The discontinuous Galerkin method for free surface and subsurface flows in geophysical applications

Das unstetige Galerkinverfahren für Strömungen mit freier Oberfläche und im Grundwasserbereich in geophysikalischen Anwendungen

Der Naturwissenschaftlichen Fakultät

der
Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur
Erlangung des Doktorgrades Dr. rer. nat

vorgelegt von

Balthasar Reuter

Als Dissertation genehmigt von der Naturwissenschaftlichen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	6. Dezember 2019
Vorsitzender des Promotionsorgans:	Prof. Dr. Georg Kreimer
Gutachter/in:	Prof. Dr. Peter Knabner Prof. Dr. Clint Dawson

Contents

Danksagung

Die vorliegende Arbeit und die darin enthaltenen Publikationen sind das Ergebnis meiner mehr als als fünf Jahre dauernden Beschäftigung am Lehrstuhl für Angewandte Mathematik I. Ich möchte den Personen, die mich in dieser Zeit begleitet, inspiriert und unterstützt haben, an dieser Stelle aufrichtig danken.

Mein größter Dank gilt Herrn Prof. Dr. Vadym Aizinger, der nicht offiziell als Betreuer dieser Arbeit fungieren durfte, diese Aufgabe in der Praxis aber vollumfänglich übernommen hat. Er war es, der mir während eines Kompaktkurses in Stockholm einen ersten Einblick in unstetige Galerkinverfahren gab und mich schließlich eingeladen hat, nach Erlangen zurück zu kehren. Als Betreuer gab Vadym stets die Richtung vor, gewährte viel Freiheit in der Umsetzung und war offen für interessante Ideen und Fragestellungen, was mir erst die Vielfalt der in dieser Arbeit behandelten Themen ermöglichte. Mit seiner großen Expertise, seiner Bereitschaft, jederzeit Fragen oder Probleme zu diskutieren, und seiner Gabe, ad hoc immer eine Lösungsidee oder einen Ansatz zu entwickeln, war das gemeinsame Arbeiten stets eine Freude, lehrreich und fruchtbar. Als Ansprechpartner war er nicht nur für fachliche Fragen sondern auch weit darüber hinaus immer verfügbar und bot Rat und Unterstützung bei jeder Gelegenheit. Für all das bin ich Vadym zutiefst dankbar und wünsche ihm und seiner Familie für den Start in Bayreuth nur das Beste.

Ebenso gilt mein Dank Herrn Prof. Dr. Peter Knabner für die Bereitschaft, die offizielle Rolle als Betreuer dieser Arbeit zu übernehmen. Ich danke ihm für die wertvollen fachlichen Impulse und Anregungen, für die Aufnahme in seinen Lehrstuhl und nicht zuletzt dafür, dass er stets für die Möglichkeit einer Weiterbeschäftigung gesorgt hat und mir so viele Sorgen abgenommen hat. Bedanken möchte ich mich auch für das in mich gesetzte Vertrauen durch die Beteiligung am Buchprojekt *Mit Mathe richtig anfangen* – die Arbeit daran, zusammen auch mit Herrn Dr. Raphael Schulz, zeigte mir einige neue Aspekte auf und ich empfand sie als rundum unterhaltsam und wertvoll. Ein gewisser Kinderreichtum im Kollegenkreis, letztendlich auch durch meinen eigenen Nachwuchs, prägte meine Zeit am Lehrstuhl, und die Toleranz und Flexibilität von Herrn Knabner gegenüber den dadurch enstandenen kurzfristigen Ausfällen oder notwendig gewordener Zwischenbetreuung am Arbeitsplatz ist bemerkenswert und verdient besonderen Dank.

Bedanken möchte ich mich bei allen Mitautoren der Zeitschriftenbeiträge für die interessante und gute Zusammenarbeit, ergiebige fachliche Diskussionen und den stets freundschaftlichen und herzlichen Umgang miteinander. Hervorheben möchte ich Herrn Prof. Dr. Florian Frank, von dem die Idee für das Softwarepaket FESTUNG ausging und der mit seiner gründlichen und fundierten Arbeitsweise jede Softwareversion und Publikation erst perfektionierte. Erwähnen möchte ich auch Herrn Dr. Andreas Rupp, dessen beeindruckendes Wissen und aufopfernde Hilfsbereitschaft ich viel zu oft in Anspruch nehmen musste. Florian und Andreas, sowie meinen ehemaligen Kommilitonen Dominik Ernst, Julian Ham-

mer, Dr. Christoph Rachinger und Dominik Thönnes danke ich auch für das kritische Lesen des Manuskripts und die hilfreichen Anmerkungen.

Meinen weiteren aktuellen und ehemaligen Kollegen am AM1, Prof. Dr. Fabian Brunner, Tobias Elbinger, Dr. Markus Gahn, Hennes Hajduk, Dr. Matthias Herz, Dr. Fabian Klingbeil, Prof. Dr. Serge Kräutle, Dr. Stefan Metzger, PD Dr. Maria Neuss-Radu, Dr. Nadja Ray, Dr. Stefan Schiessl, Dr. Raphael Schulz, Oliver Sieber, Dr. Daniel Tenbrinck, Dr. Philipp Wacker, Patrick Weiss, Philipp Werner und meinem Bürokollegen Hubertus Grillmeier danke ich für das freundschaftliche Arbeitsklima sowie unterhaltsame Diskussionen über fachliche Themen und darüber hinaus. Besonderer Dank gilt Herrn Dr. Alexander Prechtel, der, über seine eigentlichen Aufgaben hinaus, als Organisator und Mediator des Lehrstuhls stets als Ansprechpartner zur Verfügung stand. Mit vielen verbindet mich auch über den Arbeitsalltag hinaus eine freundschaftliche Beziehung. Für die Bewältigung aller bürokratischen Hürden und Probleme danke ich den Mitarbeitern des Sekretariats, Frau Astrid Bigott, Frau Monika Bittan, Herrn Sebastian Czop, Frau Gisela Jukl, Frau Juliett Kille und Frau Cornelia Weber. Erwähnen möchte ich auch Herrn Prof. Dr. Eberhard Bänsch und die Mitarbeiter seines Lehrstuhls AM3, deren Türen immer offen standen, sowohl für fachliche Fragen als auch eine gemütliche Kaffeerunde.

Abschließend will ich meiner Familie danken: Meinen Eltern dafür, dass sie mich seit jeher unterstützt, gefördert und ermunter haben, das zu tun, was mir Freude bereitet. Vor allem jedoch meiner Frau, Stefanie, und meinen Töchtern, Lorena und Jana, denen ich für ihre Unterstützung nicht genug danken kann. Sie mussten viel Geduld und Verständnis aufbringen, wenn sie wegen langer Arbeitstage oder Dienstreisen auf mich verzichten mussten, und sie konnten mich, wie nichts sonst, von Frust oder Erschöpfung ablenken, und sich mit mir über Erfolge freuen. Ganz besonders freue ich mich, dass sie bereit sind, auch weitere Abenteuer und Neuanfänge mit mir zu unternehmen.

Balthasar Reuter, im Mai 2019

Titel, Zusammenfassung und Aufbau der Arbeit

Das unstetige Galerkinverfahren für Strömungen mit freier Oberfläche und im Grundwasserbereich in geophysikalischen Anwendungen

Die vorliegende Arbeit beschäftigt sich mit der mathematischen Modellierung und numerischen Simulation von Strömungen in geophysikalischen Anwendungen mit freier Oberfläche und im gesättigten Grundwasserbereich, sowie deren Kopplung. Der Fokus der Arbeit liegt auf der Anwendung unstetiger Galerkinverfahren (*discontinuous Galerkin*, dG) zur Diskretisierung der mathematischen Modelle und deren effiziente Umsetzung auf modernen Parallelrechnerarchitekturen.

Die Bandbreite an geophysikalischen Anwendungen für Strömungen mit freier Oberfläche reicht von kleinen Seen über Flüsse und Ästuare bis hin zu regionalen Ozean- und globalen Zirkulationsmodellen. Die Tatsache, dass in solchen Situationen die vertikale Ausdehnung wesentlich geringer ist als die horizontale Größe des betrachteten Gebiets, motiviert die Verwendung der dreidimensionalen Flachwassergleichungen als mathematisches Modell. Dieses lässt sich unter Verwendung der hydrostatischen Annahme und der Boussinesq-Approximation aus den Navier-Stokes-Gleichungen herleiten. Die numerische Simulation von Strömungen mit freier Oberfläche ist wichtig für eine Vielzahl von Anwendungen, z. B. Umweltstudien, Tsunami- und Sturmwarnsysteme oder als Teil gekoppelter Simulationen in Klimaprojektionen.

Unter der Erdoberfläche fließt Wasser (oder andere Fluide) in wasserdurchlässigen Schichten bestehend aus porösen Materialien (z. B. Kalkstein, Sand oder Kies), angetrieben von Gradienten in der potentiellen Energie (Standrohrspiegelhöhe). Die vorliegende Arbeit ist auf gesättigte Situationen beschränkt, in denen der gesamte zusammenhängende Porenraum mit Wasser ausgefüllt ist. Wie allgemein üblich wird ein gemitteltes Modell basierend auf dem Darcy-Gesetz verwendet, um das Auflösen der Geometrie der Porenmatrix zu vermeiden. Die Simulation solcher Grundwasserszenarien wird beispielwseise verwendet, um Folgenabschätzungen für Trinkwasserbrunnen zu erstellen oder die Ausbreitung von Schadstoffen im Untergrund nachzuvollziehen.

Wenn die Interaktion von Gewässern mit einem darunter liegenden Grundwassersystem untersucht werden soll, erfordert dies gekoppelte Modelle. Dies ermöglicht z. B. die Simulation des Einsickerns von Hochwasser in den Untergrund oder der Ausbreitung von Schad-

stoffen zwischen Oberflächengewässern und Grundwasserleitern. In der vorliegenden Arbeit werden dazu getrennte Modelle für jedes der beiden Teilgebiete betrachtet und diese an einer scharfen Grenze zwischen den Gebieten mittels geeigneter Bedingungen gekoppelt, die Massenerhaltung und Stetigkeit des Drucks sicher stellen.

Die zugrunde liegenden mathematischen Modelle werden als Erhaltungssätze formuliert und bestehen aus Systemen partieller Differentialgleichungen. Um diese numerisch lösen zu können, werden diese in Ort und Zeit diskretisiert, wobei für die Ortsdiskretisierung ausschließlich unstetige Galerkinverfahren eingesetzt werden. Diese relativ junge Familie von Diskretisierungsverfahren bietet eine Reihe von Vorteilen, leidet aber unter dem Nachteil einer großen Anzahl lokaler Freiheitsgrade, was einen hohen Rechenaufwand im Vergleich zu anderen etablierten Verfahren nach sich zieht. Eine effiziente Implementierung und Ausnutzung moderner Parallelrechner ist daher besonders wichtig, um die Rechenzeit in einem vertretbaren Rahmen zu halten.

Unstetige Galerkinverfahren nutzen den in Finite Elemente-Verfahren üblichen Ansatz einer schwachen Formulierung, unterscheiden sich aber dadurch, dass sie Ansatz- und Testfunktionen aus Räumen verwenden, die keine Stetigkeitsbedingung über Elementgrenzen hinweg fordern. Die dadurch ebenfalls unstetige Lösung wird zwischen benachbarten Elementen nur über numerische Flüsse gekoppelt, was eine Ähnlichkeit zu Finite Volumen-Verfahren darstellt. Im Rahmen dieser Arbeit findet insbesondere die *local discontinuous Galerkin*-Methode (LDG) Anwendung, in der gemischte Formulierungen verwendet werden, um Differentialoperatoren zweiter (oder höherer) Ordnung als Systeme erster Ordnung darzustellen, bevor diese mit dem Galerkinansatz behandelt werden. Neben den primären Unbekannten erhält man so auch deren Flüsse als Hilfsunbekannte. Sowohl das System der dreidimensionalen Flachwassergleichungen als auch die Grundwassergleichung werden mit LDG diskretisiert und die Stabilität des diskreten gekoppelten Systems bewiesen.

