates a strain rate, $\dot{\epsilon}$, which depends on the crustal viscosity, η_c, and is given by the linear viscous flow law

$$\dot{\epsilon} = \frac{\tau_\mathrm{e}}{2\eta_\mathrm{c}}. \tag{3.5}$$

During gravitational collapse, the thick crust of the plateau will thin and the plateau will spread horizontally. We assume that $\dot{\epsilon}$ can be approximated by the ratio of the horizontal spreading velocity, V_e, to a characteristic length, L, over which the spreading occurs:

$$\dot{\epsilon} \approx \frac{V_e}{L}. \tag{3.6}$$

Using Equations 3.4, 3.5 and 3.6, we obtain an estimate for V_e:

$$V_\mathrm{e} \approx \frac{\rho_\mathrm{c} g h_\mathrm{a} L}{8\eta_\mathrm{c}}. \tag{3.7}$$

We will use the results of 3D numerical calculations to test the applicability of the estimate for the stress, Equation 3.4, to determine the characteristic length, L, and to test the estimate for the spreading velocity, Equation 3.7.

To assess the impact of more complicated viscous flow laws, we consider also a stress-dependent viscosity (e.g., Karato, 2008). We apply a combination of a linear and power-law viscous flow law for which the effective crustal viscosity, η_{cE}, is

$$\eta_{\mathrm{cE}} = \frac{2\eta_{\mathrm{c}}}{1 + \left(\frac{\tau_{\mathrm{e}}}{\tau_{\mathrm{C}}}\right)^{(n-1)}} \tag{3.8}$$

where n is the power-law stress exponent. For the analytical estimate, we assume that the effective viscosity is controlled by the deviatoric stress estimate, τ_{e}. τ_{C} defines the stress magnitude at which the deformation behavior changes from a linear viscous flow law to a power-law viscous flow law (Schmalholz and Podladchikov, 2013). If $n = 1$, then $\eta_{\mathrm{cE}} = \eta_{\mathrm{c}}$, and the flow is linear viscous. For $n > 1$, $\eta_{\mathrm{cE}} < \eta_{\mathrm{c}}$ if $\tau_{\mathrm{e}} > \tau_{\mathrm{C}}$, an effect often referred to as stress weakening. There is no limit for the decrease of η_{cE} if τ_{e} becomes infinitely large. Conversely, for $n > 1$, $\eta_{\mathrm{cE}} > \eta_{\mathrm{c}}$ if $\tau_{\mathrm{e}} < \tau_{\mathrm{C}}$. The maximal value of $\eta_{\mathrm{cE}} = 2\eta_{\mathrm{c}}$ if τ_{e} would be zero. The applied stress-dependent flow law, Equation 3.8, is useful to asses the impact of both weakening and hardening compared to a linear viscosity, η_{c}, for $n = 1$. The flow law of Equation 3.8 is also frequently applied without the factor of 2 in the numerator to mimic a combined flow law with diffusion and dislocation creep. The difference compared to η_{cE} in Equation 3.8 is that for $n = 1$ the effective viscosity does not correspond to the linear viscosity.

Replacing η_{c} with η_{cE}, Equation 3.8, in Equation 3.7 provides a velocity estimate of the form

$$V_{\mathrm{ePL}} \approx \frac{\rho_{\mathrm{c}} g h_{\mathrm{a}} L}{16\eta_{\mathrm{c}}} \left(1 + \left(\frac{\rho_{\mathrm{c}} g h_{\mathrm{a}}}{4\tau_{\mathrm{C}}}\right)^{(n-1)}\right). \tag{3.9}$$

We normalize V_{ePL} by the velocity estimate for linear viscous flow for $n = 1$, V_{e}, to obtain a dimensionless velocity of the form

$$\frac{V_{\mathrm{ePL}}}{V_{\mathrm{e}}} \approx \frac{1}{2}\left(1 + \left(\frac{\rho_{\mathrm{c}} g h_{\mathrm{a}}}{4\tau_{\mathrm{C}}}\right)^{(n-1)}\right). \tag{3.10}$$

We will compare the analytical estimates for spreading velocities for stress-dependent viscosities with the results of 3D numerical calculations.

3.2.2 Mathematical Model

We apply the so-called Stokes equations for slow, incompressible viscous flow under gravity in 3D (e.g., Malvern, 1969; Turcotte and Schubert, 2021). We consider a spherical coordinates system. The corresponding equations of mass conservation and linear momentum conservation are displayed in Appendix A1. As for the analytical estimate, we employ a combination of a linear and power-law viscous flow law for which the effective viscosity, η_{E}, is (e.g., Schmalholz and Podladchikov, 2013; Gerya, 2019):

$$\eta_{\mathrm{E}} = \frac{2\eta}{1 + \left(\frac{\tau_{\mathrm{II}}}{\tau_{\mathrm{C}}}\right)^{(n-1)}} \tag{3.11}$$

where η is the linear viscosity for $n = 1$, n is again the stress exponent and τ_{C} is again the characteristic stress controlling the transition from linear to power-law viscous flow. For the simulated 3D viscous flow, the stress magnitude that controls the effective viscosity is quantified by τ_{II} which is the square root of the second invariant of the deviatoric stress tensor and given by

$$\tau_{\mathrm{II}} = \sqrt{0.5\,\tau_{ij}^2} \tag{3.12}$$

where symbols τ_{ij} represent the components of the deviatoric stress tensor, indices i and j represent the spherical coordinates r, θ and φ, and the Einstein summation convention applies (Appendix A1 and Figure 3.2a). We apply the effective viscosity, η_{E}, in a flow law for the 3D viscous flow calculations:

$$\tau_{\mathrm{ij}} = 2\eta_{\mathrm{E}}\dot{\epsilon_{\mathrm{ij}}} \tag{3.13}$$

where $\dot{\epsilon_{\mathrm{ij}}}$ are the components of the devatoric strain rate tensor component. The applied flow law for the individual stress tensor and strain rate tensor components are given in Appendix A1.

3.2.3 Numerical Method

The numerical algorithm to solve the governing system of equations is programmed for spherical coordinates. To solve the governing equations, we discretize them on a staggered grid with constant spacing using the finite difference method (e.g., Virieux, 1986; Gerya, 2019; Räss et al., 2022). The numerical method we use to solve the discretized equations is the pseudo-transient method, which is an iterative solution strategy for stationary problems that allows solving the equations in a matrix-free way (e.g., Räss et al., 2022; Wang et al., 2022). The pseudo-transient method consists of adding a pseudo-time derivative to the governing equations. The initial values for the pressure and velocity fields do not solve the discretized equations. Hence, the idea of the pseudo-transient method is to iterate over the pseudo-time until the pseudo-time derivatives become smaller than a specified tolerance value for all numerical grid points, and a numerical solution is reached. More details on the applied pseudo-transient method are presented in Appendix A2.

3.2.4 Model Configuration

In the applied spherical coordinates system, r is the radial direction, θ is the polar angle ($\theta \in [0; \pi/2]$) and φ is the azimuthal angle ($\varphi \in [0; 2\pi]$; see Figure 3.2a). The geometry of the model domain represents a small 3D region from a sphere and its shape is like a rectangular cuboid that is deformed to adapt to the spherical geometry (Figure 3.2a). The model domain has a size of 1400 km × 1400 km horizontally and 565 km vertically, measured at the surface of the sphere (Figure 3.2a). A continental plateau is configured on a quarter of the model domain with a horizontal size of 600 km × 600 km and a transition zone between the plateau and the surrounding lowland with a width of 100 km (Figure 3.2b). The model domain represents a quarter of a larger, quadratic (in map view) plateau that is entirely surrounded by lowland. The lateral model sides at $Y = -700$ km and $X = -700$ km, bounding the plateau in our model domain (see Figure 3.2b), represent symmetry planes for the deformation of the larger plateau that is entirely surrounded by lowland.

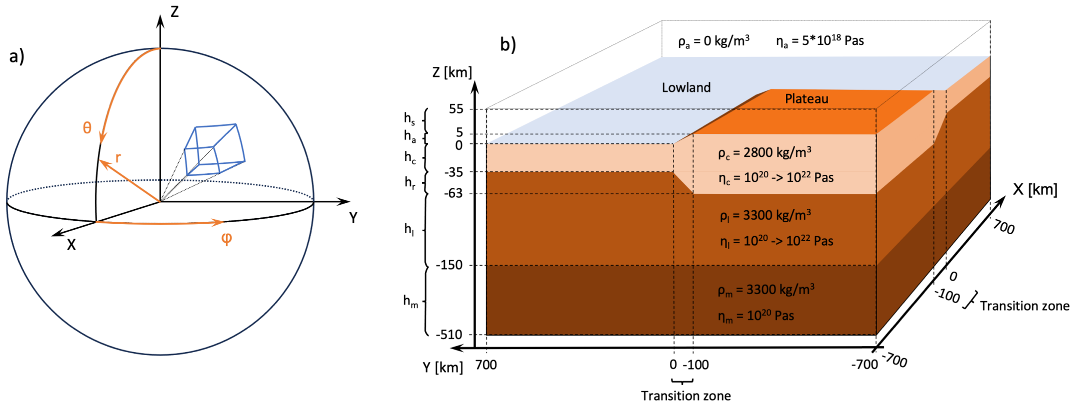

Figure 3.2: (a) Spherical coordinates system. The model domain (blue box) is defined at the surface of a sphere of various radii (for example the Earth's radius). (b) Model configuration, for simplicity sketched in Cartesian coordinates: a cube of size 565 km vertically and 1400 km × 1400 km horizontally, composed by four different layers representing the mantle, the lithospheric mantle, the continental crust, and sticky-air, from bottom to top. A continental plateau in isostatic equilibrium is defined on a quarter of the domain. Gravity acts in the vertical, r-direction.

Vertically, the model domain is divided into four different layers, or model units, representing, from bottom to top, the mantle, the lithospheric mantle, the continental crust, and a zero-density, weak layer that represents air, sometimes referred to as sticky-air layer (e.g., Crameri et al., 2012). The bottom layer has a constant thickness of 360 km, a density of 3300 kg/m^3 and a linear viscosity of 10^{20} Pa·s. The lithospheric mantle has a thickness of 115 km below the lowland and 87 km below the plateau. It has a density of 3300 kg/m^3 and we apply linear viscosities of 10^{20} Pa·s, 10^{21} Pa·s and 10^{22} Pa·s in different simulations. The continental crust in the lowland is 35 km thick, to which are added 5 km of elevation and 28 km of root to form a plateau in isostatic equilibrium (e.g., Schmalholz et al., 2014). It has a density of 2800 kg/m^3 and we vary its viscosity between 10^{20} Pa·s and 10^{22} Pa·s in different simulations.

The sticky-air layer is applied to allow the surface of the crust to behave essentially as a free surface. It has a thickness of 55 km over the lowland and of 50 km over the plateau. Its density is 0 kg/m^3 and its linear viscosity is 5×10^{18} Pa·s. The thickness and the viscosity of the sticky-air layer have been chosen following a criterion defined by Crameri et al. (2012). This criterion guarantees that the sticky-air layer is sufficiently weak and thick to mimic a free surface boundary condition. The ratio $(\eta_{\mathrm{st}}/\eta_{\mathrm{ch}})/(h_{\mathrm{st}}/L)^3$ has to be significantly smaller than one, where η_{st} and η_{ch} are the viscosity of the sticky-air and a characteristic viscosity value, respectively, h_{st} is the sticky-air thickness and L is a characteristic length of the model. For the values used in this study, $\eta_{\mathrm{st}} = 5 \times 10^{18}$ Pa·s and $h_{\mathrm{st}} = 50$ km, and we consider $\eta_{\mathrm{ch}} = 10^{20}$ to 10^{22} Pa·s and the order of magnitude of $L \sim 100$ km, the ratio $(\eta_{\mathrm{st}}/\eta_{\mathrm{ch}})/(h_{\mathrm{st}}/L)^3$ ranges then between 0.004 and 0.4. The applied sticky-air layer is, hence, suitable to mimic a free surface boundary condition. The boundary conditions applied in this study are free slip on each lateral side and at the top and bottom of the model domain. We do not prescribe any boundary velocity so that gravity, pointing towards the center of the coordinates system along direction r, is the only driving force acting in the model domain. The applied numerical resolution is 804 m vertically and 7.07 km horizontally, resulting from a numerical mesh with $703 \times 199 \times 199$ grid points.

The applied model geometry and densities generate a ΔGPE between the plateau and the lowlands of $\approx 7 \times 10^{12}$ N/m. A ΔGPE of $\approx 7 \times 10^{12}$ N/m has been typically used in theoretical GPE studies applied to the Tibetan plateau (e.g., Molnar and Lyon-Caen, 1988; Schmalholz et al., 2014) and is a representative value of ΔGPE between the Tibetan plateau and its surrounding lowlands resulting from the CRUST1.0 dataset (Laske et al., 2013) (Figure 3.1c).

Parameters related to the power-law flow law are the power-law exponent, n, and the characteristic stress, τ_{C}. In this study, we test three different values for the power-law exponent, $n = 1$ (linear viscous), $n = 3$ and $n = 6$. The values of $n > 1$ are only tested for a crustal linear viscosity $\eta = 10^{22}$ Pa·s in the equation for the effective viscosity, η_{E} (Equation 3.11). We apply a characteristic stress τ_{C} of 24 MPa. The exact value of τ_{C} is not important, but it is important to choose a value that represents average values of crustal stress so that some crustal regions exhibit stresses $> \tau_{\mathrm{C}}$ and some exhibit stresses $< \tau_{\mathrm{C}}$.

The calculations are done with values in dimensionless form. Three characteristic scales are used to scale the results back to real physical units: one characteristic scale for length, which is the total height of the model domain; one scale for viscosity, which is the linear mantle viscosity; and one scale for density, which is the mantle density.

3.3 Results

3.3.1 Overview

We begin by presenting the 3D stress field for two representative simulations: A simulation with a mechanically strong crust (10^{22} Pa·s) and a weak lithospheric mantle (10^{20} Pa·s), referred to as model SC_WL in the following, and another simulation with a strong lithospheric mantle (10^{22} Pa·s) and crust with intermediate strength (10^{21} Pa·s), referred to as IC_SL. Next, we present results that show the impact of lithospheric curvature on the stress field and then results that present the impact of a corner on the stress field. Afterward, we show simulations that evaluate the impact of the viscosity ratio between the crust and lithospheric mantle on the stress field and finally simulations that show the impact of a stress-weakening power-law flow law. Furthermore, we compare the analytical estimates presented in Section 3.2.1 with the numerical results.

For each model configuration, we have calculated a single time step to obtain the instantaneous 3D stress and velocity fields. For visualization reasons, we display the results for the spherical coordinates in a Cartesian coordinate system.

3.3.2 General Stress State Around a Continental Plateau

Figure 3.3 shows for two simulations (Figure 3.3a to d: SC_WL and Figure 3.3e to h: IC_SL) the deviatoric normal stresses in the three spatial directions and the differential stress ($\sigma_1 - \sigma_3$), that is the difference between the largest and the smallest principal stress at each numerical grid point of the model domain.

For the model SC_WL, high stress magnitudes are inside the crust and are essentially zero in the mantle (Figure 3.3a to d). Conversely, for the model IC_SL, stress magnitudes are larger in the lithospheric mantle compared to the crust (Figure 3.3e to h). In

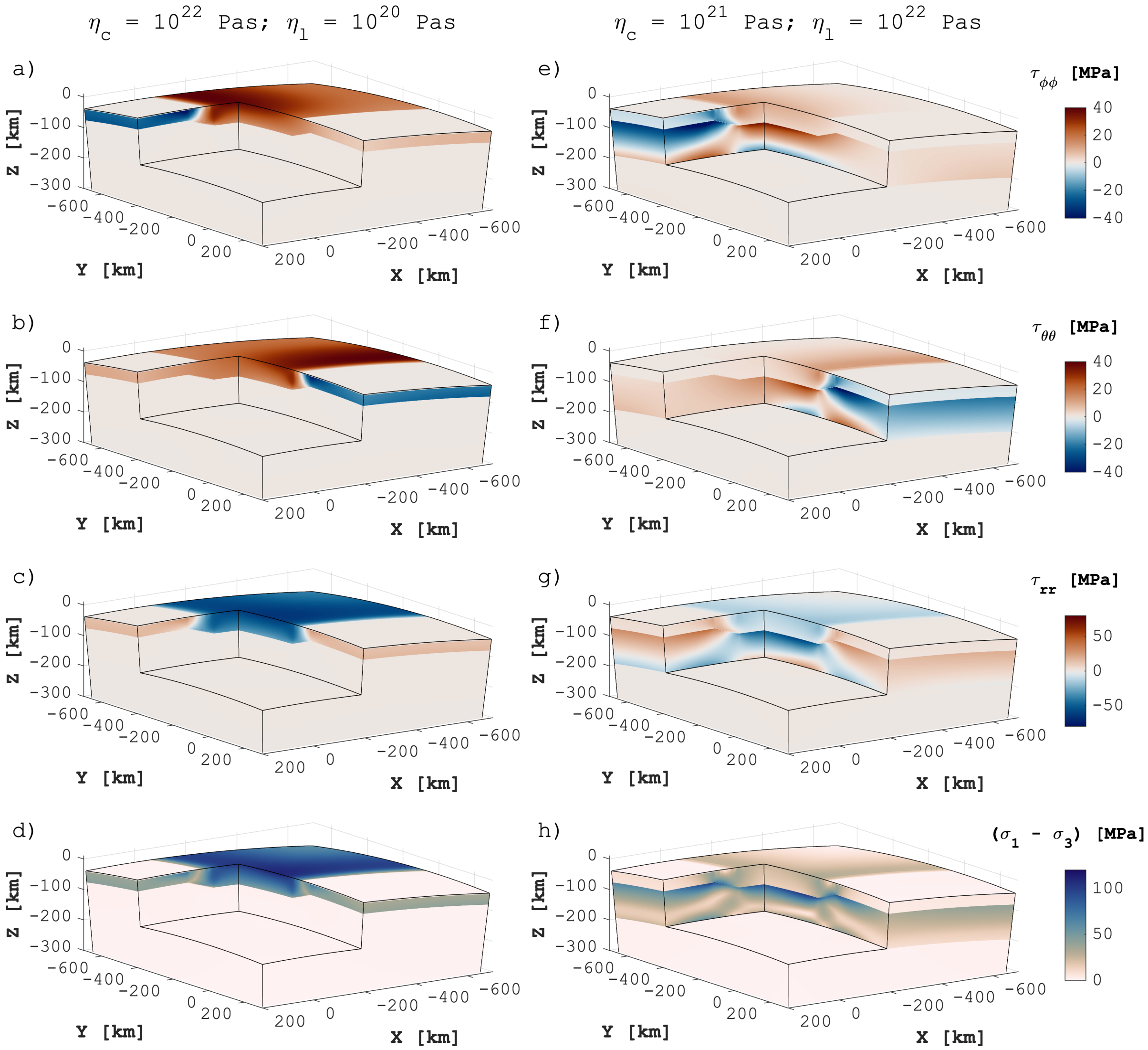

Figure 3.3: 3D visualization of deviatoric normal stresses in the direction parallel to the X-direction (panels a, e), parallel to Y (b, f), and parallel to Z (c, g). Panels (d, h) display the differential stress $(\sigma_1 - \sigma_3)$. Panels (a-d) show results for a simulation with a strong crust (10^{22} Pa·s) and a weak lithospheric mantle (10^{20} Pa·s), model SC_WL, and panels (e-h) for a simulation with an intermediate crust (10^{21} Pa·s) and a strong lithospheric mantle (10^{22} Pa·s), model IC_SL. Positive and negative deviatoric stress values indicate extensional and compressive stresses, respectively. These simulations are performed with the Earth's curvature and for linear viscous flow.

general, in both horizontal directions, the same stress pattern is visible: an extension of the plateau (positive deviatoric stresses) and a compression of the surrounding lowland (negative deviatoric stresses). This pattern represents the lateral, gravitational spreading of the continental plateau. This spreading is further confirmed by the vertical deviatoric stress (Figure 3.3c and g). The negative stress values located inside the plateau indicate a vertical thinning of the crust. Furthermore, the magnitudes of the vertical deviatoric stresses are twice larger than the ones of the horizontal deviatoric stresses. The relation between the deviatoric normal stresses is constrained by their definition that requires that the sum of these deviatoric normal stresses is zero (e.g., Turcotte and Schubert, 2021).

Figure 3.3d shows that the largest differential stresses are located inside the plateau and that the surrounding lowlands are under less stress. The largest differential stresses for the simulation with the model IC_SL are located below the plateau at the top of the lithospheric mantle (Figure 3.3h). Differential stresses are also high below the lowlands at the top of the lithospheric mantle (Figure 3.3h).

In summary, the ratio of crustal viscosity to the lithospheric mantle viscosity has a major impact on the distribution and magnitude of stresses in the crust and lithospheric mantle.

3.3.3 Impact of the Curvature

To quantify the impact of curvature on the stresses in and around the continental plateau, we modify the radius of the spherical coordinates system without changing the geometry of the model domain. In other words, the arc length of the crust stays constant, and as the radius of the sphere decreases the inner angle increases. Hence, the curvature of the domain increases.

Figure 3.4 displays the differential stress ($\sigma_1 - \sigma_3$) for four different curvatures using a radius of curvature corresponding to an infinite radius (no curvature), the radius of the

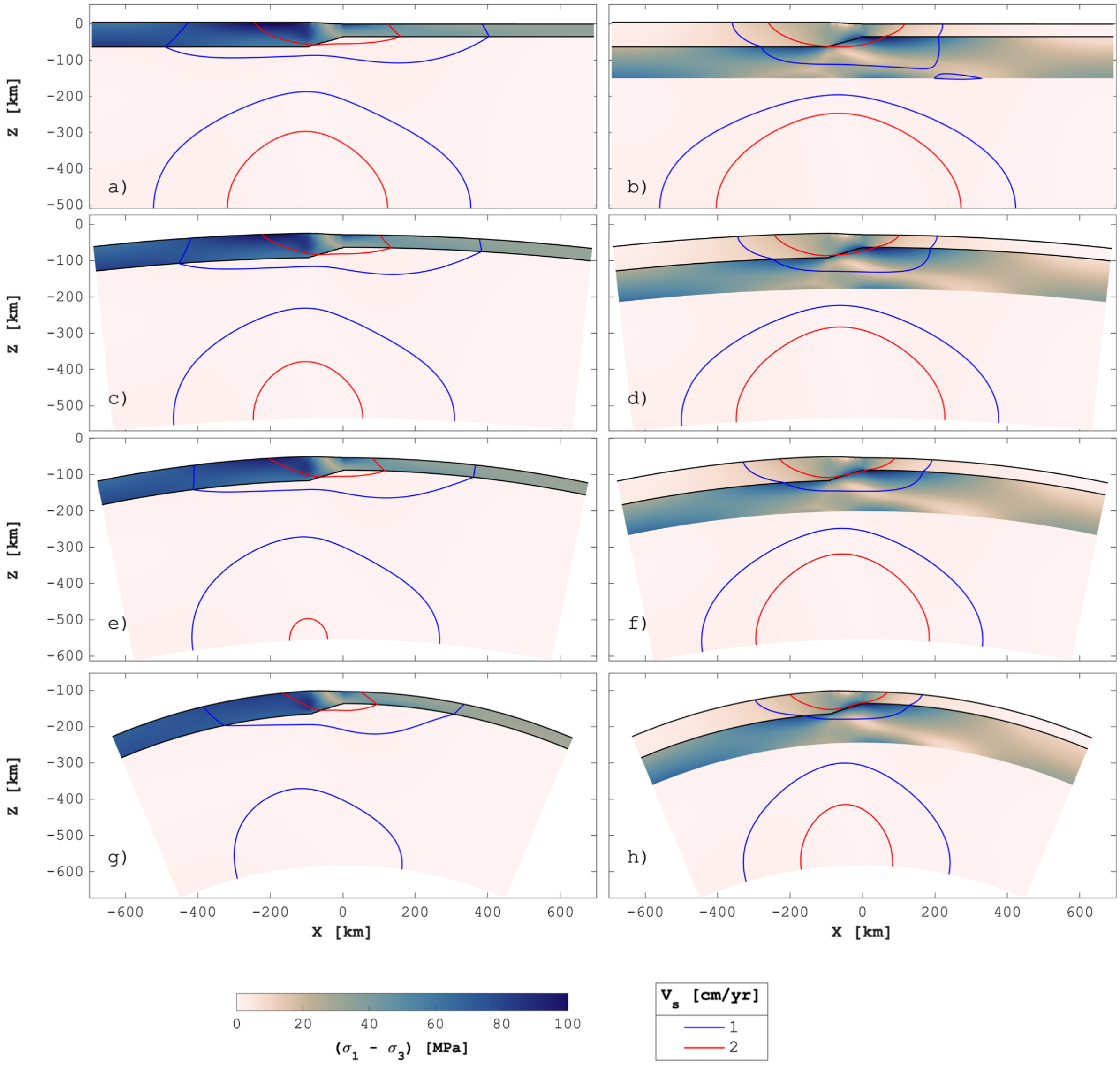

Figure 3.4: Cross-sections at 500 km from the edge of the plateau ($Y = -600$ km) showing the differential stress. Different curvatures corresponding to (a, b) negligible curvature, (c, d) the Earth, (e, f) Mars, and (g, h) the Moon are displayed. Panels (a, c, e, g) display results for model SC_WL, and panels (b, d, f, h) display results for model IC_SL. The contours mark horizontal velocities of 2 cm/yr (red) and 1 cm/yr (blue). The power-law exponent is always 1.

Earth (6371 km), of Mars (3390 km), and of the Moon (1737 km). We use these radii for Mars and the Moon simply as examples to test the impact of increasing curvature. All four curvatures have been tested with model SC_WL and model IC_SL. To first order, we do not see any significant effect of curvature on the stresses and on the horizontal velocities. Representative horizontal velocity magnitudes are indicated by the contour lines. For all curvatures, larger differential stresses are located inside the strong layer of the model, that is inside the crust for model SC_WL and inside the lithospheric mantle for model IC_SL.

Eight vertical profiles of differential stresses, from two horizontal sections, are compared in Figure 3.5 for model SC_WL and in Figure 3.6 for model IC_SL. The different curvatures are displayed for each vertical profile. There are two sets with four vertical profiles each. The two sets show vertical profiles that belong to two horizontal profiles: Profile HP1 (Figure 3.5a to d, and Figure 3.6a to d) is located far away from the corner region and its location is displayed in Figure 3.7d. Profile HP2 (Figure 3.5e to h, and Figure 3.6e to h) is located inside the transition zone between the plateau and the lowland and displayed in Figure 3.7d. For model SC_WL, the impact of the different curvatures on the differential stress profile is generally minor. There is an average difference of 0.15 MPa between the flat simulation and the one with the Earth radius and a maximum difference of less than 10 MPa. Hence, for the case of a 3D model applied to Earth, a flat (Cartesian) model provides stresses that are close to stresses resulting from a model considering the Earth's curvature. For a curvature corresponding to the radius of the Moon, some vertical profiles show opposite trends of stress profiles compared to models with a radius corresponding to Earth (Figure 3.5d, g and h). For example, in the vertical profile in the lowland close to the transition zone and the plateau corner, the differential stresses in the crust increase with depth for the radius of the Moon, but decrease with depth for all other models with larger radii (Figure 3.5g). However, the magnitudes of the differential stresses are similar.

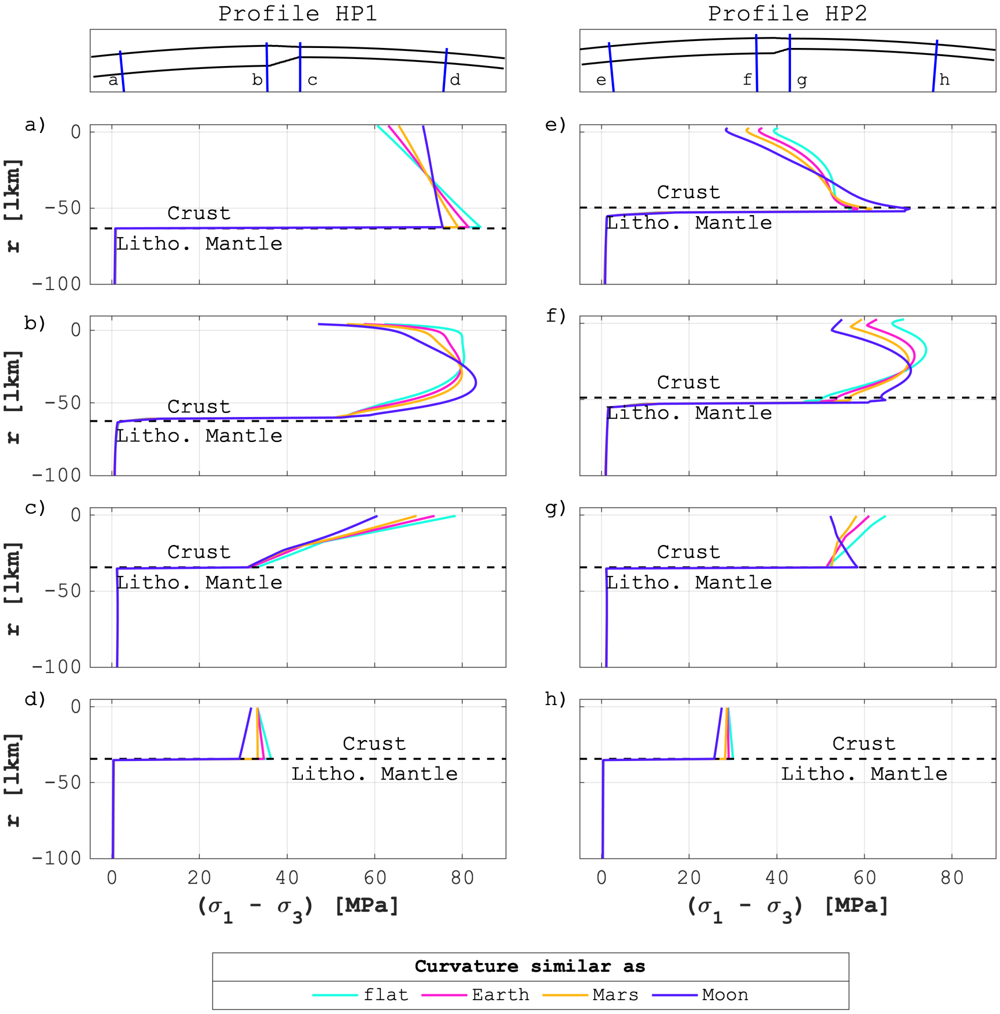

Figure 3.5: Representative vertical profiles of differential stress for model SC_WL with different curvatures. Differential stress, $(\sigma_1 - \sigma_3)$, on the horizontal axis and depth on the vertical axis. In panels (a-d) the profiles are taken at 500 km from the edge of the plateau along the Y-axis (Profile HP1 in Figure 3.7) and their position in direction X are (a) 500 km from the edge of the plateau, (b) 5 km from the edge, (c) 5 km after lowland started, and (d) 500 km after the end of the transition between the plateau and the lowland (positions are indicated in the top panel). In panels (e-h), profiles are taken in the middle of the transition zone between the plateau and its surrounding lowlands (Profile HP2 in Figure 3.7) and are distributed the same way as the left side along direction X (positions displayed in the top panel). The different lines represent different curvatures (see legend) and the dashed line marks the limit between the crust and the lithospheric mantle. All of these simulations are done with the model SC_WL. The power-law exponent is 1.

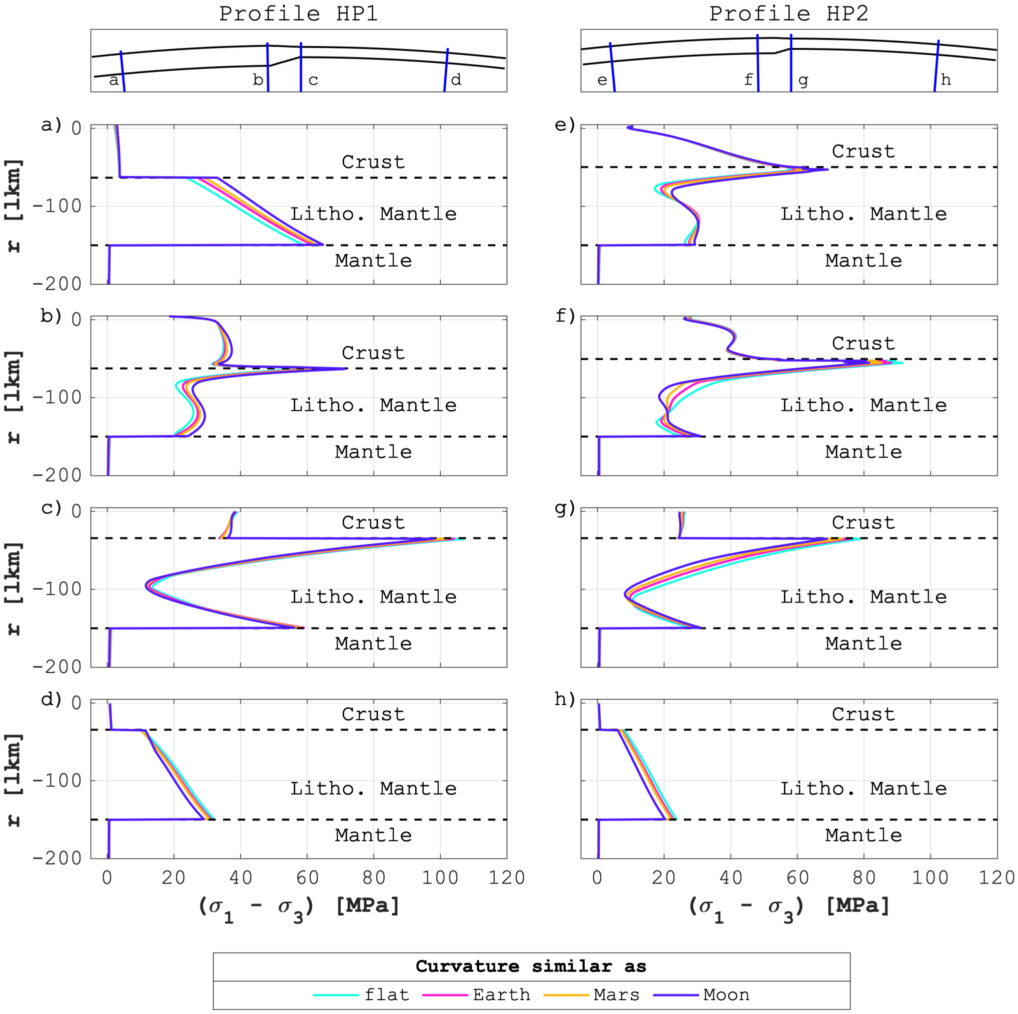

Figure 3.6: Representative vertical profiles of differential stress for model IC_SL with different curvatures. Differential stress, $(\sigma_1 - \sigma_3)$, on the horizontal axis and depth on the vertical axis. In panels (a-d) the profiles are taken at 500 km from the edge of the plateau along the Y-axis (Profile HP1 in Figure 3.7) and their position in direction X are (a) 500 km from the edge of the plateau, (b) 5 km from the edge, (c) 5 km after lowland started, and (d) 500 km after the end of the transition between the plateau and the lowland (positions are indicated in the top panel). In panels (e-h), profiles are taken in the middle of the transition zone between the plateau and its surrounding lowlands (Profile HP2 in Figure 3.7) and are distributed the same way as the left side along direction X (positions displayed in the top panel). The different lines represent different curvatures (see legend) and the dashed line marks the limit between the crust, the lithospheric mantle and the mantle. All of these simulations are done with the model IC_SL. The power-law exponent is 1.