Um nichtphysikalische Oszillationen und Verletzungen diskreter Maximumsprinzipien in dG-Lösungen zu vermeiden, werden in dieser Arbeit knotenbasierte Steigungsbegrenzer (*slope limiter*) verwendet. Diese Nutzen im Zusammenspiel mit TVD-Zeitschrittverfahren (*total variation diminishing*) die Tatsache, dass für den stückweise konstanten Teil der Lösung die Monotonität garantiert ist. Für einen bestimmten Knoten lassen sich dann aus den stückweise konstanten Teilen der Lösung auf allen Elementen, zu denen dieser Knoten gehört, obere und untere Schranken für die Lösung in diesem Punkt ermitteln. Die Steigung der Lösung wird anschließend elementweise so begrenzt, dass diese Schranken in allen Knoten des jeweiligen Elements eingehalten werden, ohne dabei die Massenerhaltung innerhalb des Elements zu verletzen. Die vorliegende Arbeit liefert eine geschlossene Formulierung für knotenbasierte Steigungsbegrenzer beliebiger Ordnung und enthält einen Vorschlag für eine striktere Limitierungstechnik sowie einen anisotropen Steigungsbegrenzer. Zusätzlich wird ein recheneffizientes und weiterhin massenerhaltendes Transportschema beschrieben, das Vorteile für Transportszenarien bietet, in denen sich die Lösung nur in einem kleinen Teil des Gebietes ändert. Dazu werden die Maximums- und Minimumsinformationen verwendet, die im Rahmen des Limitierungsalgorithmus ohnehin ermittelt werden, um die Berechnung der numerischen Flüsse nur auf den Elementen durchzuführen, auf denen eine Änderung der Lösung erwartet wird. Folglich wird das Schema als lokal gefilterter Transport (*locally filtered transport*, LFT) bezeichnet.

Viele geophysikalische Anwendungen für Strömungen mit freier Oberfläche oder im Grundwasserbereich zeichnen sich durch große Rechengebiete und lange Simulationszeiträume aus, sodass die numerische Simulation mit erheblichem Rechenaufwand verbunden ist. Um den

Zeitaufwand für Simulationen auch bei hohen Anforderungen an die örtliche Auflösung möglichst gering zu halten, wird auf Parallelrechner und Technologien des Hochleistungsrechnen (*high performance computing*, HPC) zurückgegriffen. Zu den Vorteilen der LDG-Methode zählt insbesondere die Abwesenheit einer globalen Kopplung der Freiheitsgrade, wodurch sich das Verfahren besonders gut für paralleles Rechnen eignet und hervorragende Skalierbarkeit zeigt.

Gleichzeitig spielt für HPC-Systeme und -Anwendungen die Energieeffizienz der Hardware und Applikationen eine zunehmende Rolle. Gerade die gute Skalierbarkeit von dG-Verfahren kann hier einen Beitrag leisten: Der Einsatz von Prozessorarchitekturen aus dem Niedrigenergiebereich verspricht eine geringere Leistungsaufnahme und einen reduzierten Gesamtenergiebedarf für die Berechnung. Damit einher geht allerdings die vergleichsweise geringe Rechenleistung solcher Prozessoren, die wieder eine verlängerte Rechenzeit zur Folge hat. In einer Vergleichsstudie zwischen aktuellen Intel Haswell-Prozessoren sowie ARM-Prozessoren für das regionale Ozeanmodell UTBEST3D wird gezeigt, dass dessen gute Skalierbarkeit den Einsatz von einer größeren Zahl von Niedrigenergieprozessoren erlaubt, um schließlich einen geringeren Energiebedarf bei gleichbleibender Rechenzeit zu erzielen.

Die praktische Umsetzung dieser Verfahren erfolgte in zwei Software-Paketen: Der Autor dieser Arbeit entwickelt gemeinsam mit Florian Frank und Vadym Aizinger FESTUNG (*Finite Element Simulation Toolbox for UNstructured Grids*), eine neue MATLAB / GNU Octave Toolbox für die effiziente Nutzung und Entwicklung unstetiger Galerkin-Verfahren. Durch den konsequenten Einsatz vektorisierter Operationen liefert diese gute Recheneffizienz für kleine und mittlere Problemgrößen und eignet sich besonders als *Rapid Prototyping*-Werkzeug und für den Einsatz in der Lehre. Klare Schnittstellen und eine umfangreiche Dokumentation erleichtern dabei den Umgang mit diesem Softwarepaket. In einer Reihe von Veröffentlichungen ist die Diskretisierung und Implementierung von dG-Verfahren in dieser Toolbox für verschiedene Operatoren beschrieben.

Das regionale Ozeanmodell UTBEST3D nutzt die LDG-Methode für das System der dreidimensionalen Flachwassergleichungen. Das Modell ist in C++ implementiert und unter Verwendung von MPI und OpenMP parallelisiert. Im Rahmen dieser Arbeit wurde die OpenMP-Parallelisierung neu hinzugefügt und Verbesserungen der Recheneffizienz sowie der Skalierbarkeit vorgenommen. Darüber hinaus wurde die Portierbarkeit auf eine Reihe von unterschiedlichen Architekturen untersucht.

Aufbau der Arbeit

Die vorliegende Arbeit besteht aus zwei Teilen: eine Reihe begutachteter und bereits publizierter Zeitschriftenbeiträge, sowie eine erweiterte Zusammenfassung, die diese Schriften miteinander verknüpft und in einen größeren fachwissenschaftlichen Kontext einbettet.

Den ersten Teil bildet die erweiterte Zusammenfassung bestehend aus sechs Kapiteln. Nach einer allgemeinen Einführung in das Anwendungsgebiet in Kapitel 1, werden in Kapitel 2 kompakt die mathematischen Modelle für Strömungen mit freier Oberfläche, im Grundwasserbereich und deren Kopplung beschrieben. Ein Überblick über die Geschichte und Varianten unstetiger Galerkinverfahren, sowie die Diskretisierung der zuvor eingeführten Modelle findet sich in Kapitel 3. Ebenfalls in diesem Kapitel werden Steigungsbegrenzer und das lokal gefilterte Transportschema beschrieben. Kapitel 4 liefert eine Einführung in aktuelle Rechenarchitekturen, beschreibt Metriken und Maßnahmen für effizientes Rechnen

auf der Ebene einzelner Rechenknoten und Aspekte der Parallelisierung und Skalierbarkeit. Abgeschlossen wird dieses Kapitel durch Energieeffizienz-Betrachtungen. Eine kurze Beschreibung der entwickelten Softwarepakete findet sich in Kapitel 5 und eine Zusammenfassung sowie ein Ausblick in Kapitel 6 runden den ersten Teil ab.

Den zweiten Teil der Arbeit bilden Nachdrucke von acht bereits begutachteten und publizierten Zeitschriftenbeiträgen, die im folgenden Abschnitt genannt werden.

Publizierte Zeitschriftenbeiträge

[P1] F. Frank, B. Reuter, V. Aizinger und P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part I: Diffusion operator". In: *Computers and Mathematics with Applications* 70.1 (2015), S. 11–46. DOI: 10.1016/j.camwa.2015.04.013.

[P2] B. Reuter, V. Aizinger und H. Köstler. "A multi-platform scaling study for an OpenMP parallelization of a discontinuous Galerkin ocean model". In: *Computers & Fluids* 117 (2015), S. 325–335. DOI: 10.1016/j.compfluid.2015.05.020.

[P3] B. Reuter, V. Aizinger, M. Wieland, F. Frank und P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part II: Advection operator and slope limiting". In: *Computers and Mathematics with Applications* 72.7 (2016), S. 1896–1925. DOI: 10.1016/j.camwa.2016.08.006.

[P4] M. Geveler, B. Reuter, V. Aizinger, D. Göddeke und S. Turek. "Energy efficiency of the simulation of three-dimensional coastal ocean circulation on modern commodity and mobile processors". In: *Computer Science – Research and Development* 31.4 (2016), S. 225–234. DOI: 10.1007/s00450-016-0324-5.

[P5] V. Aizinger, A. Kosík, D. Kuzmin und B. Reuter. "Anisotropic slope limiting for discontinuous Galerkin methods". In: *International Journal for Numerical Methods in Fluids* 84.9 (2017), S. 543–565. DOI: 10.1002/fld.4360.

[P6] A. Jaust, B. Reuter, V. Aizinger, J. Schütz und P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part III: Hybridized discontinuous Galerkin (HDG) formulation". In: *Computers and Mathematics with Applications* 75.12 (2018), S. 4505–4533. DOI: 10.1016/j.camwa.2018.03.045.

[P7] H. Hajduk, B. R. Hodges, V. Aizinger und B. Reuter. "Locally Filtered Transport for computational efficiency in multi-component advection-reaction models". In: *Environmental Modelling & Software* 102 (2018), S. 185–198. DOI: 10.1016/j.envsoft.2018.01.003.

[P8] B. Reuter, A. Rupp, V. Aizinger und P. Knabner. "Discontinuous Galerkin method for coupling hydrostatic free surface flows to saturated subsurface systems". In: *Computers and Mathematics with Applications* 77.9 (2019), S. 2291–2309. DOI: 10.1016/j.camwa.2018.12.020.

Die Beiträge von Balthasar Reuter zu den einzelnen Publikationen sind wie folgt:

[P1]: Der Autor dieser Arbeit diskretisierte und implementierte das Modell zu gleichen Teilen und in enger Zusammenarbeit mit Florian Frank, verfasste große Teile des

Manuskripts und war der Alleinverantwortliche für die Auswertung der Recheneffizienz des Algorithmus.

[P2]: Der Autor dieser Arbeit entwickelte und implementierte die OpenMP-Parallelisierung, führte die numerischen Experimente inklusive des Vergleichs verschiedener Architekturen und der Laufzeitanalyse durch, erstellte die Struktur und verfasste den Großteil des Artikels, inklusive aller Abbildungen.

[P3]: Der Autor dieser Arbeit diskretisierte und implementierte das Modell in FESTUNG, inklusive der bibliotheksartigen Umsetzung der Steigungsbegrenzer für beliebige Ordnungen, entwickelte das striktere Verfahren zur Steigungsbegrenzung, führte die numerischen Experimente durch, erstellte die Struktur und verfasste den Großteil des Artikels, inklusive aller Abbildungen.

[P4]: Der Autor dieser Arbeit entwickelte die numerischen Testfälle und bereitete diese vor, verfasste große Teile des Artikels und lieferte die Mehrheit der Abbildungen.

[P5]: Der Autor dieser Arbeit lieferte entscheidende Ideen zur Verbesserung des anisotropen Steigungsbegrenzers, verifizierte die Methode und verbesserte Text und Abbildungen in der Publikation.

[P6]: Der Autor dieser Arbeit beriet Alexander Jaust bei der Implementierung des Modells, verbesserte die Recheneffizienz des Programms in erheblichem Maße, führte die numerischen Experimente durch und verfasste einen Großteil des Manuskripts, inklusive aller Abbildungen.

[P7]: Der Autor dieser Arbeit beriet Hennes Hajduk bei der Implementierung des gekoppelten Modells, wobei das Transportmodell auf dem vom Autor dieser Arbeit in [P3] implementierten Standardmodell für eine einzelne Transportgröße beruht, passte den Flachwasserlöser an die überarbeitete Struktur von FESTUNG an und lieferte entscheidende Ideen für die effiziente und massenerhaltende Umsetzung des LFT-Algorithmus.

[P8]: Der Autor dieser Arbeit analysierte, in enger Zusammenarbeit mit Vadym Aizinger, das Oberflächen-Teilsystem und das gekoppelte Modell, implementierte das gekoppelte Modell in FESTUNG, führte die numerischen Experimente durch und verfasste große Teile des Artikels, inklusive aller Abbildungen.

Folgende Ergebnisberichte und Konferenzbeiträge, die im selben Zeitraum entstanden aber nicht Teil dieser Arbeit sind, liefern weitere Details zu den behandelten Themen:

[Pp1] B. Reuter, A. Rupp, V. Aizinger, F. Frank und P. Knabner. “FESTUNG: A MATLAB / GNU Octave toolbox for the discontinuous Galerkin method. Part IV: Generic problem framework and model-coupling interface”. Preprint. 2018. URL: https://arxiv.org/abs/1806.03908.

[R1] B. Reuter und V. Aizinger. *KONWIHR III-Project: UTBEST3D*. Techn. Ber. Friedrich-Alexander University Erlangen-Nürnberg, März 2015. URL: https://www1.am.uni-erlangen.de/~reuter/ReuterAizinger_2015_KONWIHR.pdf.

[C1] D. Schoenwetter, A. Ditter, V. Aizinger, B. Reuter und D. Fey. “Cache Aware Instruction Accurate Simulation of a 3-D Coastal Ocean Model on Low Power Hardware”. In: *Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications – Volume 1: SIMULTECH*. SCITEPRESS, 2016, S. 129–137. DOI: 10.5220/0006006501290137.

[C2] B. Reuter und V. Aizinger. “Boosting node-level performance with compile-time generated and evaluated lookup of loop lengths”. In: *ISC HPC Conference*. Poster. Frankfurt, Juni 2016. URL: https://www1.am.uni-erlangen.de/research/poster/ReuterAizinger2016.pdf.

[C3] B. Reuter, F. Frank und V. Aizinger. “FESTUNG – Finite Element Simulation Toolbox for UNstructured Grids”. In: *SIAM Conference on Mathematical and Computational Issues in the Geosciences*. Poster. Erlangen, Sep. 2017. URL: https://www1.am.uni-erlangen.de/research/poster/ReuterFA2017.pdf.

Preface

This thesis consists of eight previously published, peer-reviewed journal articles listed below and an extended summary of the papers. The summary is structured into four main parts, providing an introduction into the most important mathematical models, their discretization, issues related to the computational performance of these numerical models, and the software packages in which these are implemented.

The extended summary in Part A is not a sequence of the papers but takes a more wholistic point of view: It covers the relevant ideas from the articles in each part, thus highlighting the connection between the individual publications, and points to the corresponding papers for further details. Reprints of the published articles are included in Part B.

[P1] F. Frank, B. Reuter, V. Aizinger, and P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part I: Diffusion operator". In: *Computers and Mathematics with Applications* 70.1 (2015), pp. 11–46. DOI: 10.1016/j.camwa.2015.04.013.

[P2] B. Reuter, V. Aizinger, and H. Köstler. "A multi-platform scaling study for an OpenMP parallelization of a discontinuous Galerkin ocean model". In: *Computers & Fluids* 117 (2015), pp. 325–335. DOI: 10.1016/j.compfluid.2015.05.020.

[P3] B. Reuter, V. Aizinger, M. Wieland, F. Frank, and P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part II: Advection operator and slope limiting". In: *Computers and Mathematics with Applications* 72.7 (2016), pp. 1896–1925. DOI: 10.1016/j.camwa.2016.08.006.

[P4] M. Geveler, B. Reuter, V. Aizinger, D. Göddeke, and S. Turek. "Energy efficiency of the simulation of three-dimensional coastal ocean circulation on modern commodity and mobile processors". In: *Computer Science – Research and Development* 31.4 (2016), pp. 225–234. DOI: 10.1007/s00450-016-0324-5.

[P5] V. Aizinger, A. Kosík, D. Kuzmin, and B. Reuter. "Anisotropic slope limiting for discontinuous Galerkin methods". In: *International Journal for Numerical Methods in Fluids* 84.9 (2017), pp. 543–565. DOI: 10.1002/fld.4360.

[P6] A. Jaust, B. Reuter, V. Aizinger, J. Schütz, and P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part III: Hybridized discontinuous Galerkin (HDG) formulation". In: *Computers and Mathematics with Applications* 75.12 (2018), pp. 4505–4533. DOI: 10.1016/j.camwa.2018.03.045.

[P7] H. Hajduk, B. R. Hodges, V. Aizinger, and B. Reuter. "Locally Filtered Transport for computational efficiency in multi-component advection-reaction models". In: *Environmental Modelling & Software* 102 (2018), pp. 185–198. DOI: 10.1016/j.envsoft.2018.01.003.

[P8] B. Reuter, A. Rupp, V. Aizinger, and P. Knabner. "Discontinuous Galerkin method for coupling hydrostatic free surface flows to saturated subsurface systems". In: *Computers and Mathematics with Applications* 77.9 (2019), pp. 2291–2309. DOI: 10.1016/j.camwa.2018.12.020.

Balthasar Reuter's contributions to the individual publications are summarized as follows:

[P1]: The author of this thesis discretized and implemented the model in equal parts and in close collaboration with Florian Frank, wrote major parts of the paper, and was solely responsible for the evaluation of the computational performance.

[P2]: The author of this thesis designed and implemented the OpenMP parallelization, conducted the numerical experiments including the cross-platform study and performance analysis, and produced the outline and major parts of the paper, including all figures.

[P3]: The author of this thesis discretized and implemented the model in the FESTUNG framework, including the arbitrary order slope limiters in a portable and library-like fashion, proposed the stricter limiting procedure, conducted the numerical experiments, and produced the outline and major parts of the paper, including all figures.

[P4]: The author of this thesis designed and prepared the numerical testcases, wrote parts of the paper, and contributed the majority of figures.

[P5]: The author of this thesis contributed ideas to the improvement of the anisotropic limiting procedure, verified the effectiveness of the method, and enhanced text and figures in the publication.

[P6]: The author of this thesis advised Alexander Jaust on the implementation of the model, contributed significant performance improvements to the code, conducted the numerical experiments, and wrote major parts of the paper including all figures.

[P7]: The author of this thesis advised Hennes Hajduk on the implementation of the coupled model, with the transport model based on the single-component standard model implemented by the author in [P3], adapted the shallow–water solver to the updated structure of FESTUNG, and contributed essential ideas for an efficient and mass-conservative realisation of the Locally Filtered Transport algorithm in this software framework.

[P8]: The author of this thesis collaborated with Vadym Aizinger on the analysis of the free-surface subsystem and of the coupled model, implemented the coupled model in the FESTUNG framework, conducted the numerical experiments, and wrote major parts of the paper including all figures.

Further preprints, reports, and conference contributions by the author are listed below. These were published in the same period of time but not included in this thesis and provide additional details on topics covered in this work.

[Pp1] B. Reuter, A. Rupp, V. Aizinger, F. Frank, and P. Knabner. “FESTUNG: A MATLAB / GNU Octave toolbox for the discontinuous Galerkin method. Part IV: Generic problem framework and model-coupling interface”. Preprint. 2018. URL: https://arxiv.org/abs/1806.03908.

[R1] B. Reuter and V. Aizinger. *KONWIHR III-Project: UTBEST3D*. Tech. rep. Friedrich-Alexander University Erlangen-Nürnberg, Mar. 2015. URL: https://www1.am.uni-erlangen.de/~reuter/ReuterAizinger_2015_KONWIHR.pdf.

[C1] D. Schoenwetter, A. Ditter, V. Aizinger, B. Reuter, and D. Fey. “Cache Aware Instruction Accurate Simulation of a 3-D Coastal Ocean Model on Low Power Hardware”. In: *Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications – Volume 1: SIMULTECH*. SCITEPRESS, 2016, pp. 129–137. DOI: 10.5220/0006006501290137.

[C2] B. Reuter and V. Aizinger. “Boosting node-level performance with compile-time generated and evaluated lookup of loop lengths”. In: *ISC HPC Conference*. Poster. Frankfurt, June 2016. URL: https://www1.am.uni-erlangen.de/research/poster/ReuterAizinger2016.pdf.

[C3] B. Reuter, F. Frank, and V. Aizinger. “FESTUNG – Finite Element Simulation Toolbox for UNstructured Grids”. In: *SIAM Conference on Mathematical and Computational Issues in the Geosciences*. Poster. Erlangen, Sept. 2017. URL: https://www1.am.uni-erlangen.de/research/poster/ReuterFA2017.pdf.

Part A

Extended summary

Chapter

1 Introduction

Math, science, history,
unraveling the mysteries
that all started with the big bang![1]

1.1 Motivation

Free surface flows appear in a broad range of geophysical applications: small lakes, rivers, or estuaries via coastal and regional ocean to global ocean circulation. In the context of this thesis we consider shallow-water-type flows, i. e., situations that are characterized by a vertical dimension much smaller than typical horizontal scales. The underlying mathematical models are not limited to water bodies alone but also applicable to other fluids, e. g., air when considering atmospheric flows [157]. The primary driving forces largely depend on the spatial and temporal scales considered and can be as diverse as Coriolis acceleration, tidal forces, wind stress at the surface, landslides or earthquakes (triggering tsunamis), or free convection due to unstable stratifications [157]. Numerical simulations using the underlying models are required in many situations, such as environmental studies, storm surge simulations, or as part of coupled simulations in climate projections.

Subsurface flows consider the flow of water or other fluids below earth's surface, i. e., in permeable domains typically made up by porous media (such as sand or gravel), and can be further subdivided into saturated and unsaturated regimes. In the latter case, the pore space can contain a gas phase (e. g., air) next to the fluid phase; however, in this thesis, only the first case is considered where water fills the entire connected pore space. The driving forces for groundwater flow are gradients in potential energy (the hydraulic head) [58]. Understanding and predicting these types of flows plays an important role in many engineering and environmental applications such as impact assessment for water supplies or simulation of contaminant transport when planning pollutant remediation in aquifers [83].

In many environmental settings, situations arise which require the coupling of free surface and subsurface flows, for example wind influencing evaporation from soils, infiltration of overland flow into the soil during rainfall, or interaction and contaminant propagation between free surface water bodies and groundwater aquifers [134]. Treating these situations numerically is challenging: It requires employing two different models and a suitable method for coupling these at the interface.

Many application scenarios for free surface and subsurface flows (individually or in coupled settings) are characterized by large domain sizes and long simulation times. Hence, they need considerable amounts of computational work to achieve accurate solutions. In

[1] From the song *History of Everything* by *Barenaked Ladies*

order to obtain results within a reasonable time frame, the use of efficient algorithms and high performance computing resources is unavoidable. At the same time, parallelization finds its way into computing platforms on many levels, and energy efficiency becomes increasingly important. Consequently, time-to-solution is no longer the only relevant metric when judging the performance of hardware and software calling for novel approaches and computing architectures.

The scope of this thesis is the application of discontinuous Galerkin (dG) methods to mathematical models for free surface and subsurface flows with a strong focus on computational aspects. These models are formulated as conservation laws expressed in the form of partial differential equations, which make it necessary to transform the continuous descriptions into discrete approximations in order to obtain computable representations. For that, discontinuous Galerkin methods are a relatively recent technique that offers a wide flexibility. This comes at a price of high computational costs compared to other established methods, thus efficient implementation and effective use of parallel computers are mandatory to obtain accurate results in reasonable time.

1.2 Structure of this extended summary

Chapter 2 introduces the mathematical models for free surface flow, groundwater flow, and an approach to couple these two. A short overview of discontinuous Galerkin methods opens Chapter 3, which presents the dG discretization of the previously described mathematical models. It further includes a discussion of slope limiters and a computationally efficient transport scheme that builds upon information computed for slope limiters. Chapter 4 is concerned with computing aspects and includes a review of current computing architectures and relevant performance metrics. This is followed by some remarks about the software packages developed in the context of this work in Chapter 5. A short summary and outlook conclude this first part of the thesis. References to the author's publications are included throughout the text, and reprints of these articles represent Part B.

1.3 Notation

Throughout the text bold letters $\boldsymbol{v}$ denote vector-valued functions and sans-serif bold letters $\mathbf{A}$ are matrices / tensors. The outer product of two vectors $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{v} \in \mathbb{R}^n$ is written as $\boldsymbol{u} \otimes \boldsymbol{v} \coloneqq \boldsymbol{u}\,\boldsymbol{v}^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\nabla} \coloneqq (\partial_x, \partial_y, \partial_z)^{\mathrm{T}}$ is the vector differential operator. Further notation is introduced as required, in particular in Section 3.2 for notation relevant to discontinuous Galerkin methods.

Chapter

2 Mathematical models

2.1 Three-dimensional baroclinic shallow-water flow

The setting for free-surface flows in geophysical applications is a water body that is surrounded by a solid on the bottom, air on the top, and by solid or other water bodies (e. g., open sea, rivers) laterally, as depicted in Figure 2.1. The system of *three-dimensional shallow-water equations* (3D SWE), often also referred to as the *primitive (hydrostatic) equations*, is the most commonly-used model for three-dimensional free surface flows in geophysical domains. It comprises conservation equations for momentum and mass defined over a time-dependent domain with the top boundary moving in vertical direction, and boundary and initial conditions.[1] Additional equations model the transport of temperature, salinity, and turbulence quantities.

The 3D SWE are derived from *Reynolds-averaging* the *Navier–Stokes equations* defined in a rotating frame of reference

$$\partial_t(\rho\,\boldsymbol{u}) + \boldsymbol{\nabla}\cdot(\rho\,\boldsymbol{u}\otimes\boldsymbol{u}) + \boldsymbol{\nabla}p + \rho\begin{pmatrix}-f_\mathrm{c}\,v\\ f_\mathrm{c}\,u\\ g\end{pmatrix} - \boldsymbol{\nabla}\cdot\boldsymbol{\tau} = 0\,, \tag{2.1a}$$

$$\partial_t\rho + \boldsymbol{\nabla}\cdot(\rho\,\boldsymbol{u}) = 0\,, \tag{2.1b}$$

where $\boldsymbol{u} = (u, v, w)^\mathrm{T}$ is the fluid velocity, ρ its density, p the pressure, g the acceleration due to gravity, f_c the coriolis coefficient, and $\boldsymbol{\tau}$ the viscous stress tensor. The derivation uses two essential assumptions:

1. The fluid is assumed to be *incompressible*, i. e., the fluid density ρ is independent of pressure p. This filters out fast-moving acoustic waves; however, it does not imply that density is always constant as small variations (typically less than a few percent) due to salinity and temperature gradients can be included in the model [95, 157].
2. Denoting vertical and horizontal extent of the fluid domain by H and L, respectively, the aspect ratio of the domain H/L is supposed to be bounded from above by $1/20$ [157]. This ratio is typically smaller than $1/100$ in real-world applications.