For model IC_SM, there are only minor differences between the differential stress profiles for a model with flat geometry and with Earth's, Mars' and Moon's curvature (Figure 3.6).

3.3.4 Impact of the Corner

For model SC_WL, Figure 3.7a displays the devatoric normal stress at 15 km depth in map view. Horizontal velocities indicate that the plateau flows apart and the velocity direction changes by 90 degrees around the corner (Figure 3.7a). The horizontal deviatoric normal stresses away from the corner and orthogonal to the plateau boundary are highest around the plateau boundary, for example, for $\tau_{\varphi\varphi}$ along profile HP1 (Figure 3.7b). Magnitudes of $\tau_{\varphi\varphi}$ decrease toward the corner (profile HP2 in Figure 3.7a). This is visible in Figure 3.7b which shows about 10 MPa difference in absolute stress magnitudes between the two profiles. Away from the transition zone, the deviatoric normal stresses are essentially constant. For comparison, we show two profiles for two stress magnitudes: a profile of the stress at 15 km depth (solid line in Figure 3.7b and e) and a profile for the corresponding stress that is vertically averaged across the crust (dashed lines). Both stress profiles are similar (Figure 3.7b).

Figure 3.7d displays the horizontal shear stress at 15 km depth in map view. Shear stresses are significant only in the corner region with magnitudes of up to more than 30 MPa. The shear stresses increase nonlinearly toward the corner, indicated by the concave-upward curve of shear stress versus distance (Figure 3.7e). Away from the corner region, shear stresses become negligible (Figure 3.7d and e).

We also present representative vertical profiles of normal deviatoric stress and vertical shear stress (Figure 3.7c and f). Both stresses show a significant vertical variation across the crust. Vertical shear stresses are essentially zero at the top and bottom of the crust

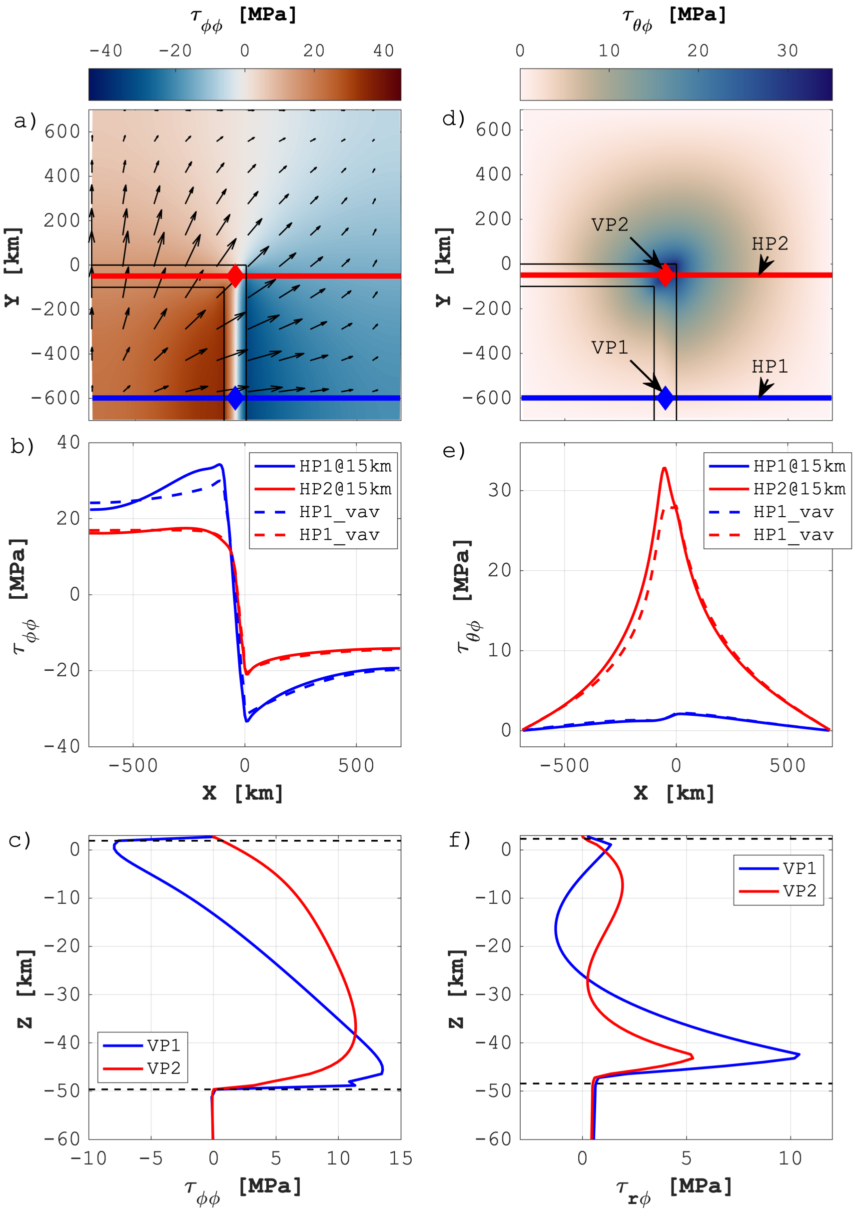

Figure 3.7: Stress distributions for model SC_WL with Earth's curvature and linear viscous flow. (a, d) Map views. Projection on a flat plane of stresses following the curvature at 15 km depth. (b, e) Horizontal profiles of stresses at 15 km depth and vertically averaged across the crust. Blue line (Profile HP1) is a profile 500 km away from the edge of the plateau and red line (Profile HP2) is inside of the transition between the plateau and the lowland. Panels (a, b) display the horizontal deviatoric stress $\tau_{\varphi\varphi}$ and panels (d, e) display the horizontal shear stress $\tau_{\theta\varphi}$. Panels (c, f) display vertical profiles of horizontal deviatoric stress and vertical shear stress, respectively. The location of the profiles are indicated in panel (d) by VP1 and VP2. The black lines on panels (a, d) represent the position of the plateau at the surface and the arrows are the velocity field.

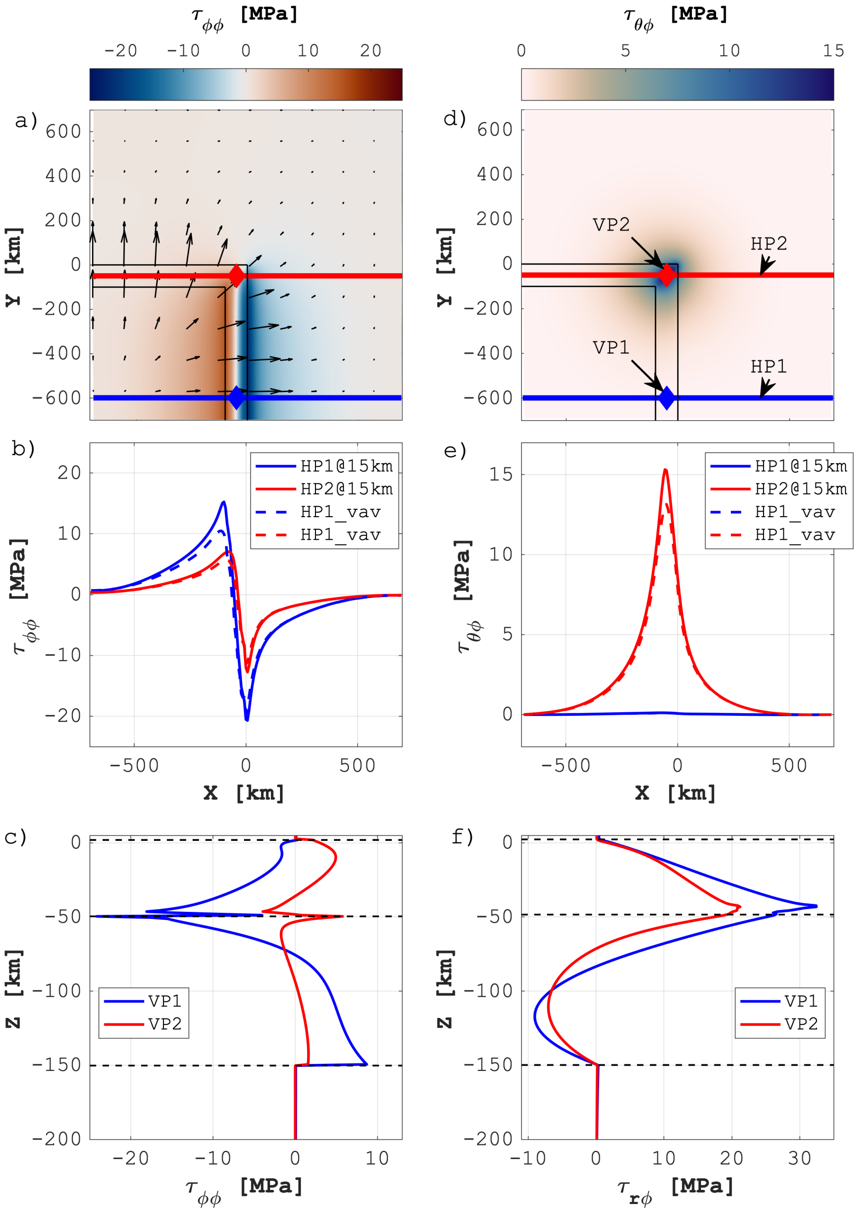

Figure 3.8: Stress distributions for model IC_SL with Earth's curvature and linear viscous flow. (a, d) Map views. Projection on a flat plane of stresses following the curvature at 15 km depth. (b, e) Horizontal profiles of stresses at 15 km depth and vertically averaged across the crust. Blue line (Profile HP1) is a profile 500 km away from the edge of the plateau and red line (Profile HP2) is inside of the transition between the plateau and the lowland. Panels (a, b) display the horizontal deviatoric stress $\tau_{\varphi\varphi}$ and panels (d, e) display the horizontal shear stress $\tau_{\theta\varphi}$. Panels (c, f) display vertical profiles of horizontal deviatoric stress and vertical shear stress, respectively. The location of the profiles are indicated in panel (d) by VP1 and VP2. The black lines on panels (a, d) represent the position of the plateau at the surface and the arrows are the velocity field.

but vary considerably vertically (Figure 3.7f). Maximum values of vertical shear stresses reach ≈10 MPa.

As for model SC_WL (Figure 3.7), we present the same map views and stress profiles for model IC_SL (Figure 3.8). Compared to model SC_WL, in model IC_SL the crustal stress magnitudes are smaller, normal deviatoric stresses reach up to only 20 MPa along profile HP1 and shear stresses reach 30 MPa in the corner region (profile HP2). However, the general crustal stress pattern is the same in both models, with decreasing normal deviatoric stresses and larger shear stresses toward the corner. The vertical profile of the vertical shear stresses shows that shear stresses are largest at the crust-mantle boundary and that maximum values are ≈35 MPa (Figure 3.8f).

Figure 3.9 displays map views of the maximum crustal differential stress and horizontal crustal velocities at each horizontal position. For each horizontal location, we select the largest crustal value along the vertical profile at this location. For model SC_WL (Figure 3.9a and b), differential stresses are highest in the plateau and maximum values are ≈100 MPa (Figure 3.9a). Conversely, for model IC_SL (Figure 3.9c and d), differential stresses are largest along the transition zone (Figure 3.9c). Inside the plateau, differential stresses are smaller than ca. 40 MPa. The comparison of model SC_WL with IC_SL shows that the viscosity ratio between the crust and lithospheric mantle has a major impact on the magnitude and distribution of crustal differential stress.

The distribution of maximum crustal horizontal spreading velocities is also different for models SC_WL and IC_SL (Figure 3.9b and d). For model SC_WL, the velocity magnitudes vary less across the model domain compared to model IC_SL for which maximum magnitudes are localized along the transition zone. The magnitudes of the spreading velocities are lager for model IC_SL compared to SC_WL.

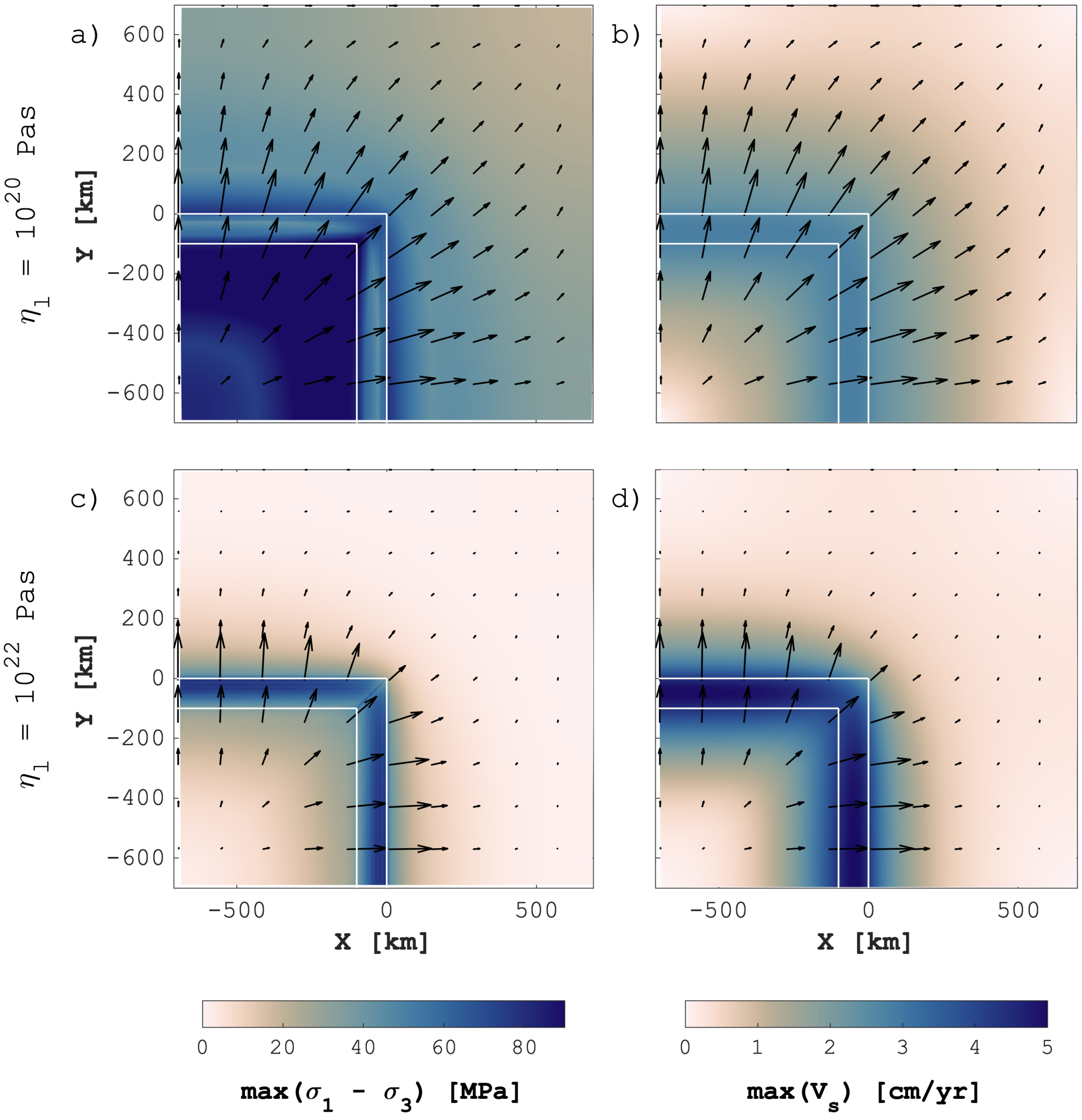

Figure 3.9: Map views showing lateral distribution of differential stress and spreading velocities for models SC_WL and IC_SL. Maximum value in the crust at every horizontal position of differential stress (a, c), and spreading velocity (b, d). Panels (a, b) display the simulation with the model SC_WL, and panels (c, d) show the simulation with the model IC_SL. The white lines represent the position of the plateau at the surface and the arrows are the velocity field. These simulations are done using the Earth's curvature and linear viscous flow.

In map view, the magnitudes of differential stress and horizontal velocities do not vary significantly along the transition zone for both simulations with strong and intermediate crust (Figure 3.9). In other words, only the directions change along the transition zone, but the magnitudes of stress and horizontal velocity change insignificantly. Far away from the corner region, the magnitude contours of normal deviatoric stresses run parallel to the plateau edge, and shear stresses are negligible. This suggests that far away from the corner region there are no significant 3D features in the stress field in that area. However, it is important to take into account the 3D characteristics of the plateau close to the corner region where horizontal shear stresses are significant.

3.3.5 Impact of the Crust and Mantle Strength

We present the results of 18 simulations with various viscosities for the crust and lithospheric mantle to determine what crustal viscosities are required to prevent the continental plateau from gravitational collapse with unrealistically fast velocities. In our viscous model, the plateau will always flow apart as long as there are lateral variations in GPE, but the horizontal velocities will be negligible if the viscosities are sufficiently large. We consider a model unrealistic if the maximum horizontal velocity exceeds 4 cm/yr. The 4 cm/yr corresponds approximately to the current indentation velocity of India (e.g., Liang et al., 2013). Such indentation velocity could theoretically balance a spreading velocity of the same magnitude. However, spreading velocities along the northern limit of the Tibetan plateau are smaller than 4 cm/yr (e.g., Liang et al., 2013), so that 4 cm/yr represents a maximal value of acceptable velocities.

Figure 3.10 displays the differential stress distribution in cross-sections at 500 km from the edge of the plateau (profile location HP1). Contours represent spreading velocities of 4, 10 and 20 cm/yr (black, blue and red contours, respectively). In the case of a weak crust (10^{20}

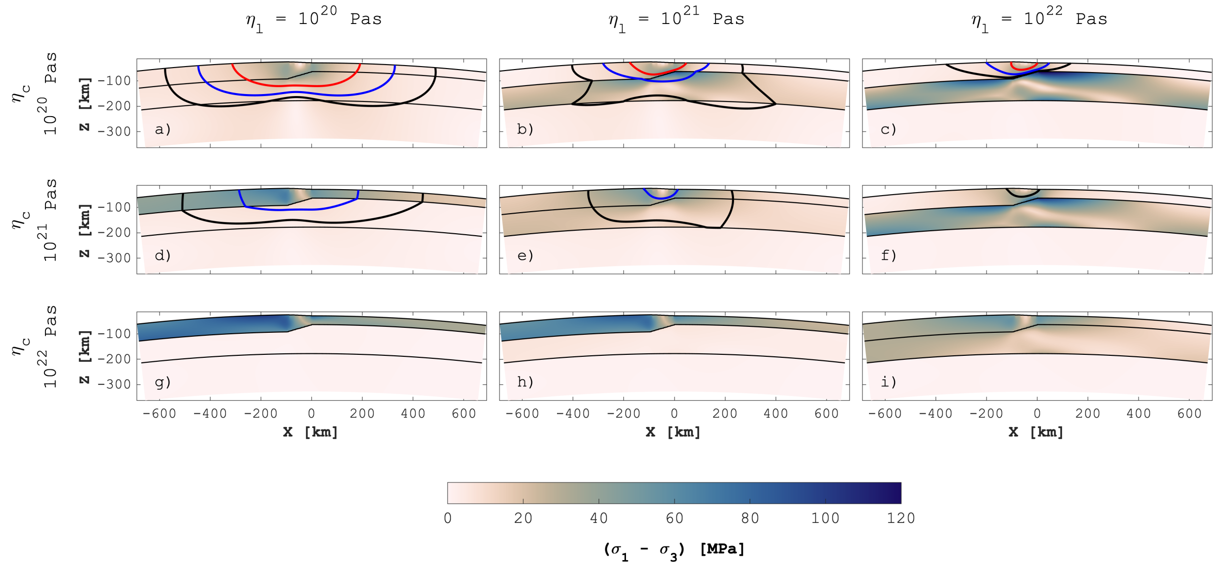

Figure 3.10: Vertical cross sections at profile location HP1 for different models showing differential stress. Differential stress for crustal viscosities of 10^{20} Pa·s (a-c), 10^{21} Pa·s (d-f) and 10^{22} Pa·s (g-i) and lithospheric mantle viscosities of 10^{20} Pa·s (a, d, g), 10^{21} Pa·s (b, e, h) and 10^{22} Pa·s (c, f, i). The contours mark horizontal velocities of black: 4 cm/yr, blue: 10 cm/yr, and red: 20 cm/yr. The black lines display the limits between the continental crust, the lithospheric mantle, and the mantle. These simulations are done using the Earth's curvature and linear viscous flow.

Pa·s) and a weak lithospheric mantle (10^{20} Pa·s), stresses are small and spreading velocities are unrealistically large (up to 70 cm/yr). An increase of viscosity of either the crust or the lithospheric mantle leads to smaller velocities (Figure 3.10). A strong lithospheric mantle associated with a weak crust still allows the plateau to spread with unrealistic velocities > 20 cm/yr (Figure 3.10c). On the other hand, a strong crust (10^{22} Pa·s) overlying a weak lithospheric mantle (10^{20} Pa·s) exhibits a maximum spreading velocity < 4 cm/yr (Figure 3.10g). Moreover, stresses are focusing in the strong layer, which leads to different stress distribution and magnitudes in the crust above a strong lithospheric mantle. Stresses are focused in the transition zone between plateau and lowland, where most of the deformation occurs.

Figure 3.11 displays the maximum spreading velocity in the entire crust as a function of the crustal viscosity. It shows more clearly the tendency observed in Figure 3.10, that the larger the crustal viscosity, the slower the spreading of the plateau. For the same crustal viscosities, larger viscosities of the lithospheric mantle cause smaller spreading velocities. Five of our 18 simulations provided velocities smaller than 4 cm/yr, all for a crustal viscosity of at least 3×10^{21} Pa·s.

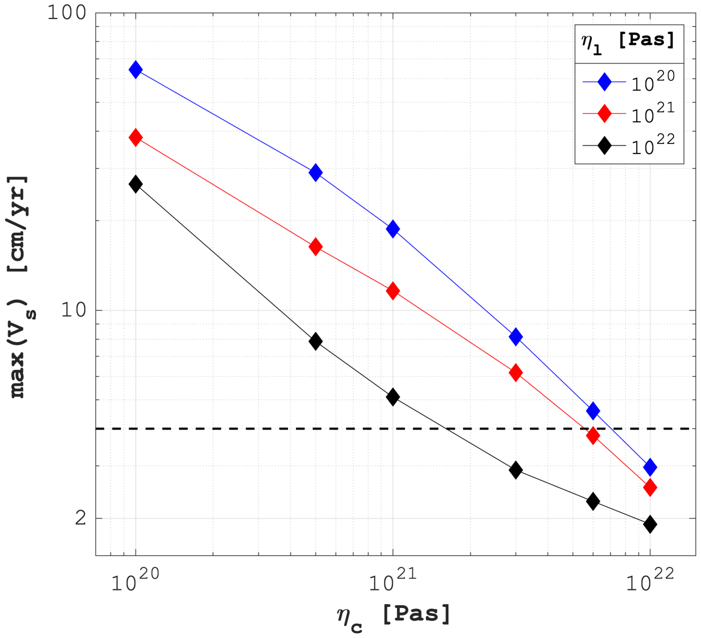

Figure 3.11: Maximum values of the spreading velocity in the entire crust for models with varying viscosities of the crust and lithospheric mantle. The horizontal axis indicates the crustal viscosity and the color the viscosity of the lithospheric mantle. All results are for models with Earth's curvature and linear viscous flow.

In addition, Figure 3.12 presents the relation between the maximum differential stress in the crust and the maximum spreading velocity (Figure 3.12a) and the maximum strain rate invariant ($\dot{\epsilon}_{\mathrm{II}} = \sqrt{1/2 * (\dot{\epsilon}^2_{\mathrm{rr}} + \dot{\epsilon}^2_{\theta\theta} + \dot{\epsilon}^2_{\varphi\varphi}) + \dot{\epsilon}^2_{\mathrm{r}\theta} + \dot{\epsilon}^2_{\mathrm{r}\varphi} + \dot{\epsilon}^2_{\theta\varphi}}$; Figure 3.12b) in the crust. There is no clear correlation between these quantities. Similar stress magnitudes can lead to considerably different velocities or strain rates. For a strong lithospheric mantle, the crustal stress magnitudes do not systematically decrease with decreasing crustal viscosities

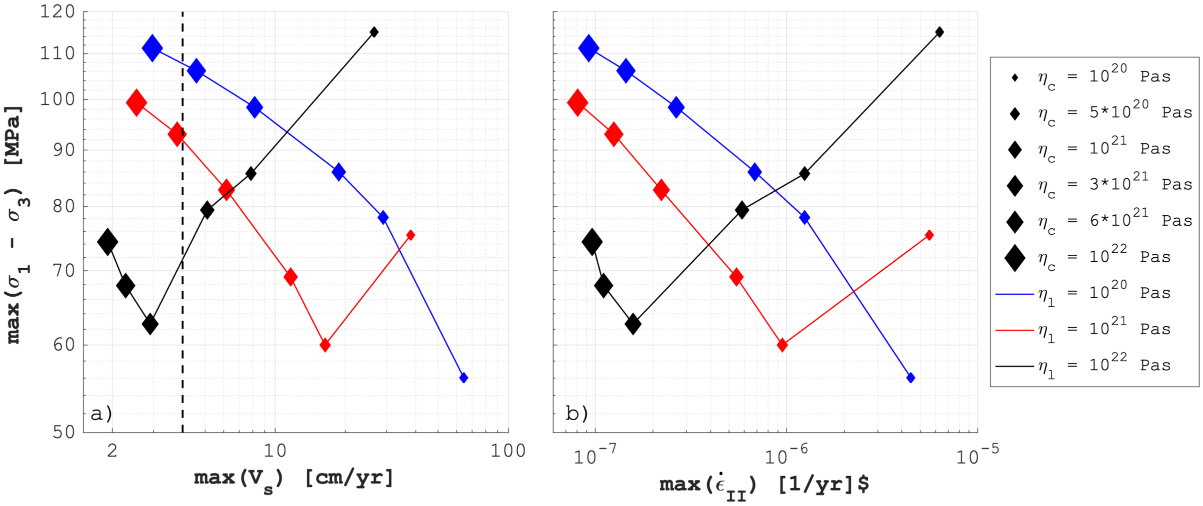

Figure 3.12: Relation between the maximum crustal differential stress and (a) the maximum spreading velocity, and (b) the maximum strain rate invariant for models with varying viscosities of the crust and lithospheric mantle. Different crustal (size of markers) and lithospheric mantle (color of markers) viscosities are indicated in the legend in panel (b). All simulations are performed with Earth's curvature and linear viscous flow.

(Figure 3.12a). The reason is that a weaker crust flows faster and hence generates higher strain rates (Figure 3.12). These higher strain rates cause higher stresses and can balance the decrease in viscosity to maintain similar stress magnitudes. Magnitudes of stress are hence not a reliable proxy for lithospheric strength which is governed by the effective viscosity.

3.3.6 Impact of a Stress-Dependent Power-Law Flow Law

In our combined linear and power-law viscous flow law, we need to specify the characteristic stress that controls the transition from linear viscous (e.g., diffusion creep) to power-law viscous flow (e.g., dislocation creep). We define τ_C at 24 MPa. As a consequence, effective viscosity decreases when stresses are larger than the characteristic stress and increases when stresses are smaller. If we refer in this section, with results for power-law viscous flow, to a viscosity value applied in a model, then we refer to the linear viscosity, η, used in the Equation 3.11 for the effective viscosity, η_{E}.

Figure 3.13 displays map views of differential stress and effective viscosity. For both quantities, we plot the maximum value of the crust at each horizontal position. Results for a strong crust (10^{22} Pa·s) and a weak lithospheric mantle (10^{20} Pa·s), model SC_WL, show that stress magnitudes decrease and the stress distribution gets smoother as the power-law exponent increases (Figure 3.13a to f). Also, the effective viscosity varies from a constant value (linear viscous) to more and more variable values (Figure 3.13b, d and f). Differences in effective viscosity reach up to one order of magnitude for $n = 6$ (Figure 3.13f). In the plateau, where stresses are high, the effective viscosity for $n = 3$ and 6 is smaller compared to the linear viscous one. However, far away from the plateau the stresses are smaller than the characteristic stress what leads to an increase in effective viscosity.

Results for a strong crust (10^{22} Pa·s) and a strong lithospheric mantle (10^{22} Pa·s) show small differences in the differential stress distribution for increasing values of n (Figure 3.13g, i and k). The reason is that crustal deviatoric stresses do not significantly exceed the characteristic stress. Hence, the effective viscosity increases nearly everywhere, except in the transition zone between plateau and lowlands (Figure 3.13h, j and l).

Figure 3.14 presents the maximal differential stress and spreading velocity of the entire crust as a function of the power-law exponent. For all lithospheric mantle viscosities, the maximum differential stress decreases as the power-law exponent increases (Figure 3.14a). For example, for the model with a lithospheric mantle viscosity of $\eta_{\mathrm{l}} = 10^{21}$ Pa·s, the maximum differential stress decreases from ≈100 MPa to ≈67 MPa when n increases from 1 to 6 (red curve in Figure 3.14a). For lithospheric mantle viscosities between $\eta_{\mathrm{l}} = 10^{21}$ Pa·s and $\eta_{\mathrm{l}} = 10^{22}$ Pa·s, the maximum spreading velocities vary little when n increaseas (Figure 3.14b). The spreading velocity increases with increasing n for a lithospheric mantle

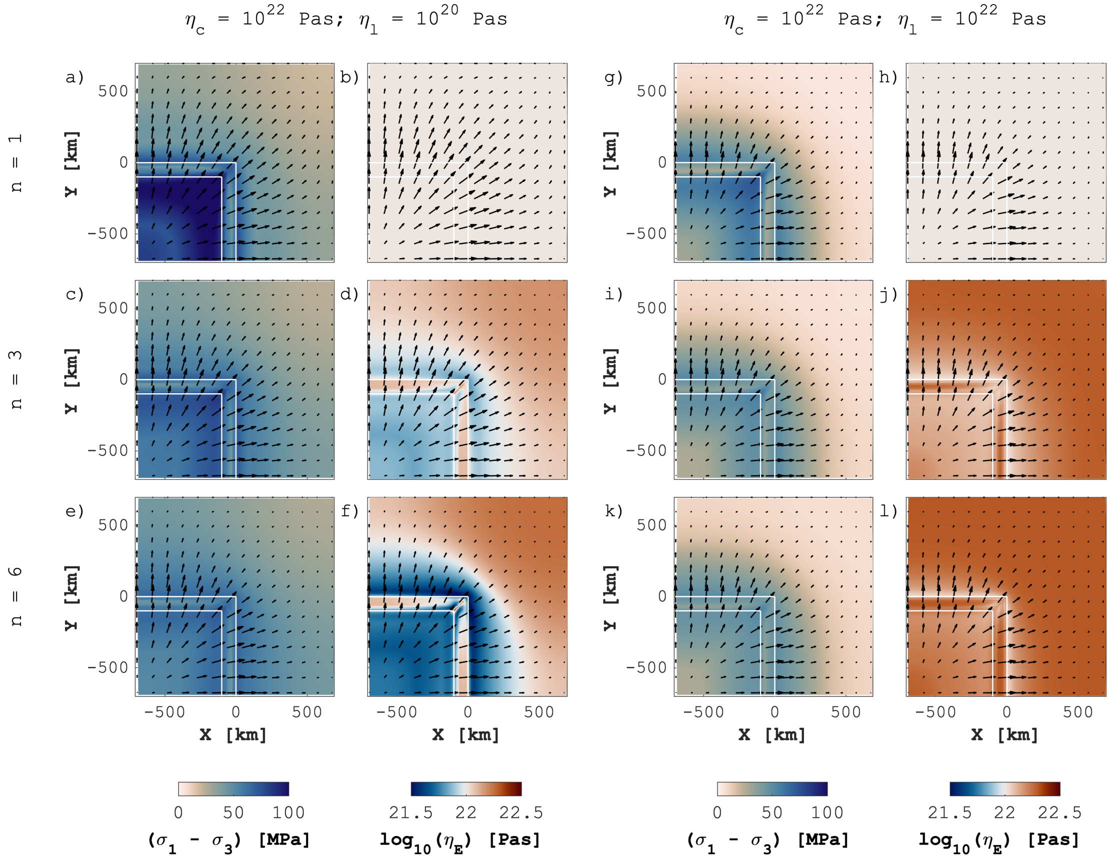

Figure 3.13: Map views of crustal maximum values of differential stress and effective viscosity for two simulations with power-law viscous flow with different stress exponents n. Panels (a-f) display results for a simulation with a strong crust (10^{22} Pa·s) and a weak lithospheric mantle (10^{20} Pa·s). Panels (g-l) display results for a simulation with a strong crust (10^{22} Pa·s) and a strong lithospheric mantle (10^{22} Pa·s). Panels (a, c, e), and panels (g, i, k) display the differential stress. Panels (b, d, f), and panels (h, j, l) display the effective viscosity. The corresponding stress exponent n is indicated on the left of each of the three rows of panels. All models are with the Earth's curvature.

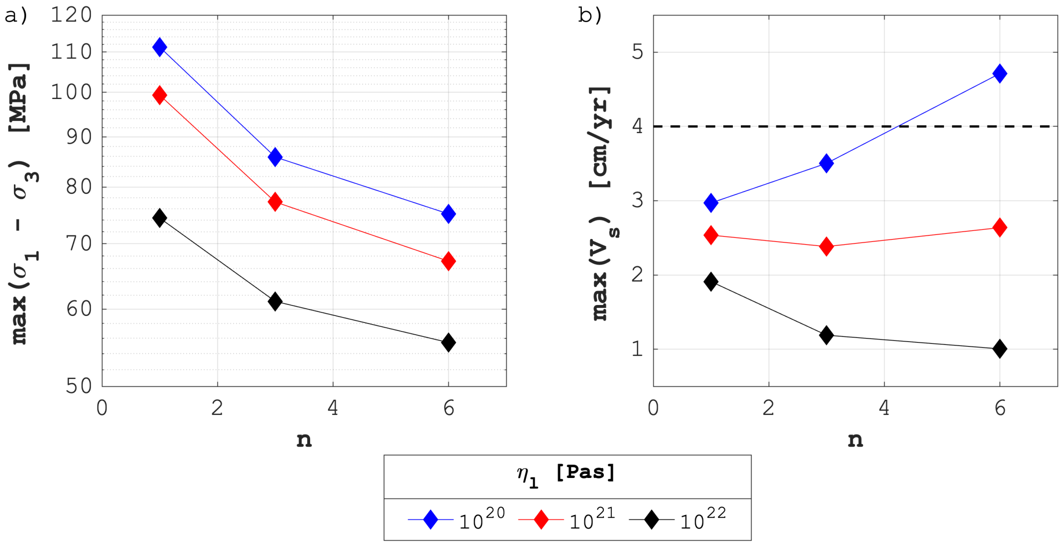

Figure 3.14: Variation of the maximum (a) differential stress and (b) spreading velocity of the entire crust with increasing power-law stress exponent n. All simulations are with the Earth's curvature, a crustal viscosity value of $\eta = 10^{22}$ Pa·s and with $\tau_C = 24$ MPa.

viscosity of $\eta_l = 10^{20}$ Pa·s because in this model the crustal stresses are largest and the stress weakening effect due to power-law flow is largest.

The presented results show the general impacts of a power-law viscous flow on the magnitudes of stress and spreading velocities. Specific results depend on the applied value of τ_C but a systematic analysis of the impact of different values of τ_C is beyond the scope of our study.