With variations in temperature and salinity assumed to be small, it follows that density variations are small, too, so that their impact on changes in mass or inertia can be neglected. Only in gravity forcing terms such variations are important and must be taken into account. This allows to replace density ρ by a reference density ρ_0 everywhere except in the hydrostatic pressure term, which is called the *Boussinesq approximation* [95].

The second assumption justifies the *hydrostatic* approximation: Let U denote the order of horizontal velocity components u, v and W the order of the vertical velocity component w.

[1] This implies elongation / shrinking of lateral boundaries according to the free surface movement. In simulations that account for wetting / drying at land boundaries, corresponding lateral boundaries are also allowed to move horizontally. However, this is not considered here.

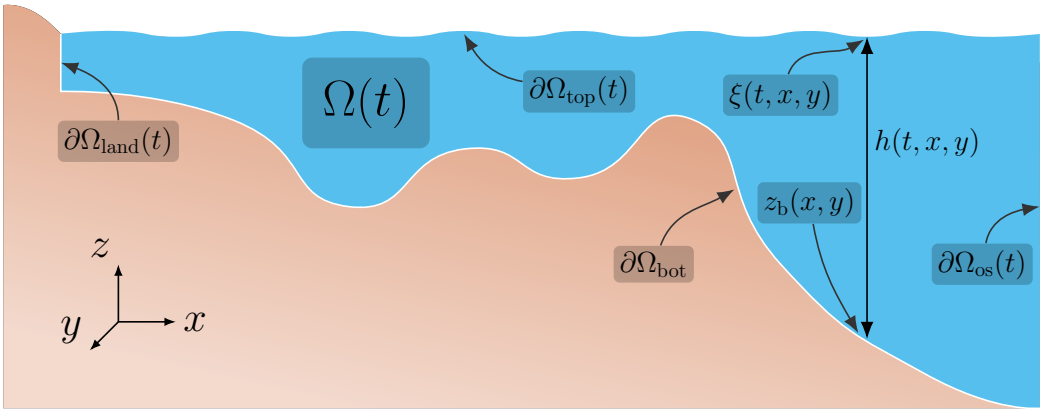

Figure 2.1: Schematic of the domain for three-dimensional shallow-water flow, here with land and open sea boundaries comprising $\partial\Omega_{\text{lat}}(t) = \partial\Omega_{\text{land}}(t) \cup \partial\Omega_{\text{os}}(t)$

Then, their derivatives (with respect to space) in the continuity equation (2.1b) are of order U/L and W/H, respectively. However, the derivatives of horizontal components typically do not cancel one another, which means that the spatial derivatives for all velocity components are of the same order $U/L = W/H$, and the ratio between orders of vertical and horizontal velocity components $W/U = H/L$ is as small as the aspect ratio of the domain [157, Sec. 2.4]. Due to this fact, a similar point can be made for orders of terms in the momentum equation (2.1a) for the vertical velocity component, revealing that the acceleration, the advection, and the stress terms are small compared to the gravitational term ρg. Neglecting these small terms simplifies the equation to the *hydrostatic pressure distribution*

$$\partial_z p = -\rho g \,.$$

Integrating over the total height of the water column and using the atmospheric pressure at the surface as the initial condition yields an expression for p that is used to replace the pressure terms in the remaining momentum equations.

With the non-linearity substantially reduced and the fact that the resulting system no longer poses a saddle-point problem, its numerical treatment is easier than for non-hydrostatic models [3]. It does, however, also result in a neglection of vertical accelerations and the loss of vertical momentum conservation. For an overview of the hierarchy of ocean models attainable when applying only parts of these approximations see the habilitation thesis by Aizinger [3], and for a detailed derivation that includes comprehensive motivation for all steps refer to the book by Vreugdenhil [157].

The full problem statement then reads [9, 157]:

For $t \in (t_0, t_{\text{end}})$, let $\Omega(t) \subset \mathbb{R}^3$ be the time-dependent domain with moving free surface, Π the standard orthogonal projection operator from $\mathbb{R}^3$ to $\mathbb{R}^2$ ($\Pi(x,y,z) = (x,y), \forall (x,y,z) \in \mathbb{R}^3$), and $\Omega_{xy} \coloneqq \Pi\Omega(t)$. The boundary of the domain $\partial\Omega(t)$ is subdivided into top $\partial\Omega_{\text{top}}(t)$, bottom $\partial\Omega_{\text{bot}}$, and strictly vertical lateral $\partial\Omega_{\text{lat}}(t)$ sections (see Figure 2.1). The primary unknowns are water depth $h(t,x,y) = \xi(t,x,y) - z_{\text{b}}(x,y)$ [m], where ξ and z_{b} are values of the vertical coordinate at free surface and sea bed, correspondingly, horizontal velocity $\boldsymbol{u}_{xy}(t,x,y,z) = (u,v)^{\text{T}}$ [$\text{m}\,\text{s}^{-1}$], temperature $\theta(t,x,y,z)$ [°C], salinity $s(t,x,y,z)$ [psu],[2]

[2] practical salinity unit, or equivalently, parts per thousand (ppt)

turbulent kinetic energy $k(t,x,y,z)$ [$\mathrm{m}^2\,\mathrm{s}^{-2}$], and turbulence parameter $\psi(t,x,y,z)$.[3]

The *momentum equations* in conservative form in $(t_0,t_{\mathrm{end}})\times\Omega(t)$ are

$$\partial_t \boldsymbol{u}_{xy} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}_{xy}\otimes\boldsymbol{u} - \mathbf{D}\boldsymbol{\nabla}\boldsymbol{u}_{xy}) + g\boldsymbol{\nabla}_{xy}(h+p) + \begin{pmatrix} -f_{\mathrm{c}}\,v \\ f_{\mathrm{c}}\,u \end{pmatrix} = \boldsymbol{F}_u - \boldsymbol{\nabla} z_{\mathrm{b}}\,, \tag{2.2}$$

where $\boldsymbol{\nabla}_{xy} \coloneqq (\partial_x,\partial_y)^{\mathrm{T}}$, $\boldsymbol{F}_u = \boldsymbol{F}_u(t,x,y,z)$ contains body forces (e. g., atmospheric pressure gradient and tidal potential), $\boldsymbol{u} = \boldsymbol{u}(t,x,y,z) = (u,v,w)^{\mathrm{T}}$ is the three-dimensional velocity vector, g is acceleration due to gravity, $f_{\mathrm{c}} = 2\,\omega\sin\phi$ is the Coriolis coefficient with angular rate of revolution ω and geographical latitude ϕ, and $\mathbf{D} = \mathbf{D}(\boldsymbol{u})$ is the tensor of eddy viscosity coefficients that may depend on the flow velocity and defined as

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_v \end{pmatrix} \quad\text{and}\quad \mathbf{D}\boldsymbol{\nabla}\boldsymbol{u}_{xy} = \begin{pmatrix} (\mathbf{D}_u\boldsymbol{\nabla}u)^{\mathrm{T}} \\ (\mathbf{D}_v\boldsymbol{\nabla}v)^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{2\times 3}\,,\quad \mathbf{D}_u = \mathbf{D}_v = \begin{pmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & \nu_t \end{pmatrix}, \tag{2.3}$$

where A_x, A_y are horizontal and ν_t the vertical eddy viscosity coefficients, all non-negative.

The baroclinic pressure correction $p(x,y,z)$ accounts for changes in the hydrostatic pressure due to variations in density and is computed as

$$p(x,y,z) = \frac{1}{\rho_0}\int_z^{\xi} (\rho(\theta,s,\xi-\tilde{z}) - \rho_0)\,\mathrm{d}\tilde{z}\,, \tag{2.4}$$

where ρ_0 is the reference density, and $\rho(\theta,s,h)$ is the density computed from the equation of state. Neglecting the baroclinic pressure correction (i. e., using constant density ρ_0) results in the *barotropic* system. The barotropic system served as the free surface flow model in the coupling to subsurface flows (see Section 2.3 and [Pp1, P8]), and both barotropic and baroclinic test cases were employed in [P2, P4].

For more than three decades, the most popular choice for the equation of state has been the *International Equation of State of Seawater* [154] commonly refered to as *UNESCO equation* and presented in some detail by Gill [69, Appendix 3]. A new, thermodynamically consistent formulation as a free energy function was introduced in 2010 [89]. Here, we use a polynomial approximation by Klinger [103], which produes a relative error less than 0.05%.

The vertical velocity component $w(x,y,z)$ is computed diagnostically from the incompressible *continuity equation*

$$\partial_x u + \partial_y v + \partial_z w = 0 \qquad \text{in}\ \ (t_0,t_{\mathrm{end}})\times\Omega(t)\,. \tag{2.5}$$

The boundary conditions specified for this system are:

- *Bottom boundary* $\partial\Omega_{\mathrm{bot}}$: No normal flow

$$\boldsymbol{u}(z_{\mathrm{b}})\cdot\boldsymbol{n} = 0 \tag{2.6a}$$

and a friction law on horizontal velocity components

$$\mathbf{D}_u\boldsymbol{\nabla}u(z_{\mathrm{b}})\cdot\boldsymbol{n} = -C_{\mathrm{f}}(\boldsymbol{u})\,u(z_{\mathrm{b}})\,,\qquad \mathbf{D}_v\boldsymbol{\nabla}v(z_{\mathrm{b}})\cdot\boldsymbol{n} = -C_{\mathrm{f}}(\boldsymbol{u})\,v(z_{\mathrm{b}})\,, \tag{2.6b}$$

where $\boldsymbol{n} = (n_x,n_y,n_z)^{\mathrm{T}}$ is an exterior unit normal to the bottom boundary and $C_{\mathrm{f}}(\boldsymbol{u}) > 0$ is the friction coefficient chosen to be either $C_{\mathrm{f}}(\boldsymbol{u}) = \text{const.}$ for a linear friction law or $C_{\mathrm{f}}(\boldsymbol{u}) = C'_{\mathrm{f}}|\boldsymbol{u}(z_{\mathrm{b}})|$ with $C'_{\mathrm{f}} = \text{const.}$ for a quadratic friction law.

[3] This parameter originates from the generic length scale model proposed by Umlauf and Burchard [153], and its units depend on the choice of the parameterization.

- *Free surface boundary* $\partial\Omega_{\text{top}}(t)$: Relative normal velocity must vanish, i. e.,

$$\partial_t \xi + u(\xi)\,\partial_x \xi + v(\xi)\,\partial_y \xi - w(\xi) = q_{\text{p}} - q_{\text{e}}\,, \tag{2.6c}$$

where $q_{\text{p}}, q_{\text{e}}$ are rates of precipitation and evaporation specified in velocity units, and

$$\boldsymbol{\nabla} u(\xi) \cdot \boldsymbol{n} = \boldsymbol{\nabla} v(\xi) \cdot \boldsymbol{n} = 0\,. \tag{2.6d}$$

- *Lateral boundaries* $\partial\Omega_{\text{lat}}(t)$: Different boundary types are defined on subsets of the boundary. Common boundary types include *land boundaries* with no normal flow

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad (t_0, t_{\text{end}}) \times \partial\Omega_{\text{land}}(t)\,, \tag{2.6e}$$

open sea boundaries with vanishing normal derivative of horizontal velocity components and prescribed Dirichlet boundary data $\xi_{\text{os}}(t, x, y)$ (e. g., tides) for the surface elevation

$$\boldsymbol{\nabla} u \cdot \boldsymbol{n} = \boldsymbol{\nabla} v \cdot \boldsymbol{n} = 0 \quad \text{and} \quad \xi = \xi_{\text{os}}(t, x, y) \qquad \text{on} \quad (t_0, t_{\text{end}}) \times \partial\Omega_{\text{os}}(t)\,, \tag{2.6f}$$

river boundaries with prescribed flow rate FR through boundary area A, from which the normal velocity is computed and the tangential velocity is set to 0,

$$\boldsymbol{u} \cdot \boldsymbol{n} = \frac{\text{FR}}{A} \qquad \text{and} \qquad \boldsymbol{u} \cdot \boldsymbol{\tau} = 0 \qquad \text{on} \quad (t_0, t_{\text{end}}) \times \partial\Omega_{\text{riv}}(t)\,, \tag{2.6g}$$

where $\boldsymbol{\tau}$ is a unit tangential vector, and *radiation boundaries* (outflow) with zero normal derivative of horizontal velocity components

$$\boldsymbol{\nabla} u \cdot \boldsymbol{n} = \boldsymbol{\nabla} v \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad (t_0, t_{\text{end}}) \times \partial\Omega_{\text{rad}}(t)\,. \tag{2.6h}$$

Integrating continuity equation (2.5) over the total height of the water column and applying top and bottom boundary conditions results in the *primitive continuity equation* from which the free surface elevation $\xi(t, x, y)$ is computed:

$$\partial_t \xi + \partial_x \int_{z_{\text{b}}}^{\xi} u \,\mathrm{d}z + \partial_y \int_{z_{\text{b}}}^{\xi} v \,\mathrm{d}z = q_{\text{p}} - q_{\text{e}} =: F_h \qquad \text{in} \quad (t_0, t_{\text{end}}) \times \Omega_{xy}\,. \tag{2.7}$$

Species transport for salinity $s(t, x, y, z)$ and temperature $\theta(t, x, y, z)$ in the baroclinic system is modeled by *advection–diffusion equations* for $r \in \{\theta, s\}$

$$\partial_t r + \boldsymbol{\nabla} \cdot (\boldsymbol{u}\, r - \mathbf{D}_r \boldsymbol{\nabla} r) = F_r \qquad \text{in} \quad (t_0, t_{\text{end}}) \times \Omega(t)\,, \tag{2.8a}$$

where

$$\mathbf{D}_r = \begin{pmatrix} A_{r,x} & 0 & 0 \\ 0 & A_{r,y} & 0 \\ 0 & 0 & \nu_r \end{pmatrix} \tag{2.8b}$$

is the diffusivity tensor with horizontal and vertical diffusivities $A_{r,x}, A_{r,y}, \nu_r \geq 0$. On inflow boundaries, we specify Dirichlet-type boundary conditions for temperature and salinity values and account for precipitation and evaporation effects on the salt balance using a virtual salinity flux boundary condition at the free surface

$$s(\xi)\, w(\xi) = -(q_{\text{p}} - q_{\text{e}})\, s(\xi) \quad \text{or} \quad s(\xi)\, \boldsymbol{u}(\xi) \cdot \boldsymbol{n} = -(q_{\text{p}} - q_{\text{e}})\, n_z\, s(\xi) \quad \text{on} \quad (t_0, t_{\text{end}}) \times \partial\Omega_{\text{top}}\,.$$

In momentum and species transport equations (2.2), (2.8), turbulent transport phenomena are parameterized as eddy viscosity to account for subgrid-scale transport. While horizontal transport terms play only a minor role due to the much larger length scales [157, Sec. 11.2] and are neglected alltogether in many applications (including the studies in [C1, P2, P4]), i. e., $A_x = A_y = A_{r,x} = A_{r,y} = 0\,[\mathrm{m^2\,s^{-1}}]$, vertical eddy viscosity plays a crucial role in vertical mixing, which serves as the mechanism accounting for the vertical transport and eventual drainage of the kinetic energy via bottom friction terms. Vertical eddy viscosity can be parameterized using a hierarchy of models of increasing complexity and accuracy [157]: By employing a constant diffusion coefficient, algebraic expressions depending on the flow velocity, or one / two-equation turbulence closure schemes that introduce unknowns for the turbulent kinetic energy and other quantities, if applicable, such as mixing length or energy dissipation.

Many popular two-equation turbulence closures are unified in the generic length scale (GLS) model proposed by Umlauf and Burchard [153] and are extensively evaluated and presented with many implementation details by Warner et al. [158]. GLS uses a generic turbulence length-scale quantity as second variable. In our model, turbulent kinetic energy $k(t,x,y,z)$ and derived quantity $\psi(t,x,y,z) = \left(c_\mu^0\right)^p k^m l^n$ are transported only in vertical direction [9] by a *diffusion equation* in $(t_0, t_{\mathrm{end}}) \times \Omega(t)$,

$$\partial_t k - \partial_z\left(\nu_k\,\partial_z k\right) = \nu_t\left((\partial_z u)^2 + (\partial_z v)^2\right) - \nu_r \frac{g}{\rho_0}\,\partial_z \rho - \varepsilon\,, \tag{2.9a}$$

$$\partial_t \psi - \partial_z\left(\nu_\psi\,\partial_z \psi\right) = \frac{\psi}{k}\left(C_1\,\nu_t\,\left((\partial_z u)^2 + (\partial_z v)^2\right) - C_3\,\nu_r\,\frac{g}{\rho_0}\,\partial_z\rho - C_2\,\varepsilon\,F_{\mathrm{wall}}\right)\,, \tag{2.9b}$$

where C_1, C_2, C_3 are dimensionless constants, l is the mixing length scale, ε the dissipation rate, and ν_t, ν_r are eddy viscosity and diffusivity coefficients defined in Equations (2.3) and (2.8b). Specific choices for p, m, n result in different closure schemes. At the free surface, zero flux is assumed for both turbulence unknowns. For the bottom boundary, an expression for turbulent kinetic energy can be derived assuming a logarithmic flow profile. However, since the viscous boundary layer is usually not resolved, this boundary condition should be applied in flux form to avoid numerical instabilities [158]. For parameter choices and computation of derived values, see works by Aizinger et al. [9] and Warner et al. [158].

The system of equations is complemented by initial conditions for all primary unknowns, which we indicate in the following by a zero subscript, e. g., ξ_0.

When horizontal momentum equations (2.2) and continuity equation (2.5) are integrated over the depth of the water column, one obtains the (two-dimensional) *shallow-water equations* that model horizontal circulation in terms of depth-integrated quantities. This was used in [P7] in combination with an advection-reaction model to introduce a novel scheme for efficient transport. The full baroclinic model for three-dimensional shallow-water flow is employed in the computational performance studies in [C1, P2, P4], and the barotropic system is coupled to subsurface flow in [P8].

2.2 Groundwater flow

For groundwater flow, we consider *saturated flow through an aquifer*, i. e., a region of the subsurface typically made up of permeable rock (e. g., sandstone, limestone) or granular material (e. g., sand, gravel) through which water can pass. The material is a type of *porous*

media with pores and cavities between impermeable solid structures. The fraction of non-solid pore space per total volume is a dimensionless quantity called *porosity* $\widetilde{\phi}$, and we assume that this space is connected and filled completely by water.[4] Flow in the pore space is then described by the *Stokes equation*; however, a direct numerical simulation of this model would require the geometry of the pore matrix to be resolved. For relevant domain sizes and time periods, this is both unfeasible and unnecessary as flow in this regime is very slow (typically in the order of decimeters per day or less), and thus averaged quantities can be used instead [83].

With the energy conservation law stated by the *Bernoulli* equation [58] we have an expression for the total energy of groundwater flow per unit weight (i. e., scaled by $1/(\widetilde{\rho}\, g\, V)$) referred to as *total head*

$$z + \frac{P}{\widetilde{\rho}\, g} + \frac{|\boldsymbol{v}|^2}{2\, g}\,, \tag{2.10}$$

where the first term is called *elevation head*, the second term *pressure head*, and the last term *velocity head* (representing potential energy, energy due to sustained pressure, and kinetic energy, respectively). Note that a tilde is used for quantities in groundwater flow to make it easier to distinguish them from those in the surface flow model. Here, z [m] is the elevation of the measuring point (with respect to some datum), P [$\mathrm{N\,m^{-2}} = \mathrm{kg\,m^{-1}\,s^{-2}}$] the pressure due to the water column above this point, $\widetilde{\rho}$ [$\mathrm{kg\,m^{-3}}$] the density of groundwater, and $\boldsymbol{v}$ [$\mathrm{m\,s^{-1}}$] the velocity. As groundwater flow is slow, the last term can be neglected and the first two form the *hydraulic head* [58]

$$\widetilde{h} \coloneqq z + \frac{P}{\widetilde{\rho}\, g}\,. \tag{2.11}$$

Groundwater flow is imposed by variations in the hydraulic head, and, in 1856, *Henry Darcy* determined experimentally the relationship between a hydraulic gradient and the specific discharge rate for one-dimensional flow through sand beds [58]. Later, this relationship was also derived from the *Stokes equation* via homogenization [160] and extended to multi-dimensional applications and anisotropic media. This extended *Darcy Law* takes the form [58]

$$\widetilde{\boldsymbol{q}} = -\widetilde{\mathbf{K}}\, \boldsymbol{\nabla} \widetilde{h}\,, \tag{2.12}$$

for *specific discharge* $\widetilde{\boldsymbol{q}}(t, x, y, z)$ [$\mathrm{m^3\,m^{-2}\,s^{-1}} = \mathrm{m\,s^{-1}}$] also called *seepage velocity* or *Darcy velocity*, which is a measure of the volumetric discharge per area and time. The symmetric and positive-definite *hydraulic conductivity* matrix $\widetilde{\mathbf{K}}(x, y, z)$ [$\mathrm{m\,s^{-1}}$] depends on the intrinsic permeability of the material and, by extension, also on the porosity.

Consider the total mass of fluid $m = \int_V \widetilde{\rho}\, \widetilde{\phi}\, \mathrm{d}\boldsymbol{x}$ with density $\widetilde{\rho}$ [$\mathrm{kg\,m^{-3}}$] stored in a representative elementary volume V with porosity $\widetilde{\phi}$. Denoting the density of mass flow by $\widetilde{\rho}\, \widetilde{\boldsymbol{q}}(t, x, y, z)$, the corresponding total mass flow through the boundaries of V is determined by $\int_{\partial V} \widetilde{\boldsymbol{n}} \cdot (\widetilde{\rho}\, \widetilde{\boldsymbol{q}})\, \mathrm{d}s$. In addition, we allow for generation / destruction of mass at some rate $\widetilde{W}_0(t, x, y, z)$ [$\mathrm{s^{-1}}$] and thus can formulate a mass balance equation that becomes after applying *Gauss*'s divergence theorem [83]

$$\partial_t \int_V \widetilde{\rho}\, \widetilde{\phi}\, \mathrm{d}\boldsymbol{x} + \int_V \boldsymbol{\nabla} \cdot (\widetilde{\rho}\, \widetilde{\boldsymbol{q}})\, \mathrm{d}\boldsymbol{x} = \int_V \widetilde{\rho}\, \widetilde{W}_0\, \mathrm{d}\boldsymbol{x}\,.$$

[4] Unsaturated situations and multiphase flow (e. g., gas or oil inclusion) require additional consideration.

Choosing an infinitesimally small V, assuming incompressibility of the fluid (thus neglecting spatial variations in density), and dividing by density yields the point-wise *continuity equation*

$$\partial_t \widetilde{\phi} + \frac{\widetilde{\phi}}{\widetilde{\rho}} \partial_t \widetilde{\rho} + \boldsymbol{\nabla} \cdot \widetilde{\boldsymbol{q}} = \widetilde{W}_0 \,.$$

With that, the full problem statement reads:

Let (t_0, t_{end}) be a finite time interval and $\widetilde{\Omega} \subset \mathbb{R}^3$ denote the polygonally bounded domain for three-dimensional groundwater flow with boundary $\partial\widetilde{\Omega}$ subdivided into Dirichlet $\partial\widetilde{\Omega}_{\text{D}}$ and Neumann $\partial\widetilde{\Omega}_{\text{N}}$ parts. With *Darcy Law* (2.12) and the chain rule, we obtain the *equation for groundwater flow* [58, 83]

$$S_0 \, \partial_t \widetilde{h} - \boldsymbol{\nabla} \cdot \left(\widetilde{\mathbf{K}} \, \boldsymbol{\nabla} \widetilde{h} \right) = \widetilde{W}_0 \qquad \text{in } (t_0, t_{\text{end}}) \times \widetilde{\Omega} \,, \tag{2.13a}$$

where $S_0 \coloneqq \partial_{\widetilde{h}} \widetilde{\phi} + \frac{\widetilde{\phi}}{\widetilde{\rho}} \partial_{\widetilde{h}} \widetilde{\rho}$ $[\mathrm{m}^{-1}]$ denotes the *specific storage*, a measure for the change of mass due to a change in head. This equation is complemented by initial conditions for hydraulic head and boundary conditions for head and flux

$$\widetilde{h} = \widetilde{h}_{\text{D}} \qquad \text{on } (t_0, t_{\text{end}}) \times \partial\widetilde{\Omega}_{\text{D}} \,, \tag{2.13b}$$

$$-\widetilde{\mathbf{K}} \, \boldsymbol{\nabla} \widetilde{h} \cdot \widetilde{\boldsymbol{n}} = \widetilde{q}_{\text{N}} \qquad \text{on } (t_0, t_{\text{end}}) \times \partial\widetilde{\Omega}_{\text{N}} \,, \tag{2.13c}$$

$$\widetilde{h} = \widetilde{h}_0 \qquad \text{in } \{t_0\} \times \widetilde{\Omega} \,. \tag{2.13d}$$

This mathematical model yields a relatively simple macroscopic description of saturated groundwater flow, where the microscopic pore structure is relevant but is not resolved explicitly and instead incorporated in averaged effective coefficients. The underlying approach based on homogenization is common to models even for more complicated situations, e. g., where saturated systems with multiphase flow (e. g., water, air, oil, etc.) [32] or growth and degression of solid structures (thus changing the porosity) are considered [138, 139].