3.3.7 Estimates of Stress and Spreading Velocity

For all the numerical calculations with linear viscous flow, we compare the maximum numerical value of τ_{II} in the crust with the analytical estimate τ_e since both stress magnitudes represent deviatoric stresses (Figure 3.15a). The value of $\tau_e \approx$34 MPa for the applied model configuration and predicts the numerical values of τ_{II} accurately within a factor of $\approx$2. The comparison shows that τ_e tends to provide a lower bound for the numerical stress magnitudes (Figure 3.15a). The numerical results also show that stress magnitudes depend only slightly on the crustal viscosity. While the viscosity of both the crust and lithospheric mantle varies by two orders of magnitude, the stress magnitudes only vary within a factor of less than ≈ 2 which supports the analytical estimate of τ_e that is estimated independently on the viscosity.

The numerical results show that there is an approximately linear correlation between the maximum horizontal spreading velocity, V_s, and the maximum strain rate invariant, $\dot{\epsilon}_\mathrm{II}$ (Figure 3.15b). We have assumed such linear relation in Equation 3.6 to derive an analytical estimate for the spreading velocity. A length scale, L, that captures the observed relation between the numerical spreading velocities and strain rates is ≈ 200 km. The 200 km corresponds to twice the width of the transition zone. For the analytical estimate of the spreading velocity, V_e, we hence use $L = 200$ km (see Equation 3.7).

The comparison between the maximum value of the numerical spreading velocity, V_s, and the corresponding analytical estimate, V_e, shows that V_e can accurately predict the spreading velocities within a factor of ≈ 3 (Figure 3.15c). Such accuracy of V_e is remarkable because in the numerical calculations both η_c and η_l vary between 10^{20} and 10^{22} Pa·s, and the numerical calculations consider all end-member combinations of η_c and η_l (Figure 3.10).

We made calculations with a stress-dependent viscosity only for a crustal linear viscosity of 10^{22} Pa·s. The applied value of τ_C is 24 MPa. The estimated value of $\tau_\mathrm{e}/\tau_\mathrm{C} \approx 1.4$ (Figure 3.15d). For simulations with $n = 6$, estimated values of $\tau_\mathrm{e}/\tau_\mathrm{C}$ are closest to the numerical values of $\max(\tau_\mathrm{II})/\tau_\mathrm{C}$ while for simulations with $n = 1$ the value of $\tau_\mathrm{e}/\tau_\mathrm{C}$ underestimates the numerical values (Figure 3.15d). Simulations with $n = 6$ show the smallest variation in $\max(\tau_\mathrm{II})/\tau_\mathrm{C}$ (from ≈ 1.2 to ≈ 1.6) but the largest variation in $\max(V_\mathrm{s})/V_\mathrm{e}$ (from ≈ 0.9 to ≈ 4.2) which is consistent with the analytical estimate (Figure 3.15d). Conversely, simulations with $n = 1$ show the largest variation in $\max(\tau_\mathrm{II})/\tau_\mathrm{C}$ (from ≈ 1.6 to ≈ 2.4)

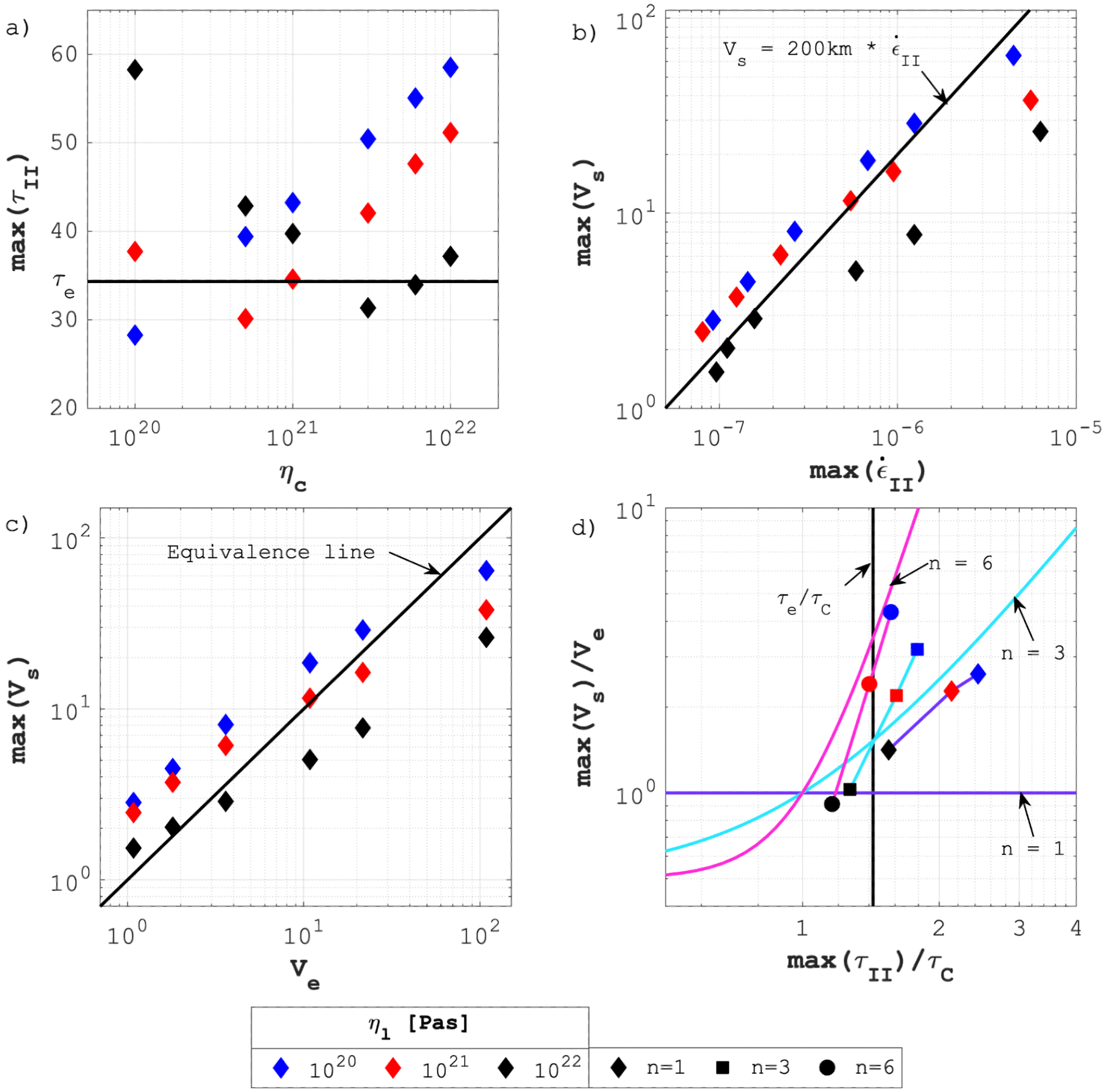

Figure 3.15: Comparison of analytical estimates with numerical results. The legend at the bottom of the figure indicates the values used in the numerical calculation. (a) Maximum value of τ_{II} in the crust from numerical calculation versus crustal viscosity, η_{c}. The horizontal black line indicates the analytical estimate for the deviatoric stress, τ_{e}. (b) Maximum value of crustal horizontal spreading velocity, V_{s}, versus the maximum value of $\dot{\epsilon}_{\mathrm{II}}$. The black line indicates the applied analytical estimate for the relation between spreading velocity, V_{e}, and strain rate, $\dot{\epsilon}$. (c) Comparison of numerical spreading velocity, V_{s}, with analytical estimate, V_{e}. The black line indicates the equivalence between numerical and analytical velocities. (d) Dimensionless spreading velocity versus the stress ratio that controls the stress-dependent effective viscosity. The solid lines indicate the analytical velocity estimate for different values of n (see Equation 3.10).

but the smallest variation in $\max(V_\mathrm{s})/V_\mathrm{e}$ (from ≈1.3 to ≈2.6). Simulations with $n = 1$ should theoretically not vary with varying $\max(\tau_\mathrm{II})/\tau_\mathrm{C}$ and the velocity variations are due to differences between $\max(\tau_\mathrm{II})$ and τ_e causing the inaccuracy in the velocity prediction.

3.4 Discussion

3.4.1 Viscous Deformation of the Lithosphere

For simplicity, we consider only a linear and a power-law viscous deformation of the lithosphere and neglect elastic and frictional-plastic deformation. We use viscosities between 10^{20} and 10^{22} Pa·s. Typical values of the elastic shear modulus for lithospheric rocks are $\approx 3 \times 10^{10}$ Pa (e.g., Turcotte and Schubert, 2021). The characteristic Maxwell viscoelastic stress relaxation time is given by the ratio of viscosity to shear modulus (e.g., Turcotte and Schubert, 2021). The duration of the relaxation of elastic stresses in a viscoelastic material is approximated by the Maxwell time. For the applied viscosities between 10^{20} and 10^{22} Pa·s, and a typical shear modulus of $\approx 3 \times 10^{10}$ Pa, the Maxwell time is between 100 and 10'000 years. Since we are interested in the long-term stability of continental plateaus, say >1 Myr, it is justifiable to neglect elastic effects in our application because elastic stresses are relaxed after ≈10′000 years for our model configuration.

The maximum differential stresses in our model are ≈110 MPa (Figure 3.12), and maximum differential stresses close to the surface are <80 MPa (Figure 3.5). Frictional-plastic yield stresses in the continental crust are >80 MPa already below a few kilometers of depth (e.g., Kohlstedt et al., 1995; Townend and Zoback, 2000). Therefore, in our model, only the uppermost few kilometers of the continental crust would be affected by frictional-plastic deformation, because most of the continental crust deeper than a few kilometers exhibits differential stresses below typical yield stresses. Furthermore, differential stresses in the lithospheric mantle in all the performed models are also <100 MPa, and stresses are, hence, below the yield stress in the lithospheric mantle (Figure 3.10). The application of an effectively viscous deformation behavior of the lithosphere is, hence, justifiable for our model and our modelling objectives.

3.4.2 Curvature

Most 3D models investigating the deformation of the lithosphere around continental plateaus are considering a flat rectangular plate. However, the Earth's lithosphere has a double curvature and is mechanically a shell. A shell is a plate that has a curvature in its undeformed state (e.g., Bower, 2009). One could expect a lithospheric plate and shell to generate different stress fields around continental plateaus due to, for example, geometrical stiffening effects (e.g., Pini et al., 2016). Another effect of the curvature is the opposite reaction of a plate and a shell to lateral, that is orthogonal to the plate's surface, loading (e.g., von Karman et al., 1940). A plate deflects in response to a lateral load, hence its length, or arc length, increases, and the plate's middle line is under extension. A shell with a convex upward curvature is shortened and compressed if the lateral loading is in the direction opposite to the upward convexity.

Our results show that the impact of the Earth's curvature on the stress field around continental plateaus is minor. The absolute magnitudes of stresses are slightly smaller in models considering the Earth's curvature compared to flat models (Figure 3.5). If the curvature becomes larger, then differences between stresses for a rectangular and spherical geometry increase. However, the differences are not large (Figure 3.5). We suggest that the reason for this minor impact of curvature is likely the isostatic state of the plateau. Despite the high topography, the plateau does not generate a vertical load and vertical displacement on the lithospheric plate. Conversely, the lateral GPE variations cause mainly stresses that are parallel to the curved lithosphere. Therefore, the stress distribution around plateaus is similar for flat and curved geometries.

3.4.3 Crustal and Lithospheric Mantle Viscosities

We use in our model a simple plateau geometry and simple distributions of densities and effective viscosities. The main reasons for the simplified model configuration are to keep the results transparent and to focus on first-order processes. Nevertheless, our model is considerably more elaborated than thin viscous sheet models, because, for example, it captures the full 3D stress field and considers different viscosities of the crust and lithospheric mantle. Our results show that the ratio of crustal to lithospheric mantle viscosity has a significant impact on the magnitude of crustal stresses. For the estimation of crustal stresses in regions with significant lateral variation of GPE, it is therefore essential to apply models that can consider different viscosities in the crust and lithospheric mantle. Thin viscous sheet models and models based on lateral variations of GPE are not suitable to estimate crustal stress magnitudes in such regions because they consider effective viscosities that are vertically averaged across the entire lithosphere.

Our results show that a strong lithospheric mantle alone is not sufficient to support the Tibetan plateau. The crust must also have significant strength to prohibit an unrealistically fast gravitational collapse of the plateau. For the strongest lithospheric mantle in our model we apply a viscosity of 10^{22} Pa·s. Below the crust-mantle boundary (Moho), effective viscosities of the lithospheric mantle may be slightly higher than 10^{22} Pa·s, especially below the lowland due to colder Moho temperatures compared to the thicker plateau (e.g., Hirth and Kohlstedt, 2003; England and Molnar, 2015). However, since we apply a constant viscosity in the lithospheric mantle, this viscosity represents a vertically averaged viscosity. We apply the viscosity of 10^{22} Pa·s for the lithospheric mantle down to a depth of 150 km. In such depth, and for the correspondingly hotter temperatures, the viscosity of the lithospheric mantle is smaller than 10^{22} Pa·s (e.g., Hirth and Kohlstedt, 2003; England and Molnar, 2015). Therefore, the modelled lithospheric mantle with a constant viscosity of 10^{22} Pa·s represents a strong lithospheric mantle compared to more realistic vertical viscosity variations based on experimentally derived flow laws.

Here, we consider a crust that causes spreading velocities larger than 4 cm/yr as mechanically too weak to support the Tibetan plateau. We chose 4 cm/yr because this value is like the present-day indentation velocity of India (e.g., Liang et al., 2013). Our results show that the vertically averaged viscosity of the crust must be at least 3×10^{21} Pa·s or larger so that spreading velocities in the crust are everywhere smaller than 4 cm/yr. This average viscosity value is quite large considering the larger-than-normal thickness of the crust forming continental plateaus ($\approx$70 km in the Tibetan plateau) and its associated large vertical temperature variation. For example, the upper limit for lower crustal viscosities, resulting from geodetic estimates (e.g., Thatcher and Pollitz, 2008), is $\approx 3 \times 10^{21}$ Pa·s, and many studies suggest lower crustal viscosities to range between 10^{18} Pa·s and 3×10^{20} Pa·s (e.g., England et al., 2013; Doin et al., 2015; Shi et al., 2015). If such low-viscosity estimates for the continental lower crust are accurate, then our results imply that the average, effective viscosity of the upper crust must be considerably larger than 3×10^{21} Pa·s in order to support the topographic variation between the Tibetan plateau and surrounding lowland for geological time scales $> \approx 1$ Myr.

3.4.4 Shear Stresses

The crustal shear stresses vary significantly in both the horizontal and vertical directions (Figures 3.7 and 3.8). In the corner region of the plateau, the magnitudes of the horizontal shear stresses are even larger than the absolute magnitudes of the horizontal deviatoric normal stresses (Figures 3.7 and 3.8). Conversely, away from the corner region, the horizontal shear stresses are essentially negligible (Figures 3.7 and 3.8).

The vertical shear stresses vary significantly in the vertical direction at the transition zone. For a strong crust and weak lithospheric mantle (SC_WL), the shear stresses have a maximum absolute value of ≈10 MPa (Figures 3.7c and f). For an intermediate strong crust and strong lithospheric mantle (IC_SL), the shear stresses reach a maximum absolute value of ≈30 MPa at the base of the crust (Figure 3.8f). The existence of such shear stresses has important implications for thin viscous sheet models because in these models the vertical shear stresses are considered to be zero (e.g., England and McKenzie, 1982; Schmalholz et al., 2014). Such thin viscous sheet approximations, that neglect vertical shear stresses, are often applied to quantify the global lithospheric stress field (e.g., Coblentz et al., 1994; Lithgow-Bertelloni and Guynn, 2004; Ghosh et al., 2009) or the stresses and forces associated with large-scale tectonic plate motion (e.g., Ghosh et al., 2006; Warners-Ruckstuhl et al., 2012, 2013). These models provide first-order accurate magnitudes of large-scale lithospheric stresses. Our results indicate that care should be taken when interpreting such stress fields in regions with significant lateral GPE variations because in these regions vertical shear stresses can be significant and can have a first-order impact on stress magnitudes. The vertical variation of the vertical shear stress is associated with a vertical variation of the horizontal deviatoric normal stresses (Figures 3.7c and f, and 3.8c and f). The horizontal deviatoric normal stress changes its sign with depth (Figures 3.7c and 3.8c) which is typically resulting from bending of the lithosphere around the transition zone Schmalholz et al. (2019).

2D lithospheric models configured for vertical cross sections provide accurate stress fields if the cross section has a distance from a corner region of at least a few hundred kilometers. If the modelled cross section is closer, then horizontal shear stresses are significant (Figures

3.7 and 3.8). The modelled 2D stress field can be, hence, considerably inaccurate because it does not take into account these horizontal shear stresses.

3.4.5 Stress Versus Strength Relationship

The viscosity controls the strength of the different model units, such as the crust. The maximum magnitudes of the differential stress in the crust do not vary significantly for a strong lithospheric mantle and different crustal viscosities (Figure 3.12a). Conversely, the maximum magnitudes of the spreading velocities and strain rates in the crust vary between one and two orders of magnitude, respectively. The minor variation of maximum crustal differential stresses, despite two orders of magnitude variation in crustal viscosities, shows that crustal stress magnitudes are not a good proxy for crustal strength (e.g., Schmalholz et al., 2009). Smaller crustal viscosities result in faster spreading velocities and higher strain rates. Therefore, stress magnitudes, which are proportional to the product of viscosity and strain rate, remain more or less constant for decreasing crustal viscosities.

3.4.6 Simplifications

We make several simplifications in our model to study first-order features of the 3D stress field around continental plateaus and to keep the results transparent. We consider a simple and idealized geometry of a continental plateau and only the instantaneous stress field. We do not apply a far-field velocity field in order to isolate the impact of topography and lateral crustal thickness variation on the stress field. For the Tibetan plateau, the indentation of the Indian plate affects the deformation field around the Himalayan range. Therefore, we consider a spreading velocity of >4 cm/yr as unrealistic.

Another limitation of our model is the simplified density and viscosity structure. 2D numerical models that consider a more realistic yield strength envelope exhibit significant vertical viscosity variations due to the temperature dependence of viscosity. These viscosity variations cause stronger vertical stress variations compared to our model because high stresses are focused in the high-viscosity levels of the crust and lithospheric mantle (e.g., Schmalholz et al., 2019). Also, several studies have proposed the existence of a low-viscosity middle or lower crustal layer in which significant crustal flow could take place (e.g., Clark and Royden, 2000; Beaumont et al., 2004). The existence of such a widespread and continuous low-viscosity crustal layer remains disputed (e.g., Nábělek et al., 2009). If such a low-viscosity layer existed, then the magnitudes of crustal differential stresses would likely be larger as predicted by our model because the vertically integrated stresses supporting the plateau would be focused on a thinner upper crustal, high-viscosity layer. Therefore, more realistic vertical viscosity variations likely result in larger maximum differential stresses compared to our model with constant viscosity in the model units.

Furthermore, the temperature dependence of viscosity will result in lateral variations of crustal and lithospheric mantle viscosites along the Moho between plateau and lowland because of different temperatures at different depths (e.g., Schmalholz et al., 2019). However, these lateral viscosity variations might be not dramatic in the lithospheric mantle because the potentially active low-temperature plasticity is significantly less sensitive to temperature than dislocation creep (e.g., England and Molnar, 2015).

3.4.7 Dimensionless Stress Ratio and Analytical Estimate for Spreading Velocity

To derive the analytical estimate V_e, we assume that the estimated deviatoric stress in the crust, τ_e, causes gravitational spreading. In other words, we assume that the gravitational stress is equal to the viscous flow stress and, hence, we assume

$$\frac{\rho_\mathrm{c} g h_\mathrm{a} L}{8\eta_\mathrm{c} V_\mathrm{e}} \approx 1. \tag{3.14}$$

The above dimensionless stress ratio is similar to the Ramberg (Ramberg, 1981; Weijermars and Schmeling, 1986) and Argand numbers (England and McKenzie, 1982). The Ramberg and Argand numbers scale the gravitational stress to a viscous stress. In the Ramberg number, the strain rate is usually expressed by the ratio of a velocity to a length scale (e.g., Medvedev, 2002). Conversely, in the Argand number the strain rate is typically assumed to be the large-scale lithospheric shortening rate during crustal thicknening (England and McKenzie, 1982) or crustal and lithospheric folding (Schmalholz et al., 2002).

To quantify the Ramberg number, one commonly assumes $L = h_\mathrm{a}$ (Ramberg, 1981; Weijermars and Schmeling, 1986). Medvedev (2002) applied the Ramberg number to a two-sided wedge of total width λ with a basal shear velocity V_bs. They estimated the strain rate with the ratio V_bs/λ. In our model, a crosssection orthogonal across the transition zone with a linear change of topography corresponds to a geometry similar to one-half of the double-sided wedge considered by Medvedev (2002). In our analytical velocity estimate, we used a length scale of $L = 200$ km which is equivalent to twice the width of the transition zone. This length scale is similar to the one used in the analytical model for two-sided wedges by Medvedev (2002) to estimate strain rates. Hence, we expect that our velocity estimate is also applicable to plateau geometries with transition widths different from the ones of our model if in the analytical estimate twice the width of the transition zone is used as length scale L.

3.4.8 Applications to the Tibetan Plateau

We can use our model result that indicates that the Ramberg number is ≈1 (Equation 3.14) to estimate the crustal viscosity in particular regions of the Tibetan plateau. For example, in the eastern Tibetan plateau the Longmen Shan orogenic belt represents the boundary between the Tibetan plateau and the Sichuan Basin (e.g., Sun et al., 2019). In the southern segment of the Longmen Shan, the altitude decreases ≈4 km over a distance of ≈50 km and in the northern segment the altitude decreases ≈3.5 km over a distance of ≈200 km (e.g., Sun et al., 2019). The crustal thickness around the Longmen Shan with ≈4 km topography is ≈60 km indicating a situation close to isostatic equilibrium (Zhang et al., 2009). The GPS velocities indicate a spreading velocity in the direction from the high to the low topographic region of ≈1 cm/yr (Gan et al., 2007; Zheng et al., 2017; Penney and Copley, 2021). We solve Equation 3.14 for the crustal viscosity which yields

$$\eta_{\mathrm{c}} \approx \frac{\rho_{\mathrm{c}} g h_{\mathrm{a}} L}{8 V_{\mathrm{e}}}. \tag{3.15}$$

Applying $V_{\mathrm{e}} = 1$ cm/yr, $\rho_{\mathrm{c}} = 2800$ kg·m^{-3}, $h_{\mathrm{a}} = 4$ km and $L = 100$ km provides an estimate for the average crustal viscosity of $\approx 5 \times 10^{21}$ Pa·s. Changing h_{a} to 3.5 km and L to 400 km provides $\eta_{\mathrm{c}} \approx 1.5 \times 10^{22}$ Pa·s. Given the uncertainties of the measured quantities such as velocities and topography as well as of the simple mathematical model, our model predicts effective average crustal viscosities of $\approx 10^{22}$ Pa·s. This viscosity value estimated with our simple Equation 3.15 agrees with viscosity estimates of other studies that applied more elaborated thin viscous sheet models (e.g., England and Molnar, 1997; Flesch et al., 2001).

The India-Asia collision and the formation of the Tibetan plateau have generated a variety of large-scale strike-slip faults, particularly around the two syntaxes regions (e.g., Molnar and Tapponnier, 1975; Harrison et al., 1992; Royden et al., 1997; Tapponnier et al., 2001). The syntaxes regions are characterized by significant horizontal changes in the direction of the transition zone between high- and low-altitude regions. In our model, the corner represents such regions in which the transition zone between high- and low-altitude regions significantly changes its horizontal direction. The modelled, strongly increased horizontal shear stresses around the plateau's corner region (Figures 3.7 and 3.8) suggest that ongoing strike-slip faulting around the Tibetan syntaxes regions might be supported, in addition to indentation and convergence, by horizontal shear stresses resulting from the significant lateral changes in topography and crustal thickness.

A stronger continental crust exhibits larger and more homogeneously distributed differential stresses compared to a weaker crust above a strong lithospheric mantle (Figure 3.9a and c). For a weak, low-viscosity crust, large differential stresses occur only along the transition zone and differential stresses significantly decrease toward the plateau center (Figure 3.9c). Earthquakes are relatively equally distributed across the Tibetan plateau (e.g., Bai et al., 2017; Li and Hou, 2019; Hetényi et al., 2023). There is no systematic trend that shows, for example, a decreasing density of earthquakes toward the center of the Tibetan plateau. Consequently, the crust of the Tibetan plateau should be at least strong enough to prohibit a considerable decrease of differential stress toward the plateau center.

Some of the most active orogenic processes on Earth occur within the syntaxes of the Himalayan chain, at the southern termination of the Tibetan plateau (e.g., Zeitler et al., 2001). These syntaxial regions are characterized by strong lateral variations in topography and crustal thickness. The applied power-law viscous flow law can locally decrease crustal viscosities in regions with stresses that are larger than those in the surroundings, due to its stress-weakening effect. Our results show that such relative stress weakening can occur in the transition zone around the corner region (Figure 3.13f). The decreased viscosity in the transition at the corner of the plateau could locally weaken the crust, focus the deformation and contribute to locally increased exhumation. Several mechanisms have been proposed to explain the localized exhumation and deformation in the syntaxes, such as crustal-scale buckle folding (Burg et al., 1997), diverted crustal flow resulting from localized river incision and erosion (Zeitler et al., 2001) or crustal deformation over a geometrically stiffened subducting plate (Bendick and Ehlers, 2014). Whatever the mechanism, a local stress-weakening of the crust in the syntaxes region due to stresses resulting from strong lateral topography and crustal thickness variations could support the localization of deformation and ongoing rapid exhumation.

3.5 Conclusions

We investigated with 3D numerical calculations the interplay of various factors governing the crustal stress distribution and magnitudes in and around continental plateaus. In our model, the stresses are caused by lateral variations in gravitational potential energy only and we do not consider any tectonic far-field deformation such as plate convergence.

For our model, we have demonstrated that Earth's curvature exerts a minor influence on stress magnitudes and distribution within the continental crust. Consequently, for spatial scales smaller than a few thousand kilometers, it is justifiable to omit the Earth's curvature in numerical models of stress distribution around continental plateaus.

Our study has emphasized the critical role played by corner regions in shaping the crustal stress field. Notably, horizontal deviatoric normal stresses consistently decrease toward the plateau's corner regions, while horizontal shear stresses significantly intensify. These corner regions exhibit elevated horizontal shear stresses, reaching approximately 30 MPa, underscoring their importance in understanding crustal stress patterns.

Varying the viscosity of both the continental crust and lithospheric mantle has unveiled several key insights. First, the stress distribution is primarily controlled by the viscosity ratio between the crust and lithospheric mantle. The highest and lowest differential stresses in the crust occur in configurations featuring a weak lithospheric mantle and a strong crust, and vice versa. Second, reducing the crustal viscosity does not automatically reduce stress magnitudes; instead, it can lead to faster crustal flow, higher strain rates, and subsequently, higher stresses. Hence, crustal stress magnitudes are not a reliable proxy for the effective crustal viscosity, and hence crustal strength. Lastly, the incorporation of a stress-weakening power-law flow law generally results in decreased stress magnitudes.

We defined a critical velocity threshold for assessing the continental crust's effective strength. For instance, modelled Tibetan plateau spreading velocities exceeding 4 cm/yr suggest an unrealistically weak continental crust. Our simulations have revealed that a lithospheric mantle with an effective viscosity of 10^{20} Pa·s necessitates a maximum differential stress in the crust exceeding 100 MPa to achieve velocities below this threshold. Similarly, a stronger lithospheric mantle with an average viscosity of 10^{22} Pa·s requires crustal differential stresses on the order of 60 to 80 MPa to maintain slower spreading velocities. In all scenarios explored, an average crustal viscosity of at least 3×10^{21} Pa·s is essential to ensure velocities below 4 cm/yr. This underscores the importance of a mechanically strong crust to sustain large topographic features and lateral crustal thickness variations over extended geological time scales.

We derived simple analytical estimates for the crustal horizontal deviatoric stress and spreading velocities of continental plateaus. These estimates can be used to estimate average crustal viscosities for natural plateaus. For the eastern Tibetan plateau we estimate a crustal viscosity of $\approx 10^{22}$ Pa·s which is in agreement with previous estimates. Furthermore, we suggest that the 3D stress field in the syntaxes regions of the Tibetan plateau, which is caused by the lateral variations in gravitational potential energy, can support ongoing strike-slip faulting and fast exhumation because of locally increased horizontal shear stresses and locally increased stress weakening, respectively.

3.6 Acknowledgements

We thank Y. Podladchikov for theoretical and programming advices. This work was supported by SNSF grant No. 200020 197218. This work was supported by the University of Lausanne.

3.7 Open Research

Current and future versions of the SphericalStokes.jl software used in this study are publicly available on GitHub at `https://github.com/PTsolvers/SphericalStokes`. The exact version used in this study is archived on Zenodo and can be access at `https://zenodo.org/doi/10.5281/zenodo.10093648` Macherel et al. (2023).

Bibliography

Allmann, B. P., Shearer, P. M., 2009. Global variations of stress drop for moderate to large earthquakes. Journal of Geophysical Research: Solid Earth 114 (B1).

Andersen, T. B., Mair, K., Austrheim, H., Podladchikov, Y. Y., Vrijmoed, J. C., 2008. Stress release in exhumed intermediate and deep earthquakes determined from ultramafic pseudotachylyte. Geology 36 (12), 995–998.

Angel, R. J., Nimis, P., Mazzucchelli, M. L., Alvaro, M., Nestola, F., 2015. How large are departures from lithostatic pressure? Constraints from host-inclusion elasticity. Journal of Metamorphic Geology 33 (8), 801–813.

Artyushkov, E., 1973. Stresses in the lithosphere caused by crustal thickness inhomogeneities. Journal of Geophysical Research 78 (32), 7675–7708.

Bai, L., Li, G., Khan, N. G., Zhao, J., Ding, L., 2017. Focal depths and mechanisms of shallow earthquakes in the Himalayan–Tibetan region. Gondwana Research 41, 390–399.

Beaumont, C., Jamieson, R. A., Nguyen, M. H., Medvedev, S., 2004. Crustal channel flows: 1. Numerical models with applications to the tectonics of the Himalayan-Tibetan orogen. Journal of Geophysical Research: Solid Earth 109 (B6).

Bendick, R., Ehlers, T. A., 2014. Extreme localized exhumation at syntaxes initiated by subduction geometry. Geophysical Research Letters 41 (16), 5861–5867.

Bird, P., Piper, K., 1980. Plane-stress finite-element models of tectonic flow in southern California. Physics of the earth and planetary interiors 21 (2-3), 158–175.

Bischoff, S., Flesch, L., 2019. Impact of lithospheric strength distribution on India-Eurasia deformation from 3-D geodynamic models. Journal of Geophysical Research: Solid Earth 124 (1), 1084–1105.

Bott, M., Kusznir, N., 1984. The origin of tectonic stress in the lithosphere. Tectonophysics 105 (1-4), 1–13.

Bower, A., 2009. Applied Mechanics of Solids. CRC Press.

Brückner, L. M., Trepmann, C. A., 2021. Stresses during pseudotachylyte formation-evidence from deformed amphibole and quartz in fault rocks from the Silvretta basal thrust (Austria). Tectonophysics 817, 229046.

Burg, J.-P., Davy, P., Nievergelt, P., Oberli, F., Seward, D., Diao, Z., Meier, M., 1997. Exhumation during crustal folding in the Namche-Barwa syntaxis. Terra Nova 9 (2), 53–56.

Campbell, L., Menegon, L., 2022. High stress deformation and short-term thermal pulse preserved in pyroxene microstructures from exhumed lower crustal seismogenic faults (Lofoten, Norway). Journal of Geophysical Research (JGR): Solid Earth 127 (7).

Chamolly, A., Ribe, N. M., 2021. Fluid mechanics of free subduction on a sphere. Part 1. The axisymmetric case. Journal of Fluid Mechanics 929, A22.

Chen, F., Davies, D. R., Goes, S., Suchoy, L., Kramer, S. C., 2022. Comparing the dynamics of free subduction in cartesian and spherical domains. Geochemistry, Geophysics, Geosystems 23 (12), e2022GC010757.

Chen, L., Liu, L., Capitanio, F. A., Gerya, T. V., Li, Y., 2020. The role of pre-existing weak zones in the formation of the Himalaya and Tibetan plateau: 3-D thermomechanical modelling. Geophysical Journal International 221 (3), 1971–1983.

Clark, M. K., Royden, L. H., 2000. Topographic ooze: Building the eastern margin of Tibet by lower crustal flow. Geology 28 (8), 703–706.

Coblentz, D. D., Richardson, R. M., Sandiford, M., 1994. On the gravitational potential of the Earth's lithosphere. Tectonics 13 (4), 929–945.

Crameri, F., Schmeling, H., Golabek, G., Duretz, T., Orendt, R., Buiter, S., May, D., Kaus, B., Gerya, T., Tackley, P., 2012. A comparison of numerical surface topography calculations in geodynamic modelling: an evaluation of the 'sticky air' method. Geophysical Journal International 189 (1), 38–54.

Dalmayrac, B., Molnar, P., 1981. Parallel thrust and normal faulting in Peru and constraints on the state of stress. Earth and Planetary Science Letters 55 (3), 473–481.

Darwin, G. H., 1882. IV. on the stresses caused in the interior of the earth by the weight of continents and mountains. Philosophical Transactions of the Royal Society of London (173), 187–230.

Doin, M.-P., Twardzik, C., Ducret, G., Lasserre, C., Guillaso, S., Jianbao, S., 2015. InSAR measurement of the deformation around Siling Co Lake: Inferences on the lower crust viscosity in central Tibet. Journal of Geophysical Research: Solid Earth 120 (7), 5290–5310.

Engelder, T., 2014. Stress regimes in the lithosphere. Vol. 151. Princeton University Press.

England, P., Houseman, G., 1986. Finite strain calculations of continental deformation: 2. comparison with the India-Asia collision zone. Journal of Geophysical Research: Solid Earth 91 (B3), 3664–3676.

England, P., Houseman, G., 1988. The mechanics of the Tibetan Plateau. Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences 326 (1589), 301–320.

England, P., McKenzie, D., 1982. A thin viscous sheet model for continental deformation. Geophysical Journal International 70 (2), 295–321.

England, P., Molnar, P., 1997. Active deformation of Asia: From kinematics to dynamics. Science 278 (5338), 647–650.

England, P., Molnar, P., 2015. Rheology of the lithosphere beneath the central and western Tien Shan. Journal of Geophysical Research: Solid Earth 120 (5), 3803–3823.

England, P., Walker, R. T., Fu, B., Floyd, M. A., 2013. A bound on the viscosity of the Tibetan crust from the horizontality of palaeolake shorelines. Earth and Planetary Science Letters 375, 44–56.

Fleitout, L., Froidevaux, C., 1982. Tectonics and topography for a lithosphere containing density heterogeneities. Tectonics 1 (1), 21–56.

Flesch, L. M., Haines, A. J., Holt, W. E., 2001. Dynamics of the India-Eurasia collision zone. Journal of Geophysical Research: Solid Earth 106 (B8), 16435–16460.

Gan, W., Zhang, P., Shen, Z.-K., Niu, Z., Wang, M., Wan, Y., Zhou, D., Cheng, J., 2007. Present-day crustal motion within the Tibetan Plateau inferred from GPS measurements. Journal of Geophysical Research: Solid Earth 112 (B8).

Gerya, T., 2019. Introduction to numerical geodynamic modelling. Cambridge University Press.

Ghosh, A., Holt, W. E., Flesch, L. M., 2009. Contribution of gravitational potential energy differences to the global stress field. Geophysical Journal International 179 (2), 787–812.

Ghosh, A., Holt, W. E., Flesch, L. M., Haines, A. J., 2006. Gravitational potential energy of the Tibetan Plateau and the forces driving the Indian plate. Geology 34 (5), 321–324.