The model (2.13) is discretized with the local discontinuous Galerkin method (see Section 3.4) and used in coupled simulations with barotropic free-surface flow in [P8]. A similar model serves as the prototype application for the presentation of discretization and efficient implementation techniques in [P1].

2.3 Coupled model

In situations where a free surface water body is located on top of a subsurface flow system, the interaction between both is of interest, e. g., for environmental applications such as infiltration of overland flow into soil during landfall, contaminant propagation, sedimentation processes, or the interaction of seas, lakes, rivers, or wetlands with groundwater aquifers. Typically, different mathematical models are used in each flow domain while they are coupled by suitable interface conditions. In the free surface flow domain, momentum conservation is usually modeled by the Navier–Stokes equations or derived models, e. g., Stokes equation [133, 134], the kinematic wave equation [117], or two- / three-dimensional shallow water equations as in the context of this thesis and in [49, 53, 114]. Darcy's model or Richards' equation is typically applied to describe flow in the subsurface possibly using the Brinkman extension for systems with high porosity [117] or the Forchheimer extension for high Reynolds number flows [91]. At the interface, either a sharp interface with a set of

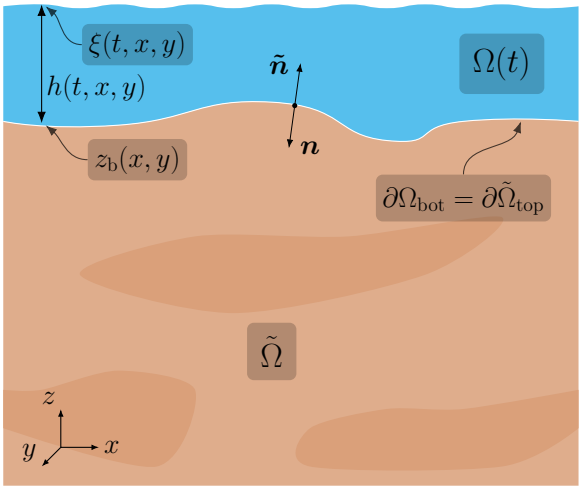

Figure 2.2: Schematic of the coupled domain

coupling conditions (with the Beavers-Joseph-Saffman [25, 135] condition or modifications thereof being the most popular choice) or a transition region is used, in which conservation equations are averaged over the transition zone [133].

[P8] considers barotropic free surface flow (i. e., (2.2)–(2.7) without the baroclinic pressure correction (2.4)), the groundwater flow equation (2.13), and a sharp interface separating the two domains. The challenge in formulating interface conditions for the coupling of surface and subsurface flows lies in finding a set of transition conditions that are physically meaningful and result in a mathematically well-posed system of equations. Here, we model the coupling of barotropic surface and subsurface flows as conservation of volume / mass and continuity of pressure (head) at the interface, which is close to conditions used in Navier–Stokes / Darcy coupled models (e. g., by Fetzer, Smits, and Helmig [63]), however with modifications that are due to the absence of vertical momentum conservation in the hydrostatic free surface flow model.

For conservation of mass, one must ensure that the mass loss from one domain due to outflow through the coupling interface must be gained by the other domain and vice versa. In our case, this means that normal fluxes across the interface must be continuous, i. e.,

$$(\boldsymbol{u} \cdot \boldsymbol{n})\Big|_{\partial\Omega_\mathrm{bot}} = -\,(\tilde{\boldsymbol{q}} \cdot \tilde{\boldsymbol{n}})\Big|_{\partial\tilde{\Omega}_\mathrm{top}}, \tag{2.14a}$$

where the sign is due to the opposing directions of exterior normal vectors (see Figure 2.2).

Continuity of pressure follows from a continuity argument for the total head (2.10), where we balance the hydraulic head $\tilde{h}$ (2.11) by the hydrostatic pressure exerted by the water depth h and the dynamic pressure due to (non neglectable) kinematic energy in the shallow-water domain. This results in a continuity requirement for the hydraulic head, i. e.,

$$\tilde{h}\Big|_{\partial\tilde{\Omega}_\mathrm{top}} = \left(\xi + \frac{\boldsymbol{u}_{xy} \cdot \boldsymbol{u}_{xy}}{2\,g}\right)\Bigg|_{\partial\Omega_\mathrm{bot}}. \tag{2.14b}$$

These choices can be justified using a weak energy estimate of the stationary coupled problem as presented in [P8], which further includes a proof of discrete energy stability for the coupled system (see Section 3.5), convergence tests for the discretization error, and a simulation of a realistic scenario.

Chapter 3

The discontinuous Galerkin method

3.1 Types of discontinuous Galerkin methods

Discontinuous Galerkin (dG) schemes are a class of numerical methods for solving differential equations that have become increasingly popular over the last three decades. They share characteristics with methods from the finite volume and finite element frameworks: dG methods employ the finite element approach for a weak formulation only with discontinuous ansatz and test spaces; this is combined with discontinuous approximate solutions, numerical fluxes, and slope limiters also used in finite volume methods [40].

The application of dG to a linear hyperbolic equation describing steady neutron transport proposed by Reed and Hill [125] in 1973 is generally reckoned as the origin of the method, which was soon extended to non-linear hyperbolic conservation laws [34]. However, at first, the presence of non-linearities required implicit time-stepping schemes to obtain higher-order accuracy rendering it computationally inefficient [40]. The breakthrough in conjunction with explicit time-stepping schemes was the *Runge–Kutta discontinuous Galerkin* (RKDG) method introduced by Cockburn and Shu [43] in 1989 with extensions to multidimensional cases and systems in a series of papers [38, 41, 42, 45]. [P3] and [P5] employ this type of time-explicit dG method for the discretization of linear advection equations and the investigation of slope-limiting techniques.

The idea of rewriting higher-order differential equations as systems of first-order equations and treating each of which with the established dG framework allowed Cockburn and Shu [44] to derive the *local discontinuous Galerkin* (LDG) method in 1998 using previous work of Bassi and Rebay [23]. With that, the dG method was able to handle higher order operators and thus has ultimately been opened to a wide range of applications including those in the present work in the context of ocean modelling or groundwater flow in [P1, C1, P2, P4, P7, P8].

Already before LDG, another avenue of dealing with second-order terms in the context of elliptic and parabolic problems was investigated independently in the 1970s. These approaches utilized weakly-imposed continuity conditions on interfaces, e. g., by Babuška [18]. The application of this technique to second-order elliptic problems by Babuška and Zlámal [19] and Wheeler [159] followed by an extension of Arnold [15] to non-linear parabolic problems in 1982 are the cornerstones for *interior penalty discontinuous Galerkin* (IP-dG) methods. These methods are further classified according to the symmetry of the resulting bilinear form into *symmetric interior penalty discontinous Galerkin* (SIP-dG) and *nonsymmetric interior penalty discontinuous Galerkin* (NIP-dG) or, when omitting the symmetrization term, *incomplete interior penalty discontinuous Galerkin* (IIP-dG) methods. Based on NIP-dG, Oden, Babuška, and Baumann [122] developed a formulation without any penalty parameters generally referred to as *OBB*-method. A comprehensive introduction into dG

methods without mixed formulations is the book by Rivière [128]. Both IP-dG and LDG methods were unified in the analysis framework of Arnold et al. [16] for elliptic problems.

For a more exhaustive overview of the historical evolution of these methods, refer to the introduction by Cockburn, Karniadakis, and Shu [40] in the proceedings of the first international symposium on dG methods [39] and more condensed overviews by Aizinger [1] or Di Pietro and Ern [54]. The latter contains also a very illustrative figure highlighting the rather recent nature of and, compared to that, high interest in dG methods [54, Fig. 1].

Ab initio, dG methods had to prove their worth compared to established finite volume and finite element methods, in which context Cockburn, Karniadakis, and Shu [40] list the following:

1. High formal orders of accuracy are easily achieved using high-order approximating polynomials.
2. The local nature of the degrees of freedom and their loose coupling make the method highly parallelizable.
3. Easy treatment of complicated geometries and boundary conditions.
4. *h*- and *p*-adaptivity (i. e., refining / coarsening the mesh or changing approximation orders locally) can be easily incorporated due to the lack of continuity requirements on interfaces.

In particular for time-dependent advection-dominated PDEs, dG methods established themselves as a popular choice as outlined by a recent survey of Shu [141]. However, the large number of degrees of freedom (in particular for the LDG method) puts them at a disadvantage for diffusion dominated problems, where continuous-in-space approximations appear to have the edge [40]. This is most pronounced in time-implicit or stationary numerical solvers, where matrices must be assembled, and large numbers of local degrees of freedom are exacerbated by a quadratic growth of matrix sizes. Attempts to overcome this drawback include the choice of a reduced approximation order for flux unknowns while retaining the original order of convergence for diffusion operators [132] or specific solvers for the linear systems. These exploit properties of the underlying approximation spaces by splitting the degrees of freedom into coarse and fine parts, such as *p*-multigrid methods [22, 64, 116] or the very recent *hierarchical scale separation* (HSS) and its variants [8, 11, 93, 149].

Another relatively recent[1] technique called *hybridization* is to introduce auxilliary unknowns supported on the skeleton of the computational mesh (i. e., edges in 2D and faces in 3D). After carrying out the dG discretization procedure, static condensation (a Schur complement reduction technique) can be applied to obtain a significantly smaller system of equations that has to be solved globally on the mesh skeleton and additional small element-local (and uncoupled) systems. Due to the reduced size of the globally coupled system, this approach can lead to considerably lower computation times for higher polynomial approximation orders and in parallelized solvers. When used in conjunction with mixed formulations, the resulting method is called *hybridized discontinuous Galerkin* (HDG) method and was described by Cockburn, Gopalakrishnan, and Lazarov [37] as part of a unifying framework in 2009. For dG methods without mixed formulations they are referred to as *Hybrid High-Order* (HHO) methods and were introduced by Di Pietro and Ern [55] for linear elasticity and by Di Pietro, Ern, and Lemaire [56] for diffusion problems. If rewritten in mixed form, HHO methods can be incorporated into the framework of HDG methods by excplicitly identifying the numerical flux, as shown by Cockburn, Di Pietro, and Ern [36]

[1] Recent in this context, as underlying ideas are much older. See [P6, 36] for historical overviews.

for a diffusion problem. The application of a hybridized dG method to a linear advection operator and its efficient implementation are discussed in [P6].

3.2 Notation and basic definitions

Before presenting the LDG discretization for free surface and subsurface flow, more notation is introduced. On domains $\Omega \subset \mathbb{R}^d$, $d \in \{1, 2, 3\}$, and $d-1$ dimensional surfaces γ, the L^2 norm of u is denoted by $\|u\|_\Omega$, and L^2 inner products of functions u, v are written as $(u, v)_\Omega$ and $\langle u, v\rangle_\gamma$, respectively.

In the following, $\mathcal{T}_\Delta = \mathcal{T}_\Delta(\Omega)$ denotes a non-overlapping d-dimensional polytopic partition of some $\Omega \in \{\Omega(t), \Omega_{xy}, \widetilde{\Omega}\}$ into elements T of characteristic size Δ, and $\mathcal{F}_\Delta(\Omega)$ is the corresponding set of faces γ. For $T \in \mathcal{T}_\Delta$, let $\boldsymbol{n}_T$ be the unit normal on ∂T exterior to T. On neighboring mesh elements $T_i, T_j \in \mathcal{T}_\Delta$, $T_i \neq T_j$, we define the average $\{\!|\cdot|\!\}$ and the jump $[\![\cdot]\!]$ on $\partial T_i \cap \partial T_j$ for a scalar function w and a vector function $\boldsymbol{v}$ as follows:

$$
\begin{aligned}
\{\!|w|\!\} &\coloneqq \tfrac{1}{2}(w|_{T_i} + w|_{T_j})\,, & [\![w]\!] &\coloneqq w|_{T_i}\,\boldsymbol{n}_{T_i} + w|_{T_j}\,\boldsymbol{n}_{T_j}\,,\\
\{\!|\boldsymbol{v}|\!\} &\coloneqq \tfrac{1}{2}(\boldsymbol{v}|_{T_i} + \boldsymbol{v}|_{T_j})\,, & [\![\boldsymbol{v}]\!] &\coloneqq \boldsymbol{v}|_{T_i}\cdot\boldsymbol{n}_{T_i} + \boldsymbol{v}|_{T_j}\cdot\boldsymbol{n}_{T_j}\,,\\
& & [\![\boldsymbol{v}]\!] &\coloneqq \boldsymbol{v}|_{T_i}\otimes\boldsymbol{n}_{T_i} + \boldsymbol{v}|_{T_j}\otimes\boldsymbol{n}_{T_j}\,,
\end{aligned}
$$

i. e., a jump in a scalar variable is a vector, and a jump in a vector is a scalar. In addition, $[\![\cdot]\!]$ produces a second-order tensor by applying the scalar jump definition component-wise to a vector.

The *test* and *trial* spaces for the LDG method are defined as the d-dimensional ($d \in \{1, 2, 3\}$) *broken polynomial spaces* of order $k \in \mathbb{N}_0$

$$
\mathbb{P}_k^d(\mathcal{T}_\Delta(\Omega)) \coloneqq \left\{ \boldsymbol{v} \in \left[L^2(\Omega)\right]^d \,:\, \boldsymbol{v}|_T \text{ is a polynomial of degree at most } k, \forall T \in \mathcal{T}_\Delta(\Omega) \right\}.
$$

The specific anisotropy of the shallow-water model presented in Section 2.1 is reflected in the choice of computational mesh employed in our formulation. We construct the 3D mesh $\mathcal{T}_\Delta(\Omega(t))$ by extending a 2D triangular mesh $\mathcal{T}_\Delta(\Omega_{xy})$ in the vertical direction with nodes placed at z-levels according to some predefined criteria, as illustrated in Figure 3.1. For a chosen maximum number of layers of 3D elements $n \in \mathbb{N}$, typical strategies for the vertical placement of nodes include:

- Globally (or within a part of the domain) uniform levels $\xi_0^{\max} = z_0 > z_1 > \cdots > z_{n+1} = z_{\mathrm{b}}^{\min}$ resulting in one or more layers of prismatic elements, where $\xi_0^{\max}$ and $z_{\mathrm{b}}^{\min}$ are the global (or within the part of the domain) maximum and minimum values of initial free-surface elevation and bathymetry, respectively.
- Locally per 2D node (x, y) determined levels $\xi_0(x, y) = z_0 > z_1 > \cdots > z_{n+1} = z_{\mathrm{b}}(x, y)$ resulting in the same number of elements per triangle everywhere.

It is important to note that in the following the resulting three-dimensional elements are called prisms in all cases, even when top and bottom faces are not parallel, in which case we have strictly speaking "truncated prisms".

Turbulence quantities for vertical eddy viscosity parameterization in the free-surface flow problem (cf. Section 2.1) are discretized on vertical one-dimensional segments[2] that pass

[2] Note that, due to the alignment of prisms in vertical columns, each quadrature point of an element in $\mathcal{T}_\Delta(\Omega(t))$ is located on a vertical line that passes through the corresponding quadrature points of all other elements in the column and the corresponding two-dimensional element in $\mathcal{T}_\Delta(\Omega_{xy})$.

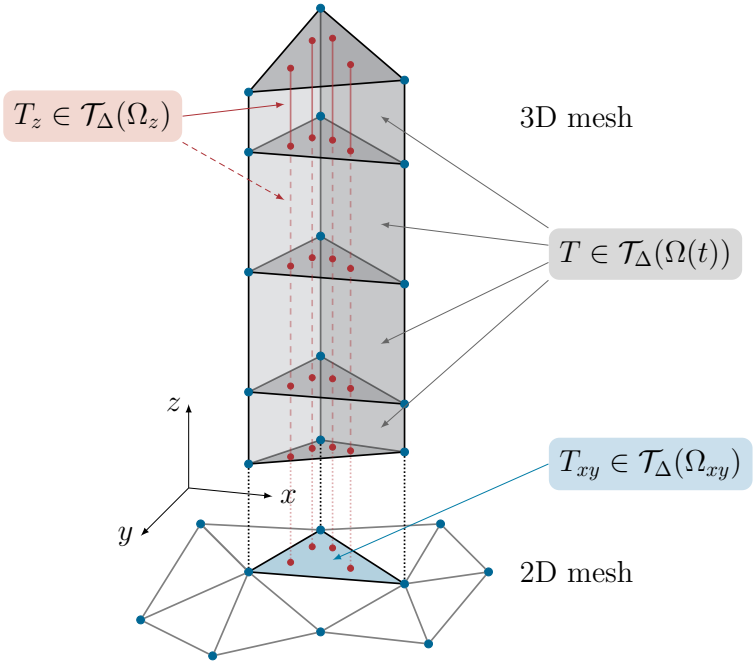

Figure 3.1: Composition of the 3D mesh $\mathcal{T}_\Delta(\Omega(t))$ from a 2D triangulation $\mathcal{T}_\Delta(\Omega_{xy})$ with the set of 1D segments $\mathcal{T}_\Delta(\Omega_z)$ aligned with two-dimensional quadrature points

through the quadrature points of the prismatic mesh (see Section 3.3 for details and Figure 3.1 for an illustration). In the following, we call this family of 1D meshes $\mathcal{T}_\Delta(\Omega_z(t))$. For simplicity, we use the same type of 3D mesh for the subsurface model from Section 2.2 as well and introduce the following sets of elements and faces:

$\mathcal{I}_{T,T_{xy}}$	set of prismatic elements vertically aligned with triangle T_{xy};
$\mathcal{I}^{\mathrm{v}}$, $\mathcal{I}^{\mathrm{h}}$	sets of lateral (vertical) faces and horizontal faces in $\mathcal{F}_\Delta(\Omega(t))$;
$\mathcal{I}_{\mathrm{int}}$, $\mathcal{I}_{\mathrm{ext}}$	sets of all interior faces and domain boundary faces in $\mathcal{F}_\Delta(\Omega(t))$;
$\mathcal{I}_{\mathrm{top}}$, $\mathcal{I}_{\mathrm{bot}}$	sets of top and bottom domain boundary faces in $\mathcal{F}_\Delta(\Omega(t))$;
$\mathcal{I}_z$	set of segment endpoints (both interior and domain boundary) in $\mathcal{F}_\Delta(\Omega_z(t))$;
$\widetilde{\mathcal{I}}_{\mathrm{int}}$	set of interior faces in $\mathcal{F}_\Delta(\widetilde{\Omega})$;
$\widetilde{\mathcal{I}}_{\mathrm{D}}$, $\widetilde{\mathcal{I}}_{\mathrm{N}}$	sets of faces on Dirichlet and Neumann boundaries in $\mathcal{F}_\Delta(\widetilde{\Omega})$.

We combine subscripts and superscripts as needed, e. g., $\mathcal{I}^{\mathrm{h}}_{\mathrm{int}}$ refers to all horizontal interior faces. When referring only to those faces of a set that belong to one element T, we write $\mathcal{I}_{*,T}$. In the shallow-water domain, all prismatic elements $T \in \mathcal{T}_\Delta(\Omega(t))$ within one column are aligned with a corresponding triangular element $T_{xy} \in \mathcal{T}_\Delta(\Omega_{xy})$ and thus, we also use this restriction for faces corresponding to an element: For example, $\mathcal{I}^{\mathrm{v}}_{\mathrm{int},T_{xy}}$ denotes the set of three-dimensional lateral faces vertically aligned with interior boundary edges of the two-dimensional element T_{xy}.

3.3 LDG discretization for three-dimensional shallow-water flow

Historically, structured grids have been the most popular choice for numerical ocean models and are still a mainstay of production codes with the most widely used structured-grid

models for global circulation being MITgcm[3], POM[4], and POP2[5]. For regional and coastal ocean, Klingbeil et al. [102] recently reviewed the state-of-the-art in structured-grid models, while, at the same time, unstructured-grid models have been becoming more and more popular for more than two decades due to their superior ability to resolve complex coastal topographies [74]. Only recently, the use of unstructured meshes has been gaining attraction also in global circulation models [48] and resulted in a number of new large-scale models, e. g., FESOM[6], ICON[7], or MPAS[8].

While the majority of ocean models employ finite volume (FV) discretizations, there are also models based on finite difference methods, the classical finite element (FE) method, or hybrid FE-FV models [97]. The use of discontinuous Galerkin schemes in ocean models is rather recent and was commenced with the first local discontinuous Galerkin scheme derived for two-dimensional shallow-water flow [1, 5]. After that, it was soon generalized to the three-dimensional barotropic model [1, 6, 51] with a stability estimate in [7] and further extended to the full baroclinic system with higher-order turbulence closure schemes [9].

In the time passed since the appearance of [5], discontinuous Galerkin methods for the system of two-dimensional shallow-water equations [52, 61, 106, 111, 127, 156, 165] gained wide popularity and were applied to atmospheric flows [20, 70, 113], flood inundation [26, 50, 100], global circulation [136], or coastal ocean [P7, 27, 28, 78, 79, 96, 163, 164], including extensions to multi-layer and non-hydrostatic models [90, 94, 123], and even motivated an entire book on this topic [101]. UTBEST3D (see Section 5.2) was the first coastal ocean model to make use of discontinuous Galerkin methods for the system of three-dimensional shallow-water equations. Later, the coastal ocean model SLIM[9] was introduced using SIP-dG and an ALE (*Arbitrary Lagrangian–Eulerian*) formulation on prismatic meshes [29], which was applied in a number of different studies [98, 155]. Recently, the Thetis[10] coastal ocean model emerged, which employs the code-generation capabilities of the Firedrake[11] framework in an attempt to achieve good computational performance [97]. Other three-dimensional studies employ mixed spectral element / dG formulations [35], non-hydrostatic models [30, 59, 152], or mixed-order / filtering approaches to increase convergence orders [46].

Our discretization of the free-flow model introduced in Section 2.1 follows the standard procedure for the LDG method with some important modifications needed to obtain a stable formulation. See works by Aizinger et. al. [1, 6, 7, 9, 51] for all details, of which relevant steps are reproduced and annotated in the following.

First, we combine advective fluxes in primitive continuity equation (2.7) and momentum equations (2.2) in the *primitive numerical fluxes*

$$\boldsymbol{C}_h(h, \boldsymbol{u}_{xy}) \coloneqq \int_{z_\mathrm{b}}^{\xi} \boldsymbol{u}_{xy} \,\mathrm{d}z\,, \tag{3.1a}$$

$$\mathbf{C}_{\boldsymbol{u}}(h, \boldsymbol{u}) \coloneqq \begin{pmatrix} \boldsymbol{C}_u(h, \boldsymbol{u}) \\ \boldsymbol{C}_v(h, \boldsymbol{u}) \end{pmatrix} \coloneqq \begin{pmatrix} u^2 + g\,(h+p) & u\,v & u\,w \\ u\,v & v^2 + g\,(h+p) & v\,w \end{pmatrix}, \tag{3.1b}$$

[3] http://mitgcm.org
[4] http://www.ccpo.odu.edu/POMWEB
[5] http://www.cesm.ucar.edu/models/cesm1.0/pop2/
[6] https://fesom.de/
[7] https://www.mpimet.mpg.de/en/science/models/icon-esm/
[8] https://mpas-dev.github.io/
[9] https://www.slim-ocean.be
[10] https://thetisproject.org/
[11] https://firedrakeproject.org/

introduce the right-hand side vector $\boldsymbol{F}(\boldsymbol{u}_{xy}) \coloneqq \boldsymbol{F}_u - g\,\boldsymbol{\nabla}_{xy} z_\mathrm{b} - f_\mathrm{c} \left(\begin{smallmatrix} -v \\ u \end{smallmatrix}\right)$, and split Equations (2.2), (2.7), and (2.8) containing second-order terms into pairs of first-order equations.[12] Introducing auxiliary variables $\mathbf{q} \in \mathbb{R}^{2\times 3}$, $\boldsymbol{q}_r \in \mathbb{R}^3$ for diffusive fluxes yields the *mixed formulation*

$$\partial_t h + \boldsymbol{\nabla}_{xy} \cdot \boldsymbol{C}_h(h, \boldsymbol{u}_{xy}) = F_h \,, \tag{3.2a}$$
$$\partial_t \boldsymbol{u}_{xy} + \boldsymbol{\nabla} \cdot (\mathbf{C}_{\boldsymbol{u}}(h, \boldsymbol{u}) + \mathbf{q}) = \boldsymbol{F}(\boldsymbol{u}_{xy}) \,, \tag{3.2b}$$
$$\mathbf{D}^{-1}\mathbf{q} + \boldsymbol{\nabla}\boldsymbol{u}_{xy} = \mathbf{0} \,, \tag{3.2c}$$
$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 \,, \tag{3.2d}$$
$$\partial_t r + \boldsymbol{\nabla} \cdot (\boldsymbol{u}\, r + \boldsymbol{q}_r) = F_r \,, \qquad \text{for } r \in \{s, \theta\} \,, \tag{3.2e}$$
$$\mathbf{D}_r^{-1}\boldsymbol{q}_r + \boldsymbol{\nabla} r = \mathbf{0} \,, \qquad \text{for } r \in \{s, \theta\} \,, \tag{3.2f}$$
$$\partial_t m + \partial_z(\nu_m\, q_m) = F_m \,, \qquad \text{for } m \in \{k, \psi\} \,, \tag{3.2g}$$
$$q_m + \partial_z m = 0 \,, \qquad \text{for } m \in \{k, \psi\} \,, \tag{3.2h}$$
$$p = \frac{1}{\rho_0} \int_z^\xi (\rho(\theta, s, \xi - \tilde{z}) - \rho_0)\, \mathrm{d}\tilde{z} \,, \tag{3.2i}$$

where $r \in \{s, \theta\}$ and $m \in \{k, \psi\}$ are transport and turbulence variables, respectively, and F_m contains the right-hand-side terms of turbulence equations (2.9). Note that Equation (3.2c) represents a system of 2×3 equations. To reduce horizontal mixing of turbulence variables k, ψ, those are discretized on vertical one-dimensional segments that pass through the quadrature points of the prismatic mesh (see Section 3.2).