Hardebeck, J. L., Okada, T., 2018. Temporal stress changes caused by earthquakes: A review. Journal of Geophysical Research: Solid Earth 123 (2), 1350–1365.

Harrison, T. M., Copeland, P., Kidd, W., Yin, A., 1992. Raising Tibet. Science 255 (5052), 1663–1670.

Hetényi, G., Vergne, J., Bollinger, L., Subedi, S., Michailos, K., Drukpa, D., 2023. Seismological imaging and current seismicity of the Himalayan arc. Himalaya, Dynamics of a Giant 1: Geodynamic Setting of the Himalayan Range, 101–128.

Hirth, G., Kohlstedt, D., 2003. Rheology of the upper mantle and the mantle wedge: A view from the experimentalists. Geophysical monograph-american geophysical union 138, 83–106.

Jeffreys, H., 1932. On the stresses in the Earth's crust required to support surface inequalities.(Second Paper.). Geophysical Journal International 3, 60–69.

Kanamori, H., 1980. The state of stress in the Earth's lithosphere.

Karato, S.-i., 2008. Deformation of Earth materials. An introduction to the rheology of Solid Earth 463.

Kind, R., Yuan, X., Saul, J., Nelson, D., Sobolev, S., Mechie, J., Zhao, W., Kosarev, G., Ni, J., Achauer, U., Jiang, M., 2002. Seismic images of crust and upper mantle beneath Tibet: Evidence for Eurasian plate subduction. science 298 (5596), 1219–1221.

Kohlstedt, D., Evans, B., Mackwell, S., 1995. Strength of the lithosphere: Constraints imposed by laboratory experiments. Journal of Geophysical Research: Solid Earth 100 (B9), 17587–17602.

Koptev, A., Ehlers, T. A., Nettesheim, M., Whipp, D. M., 2019. Response of a rheologically stratified lithosphere to subduction of an indenter-shaped plate: Insights into localized exhumation at orogen syntaxes. Tectonics 38 (6), 1908–1930.

Laske, G., Masters, G., Ma, Z., Pasyanos, M., 2013. Update on CRUST1. 0—A 1-degree global model of Earth's crust. In: Geophysical research abstracts. Vol. 15. EGU General Assembly 2013, Vienna, Austria, p. 2658.

Lechmann, S. M., Schmalholz, S. M., Hetényi, G., May, D. A., Kaus, B. J. P., 2014. Quantifying the impact of mechanical layering and underthrusting on the dynamics of the modern India-Asia collisional system with 3-D numerical models. Journal of Geophysical Research: Solid Earth 119 (1), 616–644.

Li, J., Hou, G., 2019. Stress development in heterogeneous lithosphere: insights into earthquake initiation in the Tan-Lu fault zone. Tectonophysics 750, 329–343.

Liang, S., Gan, W., Shen, C., Xiao, G., Liu, J., Chen, W., Ding, X., Zhou, D., 2013. Three-dimensional velocity field of present-day crustal motion of the Tibetan Plateau derived from GPS measurements. Journal of Geophysical Research: Solid Earth 118 (10), 5722–5732.

Lithgow-Bertelloni, C., Guynn, J. H., 2004. Origin of the lithospheric stress field. Journal of Geophysical Research: Solid Earth 109 (B1).

Liu, M., Yang, Y., 2003. Extensional collapse of the Tibetan Plateau: Results of three-dimensional finite element modeling. Journal of Geophysical Research: Solid Earth 108 (B8).

Macherel, E., Räss, L., Schmalholz, S. M., 2023. PTsolvers/SphericalStokes: SphericalStokes.jl 1.0.1.

Malvern, L. E., 1969. Introduction to the mechanics of a continuous medium. Prentice-Hall, Inc.

McGarr, A., Gay, N., 1978. State of stress in the Earth's crust. Annual Review of Earth and Planetary Sciences 6 (1), 405–436.

Medvedev, S. E., 2002. Mechanics of viscous wedges: Modeling by analytical and numerical approaches. Journal of Geophysical Research: Solid Earth 107 (B6), ETG 9–1–ETG 9–15.

Medvedev, S. E., Podladchikov, Y. Y., 1999. New extended thin-sheet approximation for geodynamic applications—I. model formulation. Geophysical Journal International 136 (3), 567–585.

Molnar, P., England, P., Martinod, J., 1993. Mantle dynamics, uplift of the Tibetan Plateau, and the Indian monsoon. Reviews of Geophysics 31 (4), 357–396.

Molnar, P., Lyon-Caen, H., 1988. Some simple physical aspects of the support, structure, and evolution of mountain belts. Processes in continental lithospheric deformation 218, 179–207.

Molnar, P., Tapponnier, P., 1975. Cenozoic tectonics of Asia: Effects of a continental collision: Features of recent continental tectonics in Asia can be interpreted as results of the India-Eurasia collision. science 189 (4201), 419–426.

Nábělek, J., Hetényi, G., Vergne, J., Sapkota, S., Kafle, B., Jiang, M., Su, H., Chen, J., Huang, B.-S., the Hi-CLIMB team, 2009. Underplating in the Himalaya-Tibet collision zone revealed by the Hi-CLIMB experiment. Science 325 (5946), 1371–1374.

Parsons, B., Richter, F. M., 1980. A relation between the driving force and geoid anomaly associated with mid-ocean ridges. Earth and Planetary Science Letters 51 (2), 445–450.

Penney, C., Copley, A., 2021. Lateral variations in lower crustal strength control the temporal evolution of mountain ranges: examples from south-east Tibet. Geochemistry, Geophysics, Geosystems 22 (2), e2020GC009092.

Philpotts, A. R., Ague, J. J., 2022. Principles of igneous and metamorphic petrology. Cambridge University Press.

Pini, V., Ruz, J. J., Kosaka, P. M., Malvar, O., Calleja, M., Tamayo, J., 2016. How two-dimensional bending can extraordinarily stiffen thin sheets. Scientific Reports 6 (1).

Plümper, O., Wallis, D., Teuling, F., Moulas, E., Schmalholz, S. M., Amiri, H., Müller, T., 2022. High-magnitude stresses induced by mineral-hydration reactions. Geology 50 (12), 1351–1355.

Pusok, A., Kaus, B. J., 2015. Development of topography in 3-D continental-collision models. Geochemistry, Geophysics, Geosystems 16 (5), 1378–1400.

Ramberg, H., 1981. Gravity, deformation and the Earth's crust: in theory, experiments, and geological application.

Räss, L., Utkin, I., Duretz, T., Omlin, S., Podladchikov, Y. Y., 2022. Assessing the robustness and scalability of the accelerated pseudo-transient method. Geoscientific Model Development 15 (14), 5757–5786.

Rey, P., Vanderhaeghe, O., Teyssier, C., 2001. Gravitational collapse of the continental crust: definition, regimes and modes. Tectonophysics 342 (3-4), 435–449, Partial Melting of Crust and Flow of Orogens.

Royden, L. H., Burchfiel, B. C., King, R. W., Wang, E., Chen, Z., Shen, F., Liu, Y., 1997. Surface deformation and lower crustal flow in eastern Tibet. science 276 (5313), 788–790.

Royden, L. H., Burchfiel, B. C., van der Hilst, R. D., 2008. The geological evolution of the Tibetan Plateau. science 321 (5892), 1054–1058.

Schmalholz, S. M., Duretz, T., Hetényi, G., Medvedev, S., 2019. Distribution and magnitude of stress due to lateral variation of gravitational potential energy between Indian lowland and Tibetan plateau. Geophysical Journal International 216 (2), 1313–1333.

Schmalholz, S. M., Kaus, B. J., Burg, J.-P., 2009. Stress-strength relationship in the lithosphere during continental collision. Geology 37 (9), 775–778.

Schmalholz, S. M., Medvedev, S., Lechmann, S. M., Podladchikov, Y., 2014. Relationship between tectonic overpressure, deviatoric stress, driving force, isostasy and gravitational potential energy. Geophysical Journal International 197 (2), 680–696.

Schmalholz, S. M., Podladchikov, Y., Burg, J.-P., 2002. Control of folding by gravity and matrix thickness: Implications for large-scale folding. Journal of Geophysical Research: Solid Earth 107 (B1), ETG–1.

Schmalholz, S. M., Podladchikov, Y. Y., 2013. Tectonic overpressure in weak crustal-scale shear zones and implications for the exhumation of high-pressure rocks. Geophysical Research Letters 40 (10), 1984–1988.

Shi, X., Kirby, E., Furlong, K. P., Meng, K., Robinson, R., Wang, E., 2015. Crustal strength in central Tibet determined from Holocene shoreline deflection around Siling Co. Earth and Planetary Science Letters 423, 145–154.

Sun, Y., Li, H., Fan, T., 2019. A numerical study of lithospheric deformation and strain partitioning across the Longmen Shan orogenic belt, eastern Tibetan Plateau. Tectonics 38 (8), 3108–3123.

Tapponnier, P., Zhiqin, X., Roger, F., Meyer, B., Arnaud, N., Wittlinger, G., Jingsui, Y., 2001. Oblique stepwise rise and growth of the Tibet Plateau. Science 294 (5547), 1671–1677.

Thatcher, W., Pollitz, F. F., 2008. Temporal evolution of continental lithospheric strength in actively deforming regions. GSA Today 18 (4/5).

Townend, J., Zoback, M. D., 2000. How faulting keeps the crust strong. Geology 28 (5), 399–402.

Turcotte, D., Oxburgh, E., 1976. Stress accumulation in the lithosphere. Tectonophysics 35 (1-3), 183–199.

Turcotte, D., Schubert, G., 2021. Geodynamics. Cambridge University Press.

Virieux, J., 1986. P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method. Geophysics 51 (4), 889–901.

von Karman, T., Dunn, L. G., Tsien, H.-S., 1940. The influence of curvature on the buckling characteristics of structures. Journal of the Aeronautical Sciences 7 (7), 276–289.

Wang, L. H., Yarushina, V. M., Alkhimenkov, Y., Podladchikov, Y., 2022. Physics-inspired pseudo-transient method and its application in modelling focused fluid flow with geological complexity. Geophysical Journal International 229 (1), 1–20.

Warners-Ruckstuhl, K. N., Govers, R., Wortel, R., 2012. Lithosphere-mantle coupling and the dynamics of the Eurasian Plate. Geophysical Journal International 189 (3), 1253–1276.

Warners-Ruckstuhl, K. N., Govers, R., Wortel, R., 2013. Tethyan collision forces and the stress field of the Eurasian Plate. Geophysical Journal International 195 (1), 1–15.

Weijermars, R., Schmeling, H., 1986. Scaling of Newtonian and non-Newtonian fluid dynamics without inertia for quantitative modelling of rock flow due to gravity (including the concept of rheological similarity). Physics of the Earth and Planetary Interiors 43 (4), 316–330.

Yang, J., Kaus, B. J., Li, Y., Leloup, P. H., Popov, A. A., Lu, G., Wang, K., Zhao, L., 2020. Lower crustal rheology controls the development of large offset strike-slip faults during the Himalayan-Tibetan orogeny. Geophysical Research Letters 47 (18), e2020GL089435.

Zeitler, P. K., Meltzer, A. S., Koons, P. O., Craw, D., Hallet, B., Chamberlain, C. P., Kidd, W. S., Park, S. K., Seeber, L., Bishop, M., et al., 2001. Erosion, Himalayan geodynamics, and the geomorphology of metamorphism. Gsa Today 11 (1), 4–9.

Zhang, P., Chen, L., Xiao, W., Zhang, J., 2022. Topographic response of Hinterland Basins in Tibet to the India–Asia convergence: 3D thermo-mechanical modeling. Frontiers in Earth Science 10, 845126.

Zhang, Z., Wang, Y., Chen, Y., Houseman, G. A., Tian, X., Wang, E., Teng, J., 2009. Crustal structure across Longmenshan fault belt from passive source seismic profiling. Geophysical Research Letters 36 (17).

Zheng, G., Wang, H., Wright, T. J., Lou, Y., Zhang, R., Zhang, W., Shi, C., Huang, J., Wei, N., 2017. Crustal deformation in the India-Eurasia collision zone from 25 years of GPS measurements. Journal of Geophysical Research: Solid Earth 122 (11), 9290–9312.

Zoback, M. L., 1992. First-and second-order patterns of stress in the lithosphere: The World Stress Map Project. Journal of Geophysical Research: Solid Earth 97 (B8), 11703–11728.

Appendix A. Method

Mathematical Model

In this study, we use a numerical algorithm to solve the Stokes equations in spherical coordinates. In this coordinates system, r is the radial direction, θ is the polar angle and φ is the azimuthal angle (see Figure 3.2a). The total stress tensor σ_{ij} is decomposed into a pressure P (or mean stress) and a deviatoric stress τ_{ij} as $\sigma_{ij} = -\delta_{ij}P + \tau_{ij}$. The indexes i and j vary between 1 and 3 and indicate the three spatial directions (r, θ, φ). Components of the deviatoric stress tensor are defined as:

$$\begin{aligned}
\tau_{rr} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{rr} = 2\eta_{\mathrm{E}}\left(\frac{\partial V_r}{\partial r}\right) \\
\tau_{\theta\theta} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\theta\theta} = 2\eta_{\mathrm{E}}\left(\frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{V_r}{r}\right) \\
\tau_{\varphi\varphi} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\varphi\varphi} = 2\eta_{\mathrm{E}}\left(\frac{1}{rsin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + \frac{V_r}{r} + \frac{V_\theta}{r}cot(\theta)\right) \\
\tau_{r\theta} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{r\theta} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_\theta}{\partial r} + \frac{1}{r}\frac{\partial V_r}{\partial \theta} - \frac{V_\theta}{r}\right)\right) \\
\tau_{r\varphi} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{r\varphi} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{\partial V_\varphi}{\partial r} + \frac{1}{rsin(\theta)}\frac{\partial V_r}{\partial \varphi} - \frac{V_\varphi}{r}\right)\right) \\
\tau_{\theta\varphi} &= 2\eta_{\mathrm{E}}\dot{\epsilon}_{\theta\varphi} = 2\eta_{\mathrm{E}}\left(\frac{1}{2}\left(\frac{1}{r}\frac{\partial V_\varphi}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial V_\theta}{\partial \varphi} - \frac{V_\varphi}{r}cot(\theta)\right)\right) \qquad (3.16)
\end{aligned}$$

where η_E is the effective viscosity, $\dot{\epsilon}_{ij}$ are the strain rate tensor components and V_i are the components of the velocity vector. The square root of the second invariant of the deviatoric stress tensor, τ_{II}, which is used in equation (3.11) for the stress-dependent viscosity, is

$$\tau_{\mathrm{II}} = \sqrt{0.5\,\tau_{ij}^2} = \sqrt{0.5(\tau_{\mathrm{rr}}^2 + \tau_{\theta\theta}^2 + \tau_{\varphi\varphi}^2) + \tau_{\mathrm{r}\theta}^2 + \tau_{\mathrm{r}\varphi}^2 + \tau_{\theta\varphi}^2}. \tag{3.17}$$

The equations for the conservation of mass and the conservation of the linear momentum in an incompressible fluid under gravity are:

$$\begin{aligned}
0 &= -\left(\frac{\partial V_r}{\partial r} + \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + 2\frac{V_r}{r} + \frac{V_\theta}{r}cot(\theta)\right) \\
0 &= \frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \tau_{r\varphi}}{\partial \varphi} + 2\frac{\sigma_{rr}}{r} - \frac{\sigma_{\theta\theta}}{r} - \frac{\sigma_{\varphi\varphi}}{r} + \frac{\tau_{r\theta}}{r}cot(\theta) - \rho g \\
0 &= \frac{\partial \tau_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \tau_{\theta\varphi}}{\partial \varphi} + 3\frac{\tau_{r\theta}}{r} + \frac{\sigma_{\theta\theta}}{r}cot(\theta) - \frac{\sigma_{\varphi\varphi}}{r}cot(\theta) \\
0 &= \frac{\partial \tau_{r\varphi}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{\theta\varphi}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \sigma_{\varphi\varphi}}{\partial \varphi} + 3\frac{\tau_{r\varphi}}{r} + 2\frac{\tau_{\theta\varphi}}{r}cot(\theta)
\end{aligned} \tag{3.18}$$

where ρ is the density and g_i is the gravitational acceleration in the direction i.

Numerical Method

To solve the governing equations, we discretize them on a staggered grid with constant spacing using the finite difference method. The numerical method used to solve the equations is the pseudo-transient method, which is an iterative solution strategy for stationary problems that allows solving the equations in a matrix-free way. The pseudo-transient method consists in adding a pseudo-time derivative to the Equations 3.16 and 3.18:

$$\frac{1}{2\tilde{G}}\frac{\partial \tau_{rr}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{rr}}{2\eta_{\mathrm{E}}} = \frac{\partial V_r}{\partial r}$$

$$
\begin{aligned}
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\theta\theta}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\theta\theta}}{2\eta_{\mathrm{E}}} &= \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{V_r}{r} \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\varphi\varphi}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\varphi\varphi}}{2\eta_{\mathrm{E}}} &= \frac{1}{rsin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + \frac{V_r}{r} + \frac{V_\theta}{r}cot(\theta) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{r\theta}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{r\theta}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{\partial V_\theta}{\partial r} + \frac{1}{r}\frac{\partial V_r}{\partial \theta} - \frac{V_\theta}{r}\right) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{r\varphi}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{r\varphi}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{\partial V_\varphi}{\partial r} + \frac{1}{rsin(\theta)}\frac{\partial V_r}{\partial \varphi} - \frac{V_\varphi}{r}\right) \\
\frac{1}{2\tilde{G}}\frac{\partial \tau_{\theta\varphi}}{\partial \tau_{\mathrm{PT}}} + \frac{\tau_{\theta\varphi}}{2\eta_{\mathrm{E}}} &= \frac{1}{2}\left(\frac{1}{r}\frac{\partial V_\varphi}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial V_\theta}{\partial \varphi} - \frac{V_\varphi}{r}cot(\theta)\right) \\
\frac{1}{\tilde{K}}\frac{\partial P}{\partial \tau_{\mathrm{PT}}} &= -\left(\frac{\partial V_r}{\partial r} + \frac{1}{r}\frac{\partial V_\theta}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial V_\varphi}{\partial \varphi} + 2\frac{V_r}{r} - \frac{V_\theta}{r}cot(\theta)\right) \\
\tilde{\rho}\frac{\partial V_r}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \sigma_{rr}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \tau_{r\varphi}}{\partial \varphi} + 2\frac{\sigma_{rr}}{r} - \frac{\sigma_{\theta\theta}}{r} - \frac{\sigma_{\varphi\varphi}}{r} + \frac{\tau_{r\theta}}{r}cot(\theta) - \rho g \\
\tilde{\rho}\frac{\partial V_\theta}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial \sigma_{\theta\theta}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \tau_{\theta\varphi}}{\partial \varphi} + 3\frac{\tau_{r\theta}}{r} + \frac{\sigma_{\theta\theta}}{r}cot(\theta) - \frac{\sigma_{\varphi\varphi}}{r}cot(\theta) \\
\tilde{\rho}\frac{\partial V_\varphi}{\partial \tau_{\mathrm{PT}}} &= \frac{\partial \tau_{r\varphi}}{\partial r} + \frac{1}{r}\frac{\partial \tau_{\theta\varphi}}{\partial \theta} + \frac{1}{rsin(\theta)}\frac{\partial \sigma_{\varphi\varphi}}{\partial \varphi} + 3\frac{\tau_{r\varphi}}{r} + 2\frac{\tau_{\theta\varphi}}{r}cot(\theta)
\end{aligned}
\tag{3.19}
$$

where $\tilde{K}$, $\tilde{\rho}$ and $\tilde{G}$ are numerical parameters, τ_{PT} is a pseudo-time. $\tilde{K}$ and $\tilde{G}$ can be considered as pseudo-bulk and pseudo-shear modulus respectively, and $\tilde{\rho}$ as a pseudo-density. With these parameters, Equation 3.19 can be considered as acoustic and inertial approximations of the mass and momentum balance equations respectively. The initial guess of pressure and velocity fields do not fulfill Equation 3.19. After a number of iterations, the pseudo-time derivatives become smaller than a specified tolerance, here 5×10^{-7}, and a numerical solution is reached.

CHAPTER 4

General Conclusion and Perspectives

Stresses are important because they control many major geological processes, such as seismicity, volcanism, folding, faulting and formation of large features such as mountain belts (Turcotte and Schubert, 2021). The aim of this thesis is to quantify and evaluate the 3D distribution and magnitude of stresses around specific geological settings, namely diapirism and continental plateaus.

In the first study (Chapter 2), the focus is on diapirism. This process is an important mechanism for mass and heat transportation (Turcotte and Schubert, 2021) and it occurs in many different geodynamic settings, time and spatial scales. Thus, quantifying the velocity of ascent is central as it determines the importance of diapirism. In this study, we investigate the rising velocity of a weak and buoyant sphere in a non-linear viscous fluid under far-field deformation. We calculated everything in dimensionless form so that our results are applicable to a wide range of geological settings.

Our results show that the rising velocity of a diapir depends on two stress ratios: (a) regional stress over characteristic stress, where regional stress is generated by far-field deformation, in this study a strike-slip shearing. The characteristic stress is a material parameter, marking the stress at which the deformation behavior changes from linear (diffusion creep) to power-law viscous flow law (dislocation creep). (b) Buoyancy stress over characteristic stress, where buoyancy stress ($\Delta\rho g R$) is induced by the diapir itself. Hence, it depends on the size R of the buoyant inclusion and on the density difference ($\Delta\rho$) between the diapir and its surrounding medium. Because of the non-linear rheology of the rocks, both stress ratios provoke stress weakening around the diapir, which allows it to rise faster than it would in a linear medium. Also, the effects of each stress ratio add up to increase the velocity of ascent up to 4 orders of magnitude faster than the calculated velocity in a linear viscous medium. We performed a systematic quantification of the rising velocity by varying both stress ratios.

Also, we compare our results with the analytical estimates of Weinberg and Podladchikov (1994). We modify them by adding a term considering the effect of the regional stresses and find that for a wide range of stress ratios, analytical estimates are accurate within a factor of 2. However, the largest discrepancies arise when buoyancy stresses are large. This can be explained by the fact that the analytical estimates consider buoyancy stresses as a proxy for deviatoric stresses. Yet, numerical results show that deviatoric stresses, used in the calculation of the effective viscosity, can be much smaller than buoyancy stresses. Therefore, we could improve the analytical estimates by adding a “shape factor”. We also compared our results with studies of sediment diapirs in subduction zones (e.g., Klein and Behn, 2021), mantle plumes (e.g., Schubert et al., 2001), (U)HP terranes (e.g., Burov et al., 2014) and plutons (Michail et al., 2021). In summary, the analytical estimates are very useful to evaluate the importance of diapirism in a wide range of geological settings.

The second study (Chapter 3) focuses on stresses inside and around continental plateaus, such as the Tibetan plateau. It is well known that stresses deviate from the lithostatic state to generate horizontal forces supporting large thickness regions (Darwin, 1882; Jeffreys, 1932). Stress and strength estimates established by lateral variations of gravitational potential energy (GPE), and thin viscous sheet models are robust, however, these methods provide depth-averaged estimations and not the vertical distribution of these quantities across the lithosphere. In this study, we calculated the 3D stress distribution inside and around an ideal geometry of a continental plateau in isostatic equilibrium. This allowed us to quantify the impact of different parameters and characteristics on the stress field.

First, curvature does not significantly impact the lithospheric stress field at this spatial scale. As a consequence, one can neglect the curvature if the area of interest is smaller than a few thousands of km.

Second, the corner region of the plateau exposes a significant effect on the stress distribution. Shear stresses are negligible far away from that area but increase, in a non-linear manner, up to about 30 MPa at the corner. Normal stresses, on the opposite, decrease towards the corner region. Our results show that velocity and stress directions rotate around the corner region.

Also, we tested different viscosities of the crust and of the lithospheric mantle. Our results show that varying these viscosities strongly affects the spreading velocities magnitude and stress distibution. Indeed, a weak crust (10^{20} Pa·s) associated with a weak lithospheric mantle (10^{20} Pa·s) presents horizontal velocities of nearly 70 cm/yr, which means the plateau would flow apart very quickly and no topography variations could be sustained on Earth. Another case, a strong lithospheric mantle (10^{22} Pa·s) associated with a weak crust (10^{20} Pa·s), also produces velocities that are unreasonably large (~ 25 cm/yr). As a consequence, our results would not fit with studies stating that the lithospheric strength lies in the lithospheric mantle exclusively (i.e., England and Molnar, 2015). On the opposite, a strong continental crust (10^{22} Pa·s) with a weak lithospheric mantle (10^{20} Pa·s) displays velocities smaller than 4 cm/yr, supporting the results of Flesch et al. (2001) stating that the lithospheric strength is mostly located inside the crust. Furthermore, comparing all simulations we performed, it appears the crustal viscosity must be at least 3×10^{21} Pa·s for a large topographic feature, such as a continental plateau, to be supported over long time scales. Our results also show that similar stress magnitudes can be associated with a wide range of horizontal velocities and strain rates. Hence, effective strength cannot be estimated based on differential stress magnitudes.

Furthermore, we derive simple analytical estimates for horizontal deviatoric stress and spreading velocities of a continental plateau. These estimates also allow to evaluate a vertically averaged viscosity of the lithosphere.

Our studies allow a better understanding of the stress distribution around diapirs and continental plateaus. However, I propose the following points to improve our model:

1. The numerical model we created can be developed by many different aspects. However, two of them are a priority:
    (a) Evolution with time: for now, the model calculates the instantaneous stress and velocity fields. Being able to calculate their time evolution would allow, for example, to model the rise of the diapir and the lateral spreading of the plateau.
    (b) Thermal coupling: currently, our model only calculates the mechanic part of geological processes. Coupling our solver with temperature dependent density and viscosity would allow to model the cooling of the diapir and its impact on the surrounding rocks. It would also improve the rheology structure of the lithosphere (Schmalholz et al., 2019).
2. Our simulations use very idealized geometries, which allow to understand the first-order processes acting on the stresses. It is necessary to apply the model to real geological settings.

Many more improvements and applications can be envisioned for this model to study more complex geological processes.

Bibliography

Burov, E., François, T., Agard, P., Le Pourhiet, L., Meyer, B., Tirel, C., Lebedev, S., Yamato, P., Brun, J.-P., 2014. Rheological and geodynamic controls on the mechanisms of subduction and HP/UHP exhumation of crustal rocks during continental collision: Insights from numerical models. Tectonophysics 631, 212–250.

Darwin, G. H., 1882. IV. on the stresses caused in the interior of the earth by the weight of continents and mountains. Philosophical Transactions of the Royal Society of London (173), 187–230.

England, P., Molnar, P., 2015. Rheology of the lithosphere beneath the central and western Tien Shan. Journal of Geophysical Research: Solid Earth 120 (5), 3803–3823.

Flesch, L. M., Haines, A. J., Holt, W. E., 2001. Dynamics of the India-Eurasia collision zone. Journal of Geophysical Research: Solid Earth 106 (B8), 16435–16460.

Jeffreys, H., 1932. On the stresses in the Earth's crust required to support surface inequalities.(Second Paper.). Geophysical Journal International 3, 60–69.

Klein, B. Z., Behn, M. D., 2021. On the evolution and fate of sediment diapirs in subduction zones. Geochemistry, Geophysics, Geosystems 22 (11).

Michail, M., Rudolf, M., Rosenau, M., Riva, A., Gianolla, P., Coltorti, M., 2021. Shape of plutons in crustal shear zones: A tectono-magmatic guide based on analogue models. Journal of Structural Geology 150, 104417.

Schmalholz, S. M., Duretz, T., Hetényi, G., Medvedev, S., 2019. Distribution and magnitude of stress due to lateral variation of gravitational potential energy between Indian lowland and Tibetan plateau. Geophysical Journal International 216 (2), 1313–1333.

Schubert, G., Turcotte, D. L., Olson, P., 2001. Mantle convection in the Earth and planets. Cambridge University Press.

Turcotte, D., Schubert, G., 2021. Geodynamics. Cambridge University Press.

Weinberg, R. F., Podladchikov, Y., 1994. Diapiric ascent of magmas through power law crust and mantle. Journal of Geophysical Research: Solid Earth 99 (B5), 9543–9559.

APPENDIX A

Numerical Grids

The finite difference method is used to discretize the equations in the numerical algorithms. Figure A.1 displays the employed staggered grids. For better readability, the grids are shown in 2D.

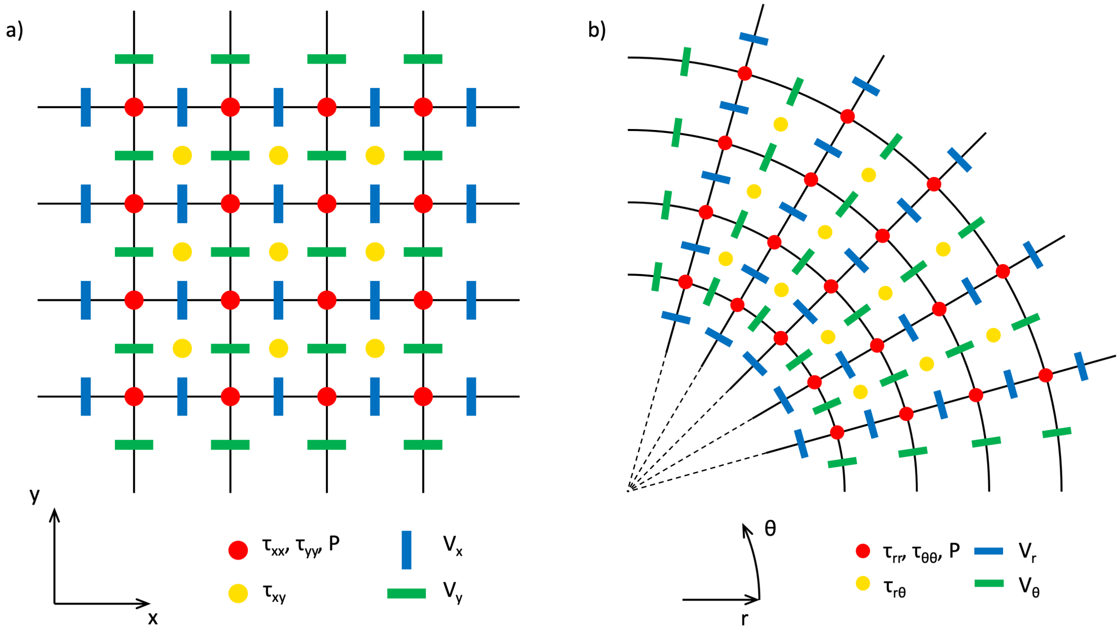

Figure A.1: Staggered grid in a) Cartesian coordinates, and b) polar coordinates.

APPENDIX B

Code for the Diapir in Cartesian Coordinates

```
# DIAPIR - CARTESIAN COORDINATES SYSTEM
const USE_GPU = true
const GPU_ID  = 0

saveflag      = true

using ParallelStencil
using ParallelStencil.FiniteDifferences3D
@static if USE_GPU
    @init_parallel_stencil(CUDA, Float64, 3)
    CUDA.device!(GPU_ID) # select GPU
else
    @init_parallel_stencil(Threads, Float64, 3)
end
using Plots, Printf, Statistics, LinearAlgebra

# SAVING FUNCTIONS ==========================================================================================
function Save_infos(num, lx, ly, lz, nx, ny, nz, εnonl, runtime; out="./output")
    fid=open(out * "/$(num)_infos.inf", "w")
    @printf(fid,"%d %f %f %f %d %d %d %d %d", num, lx, ly, lz, nx, ny, nz, εnonl, runtime); close(fid)
end

function Save_phys(num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g, sh, radius, dmp; out="./output")
    fid=open(out * "/$(num)_phys.inf", "w")
    @printf(fid,"%d   %f  %f  %f    %f     %f   %f  %f  %f  %f      %f",
                 num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g , sh, radius, dmp); close(fid)
end

@static if USE_GPU
    function SaveArray(Aname, A; out="./output")
        A_tmp = Array(A)
        fname = string(out, "/A_", Aname, ".bin");  fid = open(fname,"w"); write(fid, A_tmp); close(fid)
    end
else
    function SaveArray(Aname, A; out="./output")
        fname = string(out, "/A_", Aname, ".bin"); fid = open(fname,"w"); write(fid, A); close(fid)
    end
end

# ===========================================================================================================
@parallel_indices (ix,iy,iz) function equal!(A::Data.Array, B::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = B[ix,iy,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function multiply!(A::Data.Array, B::Data.Array, fact::Data.Number)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = B[ix,iy,iz]*fact    end
    return
end

# BOUNDARY CONDITIONS =======================================================================================
@parallel_indices (ix,iy,iz) function bc_x_lin!(A::Data.Array, COORD::Data.Array, fact::Data.Number)
    if (ix==1         && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = 2*COORD[ix  ,iy,iz]*fact -
                                                                                A[ix+1,iy,iz]    end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = 2*COORD[ix  ,iy,iz]*fact -
                                                                                A[ix-1,iy,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function bc_x_0!(A::Data.Array)
    if (ix==1         && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = 0.0    end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = 0.0    end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_x!(A::Data.Array)
    if (ix==1         && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix+1,iy,iz]    end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix-1,iy,iz]    end
    return
end
```

```
@parallel_indices (ix,iy,iz) function bc_y_fact!(A::Data.Array, COORD::Data.Array, fact::Data.Number)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = fact*COORD[ix,iy,iz]    end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = fact*COORD[ix,iy,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function bc_y_0!(A::Data.Array)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = -A[ix,iy+1,iz]    end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = -A[ix,iy-1,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function bc_z_0!(A::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1        )    A[ix,iy,iz] = -A[ix,iy,iz+1]    end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = -A[ix,iy,iz-1]    end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_z!(A::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1        )    A[ix,iy,iz] = A[ix,iy,iz+1]    end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = A[ix,iy,iz-1]    end
    return
end

# COPY BOUNDARIES =============================================================================================
@parallel_indices (ix,iy,iz) function copy_boundary!(A::Data.Array)
    if (ix==1           && iy<=size(A,2)-2 && iz<=size(A,3)-2)  A[ix  ,iy+1,iz+1] = A[ix+1,iy+1,iz+1]   end
    if (ix==size(A,1)   && iy<=size(A,2)-2 && iz<=size(A,3)-2)  A[ix  ,iy+1,iz+1] = A[ix-1,iy+1,iz+1]   end
    if (ix<=size(A,1)-2 && iy==1           && iz<=size(A,3)-2)  A[ix+1,iy  ,iz+1] = A[ix+1,iy+1,iz+1]   end
    if (ix<=size(A,1)-2 && iy==size(A,2)   && iz<=size(A,3)-2)  A[ix+1,iy  ,iz+1] = A[ix+1,iy-1,iz+1]   end
    if (ix<=size(A,1)-2 && iy<=size(A,2)-2 && iz==1          )  A[ix+1,iy+1,iz  ] = A[ix+1,iy+1,iz+1]   end
    if (ix<=size(A,1)-2 && iy<=size(A,2)-2 && iz==size(A,3)  )  A[ix+1,iy+1,iz  ] = A[ix+1,iy+1,iz-1]   end
    return
end

# INITIALIZATION ==============================================================================================
@parallel_indices (ix,iy,iz) function initialize_inclusion!(     A::Data.Array , X3D::Data.Array,
                                                               Y3D::Data.Array , Z3D::Data.Array,
                                                            radius::Data.Number,  in::Data.Number)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))
        if ((X3D[ix,iy,iz]^2 + Y3D[ix,iy,iz]^2 + Z3D[ix,iy,iz]^2) < radius)    A[ix,iy,iz] = in    end
    end
    return
end

@parallel_indices (ix,iy,iz) function initialize_velocity!(   V::Data.Array, COORD::Data.Array,
                                                           fact::Data.Number)
    if (ix<=size(V,1)-1 && iy<=size(V,2) && iz<=size(V,3))    V[ix,iy,iz] = COORD[ix  ,iy,iz]*fact    end
    if (ix==size(V,1)   && iy<=size(V,2) && iz<=size(V,3))    V[ix,iy,iz] = COORD[ix-1,iy,iz]*fact    end
    return
end

# ITERATION STRATEGY ==========================================================================================
@parallel function maxloc!(η_Max::Data.Array, η::Data.Array)
    @inn(η_Max) = @maxloc(η)
    return
end

macro KBDT(ix,iy,iz)    esc(:(    dmp * 2.0 * pi * vpdt / lx * ηSM[$ix,$iy,$iz]    ))    end
macro GSDT(ix,iy,iz)    esc(:(          4.0 * pi * vpdt / lx * ηSM[$ix,$iy,$iz]    ))    end
@parallel_indices (ix,iy,iz) function timesteps!(DT_R::Data.Array ,   η::Data.Array , ηSM::Data.Array,
                                                 vpdt::Data.Number, dmp::Data.Number,  lx::Data.Number)
    if (ix<=size(DT_R,1) && iy<=size(DT_R,2) && iz<=size(DT_R,3))
        DT_R[ix,iy,iz] = vpdt^2 / (@KBDT(ix,iy,iz) + @GSDT(ix,iy,iz)/(1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz]))
    end
    return
end
```

```
# SOLVER ======================================================================================
macro avxyi_η(ix,iy,iz)     esc(:(    0.25*(  η[$ix  ,$iy  ,$iz+1] +   η[$ix  ,$iy+1,$iz+1] +
                                              η[$ix+1,$iy  ,$iz+1] +   η[$ix+1,$iy+1,$iz+1])    ))   end
macro avxzi_η(ix,iy,iz)     esc(:(    0.25*(  η[$ix  ,$iy+1,$iz  ] +   η[$ix  ,$iy+1,$iz+1] +
                                              η[$ix+1,$iy+1,$iz  ] +   η[$ix+1,$iy+1,$iz+1])    ))   end
macro avyzi_η(ix,iy,iz)     esc(:(    0.25*(  η[$ix+1,$iy  ,$iz  ] +   η[$ix+1,$iy  ,$iz+1] +
                                              η[$ix+1,$iy+1,$iz  ] +   η[$ix+1,$iy+1,$iz+1])    ))   end
macro avxyi_ηSM(ix,iy,iz)   esc(:(    0.25*(ηSM[$ix  ,$iy  ,$iz+1] + ηSM[$ix  ,$iy+1,$iz+1] +
                                            ηSM[$ix+1,$iy  ,$iz+1] + ηSM[$ix+1,$iy+1,$iz+1])    ))   end
macro avxzi_ηSM(ix,iy,iz)   esc(:(    0.25*(ηSM[$ix  ,$iy+1,$iz  ] + ηSM[$ix  ,$iy+1,$iz+1] +
                                            ηSM[$ix+1,$iy+1,$iz  ] + ηSM[$ix+1,$iy+1,$iz+1])    ))   end
macro avyzi_ηSM(ix,iy,iz)   esc(:(    0.25*(ηSM[$ix+1,$iy  ,$iz  ] + ηSM[$ix+1,$iy  ,$iz+1] +
                                            ηSM[$ix+1,$iy+1,$iz  ] + ηSM[$ix+1,$iy+1,$iz+1])    ))   end
macro avxyi_GSDT(ix,iy,iz)  esc(:(    4.0 * pi * vpdt / lx * @avxyi_ηSM(ix,iy,iz)    ))              end
macro avxzi_GSDT(ix,iy,iz)  esc(:(    4.0 * pi * vpdt / lx * @avxzi_ηSM(ix,iy,iz)    ))              end
macro avyzi_GSDT(ix,iy,iz)  esc(:(    4.0 * pi * vpdt / lx * @avyzi_ηSM(ix,iy,iz)    ))              end
macro D_XX(ix,iy,iz)        esc(:(    (VX[$ix+1,$iy  ,$iz  ]-VX[$ix,$iy,$iz])*_dx    ))              end
macro D_YY(ix,iy,iz)        esc(:(    (VY[$ix  ,$iy+1,$iz  ]-VY[$ix,$iy,$iz])*_dy    ))              end
macro D_ZZ(ix,iy,iz)        esc(:(    (VZ[$ix  ,$iy  ,$iz+1]-VZ[$ix,$iy,$iz])*_dz    ))              end
macro D_XY(ix,iy,iz)        esc(:(    (VX[$ix+1,$iy+1,$iz+1]-VX[$ix+1,$iy  ,$iz+1])*_dy +
                                      (VY[$ix+1,$iy+1,$iz+1]-VY[$ix  ,$iy+1,$iz+1])*_dx    ))        end
macro D_XZ(ix,iy,iz)        esc(:(    (VX[$ix+1,$iy+1,$iz+1]-VX[$ix+1,$iy+1,$iz  ])*_dz +
                                      (VZ[$ix+1,$iy+1,$iz+1]-VZ[$ix  ,$iy+1,$iz+1])*_dx    ))        end
macro D_YZ(ix,iy,iz)        esc(:(    (VY[$ix+1,$iy+1,$iz+1]-VY[$ix+1,$iy+1,$iz  ])*_dz +
                                      (VZ[$ix+1,$iy+1,$iz+1]-VZ[$ix+1,$iy  ,$iz+1])*_dy    ))        end
@parallel_indices (ix,iy,iz) function compute_P!(   P::Data.Array , τ_XX::Data.Array , τ_YY::Data.Array ,
                                                 τ_ZZ::Data.Array , τ_XY::Data.Array , τ_XZ::Data.Array ,
                                                 τ_YZ::Data.Array ,   VX::Data.Array ,   VY::Data.Array ,
                                                   VZ::Data.Array ,    η::Data.Array ,  ηSM::Data.Array ,
                                                  dmp::Data.Number, vpdt::Data.Number,   lx::Data.Number,
                                                  _dx::Data.Number,  _dy::Data.Number,  _dz::Data.Number)
    if (ix<=size(P   ,1) && iy<=size(P   ,2) && iz<=size(P   ,3))
        P[ix,iy,iz]    = P[ix,iy,iz]    - @KBDT(ix,iy,iz) * (@D_XX(ix,iy,iz) + @D_YY(ix,iy,iz) +
                                                                               @D_ZZ(ix,iy,iz))      end
    if (ix<=size(τ_XX,1) && iy<=size(τ_XX,2) && iz<=size(τ_XX,3))
        τ_XX[ix,iy,iz] = (τ_XX[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@D_XX(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])                              end
    if (ix<=size(τ_YY,1) && iy<=size(τ_YY,2) && iz<=size(τ_YY,3))
        τ_YY[ix,iy,iz] = (τ_YY[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@D_YY(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])                              end
    if (ix<=size(τ_ZZ,1) && iy<=size(τ_ZZ,2) && iz<=size(τ_ZZ,3))
        τ_ZZ[ix,iy,iz] = (τ_ZZ[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@D_ZZ(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])                              end
    if (ix<=size(τ_XY,1) && iy<=size(τ_XY,2) && iz<=size(τ_XY,3))
        τ_XY[ix,iy,iz] = (τ_XY[ix,iy,iz] + @avxyi_GSDT(ix,iy,iz)*@D_XY(ix,iy,iz))/
                                    (1.0 + @avxyi_GSDT(ix,iy,iz)/@avxyi_η(ix,iy,iz))                 end
    if (ix<=size(τ_XZ,1) && iy<=size(τ_XZ,2) && iz<=size(τ_XZ,3))
        τ_XZ[ix,iy,iz] = (τ_XZ[ix,iy,iz] + @avxzi_GSDT(ix,iy,iz)*@D_XZ(ix,iy,iz))/
                                    (1.0 + @avxzi_GSDT(ix,iy,iz)/@avxzi_η(ix,iy,iz))                 end
    if (ix<=size(τ_YZ,1) && iy<=size(τ_YZ,2) && iz<=size(τ_YZ,3))
        τ_YZ[ix,iy,iz] = (τ_YZ[ix,iy,iz] + @avyzi_GSDT(ix,iy,iz)*@D_YZ(ix,iy,iz))/
                                    (1.0 + @avyzi_GSDT(ix,iy,iz)/@avyzi_η(ix,iy,iz))                 end
    return
end

@parallel_indices (ix,iy,iz) function compute_TII!( τII::Data.Array, τ_XX::Data.Array, τ_YY::Data.Array,
                                                    τ_ZZ::Data.Array, τ_XY::Data.Array, τ_XZ::Data.Array,
                                                    τ_YZ::Data.Array)
    if (ix<=size(τII,1)-2 && iy<=size(τII,2)-2 && iz<=size(τII,3)-2)
        τII[ix+1,iy+1,iz+1] = sqrt(1.0/2.0 * (τ_XX[ix+1,iy+1,iz+1]^2  + τ_YY[ix+1,iy+1,iz+1]^2 +
                                              τ_ZZ[ix+1,iy+1,iz+1]^2) +
                                       (0.25*(τ_XY[ix  ,iy  ,iz  ] + τ_XY[ix  ,iy+1,iz  ] +
                                              τ_XY[ix+1,iy  ,iz  ] + τ_XY[ix+1,iy+1,iz  ]))^2 +
                                       (0.25*(τ_XZ[ix  ,iy  ,iz  ] + τ_XZ[ix  ,iy,  iz+1] +
                                              τ_XZ[ix+1,iy  ,iz  ] + τ_XZ[ix+1,iy  ,iz+1]))^2 +
                                       (0.25*(τ_YZ[ix  ,iy  ,iz  ] + τ_YZ[ix  ,iy  ,iz+1] +
                                              τ_YZ[ix  ,iy+1,iz  ] + τ_YZ[ix  ,iy+1,iz+1]))^2)       end
    return
end
```

```
@parallel_indices (ix,iy,iz) function power_law!(η_PL::Data.Array ,        η::Data.Array ,
                                                  τII::Data.Array , η_PL_OLD::Data.Array ,
                                                η_INI::Data.Array ,      τ_C::Data.Number,
                                                n_exp::Data.Number,    relax::Data.Number)
    if (ix<=size(η_PL,1) && iy<=size(η_PL,2) && iz<=size(η_PL,3))
        η_PL[ix,iy,iz] = η_INI[ix,iy,iz] * (τII[ix,iy,iz]/τ_C)^(1.0-n_exp)                      end
    if (ix<=size(η_PL,1) && iy<=size(η_PL,2) && iz<=size(η_PL,3))
        η_PL[ix,iy,iz] = exp(log(η_PL[ix,iy,iz])*relax + log(η_PL_OLD[ix,iy,iz])*(1.0-relax))   end
    if (ix<=size(η   ,1) && iy<=size(η   ,2) && iz<=size(η   ,3))
        η[ix,iy,iz]    = 2.0/(1.0/η_INI[ix,iy,iz] + 1.0/η_PL[ix,iy,iz])                         end
    return
end

macro dVX(ix,iy,iz)        esc(:(-(   P[$ix+1,$iy+1,$iz+1]-   P[$ix,$iy+1,$iz+1])*_dx +
                                  (τ_XX[$ix+1,$iy+1,$iz+1]-τ_XX[$ix,$iy+1,$iz+1])*_dx +
                                  (τ_XY[$ix  ,$iy+1,$iz  ]-τ_XY[$ix  ,$iy,$iz  ])*_dy +
                                  (τ_XZ[$ix  ,$iy  ,$iz+1]-τ_XZ[$ix  ,$iy  ,$iz])*_dz    ))    end
macro dVY(ix,iy,iz)        esc(:( (τ_XY[$ix+1,$iy  ,$iz  ]-τ_XY[$ix,$iy  ,$iz  ])*_dx -
                                  (   P[$ix+1,$iy+1,$iz+1]-   P[$ix+1,$iy,$iz+1])*_dy +
                                  (τ_YY[$ix+1,$iy+1,$iz+1]-τ_YY[$ix+1,$iy,$iz+1])*_dy +
                                  (τ_YZ[$ix  ,$iy  ,$iz+1]-τ_YZ[$ix  ,$iy  ,$iz])*_dz    ))    end
macro dVZ(ix,iy,iz)        esc(:( (τ_XZ[$ix+1,$iy  ,$iz  ]-τ_XZ[$ix,$iy  ,$iz  ])*_dx +
                                  (τ_YZ[$ix  ,$iy+1,$iz  ]-τ_YZ[$ix  ,$iy,$iz  ])*_dy -
                                  (   P[$ix+1,$iy+1,$iz+1]-   P[$ix+1,$iy+1,$iz])*_dz +
                                  (τ_ZZ[$ix+1,$iy+1,$iz+1]-τ_ZZ[$ix+1,$iy+1,$iz])*_dz -
                                  g*(0.5*(ρ[$ix+1,$iy+1,$iz] + ρ[$ix+1,$iy+1,$iz+1]))   ))    end
macro avxi_DT_R(ix,iy,iz)  esc(:(0.5*(DT_R[$ix  ,$iy+1,$iz+1] + DT_R[$ix+1,$iy+1,$iz+1])    ))    end
macro avyi_DT_R(ix,iy,iz)  esc(:(0.5*(DT_R[$ix+1,$iy  ,$iz+1] + DT_R[$ix+1,$iy+1,$iz+1])    ))    end
macro avzi_DT_R(ix,iy,iz)  esc(:(0.5*(DT_R[$ix+1,$iy+1,$iz  ] + DT_R[$ix+1,$iy+1,$iz+1])    ))    end
@parallel_indices (ix,iy,iz) function compute_V!(  VX::Data.Array ,   VY::Data.Array ,   VZ::Data.Array ,
                                                 τ_XX::Data.Array , τ_YY::Data.Array , τ_ZZ::Data.Array,
                                                 τ_XY::Data.Array , τ_XZ::Data.Array , τ_YZ::Data.Array ,
                                                 DT_R::Data.Array ,    P::Data.Array ,    ρ::Data.Array,
                                                    η::Data.Array ,  ηSM::Data.Array , vpdt::Data.Number,
                                                  dmp::Data.Number,   lx::Data.Number,
                                                  _dx::Data.Number,  _dy::Data.Number,  _dz::Data.Number,
                                                    g::Data.Number)
    if (ix<=size(VX,1)-2 && iy<=size(VX,2)-2 && iz<=size(VX,3)-2)
        VX[ix+1,iy+1,iz+1] = VX[ix+1,iy+1,iz+1] + @dVX(ix,iy,iz) * @avxi_DT_R(ix,iy,iz)    end
    if (ix<=size(VY,1)-2 && iy<=size(VY,2)-2 && iz<=size(VY,3)-2)
        VY[ix+1,iy+1,iz+1] = VY[ix+1,iy+1,iz+1] + @dVY(ix,iy,iz) * @avyi_DT_R(ix,iy,iz)    end
    if (ix<=size(VZ,1)-2 && iy<=size(VZ,2)-2 && iz<=size(VZ,3)-2)
        VZ[ix+1,iy+1,iz+1] = VZ[ix+1,iy+1,iz+1] + @dVZ(ix,iy,iz) * @avzi_DT_R(ix,iy,iz)    end
    return
end

# CHECK ERROR ===============================================================================================
@parallel_indices (ix,iy,iz) function error!(A::Data.Array, B::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))  A[ix,iy,iz] = abs(A[ix,iy,iz] - B[ix,iy,iz])  end
    return
end
```

```
# ==========================================================================================================
@views function CART_3D_Diapir()
    num     = 1
    runtime = 0.0
    # physical parameters ------------------------------------------
    η0       = 1.0                      # Pas   , media viscosity
    vr       = 1e2                      #         viscosity ratio out/in
    ρ0       = 0.0                      # kg/m^3, media density
    ρ_in     = ρ0 - 10.0                # kg/m^3, inclusion density
    g        = 1.0                      # m/s^2 , gravitational acceleration
    n_exp    = 5.0                      #         power-law exponent
    τ_C      = 1.0                      # Pa    , characteristic stress
    relax    = 1e-3                     #         relaxation parameter (power-law)
    lx       = 6.0                      # m     , model dimension in x
    ly       = 1.0*lx                   # m     , model dimension in y
    lz       = 1.0*lx                   # m     , model dimension in z
    radius   = 1.0                      # m     , radius of the inclusion
    sh       = 1.0                      # m/s   , shearing velocity
    # numerics ---------------------------------------------
    nx      = 208 - 1                   # number of grid cells in direction x
    ny      = 208 - 1                   # number of grid cells in direction y
    nz      = 208 - 1                   # number of grid cells in direction z
    εnonl   = 5e-7                      # pseudo-transient loop exit criteria
    nt      = 1                         # number of time steps
    maxiter = 1e5                       # maximum number of pseudo-transient iterations
    nout    = 1e2                       # pseudo-transient plotting frequency
    CFL     = 1.0/(2.0 + 4.5*log10(vr)) # Courant-Friedrichs-Lewy condition
    dmp     = 4.5                       # damping parameter
    # preprocessing ----------------------------------------
    dx     = lx/(nx-1)                                      # size of cell in direction x
    dy     = ly/(ny-1)                                      # size of cell in direction y
    dz     = lz/(nz-1)                                      # size of cell in direction z
    _dx, _dy, _dz = 1.0/dx, 1.0/dy, 1.0/dz                  # 1/size of cells
    X_i    = range(-lx/2.0, lx/2.0, length=nx)              # coordinates of grid points in x
    Y_i    = range(-ly/2.0, ly/2.0, length=ny)              # coordinates of grid points in y
    Z_i    = range(-lz/2.0, lz/2.0, length=nz)              # coordinates of grid points in z
    X_i_vx = range(-(lx+dx)/2.0, (lx+dx)/2.0, length=nx+1)  # coordinates of grid points in r_vr
    Z_i_vz = range(-(lz+dz)/2.0, (lz+dz)/2.0, length=nz+1)  # coordinates of grid points in z_vz
    (X3D ,Y3D ,Z3D ) = ([xc for xc = X_i   ,yc = Y_i,zc = Z_i   ],[yc for xc = X_i   ,yc = Y_i,zc = Z_i   ],
                        [zc for xc = X_i   ,yc = Y_i,zc = Z_i   ])    # grid of coordinates
    (X_VX,Y_VX,Z_VX) = ([xc for xc = X_i_vx,yc = Y_i,zc = Z_i   ],[yc for xc = X_i_vx,yc = Y_i,zc = Z_i   ],
                        [zc for xc = X_i_vx,yc = Y_i,zc = Z_i   ])    # grid of coordinates
    (X_VZ,Y_VZ,Z_VZ) = ([xc for xc = X_i   ,yc = Y_i,zc = Z_i_vz],[yc for xc = X_i   ,yc = Y_i,zc = Z_i_vz],
                        [zc for xc = X_i   ,yc = Y_i,zc = Z_i_vz])    # grid of coordinates
    X3D  = Data.Array(X3D)
    Y3D  = Data.Array(Y3D)
    Z3D  = Data.Array(Z3D)
    X_VX = Data.Array(X_VX)
    Y_VX = Data.Array(Y_VX)
    Z_VX = Data.Array(Z_VX)
    X_VZ = Data.Array(X_VZ)
    Y_VZ = Data.Array(Y_VZ)
    Z_VZ = Data.Array(Z_VZ)
```

```
# initialization and boundary conditions --------------------------------
print("Starting initialization ... ")
P         =   @zeros(nx  , ny  , nz  )
VX        =   @zeros(nx+1, ny  , nz  )
VY        =   @zeros(nx  , ny+1, nz  )
VZ        =   @zeros(nx  , ny  , nz+1)
ERR_VX    =    @ones(nx+1, ny  , nz  )
ERR_VY    =    @ones(nx  , ny+1, nz  )
ERR_VZ    =    @ones(nx  , ny  , nz+1)
τ_XX      =   @zeros(nx  , ny  , nz  )
τ_YY      =   @zeros(nx  , ny  , nz  )
τ_ZZ      =   @zeros(nx  , ny  , nz  )
τ_XY      =   @zeros(nx-1, ny-1, nz-2)
τ_XZ      =   @zeros(nx-1, ny-2, nz-1)
τ_YZ      =   @zeros(nx-2, ny-1, nz-1)
ρ         = ρ0*@ones(nx  , ny  , nz  )
η         = η0*@ones(nx  , ny  , nz  )
η_INI     =   @zeros(nx  , ny  , nz  )
η_PL      =   @zeros(nx  , ny  , nz  )
η_PL_OLD  =   @zeros(nx  , ny  , nz  )
ηSM       =   @zeros(nx  , ny  , nz  )
DT_R      =   @zeros(nx  , ny  , nz  )
τII       =   @zeros(nx  , ny  , nz  )
err_evo   = []; iter_evo = []
iters     = 0.0
@parallel initialize_inclusion!(ρ, X3D, Y3D, Z3D, radius, ρ_in)
@parallel initialize_inclusion!(η, X3D, Y3D, Z3D, radius, η0/vr)
@parallel initialize_velocity!(VX, Y3D, sh)
vpdt = dx*CFL
@parallel equal!(η_INI, η)
@parallel equal!(η_PL , η)
# action ------------------------------------------------
print("Starting calculation ... \n")
runtime = @elapsed for it = 1:nt # time loop
    for iter = 1:maxiter # pseudo-transient loop
        # iteration strategy --------------------------------------
        @parallel equal!(ηSM, η)
        @parallel maxloc!(ηSM, η)
        @parallel copy_boundary!(ηSM)
        @parallel timesteps!(DT_R, η, ηSM, vpdt, dmp, lx)
        # SOLVER --------------------------------------------
        @parallel compute_P!(P , τ_XX, τ_YY, τ_ZZ, τ_XY, τ_XZ, τ_YZ, VX, VY, VZ, η, ηSM, dmp, vpdt,
                             lx, _dx , _dy , _dz )
        if n_exp > 1
            @parallel compute_TII!(τII, τ_XX, τ_YY, τ_ZZ, τ_XY, τ_XZ, τ_YZ)
            @parallel copy_boundary!(τII)
            @parallel equal!(η_PL_OLD, η_PL)
            @parallel power_law!(η_PL, η, τII, η_PL_OLD, η_INI, τ_C, n_exp, relax)
        end
        if mod(iter,nout) == 0
            @parallel equal!(ERR_VX, VX)
            @parallel equal!(ERR_VY, VY)
            @parallel equal!(ERR_VZ, VZ)
        end
        @parallel compute_V!(VX , VY, VZ , τ_XX, τ_YY, τ_ZZ, τ_XY, τ_XZ, τ_YZ, DT_R, P, ρ, η, ηSM, vpdt,
                             dmp, lx, _dx, _dy , _dz , g)
        @parallel bc_x_lin!(VX,Y_VX,sh)
        @parallel bc_x_0!(VY)
        @parallel bc_x_0!(VZ)
        @parallel bc_y_fact!(VX,Y_VX,sh)
        @parallel bc_y_0!(VY)
        @parallel bc_y_fact!(VZ,Y_VZ,0.0)
        @parallel copy_bc_z!(VX)
        @parallel copy_bc_z!(VY)
        @parallel bc_z_0!(VZ)
```

```
            # pseudo-transient loop exit criteria -----------------------------
            if mod(iter,nout) == 0
                @parallel error!(ERR_VX, VX)
                @parallel error!(ERR_VY, VY)
                @parallel error!(ERR_VZ, VZ)
                err_vx = maximum(ERR_VX[:])./maximum(abs.(VX[:]))
                err_vy = maximum(ERR_VY[:])./maximum(abs.(VY[:]))
                err_vz = maximum(ERR_VZ[:])./maximum(abs.(VZ[:]))
                err    = max(err_vx, err_vy, err_vz)
                if err<εnonl && iter>20
                    iters = iter
                    break
                end
                # postprocessing
                push!(err_evo, err)
                push!(iter_evo,iter)
                @printf("iter %d, err=%1.3e \n", iter, err)
            end
        end
    end
    # SAVING ------------------------------------------------
    if saveflag
        !ispath("./output") && mkdir("./output")
        err_evo  = Data.Array(err_evo)
        iter_evo = Data.Array(iter_evo)
        Save_phys(num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g, sh, radius, dmp)
        Save_infos(num, lx, ly, lz, nx, ny, nz, εnonl, runtime)
        SaveArray("X3D"     , X3D     )
        SaveArray("Y3D"     , Y3D     )
        SaveArray("Z3D"     , Z3D     )
        SaveArray("ETAS"    , η       )
        SaveArray("RHO"     , ρ       )
        SaveArray("P"       , P       )
        SaveArray("VX"      , VX      )
        SaveArray("VY"      , VY      )
        SaveArray("VZ"      , VZ      )
        SaveArray("TAU_XX"  , τ_XX    )
        SaveArray("TAU_YY"  , τ_YY    )
        SaveArray("TAU_ZZ"  , τ_ZZ    )
        SaveArray("TAU_XY"  , τ_XY    )
        SaveArray("TAU_XZ"  , τ_XZ    )
        SaveArray("TAU_YZ"  , τ_YZ    )
        SaveArray("TII"     , τII     )
        SaveArray("err_evo" , err_evo )
        SaveArray("iter_evo", iter_evo)
    end
    return
end

@time CART_3D_Diapir()
```

APPENDIX C

Code for the Diapir in Cylindrical Coordinates

```
# DIAPIR - CYLINDRICAL COORDINATES SYSTEM
const USE_GPU = true
const GPU_ID  = 0

saveflag      = true

using ParallelStencil
using ParallelStencil.FiniteDifferences3D
@static if USE_GPU
    @init_parallel_stencil(CUDA, Float64, 3)
    CUDA.device!(GPU_ID) # select GPU
else
    @init_parallel_stencil(Threads, Float64, 3)
end
using Plots, Printf, Statistics, LinearAlgebra

# SAVING FUNCTIONS ===========================================================================================
function Save_infos(num, lr, lθ, lz, nr, nθ, nz, εnonl, runtime; out="./output")
    fid=open(out * "/$(num)_infos.inf", "w")
    @printf(fid,"%d %f %f %f %d %d %d %d %d", num, lr, lθ, lz, nr, nθ, nz, εnonl, runtime); close(fid)
end

function Save_phys(num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g, sh, r, radius, dmp; out="./output")
    fid=open(out * "/$(num)_phys.inf", "w")
    @printf(fid,"%d  %f  %f  %f    %f     %f   %f  %f  %f  %f  %f      %f",
                num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g , sh, r , radius, dmp); close(fid)
end

@static if USE_GPU
    function SaveArray(Aname, A; out="./output")
        A_tmp = Array(A)
        fname = string(out, "/A_", Aname, ".bin");  fid = open(fname,"w"); write(fid, A_tmp); close(fid)
    end
else
    function SaveArray(Aname, A; out="./output")
        fname = string(out, "/A_", Aname, ".bin"); fid = open(fname,"w"); write(fid, A); close(fid)
    end
end

# ===========================================================================================================
@parallel function equal!(A::Data.Array, B::Data.Array)
    @all(A) = @all(B)
    return
end

@parallel function multiply!(A::Data.Array, B::Data.Array, fact::Data.Number)
    @all(A) = @all(B)*fact
    return
end

# BOUNDARY CONDITIONS ========================================================================================
@parallel_indices (ix,iy,iz) function bc_r_0!(A::Data.Array)
    if (ix==1         && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = -A[ix+1,iy,iz]    end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = -A[ix-1,iy,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function bc_θ_fact!(   A::Data.Array , COORD::Data.Array,
                                                 fact::Data.Number,     r::Data.Number)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = fact*COORD[ix,iy,iz]*r    end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = fact*COORD[ix,iy,iz]*r    end
    return
end

@parallel_indices (ix,iy,iz) function bc_θ_0!(A::Data.Array)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = -A[ix,iy+1,iz]    end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = -A[ix,iy-1,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function bc_z_0!(A::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1         )    A[ix,iy,iz] = 0.0    end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = 0.0    end
    return
end
```

```
@parallel_indices (ix,iy,iz) function bc_z_lin!(   A::Data.Array , COORD::Data.Array,
                                                fact::Data.Number,     r::Data.Number)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1        )    A[ix,iy,iz] = 2*COORD[ix,iy,iz  ]*r*fact -
                                                                                A[ix,iy,iz+1]    end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = 2*COORD[ix,iy,iz  ]*r*fact -
                                                                                A[ix,iy,iz-1]    end
    return
end

# COPY BOUNDARIES ===========================================================================================
@parallel_indices (ix,iy,iz) function copy_bc_r!(A::Data.Array)
    if (ix==1          && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix+1,iy,iz]    end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix-1,iy,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_θ!(A::Data.Array)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = A[ix,iy+1,iz]    end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix,iy-1,iz]    end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_z!(A::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1        )    A[ix,iy,iz] = A[ix,iy,iz+1]    end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = A[ix,iy,iz-1]    end
    return
end

# INITIALIZATION ============================================================================================
@parallel_indices (ix,iy,iz) function initialize_inclusion!( A::Data.Array ,      R::Data.Array,
                                                             θ::Data.Array ,      Z::Data.Array,
                                                             r::Data.Number, radius::Data.Number,
                                                            in::Data.Number)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))
        if (((R[ix,iy,iz]-r)^2 + (θ[ix,iy,iz]*r)^2 + Z[ix,iy,iz]^2) < radius)    A[ix,iy,iz] = in    end
    end
    return
end

@parallel_indices (ix,iy,iz) function initialize_velocity!(   V::Data.Array , COORD::Data.Array,
                                                           fact::Data.Number,     r::Data.Number)
    if (ix<=size(V,1) && iy==1         && iz<=size(V,3))    V[ix,iy,iz] = COORD[ix,iy,iz]*r*fact    end
    if (ix<=size(V,1) && iy==size(V,2) && iz<=size(V,3))    V[ix,iy,iz] = COORD[ix,iy,iz]*r*fact    end
    if (ix<=size(V,1) && iy<=size(V,2) && iz==1        )    V[ix,iy,iz] = COORD[ix,iy,iz]*r*fact    end
    if (ix<=size(V,1) && iy<=size(V,2) && iz==size(V,3))    V[ix,iy,iz] = COORD[ix,iy,iz]*r*fact    end
    return
end

# ITERATION STRATEGY ========================================================================================
@parallel function maxloc!(η_Max::Data.Array, η::Data.Array)
    @inn(η_Max) = @maxloc(η)
    return
end

@parallel function timesteps!(KBDT::Data.Array , GSDT::Data.Array , η_M::Data.Array,
                              dmp::Data.Number, vpdt::Data.Number,  lr::Data.Number)
    @all(KBDT) = dmp * 2.0 * pi * vpdt / lr * @all(η_M)
    @all(GSDT) =       4.0 * pi * vpdt / lr * @all(η_M)
    return
end

@parallel function timesteps2!(DT_R::Data.Array, KBDT::Data.Array, GSDT::Data.Array,
                                  η::Data.Array, vpdt::Data.Number)
    @all(DT_R) = vpdt^2 / (@all(KBDT) + @all(GSDT)/(1.0 + @all(GSDT)/@all(η)))
    return
end
```

```
# SOLVER ====================================================================================================
@parallel function compute_P!(D_RR::Data.Array , D_θθ::Data.Array , D_ZZ::Data.Array,
                              D_Rθ::Data.Array , D_RZ::Data.Array , D_θZ::Data.Array,
                              DIVV::Data.Array ,    P::Data.Array ,    R::Data.Array,
                              τ_RR::Data.Array , τ_θθ::Data.Array , τ_ZZ::Data.Array,
                              τ_Rθ::Data.Array , τ_RZ::Data.Array , τ_θZ::Data.Array,
                                VR::Data.Array ,   Vθ::Data.Array ,   VZ::Data.Array,
                              KBDT::Data.Array , GSDT::Data.Array ,    η::Data.Array,
                                dr::Data.Number,   dθ::Data.Number,   dz::Data.Number)
    @all(DIVV) = @d_xa(VR)/dr + 1/@all(R) * @d_ya(Vθ)/dθ + @av_xa(VR)/@all(R) + @d_za(VZ)/dz
    @all(D_RR) =                @d_xa(VR)/dr                      - 1.0/3.0 * @all(DIVV)
    @all(D_θθ) = 1/@all(R)    * @d_ya(Vθ)/dθ + @av_xa(VR)/@all(R) - 1.0/3.0 * @all(DIVV)
    @all(D_ZZ) =                @d_za(VZ)/dz                      - 1.0/3.0 * @all(DIVV)
    @all(D_Rθ) =                @d_xi(Vθ)/dr + 1/@av_xyi(R) * @d_yi(VR)/dθ - @av_xi(Vθ)/@av_xyi(R)
    @all(D_RZ) =                @d_xi(VZ)/dr +                @d_zi(VR)/dz
    @all(D_θZ) = 1/@av_yzi(R) * @d_yi(VZ)/dθ +                @d_zi(Vθ)/dz
    @all(P)    = @all(P)    - @all(KBDT) * @all(DIVV)
    @all(τ_RR) = (@all(τ_RR) +    @all(GSDT) * 2.0 * @all(D_RR))/(1.0 +    @all(GSDT) /    @all(η))
    @all(τ_θθ) = (@all(τ_θθ) +    @all(GSDT) * 2.0 * @all(D_θθ))/(1.0 +    @all(GSDT) /    @all(η))
    @all(τ_ZZ) = (@all(τ_ZZ) +    @all(GSDT) * 2.0 * @all(D_ZZ))/(1.0 +    @all(GSDT) /    @all(η))
    @all(τ_Rθ) = (@all(τ_Rθ) + @av_xyi(GSDT) *       @all(D_Rθ))/(1.0 + @av_xyi(GSDT) / @av_xyi(η))
    @all(τ_RZ) = (@all(τ_RZ) + @av_xzi(GSDT) *       @all(D_RZ))/(1.0 + @av_xzi(GSDT) / @av_xzi(η))
    @all(τ_θZ) = (@all(τ_θZ) + @av_yzi(GSDT) *       @all(D_θZ))/(1.0 + @av_yzi(GSDT) / @av_yzi(η))
    return
end

@parallel function compute_sigma!(σ_RR::Data.Array, σ_θθ::Data.Array, σ_ZZ::Data.Array, P::Data.Array,
                                  τ_RR::Data.Array, τ_θθ::Data.Array, τ_ZZ::Data.Array)
    @all(σ_RR) = -@all(P) + @all(τ_RR)
    @all(σ_θθ) = -@all(P) + @all(τ_θθ)
    @all(σ_ZZ) = -@all(P) + @all(τ_ZZ)
    return
end

@parallel function compute_TII!( τII::Data.Array, τ_RR::Data.Array, τ_θθ::Data.Array, τ_ZZ::Data.Array,
                                τ_Rθ::Data.Array, τ_RZ::Data.Array, τ_θZ::Data.Array)
    @inn(τII) = sqrt(1.0/2.0 * (@inn(τ_RR)^2 + @inn(τ_θθ)^2 + @inn(τ_ZZ)^2) +
                @av_xya(τ_Rθ)^2 + @av_xza(τ_RZ)^2 + @av_yza(τ_θZ)^2)
    return
end

@parallel function power_law!(    η_PL::Data.Array ,     η::Data.Array , τII::Data.Array ,
                              η_PL_OLD::Data.Array , η_INI::Data.Array , τ_C::Data.Number,
                                 n_exp::Data.Number, relax::Data.Number)
    @all(η_PL) = @all(η_INI) * (@all(τII)/τ_C)^(1.0-n_exp)
    @all(η_PL) = exp(log(@all(η_PL))*relax + log(@all(η_PL_OLD))*(1.0-relax))
    @all(η)    = 2.0/(1.0/@all(η_INI) + 1.0/@all(η_PL))
    return
end

@parallel function compute_dV!( dVR::Data.Array ,  dVθ::Data.Array ,  dVZ::Data.Array ,
                               σ_RR::Data.Array , σ_θθ::Data.Array , σ_ZZ::Data.Array ,
                               τ_Rθ::Data.Array , τ_RZ::Data.Array , τ_θZ::Data.Array ,
                                  R::Data.Array ,   ρG::Data.Array ,
                                 dr::Data.Number,   dθ::Data.Number,   dz::Data.Number)
    @all(dVR) = @d_xi(σ_RR)/dr + 1/@av_xi(R) * @d_ya(τ_Rθ)/dθ + @d_za(τ_RZ)/dz +
                    @av_xi(σ_RR)/@av_xi(R) - @av_xi(σ_θθ)/@av_xi(R) - @av_xi(ρG)
    @all(dVθ) = @d_xa(τ_Rθ)/dr + 1/@av_yi(R) * @d_yi(σ_θθ)/dθ + @d_za(τ_θZ)/dz +
                2 * @av_xa(τ_Rθ)/@av_yi(R)
    @all(dVZ) = @d_xa(τ_RZ)/dr + 1/@av_zi(R) * @d_ya(τ_θZ)/dθ + @d_zi(σ_ZZ)/dz +
                    @av_xa(τ_RZ)/@av_zi(R)
    return
end

@parallel function compute_V!( VR::Data.Array,  Vθ::Data.Array,  VZ::Data.Array,
                              dVR::Data.Array, dVθ::Data.Array, dVZ::Data.Array, DT_R::Data.Array)
    @inn(VR) = @inn(VR) + @all(dVR)*@av_xi(DT_R)
    @inn(Vθ) = @inn(Vθ) + @all(dVθ)*@av_yi(DT_R)
    @inn(VZ) = @inn(VZ) + @all(dVZ)*@av_zi(DT_R)
    return
end
```

```
# CHECK ERROR =================================================================================================
@parallel function err_rθ!(err_rθ::Data.Array, τ_Rθ::Data.Array, D_Rθ::Data.Array, η::Data.Array)
    @all(err_rθ) = @all(τ_Rθ) - @all(D_Rθ)*@av_xyi(η)
    return
end

# ============================================================================================================
@views function CYL_3D_Diapir()
    num     = 1
    runtime = 0.0
    # physical parameters --------------------------------------------
    η0          = 1.0                       # Pa*s  , media viscosity
    vr          = 1e2                       #         viscosity ratio out/in
    ρ0          = 0.0                       # kg/m^3, media density
    ρ_in        = ρ0 - 10.0                 # kg/m^3, inclusion density
    g           = 1.0                       # m/s^2 , gravity acceleration
    n_exp       = 1.0                       #         power law exponent
    n_exp_PL    = 5.0                       #         power-law exponent
    τ_C         = 1.0                       # Pa    , xharacteristic stress
    relax       = 1e-3                      #         relaxation parameter (power law)
    r           = 1000.0                    # m     , radius of the total cylinder
    lr          = 6.0                       # m     , model dimension in r
    lθ          = lr/r                      # m     , model dimension in θ
    lz          = 1.0*lr                    # m     , model dimension in z
    radius      = 1.0                       # m     , radius of the inclusion
    sh          = 1.0                       # m/s   , shearing velocity
    # numerics ------------------------------------------------
    nr          = 208 - 1                   # number of grid cells in direction r
    nθ          = 208 - 1                   # number of grid cells in direction θ
    nz          = 208 - 1                   # number of grid cells in direction z
    εnonl       = 5e-7                      # pseudo-transient loop exit criteria
    nt          = 1                         # number of time steps
    maxiter     = 1e5                       # maximum number of pseudo-transient iterations
    nout        = 1e2                       # pseudo-transient plotting frequency
    CFL         = 1.0/(2.0 + 4.5*log10(vr)) # Courant-Friedrichs-Lewy condition
    dmp         = 4.5                       # damping parameter
    # preprocessing ----------------------------------------------
    dr     = lr/(nr-1)                                              # size of cell in direction r
    dθ     = lθ/(nθ-1)                                              # size of cell in direction θ
    dz     = lz/(nz-1)                                              # size of cell in direction z
    R_i    = range(r- lr    /2.0, r+ lr    /2.0, length=nr  )     # coordinates of grid points in r
    θ_i    = range( - lθ    /2.0,    lθ    /2.0, length=nθ  )     # coordinates of grid points in θ
    Z_i    = range( - lz    /2.0,    lz    /2.0, length=nz  )     # coordinates of grid points in z
    R_i_vr = range(r-(lr+dr)/2.0, r+(lr+dr)/2.0, length=nr+1)     # coordinates of grid points in r_vr
    Z_i_vz = range( -(lz+dz)/2.0,   (lz+dz)/2.0, length=nz+1)     # coordinates of grid points in z_vz
    (R   ,θ   ,Z   ) = ([xc for xc = R_i   ,yc = θ_i,zc = Z_i   ],[yc for xc = R_i   ,yc = θ_i,zc = Z_i   ],
                        [zc for xc = R_i   ,yc = θ_i,zc = Z_i   ])    # grid of coordinates
    (R_VR,θ_VR,Z_VR) = ([xc for xc = R_i_vr,yc = θ_i,zc = Z_i   ],[yc for xc = R_i_vr,yc = θ_i,zc = Z_i   ],
                        [zc for xc = R_i_vr,yc = θ_i,zc = Z_i   ])    # grid of coordinates
    (R_VZ,θ_VZ,Z_VZ) = ([xc for xc = R_i   ,yc = θ_i,zc = Z_i_vz],[yc for xc = R_i   ,yc = θ_i,zc = Z_i_vz],
                        [zc for xc = R_i   ,yc = θ_i,zc = Z_i_vz])    # grid of coordinates
    R                = Data.Array(R   )
    θ                = Data.Array(θ   )
    Z                = Data.Array(Z   )
    R_VR             = Data.Array(R_VR)
    θ_VR             = Data.Array(θ_VR)
    Z_VR             = Data.Array(Z_VR)
    R_VZ             = Data.Array(R_VZ)
    θ_VZ             = Data.Array(θ_VZ)
    Z_VZ             = Data.Array(Z_VZ)
```

```
# initialization and boundary conditions --------------------------------
print("Starting initialization ... ")
P           =   @zeros(nr  , nθ  , nz  )
DIVV        =   @zeros(nr  , nθ  , nz  )
VR          =   @zeros(nr+1, nθ  , nz  )
Vθ          =   @zeros(nr  , nθ+1, nz  )
VZ          =   @zeros(nr  , nθ  , nz+1)
dVR         =   @zeros(nr-1, nθ-2, nz-2)
dVθ         =   @zeros(nr-2, nθ-1, nz-2)
dVZ         =   @zeros(nr-2, nθ-2, nz-1)
VR_OLD      =   @zeros(nr+1, nθ  , nz  )
Vθ_OLD      =   @zeros(nr  , nθ+1, nz  )
VZ_OLD      =   @zeros(nr  , nθ  , nz+1)
D_RR        =   @zeros(nr  , nθ  , nz  )
D_θθ        =   @zeros(nr  , nθ  , nz  )
D_ZZ        =   @zeros(nr  , nθ  , nz  )
D_Rθ        =   @zeros(nr-1, nθ-1, nz-2)
D_RZ        =   @zeros(nr-1, nθ-2, nz-1)
D_θZ        =   @zeros(nr-2, nθ-1, nz-1)
τ_RR        =   @zeros(nr  , nθ  , nz  )
τ_θθ        =   @zeros(nr  , nθ  , nz  )
τ_ZZ        =   @zeros(nr  , nθ  , nz  )
τ_Rθ        =   @zeros(nr-1, nθ-1, nz-2)
τ_RZ        =   @zeros(nr-1, nθ-2, nz-1)
τ_θZ        =   @zeros(nr-2, nθ-1, nz-1)
σ_RR        =   @zeros(nr  , nθ  , nz  )
σ_θθ        =   @zeros(nr  , nθ  , nz  )
σ_ZZ        =   @zeros(nr  , nθ  , nz  )
ρ           = ρ0*@ones(nr  , nθ  , nz  )
ρG          =   @zeros(nr  , nθ  , nz  )
η           = η0*@ones(nr  , nθ  , nz  )
η_INI       =   @zeros(nr  , nθ  , nz  )
η_PL        =   @zeros(nr  , nθ  , nz  )
η_PL_OLD    =   @zeros(nr  , nθ  , nz  )
ηSM         =   @zeros(nr  , nθ  , nz  )
KBDT        =   @zeros(nr  , nθ  , nz  )
GSDT        =   @zeros(nr  , nθ  , nz  )
DT_R        =   @zeros(nr  , nθ  , nz  )
τII         =   @zeros(nr  , nθ  , nz  )
err_evo     = []; iter_evo = []
iters       = 0.0
ERR_Rθ      = @zeros(nr-1, nθ-1, nz-2)
@parallel initialize_inclusion!(ρ, R, θ, Z, r, radius, ρ_in)
@parallel initialize_inclusion!(η, R, θ, Z, r, radius, η0/vr)
@parallel multiply!(ρG, ρ, g)
@parallel initialize_velocity!(VZ, θ_VZ, sh, r)
vpdt        = dr*CFL
err         = 1.0
@parallel equal!(η_INI, η)
@parallel equal!(η_PL , η)
# action ------------------------------------------------
print("Starting calculation ... \n")
runtime = @elapsed for it = 1:nt # time loop
    for iter = 1:maxiter # pseudo-transient loop
        # iteration strategy --------------------------------------
        @parallel equal!(ηSM, η)
        @parallel maxloc!(ηSM, η)
        @parallel copy_bc_r!(ηSM)
        @parallel copy_bc_θ!(ηSM)
        @parallel copy_bc_z!(ηSM)
        @parallel timesteps!(KBDT, GSDT, ηSM, dmp, vpdt, lr)
        @parallel timesteps2!(DT_R, KBDT, GSDT, η, vpdt)
        # SOLVER --------------------------------------------
        @parallel compute_P!(D_RR, D_θθ, D_ZZ, D_Rθ, D_RZ, D_θZ, DIVV, P   , R, τ_RR, τ_θθ, τ_ZZ,
                             τ_Rθ, τ_RZ, τ_θZ, VR  , Vθ  , VZ  , KBDT, GSDT, η, dr  , dθ  , dz  )
        @parallel compute_sigma!(σ_RR, σ_θθ, σ_ZZ, P, τ_RR, τ_θθ, τ_ZZ)
        if err < 1e-5
            n_exp = n_exp_PL
        end
        if n_exp > 1
            @parallel compute_TII!(τII, τ_RR, τ_θθ, τ_ZZ, τ_Rθ, τ_RZ, τ_θZ)
            @parallel copy_bc_r!(τII)
            @parallel copy_bc_θ!(τII)
            @parallel copy_bc_z!(τII)
            @parallel equal!(η_PL_OLD, η_PL)
```

```
                @parallel power_law!(η_PL, η, τII, η_PL_OLD, η_INI, τ_C, n_exp, relax)
            end
            @parallel equal!(VR_OLD, VR)
            @parallel equal!(Vθ_OLD, Vθ)
            @parallel equal!(VZ_OLD, VZ)
            @parallel compute_dV!(dVR, dVθ, dVZ, σ_RR, σ_θθ, σ_ZZ, τ_Rθ, τ_RZ, τ_θZ, R, ρG, dr, dθ, dz)
            @parallel compute_V!(VR, Vθ, VZ, dVR, dVθ, dVZ, DT_R)
            @parallel copy_bc_r!(VZ)
            @parallel copy_bc_r!(Vθ)
            @parallel bc_r_0!(VR)
            @parallel bc_θ_fact!(VZ,θ_VZ,sh,r)
            @parallel bc_θ_fact!(VR,θ_VR,0.0,r)
            @parallel bc_θ_0!(Vθ)
            @parallel bc_z_0!(Vθ)
            @parallel bc_z_0!(VR)
            @parallel bc_z_lin!(VZ,θ_VZ,sh,r)
            # pseudo-transient loop exit criteria ------------------------------
            if mod(iter,nout) == 0
                err_vr = maximum(abs.(VR[:].-VR_OLD[:]))./maximum(abs.(VR[:]))
                err_vθ = maximum(abs.(Vθ[:].-Vθ_OLD[:]))./maximum(abs.(Vθ[:]))
                err_vz = maximum(abs.(VZ[:].-VZ_OLD[:]))./maximum(abs.(VZ[:]))
                err    = max(err_vr, err_vθ, err_vz)
                @parallel err_rθ!(ERR_Rθ, τ_Rθ, D_Rθ, η)
                err_rθ = maximum(abs.(ERR_Rθ[:]))
                if err<εnonl && iter>20
                    iters = iter
                    break
                end
                # post-processing
                push!(err_evo, err)
                push!(iter_evo,iter)
                @printf("iter %d, err=%1.3e, err_rθ=%1.3e \n", iter, err, err_rθ)
            end
        end
    end
    # SAVING ------------------------------------------------
    if saveflag
        !ispath("./output") && mkdir("./output")
        err_evo  = Data.Array(err_evo)
        iter_evo = Data.Array(iter_evo)
        Save_phys(num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g, sh, r, radius, dmp)
        Save_infos(num, lr, lθ, lz, nr, nθ, nz, εnonl, runtime)
        SaveArray("R"       , R       )
        SaveArray("TH"      , θ       )
        SaveArray("Z"       , Z       )
        SaveArray("ETAS"    , η       )
        SaveArray("RHO"     , ρ       )
        SaveArray("RHOG"    , ρG      )
        SaveArray("P"       , P       )
        SaveArray("DIVV"    , DIVV    )
        SaveArray("VR"      , VR      )
        SaveArray("VTH"     , Vθ      )
        SaveArray("VZ"      , VZ      )
        SaveArray("D_RR"    , D_RR    )
        SaveArray("D_THTH"  , D_θθ    )
        SaveArray("D_ZZ"    , D_ZZ    )
        SaveArray("D_RTH"   , D_Rθ    )
        SaveArray("D_RZ"    , D_RZ    )
        SaveArray("D_THZ"   , D_θZ    )
        SaveArray("TAU_RR"  , τ_RR    )
        SaveArray("TAU_THTH", τ_θθ    )
        SaveArray("TAU_ZZ"  , τ_ZZ    )
        SaveArray("TAU_RTH" , τ_Rθ    )
        SaveArray("TAU_RZ"  , τ_RZ    )
        SaveArray("TAU_THZ" , τ_θZ    )
        SaveArray("TII"     , τII     )
        SaveArray("err_evo" , err_evo )
        SaveArray("iter_evo", iter_evo)
    end
    return
end

@time CYL_3D_Diapir()
```

APPENDIX D

Code for the Diapir in Spherical Coordinates

```
# DIAPIR - SPHERICAL COORDINATES SYSTEM
const USE_GPU   = haskey(ENV, "USE_GPU"  ) ? parse(Bool, ENV["USE_GPU"]  ) : true#false
const do_viz    = haskey(ENV, "DO_VIZ"   ) ? parse(Bool, ENV["DO_VIZ"]   ) : false
const do_save   = haskey(ENV, "DO_SAVE"  ) ? parse(Bool, ENV["DO_SAVE"]  ) : true
const do_save_p = haskey(ENV, "DO_SAVE_P") ? parse(Bool, ENV["DO_SAVE_P"]) : false
const nr        = haskey(ENV, "NR"       ) ? parse(Int , ENV["NR"]       ) : 208 - 1
const nθ        = haskey(ENV, "NTH"      ) ? parse(Int , ENV["NTH"]      ) : 208 - 1
const nφ        = haskey(ENV, "NPH"      ) ? parse(Int , ENV["NPH"]      ) : 208 - 1
const GPU_ID    = haskey(ENV, "GPU_ID"   ) ? parse(Int , ENV["GPU_ID"]   ) : 0
const dmp       = haskey(ENV, "DMP"      ) ? parse(Float64, ENV["DMP"]   ) : 4.5

using ParallelStencil
using ParallelStencil.FiniteDifferences3D
@static if USE_GPU
    @init_parallel_stencil(CUDA, Float64, 3)
    CUDA.device!(GPU_ID) # select GPU
else
    @init_parallel_stencil(Threads, Float64, 3)
end
using Plots, Printf, Statistics, LinearAlgebra

# SAVING FUNCTIONS ==========================================================================================
function Save_infos(num, lr, lθ, lz, nr, nθ, nz, εnonl, runtime; out="./output")
    fid=open(out * "/$(num)_infos.inf", "w")
    @printf(fid,"%d %f %f %f %d %d %d %d %d", num, lr, lθ, lz, nr, nθ, nz, εnonl, runtime); close(fid)
end

function Save_phys(num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g, sh, r, radius, dmp; out="./output")
    fid=open(out * "/$(num)_phys.inf", "w")
    @printf(fid,"%d   %f  %f  %f    %f     %f   %f  %f, %f, %f, %f    , %f" ,
                 num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g , sh, r , radius, dmp); close(fid)
end

@static if USE_GPU
    function SaveArray(Aname, A; out="./output")
        A_tmp = Array(A)
        fname = string(out, "/A_", Aname, ".bin");  fid = open(fname,"w"); write(fid, A_tmp); close(fid)
    end
else
    function SaveArray(Aname, A; out="./output")
        fname = string(out, "/A_", Aname, ".bin"); fid = open(fname,"w"); write(fid, A); close(fid)
    end
end

# =========================================================================================================
@parallel_indices (ix,iy,iz) function equal3!(VR_err::Data.Array, Vθ_err::Data.Array, Vφ_err::Data.Array,
                                                 VR::Data.Array,     Vθ::Data.Array,     Vφ::Data.Array)
    if (ix<=size(VR_err,1) && iy<=size(VR_err,2) && iz<=size(VR_err,3))
        VR_err[ix,iy,iz] = VR[ix,iy,iz]
    end
    if (ix<=size(Vθ_err,1) && iy<=size(Vθ_err,2) && iz<=size(Vθ_err,3))
        Vθ_err[ix,iy,iz] = Vθ[ix,iy,iz]
    end
    if (ix<=size(Vφ_err,1) && iy<=size(Vφ_err,2) && iz<=size(Vφ_err,3))
        Vφ_err[ix,iy,iz] = Vφ[ix,iy,iz]
    end
    return
end

# BOUNDARY CONDITIONS =======================================================================================
@parallel_indices (ix,iy,iz) function BC_R!(VR::Data.Array, Vθ::Data.Array, Vφ::Data.Array)
    if (ix==1          && iy<=size(VR,2) && iz<=size(VR,3)) VR[ix,iy,iz] = -VR[ix+1,iy,iz]  end
    if (ix==size(VR,1) && iy<=size(VR,2) && iz<=size(VR,3)) VR[ix,iy,iz] = -VR[ix-1,iy,iz]  end
    if (ix==1          && iy<=size(Vθ,2) && iz<=size(Vθ,3)) Vθ[ix,iy,iz] =  Vθ[ix+1,iy,iz]  end
    if (ix==size(Vθ,1) && iy<=size(Vθ,2) && iz<=size(Vθ,3)) Vθ[ix,iy,iz] =  Vθ[ix-1,iy,iz]  end
    if (ix==1          && iy<=size(Vφ,2) && iz<=size(Vφ,3)) Vφ[ix,iy,iz] =  Vφ[ix+1,iy,iz]  end
    if (ix==size(Vφ,1) && iy<=size(Vφ,2) && iz<=size(Vφ,3)) Vφ[ix,iy,iz] =  Vφ[ix-1,iy,iz]  end
    return
end
```

```
@parallel_indices (ix,iy,iz) function BC_θ!(VR::Data.Array, Vθ::Data.Array , Vφ::Data.Array,
                                            θ::Data.Array,  r::Data.Number, sh::Data.Number)
    if (ix<=size(VR,1) && iy==1          && iz<=size(VR,3)  )   VR[ix,iy,iz] =  0.0                 end
    if (ix<=size(VR,1) && iy==size(VR,2) && iz<=size(VR,3)  )   VR[ix,iy,iz] =  0.0                 end
    if (ix<=size(Vθ,1) && iy==1          && iz<=size(Vθ,3)  )   Vθ[ix,iy,iz] = -Vθ[ix,iy+1,iz]      end
    if (ix<=size(Vθ,1) && iy==size(Vθ,2) && iz<=size(Vθ,3)  )   Vθ[ix,iy,iz] = -Vθ[ix,iy-1,iz]      end
    if (ix<=size(Vφ,1) && iy==1          && iz<=size(Vφ,3)-1)   Vφ[ix,iy,iz] = ( θ[ix,iy,iz  ] -
                                                                                pi/2.0)*r*sh       end
    if (ix<=size(Vφ,1) && iy==1          && iz==size(Vφ,3)  )   Vφ[ix,iy,iz] = ( θ[ix,iy,iz-1] -
                                                                                pi/2.0)*r*sh       end
    if (ix<=size(Vφ,1) && iy==size(Vφ,2) && iz<=size(Vφ,3)-1)   Vφ[ix,iy,iz] = ( θ[ix,iy,iz  ] -
                                                                                pi/2.0)*r*sh       end
    if (ix<=size(Vφ,1) && iy==size(Vφ,2) && iz==size(Vφ,3)  )   Vφ[ix,iy,iz] = ( θ[ix,iy,iz-1] -
                                                                                pi/2.0)*r*sh       end
    return
end

@parallel_indices (ix,iy,iz) function BC_φ!(VR::Data.Array, Vθ::Data.Array , Vφ::Data.Array,
                                            θ::Data.Array,  r::Data.Number, sh::Data.Number)
    if (ix<=size(VR,1) && iy<=size(VR,2) && iz==1         ) VR[ix,iy,iz] = 0.0                             end
    if (ix<=size(VR,1) && iy<=size(VR,2) && iz==size(VR,3)) VR[ix,iy,iz] = 0.0                             end
    if (ix<=size(Vθ,1) && iy<=size(Vθ,2) && iz==1         ) Vθ[ix,iy,iz] = 0.0                             end
    if (ix<=size(Vθ,1) && iy<=size(Vθ,2) && iz==size(Vθ,3)) Vθ[ix,iy,iz] = 0.0                             end
    if (ix<=size(Vφ,1) && iy<=size(Vφ,2) && iz==1         ) Vφ[ix,iy,iz] = 2*(θ[ix,iy,iz  ] -
                                                                          pi/2.0)*r*sh - Vφ[ix,iy,iz+1]   end
    if (ix<=size(Vφ,1) && iy<=size(Vφ,2) && iz==size(Vφ,3)) Vφ[ix,iy,iz] = 2*(θ[ix,iy,iz-1] -
                                                                          pi/2.0)*r*sh - Vφ[ix,iy,iz-1]   end
    return
end

# COPY BOUNDARIES ==========================================================================================
@parallel_indices (ix,iy,iz) function copy_bc_r!(A::Data.Array)
    if (ix==1         && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix+1,iy,iz] end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix-1,iy,iz] end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_θ!(A::Data.Array)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = A[ix,iy+1,iz] end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix,iy-1,iz] end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_φ!(A::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1        )    A[ix,iy,iz] = A[ix,iy,iz+1] end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = A[ix,iy,iz-1] end
    return
end

function copy_BC!(A::Data.Array)
    @parallel copy_bc_r!(A)
    @parallel copy_bc_θ!(A)
    @parallel copy_bc_φ!(A)
end

# INITIALIZATION ===========================================================================================
@parallel_indices (ix,iy,iz) function initialize_inclusion!( A::Data.Array ,      R::Data.Array ,
                                                             θ::Data.Array ,      φ::Data.Array,
                                                             r::Data.Number, radius::Data.Number,
                                                            in::Data.Number)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))
        if (((R[ix,iy,iz]-r)^2 + ((θ[ix,iy,iz]-pi/2.0)*r)^2 + (φ[ix,iy,iz]*r)^2) < radius)
            A[ix,iy,iz] = in
        end
    end
    return
end
```

```
@parallel_indices (ix,iy,iz) function initialize_velocity!(   V::Data.Array, COORD::Data.Array,
                                                           fact::Data.Number,    r::Data.Number)
    if (ix<=size(V,1) && iy<=size(V,2) && iz<=size(V,3)-1)  V[ix,iy,iz] = (COORD[ix,iy,iz  ] -
                                                                           pi/2.0)*r*fact   end
    if (ix<=size(V,1) && iy<=size(V,2) && iz==size(V,3)  )  V[ix,iy,iz] = (COORD[ix,iy,iz-1] -
                                                                           pi/2.0)*r*fact   end

    return
end

# =============================================================================================================
@parallel function maxloc!(ηSM::Data.Array, η::Data.Array)
    @inn(ηSM) = @maxloc(η)
    return
end

# SOLVER =====================================================================================================
macro KBDT(ix,iy,iz)        esc(:(  dmp * 2.0 * pi * vpdt / lr * ηSM[$ix,$iy,$iz]   ))                     end
macro GSDT(ix,iy,iz)        esc(:(        4.0 * pi * vpdt / lr * ηSM[$ix,$iy,$iz]   ))                     end
macro avxa_VR(ix,iy,iz)     esc(:(  ((   VR[$ix  ,$iy  ,$iz  ] +    VR[$ix+1,$iy  ,$iz  ])*0.5) ))         end
macro avxi_Vθ(ix,iy,iz)     esc(:(  ((   Vθ[$ix  ,$iy+1,$iz+1] +    Vθ[$ix+1,$iy+1,$iz+1])*0.5) ))         end
macro avya_Vθ(ix,iy,iz)     esc(:(  ((   Vθ[$ix  ,$iy  ,$iz  ] +    Vθ[$ix  ,$iy+1,$iz  ])*0.5) ))         end
macro avxi_Vφ(ix,iy,iz)     esc(:(  ((   Vφ[$ix  ,$iy+1,$iz+1] +    Vφ[$ix+1,$iy+1,$iz+1])*0.5) ))         end
macro avyi_Vφ(ix,iy,iz)     esc(:(  ((   Vφ[$ix+1,$iy  ,$iz+1] +    Vφ[$ix+1,$iy+1,$iz+1])*0.5) ))         end
macro avxyi_R(ix,iy,iz)     esc(:(  ((    R[$ix  ,$iy  ,$iz+1] +     R[$ix  ,$iy+1,$iz+1] +
                                          R[$ix+1,$iy  ,$iz+1] +     R[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avxzi_R(ix,iy,iz)     esc(:(  ((    R[$ix  ,$iy+1,$iz  ] +     R[$ix  ,$iy+1,$iz+1] +
                                          R[$ix+1,$iy+1,$iz  ] +     R[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avyzi_R(ix,iy,iz)     esc(:(  ((    R[$ix+1,$iy  ,$iz  ] +     R[$ix+1,$iy  ,$iz+1] +
                                          R[$ix+1,$iy+1,$iz  ] +     R[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avxzi_θ(ix,iy,iz)     esc(:(  ((    θ[$ix  ,$iy+1,$iz  ] +     θ[$ix  ,$iy+1,$iz+1] +
                                          θ[$ix+1,$iy+1,$iz  ] +     θ[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avyzi_θ(ix,iy,iz)     esc(:(  ((    θ[$ix+1,$iy  ,$iz  ] +     θ[$ix+1,$iy  ,$iz+1] +
                                          θ[$ix+1,$iy+1,$iz  ] +     θ[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avxyi_GSDT(ix,iy,iz)  esc(:(  ((@GSDT($ix  ,$iy  ,$iz+1) + @GSDT($ix  ,$iy+1,$iz+1) +
                                      @GSDT($ix+1,$iy  ,$iz+1) + @GSDT($ix+1,$iy+1,$iz+1))*0.25)  ))       end
macro avxzi_GSDT(ix,iy,iz)  esc(:(  ((@GSDT($ix  ,$iy+1,$iz  ) + @GSDT($ix  ,$iy+1,$iz+1) +
                                      @GSDT($ix+1,$iy+1,$iz  ) + @GSDT($ix+1,$iy+1,$iz+1))*0.25)  ))       end
macro avyzi_GSDT(ix,iy,iz)  esc(:(  ((@GSDT($ix+1,$iy  ,$iz  ) + @GSDT($ix+1,$iy  ,$iz+1) +
                                      @GSDT($ix+1,$iy+1,$iz  ) + @GSDT($ix+1,$iy+1,$iz+1))*0.25)  ))       end
macro avxyi_η(ix,iy,iz)     esc(:(  ((    η[$ix  ,$iy  ,$iz+1] +     η[$ix  ,$iy+1,$iz+1] +
                                          η[$ix+1,$iy  ,$iz+1] +     η[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avxzi_η(ix,iy,iz)     esc(:(  ((    η[$ix  ,$iy+1,$iz  ] +     η[$ix  ,$iy+1,$iz+1] +
                                          η[$ix+1,$iy+1,$iz  ] +     η[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro avyzi_η(ix,iy,iz)     esc(:(  ((    η[$ix+1,$iy  ,$iz  ] +     η[$ix+1,$iy  ,$iz+1] +
                                          η[$ix+1,$iy+1,$iz  ] +     η[$ix+1,$iy+1,$iz+1])*0.25)  ))       end
macro DIVV(ix,iy,iz)        esc(:((                     VR[$ix+1,$iy,$iz] - VR[$ix,$iy,$iz])*_dr +
                                    1.0/R[$ix,$iy,$iz]*(Vθ[$ix,$iy+1,$iz] - Vθ[$ix,$iy,$iz])*_dθ +
                                    1.0/R[$ix,$iy,$iz]/sin(θ[$ix,$iy,$iz])*
                                                       (Vφ[$ix,$iy,$iz+1] - Vφ[$ix,$iy,$iz])*_dφ +
                                    2.0*@avxa_VR(ix,iy,iz)/R[$ix,$iy,$iz] +
                                    @avya_Vθ(ix,iy,iz)/R[$ix,$iy,$iz]*cot(θ[$ix,$iy,$iz])   ))                end
macro DRR(ix,iy,iz)         esc(:(  (VR[$ix+1,$iy,$iz] - VR[$ix,$iy,$iz])*_dr - 1.0/3.0*@DIVV(ix,iy,iz) )) end
macro Dθθ(ix,iy,iz)         esc(:(  1.0/R[$ix,$iy,$iz]*(Vθ[$ix,$iy+1,$iz] - Vθ[$ix,$iy,$iz])*_dθ +
                                    @avxa_VR(ix,iy,iz)/R[$ix,$iy,$iz] - 1.0/3.0*@DIVV(ix,iy,iz) ))        end
macro Dφφ(ix,iy,iz)         esc(:(  1.0/R[$ix,$iy,$iz]/sin(θ[$ix,$iy,$iz])*
                                    (Vφ[$ix,$iy,$iz+1] - Vφ[$ix,$iy,$iz])*_dφ +
                                    @avxa_VR(ix,iy,iz)/R[$ix,$iy,$iz] +
                                    @avya_Vθ($ix,$iy,$iz)/R[$ix,$iy,$iz]*cot(θ[$ix,$iy,$iz]) -
                                    1.0/3.0*@DIVV(ix,iy,iz)  ))                                           end
macro DRθ(ix,iy,iz)         esc(:(                          (Vθ[$ix+1,$iy+1,$iz+1] - Vθ[$ix,$iy+1,$iz+1])*_dr +
                                    1.0/@avxyi_R(ix,iy,iz)*(VR[$ix+1,$iy+1,$iz+1] - VR[$ix+1,$iy,$iz+1])*_dθ -
                                    @avxi_Vθ(ix,iy,iz)/@avxyi_R(ix,iy,iz)   ))                            end
macro DRφ(ix,iy,iz)         esc(:(  (Vφ[$ix+1,$iy+1,$iz+1] - Vφ[$ix,$iy+1,$iz+1])*_dr +
                                    1.0/@avxzi_R(ix,iy,iz)/sin(@avxzi_θ(ix,iy,iz))*
                                    (VR[$ix+1,$iy+1,$iz+1] - VR[$ix+1,$iy+1,$iz])*_dφ -
                                    @avxi_Vφ(ix,iy,iz)/@avxzi_R(ix,iy,iz)   ))                            end
macro Dθφ(ix,iy,iz)         esc(:(  1.0/@avyzi_R(ix,iy,iz)*(Vφ[$ix+1,$iy+1,$iz+1] - Vφ[$ix+1,$iy,$iz+1])*_dθ+
                                    1.0/@avyzi_R(ix,iy,iz)/sin(@avyzi_θ(ix,iy,iz))*
                                    (Vθ[$ix+1,$iy+1,$iz+1] - Vθ[$ix+1,$iy+1,$iz])*_dφ -
                                    @avyi_Vφ(ix,iy,iz)/@avyzi_R(ix,iy,iz)*cot(@avyzi_θ(ix,iy,iz))  ))     end
```

```
@parallel_indices (ix,iy,iz) function compute_P!(   P::Data.Array , DT_R::Data.Array ,
                                                 τ_RR::Data.Array , τ_θθ::Data.Array , τ_φφ::Data.Array ,
                                                 τ_Rθ::Data.Array , τ_Rφ::Data.Array , τ_θφ::Data.Array ,
                                                   VR::Data.Array ,   Vθ::Data.Array ,   Vφ::Data.Array ,
                                                    R::Data.Array ,    θ::Data.Array ,    η::Data.Array ,
                                                  ηSM::Data.Array,
                                                  _dr::Data.Number,  _dθ::Data.Number,  _dφ::Data.Number,
                                                  dmp::Data.Number, vpdt::Data.Number,   lr::Data.Number)
    if (ix<=size(P,1) && iy<=size(P,2) && iz<=size(P,3))
        P[ix,iy,iz] = P[ix,iy,iz] - @KBDT(ix,iy,iz) * @DIVV(ix,iy,iz)
        τ_RR[ix,iy,iz] = (τ_RR[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@DRR(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])
        τ_θθ[ix,iy,iz] = (τ_θθ[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@Dθθ(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])
        τ_φφ[ix,iy,iz] = (τ_φφ[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@Dφφ(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])
        DT_R[ix,iy,iz] = vpdt^2 / (@KBDT(ix,iy,iz) + @GSDT(ix,iy,iz)/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz]))
    end
    if (ix<=size(τ_Rθ,1) && iy<=size(τ_Rθ,2) && iz<=size(τ_Rθ,3))
        τ_Rθ[ix,iy,iz] = (τ_Rθ[ix,iy,iz] + @avxyi_GSDT(ix,iy,iz)*@DRθ(ix,iy,iz))/
                                    (1.0 + @avxyi_GSDT(ix,iy,iz)/@avxyi_η(ix,iy,iz))
    end
    if (ix<=size(τ_Rφ,1) && iy<=size(τ_Rφ,2) && iz<=size(τ_Rφ,3))
        τ_Rφ[ix,iy,iz] = (τ_Rφ[ix,iy,iz] + @avxzi_GSDT(ix,iy,iz)*@DRφ(ix,iy,iz))/
                                    (1.0 + @avxzi_GSDT(ix,iy,iz)/@avxzi_η(ix,iy,iz))
    end
    if (ix<=size(τ_θφ,1) && iy<=size(τ_θφ,2) && iz<=size(τ_θφ,3))
        τ_θφ[ix,iy,iz] = (τ_θφ[ix,iy,iz] + @avyzi_GSDT(ix,iy,iz)*@Dθφ(ix,iy,iz))/
                                    (1.0 + @avyzi_GSDT(ix,iy,iz)/@avyzi_η(ix,iy,iz))
    end
    return
end

@parallel_indices (ix,iy,iz) function compute_TII!( τII::Data.Array,
                                                   τ_RR::Data.Array, τ_θθ::Data.Array, τ_φφ::Data.Array,
                                                   τ_Rθ::Data.Array, τ_Rφ::Data.Array, τ_θφ::Data.Array)
    if (ix<=size(τII,1)-2 && iy<=size(τII,2)-2 && iz<=size(τII,3)-2)
        τII[ix+1,iy+1,iz+1] = sqrt(1.0/2.0 * (τ_RR[ix+1,iy+1,iz+1]^2.0 + τ_θθ[ix+1,iy+1,iz+1]^2.0 +
                                τ_φφ[ix+1,iy+1,iz+1]^2.0) +
                              ((τ_Rθ[ix  ,iy  ,iz  ] + τ_Rθ[ix  ,iy+1,iz  ] +
                                τ_Rθ[ix+1,iy  ,iz  ] + τ_Rθ[ix+1,iy+1,iz  ])*0.25)^2.0 +
                              ((τ_Rφ[ix  ,iy  ,iz  ] + τ_Rφ[ix  ,iy  ,iz+1] +
                                τ_Rφ[ix+1,iy  ,iz  ] + τ_Rφ[ix+1,iy  ,iz+1])*0.25)^2.0 +
                              ((τ_θφ[ix  ,iy  ,iz  ] + τ_θφ[ix  ,iy  ,iz+1] +
                                τ_θφ[ix  ,iy+1,iz  ] + τ_θφ[ix  ,iy+1,iz+1])*0.25)^2.0)
    end
    return
end

@parallel_indices (ix,iy,iz) function power_law!(  η::Data.Array ,   τII::Data.Array , η_ini::Data.Array,
                                                 τ_C::Data.Number, n_exp::Data.Number)
    if (ix<=size(η,1) && iy<=size(η,2) && iz<=size(η,3))
        η[ix,iy,iz] = 2.0/(1.0/η_ini[ix,iy,iz] + 1.0/(η_ini[ix,iy,iz] * (τII[ix,iy,iz]/τ_C)^(1.0-n_exp)))
    end
    return
end
```

```
macro σRR(ix,iy,iz)         esc(:(  -P[$ix,$iy,$iz] + τ_RR[$ix,$iy,$iz] ))  end
macro σθθ(ix,iy,iz)         esc(:(  -P[$ix,$iy,$iz] + τ_θθ[$ix,$iy,$iz] ))  end
macro σφφ(ix,iy,iz)         esc(:(  -P[$ix,$iy,$iz] + τ_φφ[$ix,$iy,$iz] ))  end
macro ρG(ix,iy,iz)          esc(:(   ρ[$ix,$iy,$iz]*g   ))  end
macro avxi_R(ix,iy,iz)      esc(:(  ((   R[$ix  ,$iy+1,$iz+1] +    R[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avyi_R(ix,iy,iz)      esc(:(  ((   R[$ix+1,$iy  ,$iz+1] +    R[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avzi_R(ix,iy,iz)      esc(:(  ((   R[$ix+1,$iy+1,$iz  ] +    R[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avxi_θ(ix,iy,iz)      esc(:(  ((   θ[$ix  ,$iy+1,$iz+1] +    θ[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avyi_θ(ix,iy,iz)      esc(:(  ((   θ[$ix+1,$iy  ,$iz+1] +    θ[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avzi_θ(ix,iy,iz)      esc(:(  ((   θ[$ix+1,$iy+1,$iz  ] +    θ[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avxi_σRR(ix,iy,iz)    esc(:(  ((@σRR( ix  , iy+1, iz+1) + @σRR( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avxi_σθθ(ix,iy,iz)    esc(:(  ((@σθθ( ix  , iy+1, iz+1) + @σθθ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avyi_σθθ(ix,iy,iz)    esc(:(  ((@σθθ( ix+1, iy  , iz+1) + @σθθ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avxi_σφφ(ix,iy,iz)    esc(:(  ((@σφφ( ix  , iy+1, iz+1) + @σφφ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avyi_σφφ(ix,iy,iz)    esc(:(  ((@σφφ( ix+1, iy  , iz+1) + @σφφ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avxa_τRθ(ix,iy,iz)    esc(:(  ((τ_Rθ[$ix  ,$iy  ,$iz  ] + τ_Rθ[$ix+1,$iy  ,$iz  ])*0.5)   ))  end
macro avya_τRθ(ix,iy,iz)    esc(:(  ((τ_Rθ[$ix  ,$iy  ,$iz  ] + τ_Rθ[$ix  ,$iy+1,$iz  ])*0.5)   ))  end
macro avxa_τRφ(ix,iy,iz)    esc(:(  ((τ_Rφ[$ix  ,$iy  ,$iz  ] + τ_Rφ[$ix+1,$iy  ,$iz  ])*0.5)   ))  end
macro avya_τθφ(ix,iy,iz)    esc(:(  ((τ_θφ[$ix  ,$iy  ,$iz  ] + τ_θφ[$ix  ,$iy+1,$iz  ])*0.5)   ))  end
macro avxi_ρG(ix,iy,iz)     esc(:(  (( @ρG( ix  , iy+1, iz+1) +  @ρG( ix+1, iy+1, iz+1))*0.5)   ))  end
macro dVR(ix,iy,iz)         esc(:(                      (@σRR(ix+1,iy+1,iz+1) - @σRR(ix,iy+1,iz+1))*_dr +
                                    1.0/@avxi_R(ix,iy,iz)*(τ_Rθ[ix  ,iy+1,iz  ] - τ_Rθ[ix,iy  ,iz  ])*_dθ +
                                    1.0/@avxi_R(ix,iy,iz)/sin(@avxi_θ(ix,iy,iz))*
                                                          (τ_Rφ[ix  ,iy  ,iz+1] - τ_Rφ[ix,iy  ,iz  ])*_dφ +
                                    2.0*@avxi_σRR(ix,iy,iz)                      /@avxi_R(ix,iy,iz) -
                                       (@avxi_σθθ(ix,iy,iz)+@avxi_σφφ(ix,iy,iz))/@avxi_R(ix,iy,iz) +
                                        @avya_τRθ(ix,iy,iz)/@avxi_R(ix,iy,iz)*cot(@avxi_θ(ix,iy,iz)) -
                                        @avxi_ρG(ix,iy,iz)  ))  end
macro dVθ(ix,iy,iz)         esc(:(                      (   τ_Rθ[ix+1,iy  ,iz  ] - τ_Rθ[ix  ,iy,iz  ])*_dr +
                                    1.0/@avyi_R(ix,iy,iz)*(   @σθθ(ix+1,iy+1,iz+1) - @σθθ(ix+1,iy,iz+1))*_dθ +
                                    1.0/@avyi_R(ix,iy,iz)/sin(@avyi_θ(ix,iy,iz))*
                                                         (   τ_θφ[ix  ,iy  ,iz+1] - τ_θφ[ix  ,iy,iz  ])*_dφ +
                                    3.0*@avxa_τRθ(ix,iy,iz)                        /@avyi_R(ix  ,iy  ,iz  ) +
                                       (@avyi_σθθ(ix,iy,iz) - @avyi_σφφ(ix,iy,iz))/
                                        @avyi_R(ix  ,iy  ,iz  )*cot(@avyi_θ(ix,iy,iz)) ))  end
macro dVφ(ix,iy,iz)         esc(:(                      (τ_Rφ[ix+1,iy  ,iz  ] - τ_Rφ[ix  ,iy  ,iz])*_dr +
                                    1.0/@avzi_R(ix,iy,iz)*(τ_θφ[ix  ,iy+1,iz  ] - τ_θφ[ix  ,iy  ,iz])*_dθ +
                                    1.0/@avzi_R(ix,iy,iz)/sin(@avzi_θ(ix,iy,iz))*
                                                          (@σφφ(ix+1,iy+1,iz+1) - @σφφ(ix+1,iy+1,iz))*_dφ +
                                    3.0*@avxa_τRφ(ix,iy,iz)/@avzi_R(ix,iy,iz) +
                                    2.0*@avya_τθφ(ix,iy,iz)/@avzi_R(ix,iy,iz)*cot(@avzi_θ(ix,iy,iz))    )) end
@parallel_indices (ix,iy,iz) function compute_V!(  VR::Data.Array ,   Vθ::Data.Array ,   Vφ::Data.Array ,
                                                   DT_R::Data.Array ,
                                                   τ_RR::Data.Array , τ_θθ::Data.Array , τ_φφ::Data.Array ,
                                                   τ_Rθ::Data.Array , τ_Rφ::Data.Array , τ_θφ::Data.Array ,
                                                      P::Data.Array ,     R::Data.Array ,    θ::Data.Array ,
                                                      ρ::Data.Array ,
                                                    _dr::Data.Number,  _dθ::Data.Number,  _dφ::Data.Number,
                                                      g::Data.Number)
    if (ix<=size(VR,1)-2 && iy<=size(VR,2)-2 && iz<=size(VR,3)-2)
        VR[ix+1,iy+1,iz+1] =  VR[ix+1,iy+1,iz+1] + @dVR(ix,iy,iz)*
                            ((DT_R[ix,iy+1,iz+1] + DT_R[ix+1,iy+1,iz+1])*0.5)
    end
    if (ix<=size(Vθ,1)-2 && iy<=size(Vθ,2)-2 && iz<=size(Vθ,3)-2)
        Vθ[ix+1,iy+1,iz+1] =  Vθ[ix+1,iy+1,iz+1] + @dVθ(ix,iy,iz)*
                            ((DT_R[ix+1,iy,iz+1] + DT_R[ix+1,iy+1,iz+1])*0.5)
    end
    if (ix<=size(Vφ,1)-2 && iy<=size(Vφ,2)-2 && iz<=size(Vφ,3)-2)
        Vφ[ix+1,iy+1,iz+1] =  Vφ[ix+1,iy+1,iz+1] + @dVφ(ix,iy,iz)*
                            ((DT_R[ix+1,iy+1,iz] + DT_R[ix+1,iy+1,iz+1])*0.5)
    end
    return
end
```

```
# CHECK ERROR ==============================================================================================
@parallel_indices (ix,iy,iz) function check_err!(VR_err::Data.Array, Vθ_err::Data.Array, Vφ_err::Data.Array,
                                                     VR::Data.Array,     Vθ::Data.Array,     Vφ::Data.Array)
    if (ix<=size(VR,1) && iy<=size(VR,2) && iz<=size(VR,3))
        VR_err[ix,iy,iz] = VR[ix,iy,iz] - VR_err[ix,iy,iz]
    end
    if (ix<=size(Vθ,1) && iy<=size(Vθ,2) && iz<=size(Vθ,3))
        Vθ_err[ix,iy,iz] = Vθ[ix,iy,iz] - Vθ_err[ix,iy,iz]
    end
    if (ix<=size(Vφ,1) && iy<=size(Vφ,2) && iz<=size(Vφ,3))
        Vφ_err[ix,iy,iz] = Vφ[ix,iy,iz] - Vφ_err[ix,iy,iz]
    end
    return
end
```

```
# ==========================================================================================
@views function SPH_3D_Diapir()
    num       = 1
    # physical parameters -------------------------------------------
    η0        = 1.0               # Pa*s   , media viscosity
    vr        = 1e2               #         viscosity ratio out/in
    ρ0        = 0.0               # kg/m^3, media density
    ρ_in      = ρ0 - 10.0         # kg/m^3, inclusion density
    g         = 1.0               # m/s^2 , gravity acceleration
    n_exp     = 5.0               #         power law exponent
    τ_C       = 1.0               # Pa     , characteristic stress
    r         = 1000.0            # m      , radius of the total sphere
    lr        = 6.0               # m      , model dimension in r
    lθ, lφ    = lr/r, lr/r        # m      , model dimension in θ and φ
    radius    = 1.0               # m      , radius of the inclusion
    sh        = 1.0               # m/s    , shearing velocity
    # numerics ---------------------------------------------
    εnonl     = 5e-7                          # pseudo-transient loop exit criteria
    nt        = 1                             # number of time steps
    maxiter   = 1e5                           # maximum number of pseudo-transient iterations
    nout      = 1e2                           # pseudo-transient plotting frequency
    CFL       = 1.0/(2.0 + 4.5*log10(vr))     # Courant-Friedrichs-Lewy condition
    # preprocessing ------------------------------------------
     dr           = lr/(nr-1)
    _dr, _dθ, _dφ = (nr-1)/lr, (nθ-1)/lθ, (nφ-1)/lφ
    R_i       = range(     r - lr/2.0,      r + lr/2.0, length=nr)    # coordinates of grid points in r
    θ_i       = range(pi/2.0 - lθ/2.0, pi/2.0 + lθ/2.0, length=nθ)    # coordinates of grid points in θ
    φ_i       = range(       - lφ/2.0,          lφ/2.0, length=nφ)    # coordinates of grid points in z
    (R,θ,φ)   = ([xc for xc = R_i, yc = θ_i, zc = φ_i],[yc for xc = R_i, yc = θ_i, zc = φ_i],
                 [zc for xc = R_i, yc = θ_i, zc = φ_i])                # grid of coordinates
    R         = Data.Array(R)
    θ         = Data.Array(θ)
    φ         = Data.Array(φ)
    # initialization and boundary conditions --------------------------------
    print("Starting initialization ... ")
    P         =   @zeros(nr  , nθ  , nφ  )
    VR        =   @zeros(nr+1, nθ  , nφ  )
    Vθ        =   @zeros(nr  , nθ+1, nφ  )
    Vφ        =   @zeros(nr  , nθ  , nφ+1)
    VR_err    =   @zeros(nr+1, nθ  , nφ  )
    Vθ_err    =   @zeros(nr  , nθ+1, nφ  )
    Vφ_err    =   @zeros(nr  , nθ  , nφ+1)
    τ_RR      =   @zeros(nr  , nθ  , nφ  )
    τ_θθ      =   @zeros(nr  , nθ  , nφ  )
    τ_φφ      =   @zeros(nr  , nθ  , nφ  )
    τ_Rθ      =   @zeros(nr-1, nθ-1, nφ-2)
    τ_Rφ      =   @zeros(nr-1, nθ-2, nφ-1)
    τ_θφ      =   @zeros(nr-2, nθ-1, nφ-1)
    ρ         = ρ0*@ones(nr  , nθ  , nφ  )
    η         = η0*@ones(nr  , nθ  , nφ  )
    η_ini     =   @zeros(nr  , nθ  , nφ  )
    ηSM       =   @zeros(nr  , nθ  , nφ  )
    DT_R      =   @zeros(nr  , nθ  , nφ  )
    τII       =   @zeros(nr  , nθ  , nφ  )
    vpdt      = dr*CFL
    @parallel initialize_inclusion!(ρ, R, θ, φ, r, radius, ρ_in)
    @parallel initialize_inclusion!(η, R, θ, φ, r, radius, η0/vr)
    @parallel initialize_velocity!(Vφ, θ, sh, r)
    η_ini    .= η
    println("done.")
    # action ----------------------------------------------
    println("Starting calculation (nr=$nr, nθ=$nθ , nφ=$nφ)")
    err_evo = []; iter_evo = []; t_tic = 0.0; ittot = 0
    for it = 1:nt # time loop
        for iter = 1:maxiter # pseudo-transient loop
            if (it==1 && iter==11) GC.gc(); t_tic = Base.time() end
            # SOLVER ------------------------------------------
            @parallel equal3!(VR_err, Vθ_err, Vφ_err, VR, Vθ, Vφ)
            @parallel maxloc!(ηSM, η)
            copy_BC!(ηSM)
            @parallel compute_P!(P, DT_R, τ_RR, τ_θθ, τ_φφ, τ_Rθ, τ_Rφ, τ_θφ, VR, Vθ, Vφ, R, θ,
                                 η, ηSM , _dr , _dθ , _dφ , dmp , vpdt, lr)
            @parallel compute_TII!(τII, τ_RR, τ_θθ, τ_φφ, τ_Rθ, τ_Rφ, τ_θφ)
            copy_BC!(τII)
            @parallel power_law!(η, τII, η_ini, τ_C, n_exp)
```

```
            @parallel compute_V!(VR , Vθ , Vφ , DT_R, τ_RR, τ_θθ, τ_φφ, τ_Rθ, τ_Rφ, τ_θφ, P, R, θ, ρ,
                                 _dr, _dθ, _dφ, g)
            @parallel BC_R!(VR, Vθ, Vφ)
            @parallel BC_θ!(VR, Vθ, Vφ, θ, r, sh)
            @parallel BC_φ!(VR, Vθ, Vφ, θ, r, sh)
            # pseudo-transient loop exit criteria ------------------------------
            if iter % nout == 0
                @parallel check_err!(VR_err, Vθ_err, Vφ_err, VR, Vθ, Vφ)
                err_vr = maximum(abs.(VR_err))./maximum(abs.(VR))
                err_vθ = maximum(abs.(Vθ_err))./maximum(abs.(Vθ))
                err_vφ = maximum(abs.(Vφ_err))./maximum(abs.(Vφ))
                err    = max(err_vr, err_vθ, err_vφ)
                # post-processing
                push!(err_evo, err)
                push!(iter_evo,iter)
                if do_viz  display(scatter(iter_evo, err_evo, xaxis=:log,yaxis=:log,legend=false))  end
                @printf("iter %d, err=%1.3e \n", iter, err)
                # pseudo-transient loop exit criteria
                if (err<εnonl && iter>20)  iter_end = iter; break; end
            end
            ittot = iter
        end
    end
    # Performance ----------------------------------------------
    wtime    = Base.time() - t_tic
    # Effective main memory access per iteration [GB]
    A_eff    = (4*2 + 6*2 + 2 + 2*2)/1e9*nr*nθ*nφ*sizeof(Data.Number)
    # (Lower bound of required memory access: Te has to be read and written: 2 whole-array memaccess;
    # Ci has to be read : 1 whole-array memaccess)
    wtime_it = wtime/(ittot-10)                                       # Execution time per iteration [s]
    T_eff    = A_eff/wtime_it                                         # Effective memory throughput [GB/s]
    @printf("Total iters = %d (%d steps), time = %1.3e sec (@ T_eff = %1.2f GB/s) \n",
            ittot, nt, wtime, round(T_eff, sigdigits=3))
    if do_save_p
        !ispath("./out_perf") && mkdir("./out_perf")
        open("./out_perf/out_SPH_3D_PERF.txt","a") do io
            println(io, "$(nr) $(nθ) $(nφ) $(dmp) $(ittot) $(wtime) $(A_eff) $(wtime_it) $(T_eff)")
        end
    end
    # SAVING ------------------------------------------------
    if do_save
        !ispath("./output") && mkdir("./output")
        err_evo  = Data.Array(err_evo)
        iter_evo = Data.Array(iter_evo)
        Save_phys(num, η0, ρ0, ρ_in, n_exp, τ_C, vr, g, sh, r, radius, dmp)
        Save_infos(num, lr, lθ, lφ, nr, nθ, nφ, εnonl, wtime)
        SaveArray("err_evo" , err_evo )
        SaveArray("iter_evo", iter_evo)
        SaveArray("R"       , R       )
        SaveArray("TH"      , θ       )
        SaveArray("PH"      , φ       )
        SaveArray("ETAS"    , η       )
        SaveArray("RHO"     , ρ       )
        SaveArray("P"       , P       )
        SaveArray("VR"      , VR      )
        SaveArray("VTH"     , Vθ      )
        SaveArray("VPH"     , Vφ      )
        SaveArray("TAU_RR"  , τ_RR    )
        SaveArray("TAU_THTH", τ_θθ    )
        SaveArray("TAU_PHPH", τ_φφ    )
        SaveArray("TAU_RTH" , τ_Rθ    )
        SaveArray("TAU_RPH" , τ_Rφ    )
        SaveArray("TAU_THPH", τ_θφ    )
        SaveArray("TII"     , τII     )
    end
    return
end

@time SPH_3D_Diapir()
```

APPENDIX E

Code for the Continental Plateau in Spherical Coordinates

```
# PLATEAU - SPHERICAL COORDINATES
const USE_GPU   = haskey(ENV, "USE_GPU"  ) ? parse(Bool, ENV["USE_GPU"]  ) : true#false
const do_viz    = haskey(ENV, "DO_VIZ"   ) ? parse(Bool, ENV["DO_VIZ"]   ) : false
const do_save   = haskey(ENV, "DO_SAVE"  ) ? parse(Bool, ENV["DO_SAVE"]  ) : true
const do_save_p = haskey(ENV, "DO_SAVE_P") ? parse(Bool, ENV["DO_SAVE_P"]) : false
const nr        = haskey(ENV, "NR"       ) ? parse(Int , ENV["NR"]       ) : 703
const nθ        = haskey(ENV, "NTH"      ) ? parse(Int , ENV["NTH"]      ) : 199
const nφ        = haskey(ENV, "NPH"      ) ? parse(Int , ENV["NPH"]      ) : 199
const GPU_ID    = haskey(ENV, "GPU_ID"   ) ? parse(Int , ENV["GPU_ID"]   ) : 0
const dmp       = haskey(ENV, "DMP"      ) ? parse(Float64, ENV["DMP"]   ) : 4.5

using ParallelStencil
using ParallelStencil.FiniteDifferences3D
@static if USE_GPU
    @init_parallel_stencil(CUDA, Float64, 3)
    CUDA.device!(GPU_ID) # select GPU
else
    @init_parallel_stencil(Threads, Float64, 3)
end
using Plots, Printf, Statistics, LinearAlgebra

# SAVING FUNCTIONS ===========================================================================================
function Save_infos(num, it, lr_d, lθ_d, lφ_d, nr, nθ, nφ, εnonl, runtime; out="./output_$(it)")
    fid=open(out * "/$(num)_infos.inf", "w")
    @printf(fid,"%d %f %f %f %d %d %d %d %d", num, lr_d, lθ_d, lφ_d, nr, nθ, nφ, εnonl, runtime); close(fid)
end

function Save_phys(num , it   , sl   , sη   , sρ   , ηm_d , ηl_d, ηc_d, ηa_d, ρm_d, ρl_d, ρc_d, ρa_d,
                   hm_d, hl_d, hr_d, hc_d, ha_d, n_exp, τ_C , vr   , g    , r    , dmp ; out="./output_$(it)")
    fid=open(out * "/$(num)_phys.inf", "w")
    @printf(fid,"%d %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f",
            num, sl, sη, sρ, ηm_d, ηl_d, ηc_d, ηa_d, ρm_d, ρl_d, ρc_d, ρa_d, hm_d, hl_d, hr_d, hc_d, ha_d,
            n_exp, τ_C, vr, g, r, dmp);
    close(fid)
end

@static if USE_GPU
    function SaveArray(Aname, A, it; out="./output_$(it)")
        A_tmp = Array(A)
        fname = string(out, "/A_", Aname, ".bin");  fid = open(fname,"w"); write(fid, A_tmp); close(fid)
    end
else
    function SaveArray(Aname, A, it; out="./output_$(it)")
        fname = string(out, "/A_", Aname, ".bin"); fid = open(fname,"w"); write(fid, A); close(fid)
    end
end

# ===========================================================================================================
@parallel_indices (ix,iy,iz) function equal3!(VR_err::Data.Array, Vθ_err::Data.Array, Vφ_err::Data.Array,
                                                  VR::Data.Array,     Vθ::Data.Array,     Vφ::Data.Array)
    if (ix<=size(VR_err,1) && iy<=size(VR_err,2) && iz<=size(VR_err,3))
        VR_err[ix,iy,iz] = VR[ix,iy,iz]
    end
    if (ix<=size(Vθ_err,1) && iy<=size(Vθ_err,2) && iz<=size(Vθ_err,3))
        Vθ_err[ix,iy,iz] = Vθ[ix,iy,iz]
    end
    if (ix<=size(Vφ_err,1) && iy<=size(Vφ_err,2) && iz<=size(Vφ_err,3))
        Vφ_err[ix,iy,iz] = Vφ[ix,iy,iz]
    end
    return
end

# BOUNDARY CONDITIONS ========================================================================================
@parallel_indices (ix,iy,iz) function BC_R!(VR::Data.Array, Vθ::Data.Array, Vφ::Data.Array)
    if (ix==1          && iy<=size(VR,2) && iz<=size(VR,3)) VR[ix,iy,iz] = -VR[ix+1,iy,iz]  end
    if (ix==size(VR,1) && iy<=size(VR,2) && iz<=size(VR,3)) VR[ix,iy,iz] = -VR[ix-1,iy,iz]  end
    if (ix==1          && iy<=size(Vθ,2) && iz<=size(Vθ,3)) Vθ[ix,iy,iz] =  Vθ[ix+1,iy,iz]  end
    if (ix==size(Vθ,1) && iy<=size(Vθ,2) && iz<=size(Vθ,3)) Vθ[ix,iy,iz] =  Vθ[ix-1,iy,iz]  end
    if (ix==1          && iy<=size(Vφ,2) && iz<=size(Vφ,3)) Vφ[ix,iy,iz] =  Vφ[ix+1,iy,iz]  end
    if (ix==size(Vφ,1) && iy<=size(Vφ,2) && iz<=size(Vφ,3)) Vφ[ix,iy,iz] =  Vφ[ix-1,iy,iz]  end
    return
end
```

```
@parallel_indices (ix,iy,iz) function BC_θ!(VR::Data.Array, Vθ::Data.Array, Vφ::Data.Array)
    if (ix<=size(VR,1) && iy==1          && iz<=size(VR,3)) VR[ix,iy,iz] =  VR[ix,iy+1,iz]  end
    if (ix<=size(VR,1) && iy==size(VR,2) && iz<=size(VR,3)) VR[ix,iy,iz] =  VR[ix,iy-1,iz]  end
    if (ix<=size(Vθ,1) && iy==1          && iz<=size(Vθ,3)) Vθ[ix,iy,iz] = -Vθ[ix,iy+1,iz]  end
    if (ix<=size(Vθ,1) && iy==size(Vθ,2) && iz<=size(Vθ,3)) Vθ[ix,iy,iz] = -Vθ[ix,iy-1,iz]  end
    if (ix<=size(Vφ,1) && iy==1          && iz<=size(Vφ,3)) Vφ[ix,iy,iz] =  Vφ[ix,iy+1,iz]  end
    if (ix<=size(Vφ,1) && iy==size(Vφ,2) && iz<=size(Vφ,3)) Vφ[ix,iy,iz] =  Vφ[ix,iy-1,iz]  end
    return
end

@parallel_indices (ix,iy,iz) function BC_φ!(VR::Data.Array, Vθ::Data.Array, Vφ::Data.Array)
    if (ix<=size(VR,1) && iy<=size(VR,2) && iz==1          ) VR[ix,iy,iz] =  VR[ix,iy,iz+1]  end
    if (ix<=size(VR,1) && iy<=size(VR,2) && iz==size(VR,3)) VR[ix,iy,iz] =  VR[ix,iy,iz-1]  end
    if (ix<=size(Vθ,1) && iy<=size(Vθ,2) && iz==1          ) Vθ[ix,iy,iz] =  Vθ[ix,iy,iz+1]  end
    if (ix<=size(Vθ,1) && iy<=size(Vθ,2) && iz==size(Vθ,3)) Vθ[ix,iy,iz] =  Vθ[ix,iy,iz-1]  end
    if (ix<=size(Vφ,1) && iy<=size(Vφ,2) && iz==1          ) Vφ[ix,iy,iz] = -Vφ[ix,iy,iz+1]  end
    if (ix<=size(Vφ,1) && iy<=size(Vφ,2) && iz==size(Vφ,3)) Vφ[ix,iy,iz] = -Vφ[ix,iy,iz-1]  end
    return
end

# COPY BOUNDARIES ==========================================================================================
@parallel_indices (ix,iy,iz) function copy_bc_r!(A::Data.Array)
    if (ix==1         && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix+1,iy,iz] end
    if (ix==size(A,1) && iy<=size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix-1,iy,iz] end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_θ!(A::Data.Array)
    if (ix<=size(A,1) && iy==1         && iz<=size(A,3))    A[ix,iy,iz] = A[ix,iy+1,iz] end
    if (ix<=size(A,1) && iy==size(A,2) && iz<=size(A,3))    A[ix,iy,iz] = A[ix,iy-1,iz] end
    return
end

@parallel_indices (ix,iy,iz) function copy_bc_φ!(A::Data.Array)
    if (ix<=size(A,1) && iy<=size(A,2) && iz==1         )    A[ix,iy,iz] = A[ix,iy,iz+1] end
    if (ix<=size(A,1) && iy<=size(A,2) && iz==size(A,3))    A[ix,iy,iz] = A[ix,iy,iz-1] end
    return
end

function copy_BC!(A::Data.Array)
    @parallel copy_bc_r!(A)
    @parallel copy_bc_θ!(A)
    @parallel copy_bc_φ!(A)
end
```

```
# INITIALIZATION ============================================================================================
@parallel_indices (ix,iy,iz) function initialize_geometry!(  A::Data.Array ,  R::Data.Array ,
                                                             θ::Data.Array ,  φ::Data.Array ,
                                                             r::Data.Number, ha::Data.Number,
                                                            hc::Data.Number, hr::Data.Number,
                                                            hl::Data.Number, dl::Data.Number,
                                                            dh::Data.Number, dt::Data.Number,
                                                            gl::Data.Number, gc::Data.Number,
                                                            ga::Data.Number)
    if (ix<=size(A,1) && iy<=size(A,2) && iz<=size(A,3))
        if (R[ix,iy,iz]-r        > -hc-hr-hl                                                               )
            A[ix,iy,iz] = gl  end
        if (R[ix,iy,iz]-r        > -hc                                  && (θ[ix,iy,iz]-pi/2.0)*r<  0      )
            A[ix,iy,iz] = gc  end
        if (R[ix,iy,iz]-r        > -hc                                  &&  φ[ix,iy,iz]*r        >  0      )
            A[ix,iy,iz] = gc  end
        if (R[ix,iy,iz]-r        > -hc-hr                               && (θ[ix,iy,iz]-pi/2.0)*r>= dt    &&
            φ[ix,iy,iz]*r        <=-dt                                                                     )
            A[ix,iy,iz] = gc  end
        if (R[ix,iy,iz]-r        > -hc+((hr/dt)*φ[ix,iy,iz]*r)          && (θ[ix,iy,iz]-pi/2.0)*r>= dt    &&
            φ[ix,iy,iz]*r        <= 0                                   &&  φ[ix,iy,iz]*r        >=-dt     )
            A[ix,iy,iz] = gc  end
        if (R[ix,iy,iz]-r        > -hc+((hr/dt)*(pi/2.0-θ[ix,iy,iz])*r) && (θ[ix,iy,iz]-pi/2.0)*r>= 0     &&
           (θ[ix,iy,iz]-pi/2.0)*r<= dt                                  &&  φ[ix,iy,iz]*r        <=-dt     )
            A[ix,iy,iz] = gc  end
        if (R[ix,iy,iz]-r        >  max(-hc+((hr/dt)*(pi/2.0-θ[ix,iy,iz])*r),-hc+((hr/dt)*φ[ix,iy,iz]*r)) &&
           (θ[ix,iy,iz]-pi/2.0)*r>= 0                                   && (θ[ix,iy,iz]-pi/2.0)*r<= dt    &&
            φ[ix,iy,iz]*r        <= 0                                   &&  φ[ix,iy,iz]*r        >=-dt     )
            A[ix,iy,iz] = gc  end
        if (R[ix,iy,iz]-r        >  0                                   && (θ[ix,iy,iz]-pi/2.0)*r<  0      )
            A[ix,iy,iz] = ga  end
        if (R[ix,iy,iz]-r        >  0                                   &&  φ[ix,iy,iz]*r        >  0      )
            A[ix,iy,iz] = ga  end
        if (R[ix,iy,iz]-r        >  ha                                  && (θ[ix,iy,iz]-pi/2.0)*r>= dt    &&
            φ[ix,iy,iz]*r        <=-dt                                                                     )
            A[ix,iy,iz] = ga  end
        if (R[ix,iy,iz]-r        > -((ha/dt)*φ[ix,iy,iz]*r)             && (θ[ix,iy,iz]-pi/2.0)*r>= dt    &&
            φ[ix,iy,iz]*r        <= 0                                   &&  φ[ix,iy,iz]*r        >=-dt     )
            A[ix,iy,iz] = ga  end
        if (R[ix,iy,iz]-r        > -((ha/dt)*(pi/2.0-θ[ix,iy,iz])*r)    && (θ[ix,iy,iz]-pi/2.0)*r>= 0     &&
           (θ[ix,iy,iz]-pi/2.0)*r<= dt                                  &&  φ[ix,iy,iz]*r        <=-dt     )
            A[ix,iy,iz] = ga  end
        if (R[ix,iy,iz]-r        >  min(-((ha/dt)*(pi/2.0-θ[ix,iy,iz])*r),-((ha/dt)*φ[ix,iy,iz]*r))       &&
           (θ[ix,iy,iz]-pi/2.0)*r>= 0                                   && (θ[ix,iy,iz]-pi/2.0)*r<= dt    &&
            φ[ix,iy,iz]*r        <= 0                                   &&  φ[ix,iy,iz]*r        >=-dt     )
            A[ix,iy,iz] = ga  end
    end
    return
end

# ===========================================================================================================
@parallel function maxloc!(ηSM::Data.Array, η::Data.Array)
    @inn(ηSM) = @maxloc(η)
    return
end
```

```
#  SOLVER ===================================================================================================
macro KBDT(ix,iy,iz)        esc(:(  dmp * 2.0 * pi * vpdt / lr * ηSM[$ix,$iy,$iz]   ))  end
macro GSDT(ix,iy,iz)        esc(:(        4.0 * pi * vpdt / lr * ηSM[$ix,$iy,$iz]   ))  end
macro avxa_VR(ix,iy,iz)     esc(:(  ((   VR[$ix  ,$iy  ,$iz  ] +    VR[$ix+1,$iy  ,$iz  ])*0.5 )  ))  end
macro avxi_Vθ(ix,iy,iz)     esc(:(  ((   Vθ[$ix  ,$iy+1,$iz+1] +    Vθ[$ix+1,$iy+1,$iz+1])*0.5 )  ))  end
macro avya_Vθ(ix,iy,iz)     esc(:(  ((   Vθ[$ix  ,$iy  ,$iz  ] +    Vθ[$ix  ,$iy+1,$iz  ])*0.5 )  ))  end
macro avxi_Vφ(ix,iy,iz)     esc(:(  ((   Vφ[$ix  ,$iy+1,$iz+1] +    Vφ[$ix+1,$iy+1,$iz+1])*0.5 )  ))  end
macro avyi_Vφ(ix,iy,iz)     esc(:(  ((   Vφ[$ix+1,$iy  ,$iz+1] +    Vφ[$ix+1,$iy+1,$iz+1])*0.5 )  ))  end
macro avxyi_R(ix,iy,iz)     esc(:(  ((    R[$ix  ,$iy  ,$iz+1] +     R[$ix  ,$iy+1,$iz+1] +
                                          R[$ix+1,$iy  ,$iz+1] +     R[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avxzi_R(ix,iy,iz)     esc(:(  ((    R[$ix  ,$iy+1,$iz  ] +     R[$ix  ,$iy+1,$iz+1] +
                                          R[$ix+1,$iy+1,$iz  ] +     R[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avyzi_R(ix,iy,iz)     esc(:(  ((    R[$ix+1,$iy  ,$iz  ] +     R[$ix+1,$iy  ,$iz+1] +
                                          R[$ix+1,$iy+1,$iz  ] +     R[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avxzi_θ(ix,iy,iz)     esc(:(  ((    θ[$ix  ,$iy+1,$iz  ] +     θ[$ix  ,$iy+1,$iz+1] +
                                          θ[$ix+1,$iy+1,$iz  ] +     θ[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avyzi_θ(ix,iy,iz)     esc(:(  ((    θ[$ix+1,$iy  ,$iz  ] +     θ[$ix+1,$iy  ,$iz+1] +
                                          θ[$ix+1,$iy+1,$iz  ] +     θ[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avxyi_GSDT(ix,iy,iz)  esc(:(  ((@GSDT($ix  ,$iy  ,$iz+1) + @GSDT($ix  ,$iy+1,$iz+1) +
                                      @GSDT($ix+1,$iy  ,$iz+1) + @GSDT($ix+1,$iy+1,$iz+1))*0.25)  ))  end
macro avxzi_GSDT(ix,iy,iz)  esc(:(  ((@GSDT($ix  ,$iy+1,$iz  ) + @GSDT($ix  ,$iy+1,$iz+1) +
                                      @GSDT($ix+1,$iy+1,$iz  ) + @GSDT($ix+1,$iy+1,$iz+1))*0.25)  ))  end
macro avyzi_GSDT(ix,iy,iz)  esc(:(  ((@GSDT($ix+1,$iy  ,$iz  ) + @GSDT($ix+1,$iy  ,$iz+1) +
                                      @GSDT($ix+1,$iy+1,$iz  ) + @GSDT($ix+1,$iy+1,$iz+1))*0.25)  ))  end
macro avxyi_η(ix,iy,iz)     esc(:(  ((    η[$ix  ,$iy  ,$iz+1] +     η[$ix  ,$iy+1,$iz+1] +
                                          η[$ix+1,$iy  ,$iz+1] +     η[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avxzi_η(ix,iy,iz)     esc(:(  ((    η[$ix  ,$iy+1,$iz  ] +     η[$ix  ,$iy+1,$iz+1] +
                                          η[$ix+1,$iy+1,$iz  ] +     η[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro avyzi_η(ix,iy,iz)     esc(:(  ((    η[$ix+1,$iy  ,$iz  ] +     η[$ix+1,$iy  ,$iz+1] +
                                          η[$ix+1,$iy+1,$iz  ] +     η[$ix+1,$iy+1,$iz+1])*0.25)  ))  end
macro DIVV(ix,iy,iz)        esc(:((  VR[$ix+1,$iy,$iz] - VR[$ix,$iy,$iz])*_dr +
                                    1.0/R[$ix,$iy,$iz]                        *
                                    (Vθ[$ix,$iy+1,$iz] - Vθ[$ix,$iy,$iz])*_dθ +
                                    1.0/R[$ix,$iy,$iz]  /sin(θ[$ix,$iy  ,$iz])*
                                    (Vφ[$ix,$iy,$iz+1] - Vφ[$ix,$iy,$iz])*_dφ +
                                    2.0*@avxa_VR(ix,iy,iz)/  R[$ix,$iy  ,$iz] +
                                        @avya_Vθ(ix,iy,iz)/  R[$ix,$iy  ,$iz] * cot(θ[$ix,$iy,$iz]) )) end
macro DRR(ix,iy,iz)     esc(:(  (  VR[$ix+1,$iy,$iz]  - VR[$ix,$iy  ,$iz])*_dr -
                                1.0/3.0*@DIVV(ix,iy,iz) ))  end
macro Dθθ(ix,iy,iz)     esc(:(  1.0/R[$ix,$iy,$iz]    *(Vθ[$ix,$iy+1,$iz] - Vθ[$ix,$iy,$iz])*_dθ +
                                @avxa_VR(ix,iy,iz)    /  R[$ix,$iy  ,$iz] -
                                1.0/3.0*@DIVV(ix,iy,iz) ))  end
macro Dφφ(ix,iy,iz)     esc(:(  1.0/ R[$ix,$iy,$iz  ] /sin(θ[$ix,$iy,$iz])*
                                   (Vφ[$ix,$iy,$iz+1] -   Vφ[$ix,$iy,$iz])*_dφ +
                                @avxa_VR(ix,iy,iz)    /    R[$ix,$iy,$iz] +
                                @avya_Vθ(ix,iy,iz)    /    R[$ix,$iy,$iz]*cot(θ[$ix,$iy,$iz]) -
                                1.0/3.0*@DIVV(ix,iy,iz) ))  end
macro DRθ(ix,iy,iz)     esc(:(                         (Vθ[$ix+1,$iy+1,$iz+1] - Vθ[$ix,$iy+1,$iz+1])*_dr +
                                1.0/@avxyi_R(ix,iy,iz)*(VR[$ix+1,$iy+1,$iz+1] - VR[$ix+1,$iy,$iz+1])*_dθ -
                                    @avxi_Vθ(ix,iy,iz)/@avxyi_R(ix,iy,iz)   ))  end
macro DRφ(ix,iy,iz)     esc(:(  (Vφ[$ix+1,$iy+1,$iz+1] - Vφ[$ix,$iy+1,$iz+1])*_dr +
                                1.0/@avxzi_R(ix,iy,iz)/sin(@avxzi_θ(ix,iy,iz))*
                                (VR[$ix+1,$iy+1,$iz+1] - VR[$ix+1,$iy+1,$iz])*_dφ -
                                @avxi_Vφ(ix,iy,iz)/@avxzi_R(ix,iy,iz)   ))  end
macro Dθφ(ix,iy,iz)     esc(:(  1.0/@avyzi_R(ix,iy,iz)*(Vφ[$ix+1,$iy+1,$iz+1] - Vφ[$ix+1,$iy,$iz+1])*_dθ +
                                1.0/@avyzi_R(ix,iy,iz)/sin(@avyzi_θ(ix,iy,iz))*
                                (Vθ[$ix+1,$iy+1,$iz+1] - Vθ[$ix+1,$iy+1,$iz])*_dφ -
                                @avyi_Vφ(ix,iy,iz)/@avyzi_R(ix,iy,iz)*cot(@avyzi_θ(ix,iy,iz))  ))  end
```

```
@parallel_indices (ix,iy,iz) function compute_P!(   P::Data.Array , DT_R::Data.Array ,
                                                 τ_RR::Data.Array , τ_θθ::Data.Array , τ_φφ::Data.Array ,
                                                 τ_Rθ::Data.Array , τ_Rφ::Data.Array , τ_θφ::Data.Array ,
                                                   VR::Data.Array ,   Vθ::Data.Array ,   Vφ::Data.Array ,
                                                    R::Data.Array ,    θ::Data.Array ,    η::Data.Array ,
                                                  ηSM::Data.Array,
                                                  _dr::Data.Number,  _dθ::Data.Number,  _dφ::Data.Number,
                                                  dmp::Data.Number, vpdt::Data.Number,   lr::Data.Number)
    if (ix<=size(P,1) && iy<=size(P,2) && iz<=size(P,3))
        P[ix,iy,iz] = P[ix,iy,iz] - @KBDT(ix,iy,iz) * @DIVV(ix,iy,iz)
        τ_RR[ix,iy,iz] = (τ_RR[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@DRR(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])
        τ_θθ[ix,iy,iz] = (τ_θθ[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@Dθθ(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])
        τ_φφ[ix,iy,iz] = (τ_φφ[ix,iy,iz] + @GSDT(ix,iy,iz)*2.0*@Dφφ(ix,iy,iz))/
                                    (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz])
        DT_R[ix,iy,iz] = vpdt^2 / (@KBDT(ix,iy,iz) + @GSDT(ix,iy,iz)/
                            (1.0 + @GSDT(ix,iy,iz)/η[ix,iy,iz]))
    end
    if (ix<=size(τ_Rθ,1) && iy<=size(τ_Rθ,2) && iz<=size(τ_Rθ,3))
        τ_Rθ[ix,iy,iz] = (τ_Rθ[ix,iy,iz] + @avxyi_GSDT(ix,iy,iz)*@DRθ(ix,iy,iz))/
                                    (1.0 + @avxyi_GSDT(ix,iy,iz)/@avxyi_η(ix,iy,iz))
    end
    if (ix<=size(τ_Rφ,1) && iy<=size(τ_Rφ,2) && iz<=size(τ_Rφ,3))
        τ_Rφ[ix,iy,iz] = (τ_Rφ[ix,iy,iz] + @avxzi_GSDT(ix,iy,iz)*@DRφ(ix,iy,iz))/
                                    (1.0 + @avxzi_GSDT(ix,iy,iz)/@avxzi_η(ix,iy,iz))
    end
    if (ix<=size(τ_θφ,1) && iy<=size(τ_θφ,2) && iz<=size(τ_θφ,3))
        τ_θφ[ix,iy,iz] = (τ_θφ[ix,iy,iz] + @avyzi_GSDT(ix,iy,iz)*@Dθφ(ix,iy,iz))/
                                    (1.0 + @avyzi_GSDT(ix,iy,iz)/@avyzi_η(ix,iy,iz))
    end
    return
end

@parallel_indices (ix,iy,iz) function compute_TII!( τII::Data.Array, τ_RR::Data.Array, τ_θθ::Data.Array,
                                                     τ_φφ::Data.Array, τ_Rθ::Data.Array, τ_Rφ::Data.Array,
                                                     τ_θφ::Data.Array)
    if (ix<=size(τII,1)-2 && iy<=size(τII,2)-2 && iz<=size(τII,3)-2)
        τII[ix+1,iy+1,iz+1] = sqrt(1.0/2.0 * (τ_RR[ix+1,iy+1,iz+1]^2.0  + τ_θθ[ix+1,iy+1,iz+1]^2.0 +
                                        τ_φφ[ix+1,iy+1,iz+1]^2.0) +
                                      ((τ_Rθ[ix  ,iy  ,iz  ]     + τ_Rθ[ix  ,iy+1,iz  ] +
                                        τ_Rθ[ix+1,iy  ,iz  ]     + τ_Rθ[ix+1,iy+1,iz  ])*0.25)^2.0 +
                                      ((τ_Rφ[ix  ,iy  ,iz  ]     + τ_Rφ[ix  ,iy  ,iz+1] +
                                        τ_Rφ[ix+1,iy  ,iz  ]     + τ_Rφ[ix+1,iy  ,iz+1])*0.25)^2.0 +
                                      ((τ_θφ[ix  ,iy  ,iz  ]     + τ_θφ[ix  ,iy  ,iz+1] +
                                        τ_θφ[ix  ,iy+1,iz  ]     + τ_θφ[ix  ,iy+1,iz+1])*0.25)^2.0)
    end
    return
end

@parallel_indices (ix,iy,iz) function power_law!(  η::Data.Array ,   τII::Data.Array , η_ini::Data.Array,
                                                 τ_C::Data.Number, n_exp::Data.Number)
    if (ix<=size(η,1) && iy<=size(η,2) && iz<=size(η,3))
        η[ix,iy,iz] = 2.0/(1.0/η_ini[ix,iy,iz] + 1.0/(η_ini[ix,iy,iz] * (τII[ix,iy,iz]/τ_C)^(1.0-n_exp)))
    end
    return
end
```

```
macro σRR(ix,iy,iz)         esc(:(  -P[$ix,$iy,$iz] + τ_RR[$ix,$iy,$iz] ))  end
macro σθθ(ix,iy,iz)         esc(:(  -P[$ix,$iy,$iz] + τ_θθ[$ix,$iy,$iz] ))  end
macro σφφ(ix,iy,iz)          esc(:(  -P[$ix,$iy,$iz] + τ_φφ[$ix,$iy,$iz] ))  end
macro ρG(ix,iy,iz)          esc(:(   ρ[$ix,$iy,$iz]*g   ))  end
macro avxi_R(ix,iy,iz)      esc(:(  ((   R[$ix  ,$iy+1,$iz+1] +    R[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avyi_R(ix,iy,iz)      esc(:(  ((   R[$ix+1,$iy  ,$iz+1] +    R[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avzi_R(ix,iy,iz)      esc(:(  ((   R[$ix+1,$iy+1,$iz  ] +    R[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avxi_θ(ix,iy,iz)      esc(:(  ((   θ[$ix  ,$iy+1,$iz+1] +    θ[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avyi_θ(ix,iy,iz)      esc(:(  ((   θ[$ix+1,$iy  ,$iz+1] +    θ[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avzi_θ(ix,iy,iz)      esc(:(  ((   θ[$ix+1,$iy+1,$iz  ] +    θ[$ix+1,$iy+1,$iz+1])*0.5)   ))  end
macro avxi_σRR(ix,iy,iz)    esc(:(  ((@σRR( ix  , iy+1, iz+1) + @σRR( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avxi_σθθ(ix,iy,iz)    esc(:(  ((@σθθ( ix  , iy+1, iz+1) + @σθθ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avyi_σθθ(ix,iy,iz)    esc(:(  ((@σθθ( ix+1, iy  , iz+1) + @σθθ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avxi_σφφ(ix,iy,iz)     esc(:(  ((@σφφ( ix  , iy+1, iz+1) + @σφφ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avyi_σφφ(ix,iy,iz)     esc(:(  ((@σφφ( ix+1, iy  , iz+1) + @σφφ( ix+1, iy+1, iz+1))*0.5)   ))  end
macro avxa_τRθ(ix,iy,iz)    esc(:(  ((τ_Rθ[$ix  ,$iy  ,$iz  ] + τ_Rθ[$ix+1,$iy  ,$iz  ])*0.5)   ))  end
macro avya_τRθ(ix,iy,iz)    esc(:(  ((τ_Rθ[$ix  ,$iy  ,$iz  ] + τ_Rθ[$ix  ,$iy+1,$iz  ])*0.5)   ))  end
macro avxa_τRφ(ix,iy,iz)     esc(:(  ((τ_Rφ[$ix  ,$iy  ,$iz  ] + τ_Rφ[$ix+1,$iy  ,$iz  ])*0.5)   ))  end
macro avya_τθφ(ix,iy,iz)     esc(:(  ((τ_θφ[$ix  ,$iy  ,$iz  ] + τ_θφ[$ix  ,$iy+1,$iz  ])*0.5)   ))  end
macro avxi_ρG(ix,iy,iz)     esc(:(  (( @ρG( ix  , iy+1, iz+1) +  @ρG( ix+1, iy+1, iz+1))*0.5)   ))  end
macro dVR(ix,iy,iz)     esc(:(                       (@σRR(ix+1,iy+1,iz+1) - @σRR(ix,iy+1,iz+1))*_dr +
                               1.0/@avxi_R(ix,iy,iz)*(τ_Rθ[ix  ,iy+1,iz  ] - τ_Rθ[ix,iy  ,iz  ])*_dθ +
                               1.0/@avxi_R(ix,iy,iz)/sin(@avxi_θ(ix,iy,iz))*
                                                     (τ_Rφ[ix  ,iy  ,iz+1] - τ_Rφ[ix,iy  ,iz  ])*_dφ +
                               2.0*@avxi_σRR(ix,iy,iz)                     / @avxi_R(ix,iy,iz) -
                                  (@avxi_σθθ(ix,iy,iz)+@avxi_σφφ(ix,iy,iz))/ @avxi_R(ix,iy,iz) +
                                   @avya_τRθ(ix,iy,iz)                     / @avxi_R(ix,iy,iz)*
                                   cot(@avxi_θ(ix,iy,iz))                  - @avxi_ρG(ix,iy,iz)     ))  end
macro dVθ(ix,iy,iz)     esc(:(                       (    τ_Rθ[ix+1,iy  ,iz  ] - τ_Rθ[ix  ,iy,iz  ])*_dr +
                               1.0/@avyi_R(ix,iy,iz)*(    @σθθ(ix+1,iy+1,iz+1) - @σθθ(ix+1,iy,iz+1))*_dθ +
                               1.0/@avyi_R(ix,iy,iz)/sin(@avyi_θ(ix,iy,iz))*
                                                     (    τ_θφ[ix  ,iy  ,iz+1] - τ_θφ[ix  ,iy,iz  ])*_dφ +
                               3.0*@avxa_τRθ(ix,iy,iz)                       /@avyi_R(ix  ,iy  ,iz  ) +
                                  (@avyi_σθθ(ix,iy,iz) - @avyi_σφφ(ix,iy,iz))/@avyi_R(ix  ,iy  ,iz  )*
                                   cot(@avyi_θ(ix,iy,iz)) ))  end
macro dVφ(ix,iy,iz)     esc(:(                       (τ_Rφ[ix+1,iy  ,iz  ] - τ_Rφ[ix  ,iy  ,iz])*_dr +
                               1.0/@avzi_R(ix,iy,iz)*(τ_θφ[ix  ,iy+1,iz  ] - τ_θφ[ix  ,iy  ,iz])*_dθ +
                               1.0/@avzi_R(ix,iy,iz)/sin(@avzi_θ(ix,iy,iz))*
                                                     (@σφφ(ix+1,iy+1,iz+1) - @σφφ(ix+1,iy+1,iz))*_dφ +
                               3.0*@avxa_τRφ(ix,iy,iz)/@avzi_R(ix,iy,iz) +
                               2.0*@avya_τθφ(ix,iy,iz)/@avzi_R(ix,iy,iz)*cot(@avzi_θ(ix,iy,iz))    ))  end
@parallel_indices (ix,iy,iz) function compute_V!(  VR::Data.Array ,   Vθ::Data.Array ,   Vφ::Data.Array ,
                                                 DT_R::Data.Array , τ_RR::Data.Array , τ_θθ::Data.Array ,
                                                 τ_φφ::Data.Array , τ_Rθ::Data.Array , τ_Rφ::Data.Array ,
                                                 τ_θφ::Data.Array ,    P::Data.Array ,    R::Data.Array ,
                                                    θ::Data.Array ,    ρ::Data.Array ,  _dr::Data.Number,
                                                  _dθ::Data.Number,  _dφ::Data.Number,    g::Data.Number)
    if (ix<=size(VR,1)-2 && iy<=size(VR,2)-2 && iz<=size(VR,3)-2)
        VR[ix+1,iy+1,iz+1] =    VR[ix+1,iy+1,iz+1] + @dVR(ix  ,iy  ,iz  )*
                            ((DT_R[ix  ,iy+1,iz+1] + DT_R[ix+1,iy+1,iz+1])*0.5)
    end
    if (ix<=size(Vθ,1)-2 && iy<=size(Vθ,2)-2 && iz<=size(Vθ,3)-2)
        Vθ[ix+1,iy+1,iz+1] =    Vθ[ix+1,iy+1,iz+1] + @dVθ(ix  ,iy  ,iz  )*
                            ((DT_R[ix+1,iy  ,iz+1] + DT_R[ix+1,iy+1,iz+1])*0.5)
    end
    if (ix<=size(Vφ,1)-2 && iy<=size(Vφ,2)-2 && iz<=size(Vφ,3)-2)
        Vφ[ix+1,iy+1,iz+1] =    Vφ[ix+1,iy+1,iz+1] + @dVφ(ix  ,iy  ,iz  )*
                            ((DT_R[ix+1,iy+1,iz  ] + DT_R[ix+1,iy+1,iz+1])*0.5)
    end
    return
end
```

```
# CHECK ERROR ==============================================================================================
@parallel_indices (ix,iy,iz) function check_err!(VR_err::Data.Array, Vθ_err::Data.Array, Vφ_err::Data.Array,
                                                     VR::Data.Array,     Vθ::Data.Array,     Vφ::Data.Array)
    if (ix<=size(VR,1) && iy<=size(VR,2) && iz<=size(VR,3))
        VR_err[ix,iy,iz] = VR[ix,iy,iz] - VR_err[ix,iy,iz]
    end
    if (ix<=size(Vθ,1) && iy<=size(Vθ,2) && iz<=size(Vθ,3))
        Vθ_err[ix,iy,iz] = Vθ[ix,iy,iz] - Vθ_err[ix,iy,iz]
    end
    if (ix<=size(Vφ,1) && iy<=size(Vφ,2) && iz<=size(Vφ,3))
        Vφ_err[ix,iy,iz] = Vφ[ix,iy,iz] - Vφ_err[ix,iy,iz]
    end
    return
end
```

```
# ===========================================================================================================
@views function SPH_3D_Plateau()
    num      = 1
    # physical parameters ------------------------------------------
    ηm_d      = 1e20                            # Pa*s   , mantle viscosity
    ηl_d      = 1e20                            # Pa*s   , lithospheric mantle viscosity
    ηc_d      = 1e20                            # Pa*s   , crust viscosity - step 1
    ηc_2_d    = 5e20                            # Pa*s   , crust viscosity - step 2
    ηc_3_d    = 1e21                            # Pa*s   , crust viscosity - step 3
    ηc_4_d    = 3e21                            # Pa*s   , crust viscosity - step 4
    ηc_5_d    = 6e21                            # Pa*s   , crust viscosity - step 5
    ηc_6_d    = 1e22                            # Pa*s   , crust viscosity - step 6
    ηs_d      = 5e18                            # Pa*s   , sticky-air viscosity
    ρm_d      = 3300.0                          # kg/m^3, mantle density
    ρl_d      = 3300.0                          # kg/m^3, lithospheric mantle density
    ρc_d      = 2800.0                          # kg/m^3, crust density
    ρs_d      = 0.0                             # kg/m^3, sticky-air density
    hm_d      = 360e3                           # m      , mantle thickness
    hl_d      = 87e3                            # m      , lithospheric mantle thickness
    hr_d      = 28e3                            # m      , root thickness
    hc_d      = 35e3                            # m      , crust thickness
    ha_d      = 5e3                             # m      , elevation thickness
    hs_d      = 50e3                            # m      , sticky-air thickness
    dl_d      = 700e3                           # m      , distance lowland
    dt_d      = 100e3                           # m      , distance transition zone
    dh_d      = 600e3                           # m      , distance highland
    g_d       = 9.81                            # m/s^2 , gravity acceleration
    n_exp     = 1.0                             #         power law exponent - step 1-to-6
    n_exp_PL  = 3.0                             #         power law exponent - step 7
    n_exp_PL2 = 6.0                             #         power law exponent - step 8
    τC_d      = 24e6                            # Pa     , reference stress (power law)
    r_d       = 6371e3                          # m      , radius of the total sphere
    # (flat: 6371000e3 / Earth: 6371e3 / Mars: 3396e3 / Moon: 1737e3)
    lr_d      = hm_d+hl_d+hr_d+hc_d+ha_d+hs_d   # m      , model dimension in r
    lθ_d      = (dl_d+dt_d+dh_d)/r_d            # rad    , model dimension in θ
    lφ_d      = (dl_d+dt_d+dh_d)/r_d            # rad    , model dimension in z
    # scaling ------------------------------------------------
    # scales
    sl = lr_d                                   # m      , scale=hight of model domain
    sη = ηm_d                                   # Pas    , scale=mantle viscosity
    sρ = ρm_d                                   # kg/m^3, scale=mantle density
    # non-dimensionalization
    lr     = lr_d  /sl
    lθ     = lθ_d
    lφ     = lφ_d
    ρm     = ρm_d  /sρ
    ρl     = ρl_d  /sρ
    ρc     = ρc_d  /sρ
    ρs     = ρs_d  /sρ
    ηm     = ηm_d  /sη
    ηl     = ηl_d  /sη
    ηc     = ηc_d  /sη
    ηc_2   = ηc_2_d/sη
    ηc_3   = ηc_3_d/sη
    ηc_4   = ηc_4_d/sη
    ηc_5   = ηc_5_d/sη
    ηc_6   = ηc_6_d/sη
    ηs     = ηs_d  /sη
    r      = r_d   /sl
    hm     = hm_d  /sl
    hl     = hl_d  /sl
    hr     = hr_d  /sl
    hc     = hc_d  /sl
    ha     = ha_d  /sl
    hs     = hs_d  /sl
    dl     = dl_d  /sl
    dt     = dt_d  /sl
    dh     = dh_d  /sl
    g      = g_d   /sη^2*sρ^2*sl^3
    τ_C    = τC_d  /sη^2*sρ  *sl^2
    vr     = max(ηm,ηl,ηc,ηs)/min(ηm,ηl,ηc,ηs)  # viscosity ratio max/min
```

```
# numerics -----------------------------------------------
εnonl    = 5e-7                            # pseudo-transient loop exit criteria
nt       = 8                               # number of convergence steps
maxiter  = 200e6                           # maximum number of pseudo-transient iterations
nout     = 1e2                             # pseudo-transient error check frequency
CFL      = 1.0/(2.0 + 4.5*log10(vr))       # Courant-Friedrichs-Lewy condition
# preprocessing -------------------------------------------
 dr      = lr/(nr-1)                                 # grid spacing in direction r
_dr, _dθ, _dφ = (nr-1)/lr, (nθ-1)/lθ, (nφ-1)/lφ        # 1/grid spacing
R_i      = range(     r - (hc+hr+hl+hm),     r + lr-(hc+hr+hl+hm),length=nr) # coordinates of grid points
θ_i      = range(pi/2.0 - dl/r         ,pi/2.0 + (dt+dh)/r       ,length=nθ) # coordinates of grid points
φ_i      = range(       - dl/r         ,          (dt+dh)/r       ,length=nφ) # coordinates of grid points
(R,θ,φ)  = ([xc for xc = R_i, yc = θ_i, zc = φ_i],[yc for xc = R_i, yc = θ_i, zc = φ_i],
           [zc for xc = R_i, yc = θ_i, zc = φ_i])     # grid of coordinates
R        = Data.Array(R)                             # format of arrays, necessary for GPU computation
θ        = Data.Array(θ)                             # format of arrays, necessary for GPU computation
φ        = Data.Array(φ)                             # format of arrays, necessary for GPU computation
# initialization and boundary conditions -------------------------------
print("Starting initialization ... ")
P         =   @zeros(nr  , nθ  , nφ  )
VR        =   @zeros(nr+1, nθ  , nφ  )
Vθ        =   @zeros(nr  , nθ+1, nφ  )
Vφ        =   @zeros(nr  , nθ  , nφ+1)
VR_err    =   @zeros(nr+1, nθ  , nφ  )
Vθ_err    =   @zeros(nr  , nθ+1, nφ  )
Vφ_err    =   @zeros(nr  , nθ  , nφ+1)
τ_RR      =   @zeros(nr  , nθ  , nφ  )
τ_θθ      =   @zeros(nr  , nθ  , nφ  )
τ_φφ      =   @zeros(nr  , nθ  , nφ  )
τ_Rθ      =   @zeros(nr-1, nθ-1, nφ-2)
τ_Rφ      =   @zeros(nr-1, nθ-2, nφ-1)
τ_θφ      =   @zeros(nr-2, nθ-1, nφ-1)
ρ         = ρm*@ones(nr  , nθ  , nφ  )
η         = ηm*@ones(nr  , nθ  , nφ  )
η_ini     =   @zeros(nr  , nθ  , nφ  )
ηSM       =   @zeros(nr  , nθ  , nφ  )
DT_R      =   @zeros(nr  , nθ  , nφ  )
τII       =   @zeros(nr  , nθ  , nφ  )
vpdt      = dr*CFL       # numerical parameter
@parallel initialize_geometry!(ρ, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ρl, ρc, ρs)
@parallel initialize_geometry!(η, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ηl, ηc, ηs)
η_ini    .= η            # save initial (linear) viscosity field
println("done.")
# action -------------------------------------------------
println("Starting calculation (nr=$nr, nθ=$nθ , nφ=$nφ)")
err_evo = []; iter_evo = []; t_tic = 0.0; ittot = 0
for it = 1:nt # time loop
    if (it == 2)    ηc    = ηc_2                   # convergence step 2
        @parallel initialize_geometry!(η, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ηl, ηc_2, ηs)
        η_ini .= η                                 # save initial (linear) viscosity
        vr     = max(ηm,ηl,ηc,ηs)/min(ηm,ηl,ηc,ηs)  # viscosity ratio max/min
        CFL    = 1.0/(2.0 + 4.5*log10(vr))         # Courant-Friedrichs-Lewy condition
        vpdt   = dr*CFL                            # numerical parameter
    end
    if (it == 3)    ηc    = ηc_3                   # convergence step 3
        @parallel initialize_geometry!(η, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ηl, ηc_3, ηs)
        η_ini .= η                                 # save initial (linear) viscosity
        vr     = max(ηm,ηl,ηc,ηs)/min(ηm,ηl,ηc,ηs)  # viscosity ratio max/min
        CFL    = 1.0/(2.0 + 4.5*log10(vr))         # Courant-Friedrichs-Lewy condition
        vpdt   = dr*CFL                            # numerical parameter
    end
    if (it == 4)    ηc    = ηc_4                   # convergence step 4
        @parallel initialize_geometry!(η, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ηl, ηc_4, ηs)
        η_ini .= η                                 # save initial (linear) viscosity
        vr     = max(ηm,ηl,ηc,ηs)/min(ηm,ηl,ηc,ηs)  # viscosity ratio max/min
        CFL    = 1.0/(2.0 + 4.5*log10(vr))         # Courant-Friedrichs-Lewy condition
        vpdt   = dr*CFL                            # numerical parameter
    end
    if (it == 5)    ηc    = ηc_5                   # convergence step 5
        @parallel initialize_geometry!(η, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ηl, ηc_5, ηs)
        η_ini .= η                                 # save initial (linear) viscosity
        vr     = max(ηm,ηl,ηc,ηs)/min(ηm,ηl,ηc,ηs)  # viscosity ratio max/min
        CFL    = 1.0/(2.0 + 4.5*log10(vr))         # Courant-Friedrichs-Lewy condition
        vpdt   = dr*CFL                            # numerical parameter
    end
```

```
    if (it == 6)    ηc    = ηc_6                    # convergence step 6
        @parallel initialize_geometry!(η, R, θ, φ, r, ha, hc, hr, hl, dl, dh, dt, ηl, ηc_6, ηs)
        η_ini .= η                                  # save initial (linear) viscosity
        vr     = max(ηm,ηl,ηc,ηs)/min(ηm,ηl,ηc,ηs)  # viscosity ratio max/min
        CFL    = 1.0/(2.0 + 4.5*log10(vr))          # Courant-Friedrichs-Lewy condition
        vpdt   = dr*CFL                             # numerical parameter
    end
    if (it == 7)    n_exp = n_exp_PL     end        # convergence step 7
    if (it == 8)    n_exp = n_exp_PL2    end        # convergence step 8
    for iter = 1:maxiter        # pseudo-transient loop
        if (it==1 && iter==11) GC.gc(); t_tic = Base.time() end     # chronometer
        # SOLVER --------------------------------------------
        @parallel equal3!(VR_err, Vθ_err, Vφ_err, VR, Vθ, Vφ)
        @parallel maxloc!(ηSM, η)
        copy_BC!(ηSM)
        @parallel compute_P!(P   , DT_R, τ_RR, τ_θθ, τ_φφ, τ_Rθ, τ_Rφ, τ_θφ, VR, Vθ, Vφ, R, θ, η, ηSM,
                             _dr, _dθ , _dφ , dmp , vpdt, lr)
        @parallel compute_TII!(τII, τ_RR, τ_θθ, τ_φφ, τ_Rθ, τ_Rφ, τ_θφ)
        copy_BC!(τII)
        @parallel power_law!(η, τII, η_ini, τ_C, n_exp)
        @parallel compute_V!(VR , Vθ , Vφ , DT_R, τ_RR, τ_θθ, τ_φφ, τ_Rθ, τ_Rφ, τ_θφ, P, R, θ, ρ,
                             _dr, _dθ, _dφ, g)
        @parallel BC_R!(VR, Vθ, Vφ)
        @parallel BC_θ!(VR, Vθ, Vφ)
        @parallel BC_φ!(VR, Vθ, Vφ)
        # pseudo-transient loop exit criteria ------------------------------
        if iter % nout == 0
            @parallel check_err!(VR_err, Vθ_err, Vφ_err, VR, Vθ, Vφ)
            err_vr = maximum(abs.(VR_err))./maximum(abs.(VR))
            err_vθ = maximum(abs.(Vθ_err))./maximum(abs.(Vθ))
            err_vφ = maximum(abs.(Vφ_err))./maximum(abs.(Vφ))
            if isnan(err_vφ)
                err    = max(err_vr, err_vθ)
            else
                err    = max(err_vr, err_vθ, err_vφ)
            end
            # post-processing
            push!(err_evo, err)
            push!(iter_evo,iter)
            if do_viz  display(scatter(iter_evo, err_evo, xaxis=:log,yaxis=:log,legend=false))  end
            @printf("it %d, iter %d, err=%1.3e \n", it, iter, err)
            # pseudo-transient loop exit criteria
            if (err<εnonl && iter>20)  iter_end = iter; break; end
        end
        ittot = iter
    end
    # SAVING ----------------------------------------------
    if do_save
        !ispath("./output_$(it)") && mkdir("./output_$(it)")
        Save_phys(num  , it , sl, sη    , sρ, ηm , ηl, ηc, ηs, ρm, ρl, ρc, ρs, hm, hl, hr, hc, ha,
                  n_exp, τ_C, vr, g*1e15, r , dmp)
        Save_infos(num, it, lr, lθ, lφ, nr, nθ, nφ, εnonl*1e7, 0.0)
        SaveArray("R"       , R       , it)
        SaveArray("TH"      , θ       , it)
        SaveArray("PH"      , φ       , it)
        SaveArray("ETAS"    , η       , it)
        SaveArray("RHO"     , ρ       , it)
        SaveArray("P"       , P       , it)
        SaveArray("VR"      , VR      , it)
        SaveArray("VTH"     , Vθ      , it)
        SaveArray("VPH"     , Vφ      , it)
        SaveArray("TAU_RR"  , τ_RR    , it)
        SaveArray("TAU_THTH", τ_θθ    , it)
        SaveArray("TAU_PHPH", τ_φφ     , it)
        SaveArray("TAU_RTH" , τ_Rθ    , it)
        SaveArray("TAU_RPH" , τ_Rφ    , it)
        SaveArray("TAU_THPH", τ_θφ    , it)
        SaveArray("TII"     , τII     , it)
    end
end
```

```
    # Performance ----------------------------------------------
    wtime    = Base.time() - t_tic
    # Effective main memory access per iteration [GB]
    A_eff    = (4*2 + 6*2 + 2 + 2*2)/1e9*nr*nθ*nφ*sizeof(Data.Number)
    # (Lower bound of required memory access: Te has to be read and written: 2 whole-array memaccess;
    #  Ci has to be read: : 1 whole-array memaccess)
    wtime_it = wtime/(ittot-10)                                       # Execution time per iteration [s]
    T_eff    = A_eff/wtime_it                                         # Effective memory throughput [GB/s]
    @printf("Total iters = %d (%d steps), time = %1.3e sec (@ T_eff = %1.2f GB/s) \n",
            ittot, nt, wtime, round(T_eff, sigdigits=3))
    if do_save_p
        !ispath("./out_perf") && mkdir("./out_perf")
        open("./out_perf/out_SPH_3D_PERF.txt","a") do io
            println(io, "$(nr) $(nθ) $(nφ) $(dmp) $(ittot) $(wtime) $(A_eff) $(wtime_it) $(T_eff)")
        end
    end
    # SAVING ------------------------------------------------
    if do_save
        !ispath("./output_999") && mkdir("./output_999")
        err_evo  = Data.Array(err_evo)
        iter_evo = Data.Array(iter_evo)
        Save_phys(num  , 999, sl, sη    , sρ, ηm , ηl, ηc, ηs, ρm, ρl, ρc, ρs, hm, hl, hr, hc, ha,
                  n_exp, τ_C, vr, g*1e15, r , dmp)
        Save_infos(num, 999, lr, lθ, lφ, nr, nθ, nφ, εnonl*1e7, wtime)
        SaveArray("err_evo" , err_evo , 999)
        SaveArray("iter_evo", iter_evo, 999)
        SaveArray("R"       , R       , 999)
        SaveArray("TH"      , θ       , 999)
        SaveArray("PH"      , φ       , 999)
        SaveArray("ETAS"    , η       , 999)
        SaveArray("RHO"     , ρ       , 999)
        SaveArray("P"       , P       , 999)
        SaveArray("VR"      , VR      , 999)
        SaveArray("VTH"     , Vθ      , 999)
        SaveArray("VPH"     , Vφ      , 999)
        SaveArray("TAU_RR"  , τ_RR    , 999)
        SaveArray("TAU_THTH", τ_θθ    , 999)
        SaveArray("TAU_PHPH", τ_φφ    , 999)
        SaveArray("TAU_RTH" , τ_Rθ    , 999)
        SaveArray("TAU_RPH" , τ_Rφ    , 999)
        SaveArray("TAU_THPH", τ_θφ    , 999)
        SaveArray("TII"     , τII     , 999)
    end
    return
end

@time SPH_3D_Plateau()
```