By multiplication with smooth test functions $\delta_h, \delta_w, \delta_r, \delta_m, \delta_{q,m}, \boldsymbol{\varphi}_u, \boldsymbol{\varphi}_r, \boldsymbol{\Phi}$, and partial integration over elements $T \in \mathcal{T}_\Delta(\Omega(t))$, $T_{xy} \in \mathcal{T}_\Delta(\Omega_{xy})$, or $T_z \in \mathcal{T}_\Delta(\Omega_z(t))$, respectively, we obtain the *weak formulation*

$$(\partial_t h, \delta_h)_{T_{xy}} - (\boldsymbol{C}_h(h, \boldsymbol{u}_{xy}) \cdot \boldsymbol{\nabla}_{xy}, \delta_h)_{T_{xy}} + \langle \boldsymbol{C}_h(h, \boldsymbol{u}_{xy}) \cdot \boldsymbol{n}_{xy}, \delta_h \rangle_{\partial T_{xy}} = (F_h, \delta_h)_{T_{xy}} \,, \tag{3.3a}$$
$$(\partial_t \boldsymbol{u}_{xy}, \boldsymbol{\varphi}_u)_T - ((\mathbf{C}_{\boldsymbol{u}}(h, \boldsymbol{u}) + \mathbf{q})\, \boldsymbol{\nabla}, \boldsymbol{\varphi}_u)_T + \langle (\mathbf{C}_{\boldsymbol{u}}(h, \boldsymbol{u}) + \mathbf{q})\, \boldsymbol{n}, \boldsymbol{\varphi}_u \rangle_{\partial T} = (\boldsymbol{F}(\boldsymbol{u}_{xy}), \boldsymbol{\varphi}_u)_T \,, \tag{3.3b}$$
$$\left(\mathbf{D}^{-1}\,\mathbf{q}, \boldsymbol{\Phi}\right)_T - (\boldsymbol{u}_{xy} \otimes \boldsymbol{\nabla}, \boldsymbol{\Phi})_T + \langle \boldsymbol{u}_{xy} \otimes \boldsymbol{n}, \boldsymbol{\Phi} \rangle_{\partial T} = 0 \,, \tag{3.3c}$$
$$-(\boldsymbol{u} \cdot \boldsymbol{\nabla}, \delta_w)_T + \langle \boldsymbol{u} \cdot \boldsymbol{n}, \delta_w \rangle_{\partial T} = 0 \,, \tag{3.3d}$$
$$(\partial_t r, \delta_r)_T - ((\boldsymbol{u}\, r + \boldsymbol{q}_r) \cdot \boldsymbol{\nabla}, \delta_r)_T + \langle (\boldsymbol{u}\, r + \boldsymbol{q}_r) \cdot \boldsymbol{n}, \delta_r \rangle_{\partial T} = (F_r, \delta_r)_T \,, \tag{3.3e}$$
$$\left(\mathbf{D}_r^{-1}\, \boldsymbol{q}_r, \boldsymbol{\varphi}_r\right)_T - (r\, \boldsymbol{\nabla}, \boldsymbol{\varphi}_r)_T + \langle r\, \boldsymbol{n}, \boldsymbol{\varphi}_r \rangle_{\partial T} = 0 \,, \tag{3.3f}$$
$$(\partial_t m, \delta_m)_{T_z} - (\nu_m\, q_m\, \partial_z, \delta_m)_{T_z} + \langle \nu_m\, q_m\, n_z, \delta_m \rangle_{\partial T_z} = (F_m, \delta_m)_{T_z} \,, \tag{3.3g}$$
$$(q_m, \delta_{q,m})_{T_z} - (m\, \partial_z, \delta_{q,m})_{T_z} + \langle m\, n_z, \delta_{q,m} \rangle_{\partial T_z} = 0 \,. \tag{3.3h}$$

To keep the notation uniform, L^2 products over zero-dimensional domains ∂T_z denote evaluation at the respective point. This weak form is well-defined for $h, \delta_h \in H^1(\Omega_{xy})$, $\boldsymbol{u}_{xy}, \boldsymbol{\varphi}_u \in [H^1(\Omega(t))]^2$, $\boldsymbol{q}_r, \boldsymbol{\varphi}_r \in [H^1(\Omega(t))]^3$, $\mathbf{q}, \boldsymbol{\Phi} \in [H^1(\Omega(t))]^{2\times 3}$, $w, s, \theta, \delta_r \in H^1(\Omega(t))$, $k, \psi, \delta_m, \delta_{q,m} \in H^1(\Omega_z(t))$, and for a. e. $t \in [t_0, t_\mathrm{end}]$.

Due to the fact that all prismatic elements T within one column are vertically aligned with a corresponding triangular element T_{xy} (cf. Figure 3.1), and, in the same way, lateral

[12] Corresponding changes to boundary conditions are omitted in the interest of a compact presentation.

boundaries ∂T_{lat} are aligned with corresponding two-dimensional edges ∂T_{xy}, the weak form of the primitive continuity equation (3.3a) can be rewritten as

$$(\partial_t h, \delta_h)_{T_{xy}} - \sum_{T \in \mathcal{I}_{T,T_{xy}}} (\boldsymbol{u}_{xy} \cdot \boldsymbol{\nabla}_{xy}, \delta_h)_T + \sum_{T \in \mathcal{I}_{T,T_{xy}}} \langle \boldsymbol{u}_{xy} \cdot \boldsymbol{n}_{xy}, \delta_h \rangle_{\partial T_{\text{lat}}} = (F_h, \delta_h)_{T_{xy}} \,, \tag{3.4}$$

where set $\mathcal{I}_{T,T_{xy}}$ contains all 3D elements T in the column corresponding to the 2D element T_{xy}, and ∂T_{lat} is the lateral boundary of element T.

Again, we approximate the solution to problem (3.3) by finite-dimensional functions, choose suitable numerical fluxes across element boundaries, and incorporate boundary conditions to obtain the *semi-discrete problem*:

Let $\mathcal{V}^d \coloneqq \mathbb{P}_k^d(\mathcal{T}_\Delta(\Omega(t)))$, $\mathcal{X} \coloneqq \mathbb{P}_k^1(\mathcal{T}_\Delta(\Omega_{xy}))$, and $\mathcal{Z} \coloneqq \mathbb{P}_k^1(\mathcal{T}_\Delta(\Omega_z(t)))$ be the dG spaces of d-dimensional functions from the broken polynomial space of order k defined on the triangulation of the time-dependent three-dimensional shallow-water domain $\mathcal{T}_\Delta(\Omega(t))$ or the corresponding two-dimensional domain $\mathcal{T}_\Delta(\Omega_{xy})$ and one-dimensional segments $\mathcal{T}_\Delta(\Omega_z(t))$, respectively. Seek $(H, \boldsymbol{U}_{xy}, W, \mathbf{Q}, \Theta, \boldsymbol{Q}_\Theta, S, \boldsymbol{Q}_S, K, Q_k, \Psi, Q_\psi) \in \mathcal{X} \times \mathcal{V}^2 \times \mathcal{V}^1 \times [\mathcal{V}^3]^2 \times \mathcal{V}^1 \times \mathcal{V}^3 \times \mathcal{V}^1 \times \mathcal{V}^3 \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}$ such that, for all $T \in \mathcal{T}_\Delta(\Omega(t))$, $T_{xy} \in \mathcal{T}_\Delta(\Omega_{xy})$, $T_z \in \mathcal{T}_\Delta(\Omega_z(t))$, the following holds for a. e. $t \in [t_0, t_{\text{end}}]$ and for all $(\delta_h, \boldsymbol{\varphi}_u, \boldsymbol{\Phi}, \delta_w, \delta_\Theta, \delta_S, \boldsymbol{\varphi}_\Theta, \boldsymbol{\varphi}_S, \delta_k, \delta_{q,k}, \delta_\psi, \delta_{q,\psi}) \in \mathcal{X} \times \mathcal{V}^2 \times [\mathcal{V}^3]^2 \times \mathcal{V}^1 \times \mathcal{V}^1 \times \mathcal{V}^1 \times \mathcal{V}^3 \times \mathcal{V}^3 \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}$:

$$(\partial_t H, \delta_h)_{T_{xy}} + \sum_{T \in \mathcal{I}_{T,T_{xy}}} \left(A_{H,T}(H, H_s, \boldsymbol{U}_{xy}, \delta_h) - \left(\tfrac{\boldsymbol{U}_{xy} H}{H_s} \cdot \boldsymbol{\nabla}_{xy}, \delta \right)_T \right) = (F_h, \delta_h)_{T_{xy}} \,, \tag{3.5a}$$

$$(\partial_t \boldsymbol{U}_{xy}, \boldsymbol{\varphi}_u)_T + A_{\boldsymbol{U},T}(H, \boldsymbol{U}, \boldsymbol{\varphi}_u) + E_{\boldsymbol{U},T}(\mathbf{Q}, \boldsymbol{\varphi}_u) = (\boldsymbol{F}(\boldsymbol{U}_{xy}), \boldsymbol{\varphi}_u)_T \,, \tag{3.5b}$$

$$\left(\mathbf{D}^{-1} \mathbf{Q}, \boldsymbol{\Phi} \right)_T + E_{\mathbf{Q},T}(\boldsymbol{U}_{xy}, \boldsymbol{\Phi}) = 0 \,, \tag{3.5c}$$

$$A_{H,T}(H, H_s, \boldsymbol{U}, \delta_w) + A_{W,T}(\boldsymbol{U}, \delta_w) - (\boldsymbol{U} \cdot \boldsymbol{\nabla}, \delta_w)_T = 0 \,, \tag{3.5d}$$

$$(\partial_t R, \delta_R)_T + A_{R,T}(\boldsymbol{U}, R, \delta_R) + E_{R,T}(\boldsymbol{Q}_R, \delta_R) = (F_R, \delta_R)_T \,, \tag{3.5e}$$

$$(\mathbf{D}_R^{-1} \boldsymbol{Q}_R, \boldsymbol{\varphi}_R)_T + E_{\boldsymbol{q},T}(R, \boldsymbol{\varphi}_R) = 0 \,, \tag{3.5f}$$

$$(\partial_t M, \delta_M)_{T_z} - (\nu_M \, Q_M \, \partial_z, \delta_M)_{T_z} + \sum_{\gamma \in \mathcal{I}_{z,T_z}} \left\langle \nu_M \, Q_M^{\downarrow} \, n_z, \delta_M \right\rangle_\gamma = (F_M, \delta_M)_{T_z} \,, \tag{3.5g}$$

$$(Q_M, \delta_{q,M})_{T_z} - (M \, \partial_z, \delta_{q,M})_{T_z} + \sum_{\gamma \in \mathcal{I}_{z,T_z}} \left\langle M^{\uparrow} \, n_z, \delta_{q,M} \right\rangle_{\partial T_z} = 0 \,, \tag{3.5h}$$

for $R \in \{S, \Theta\}$, $M \in \{K, \Psi\}$ with forms

$$A_{H,T}(H, H_s, \boldsymbol{U}_{xy}, \delta) \coloneqq \sum_{\gamma \in \mathcal{I}^{\text{v}}_{\text{ext},T}} \left\langle \frac{(\boldsymbol{U}_{xy} \, H)_{\text{bdr}}}{H_s} \cdot \boldsymbol{n}_{xy}, \delta \right\rangle_\gamma + \sum_{\gamma \in \mathcal{I}^{\text{v}}_{\text{int},T}} \left\langle \widehat{C}_h(H, H_s, \boldsymbol{U}_{xy}), \delta \right\rangle_\gamma \,,$$

$$\begin{aligned} A_{\boldsymbol{U},T}(H, \boldsymbol{U}, \boldsymbol{\varphi}_u) \coloneqq & - (\mathbf{C}_{\boldsymbol{u}}(H, \boldsymbol{U}) \, \boldsymbol{\nabla}, \boldsymbol{\varphi}_u)_T + \sum_{\gamma \in \mathcal{I}_{\text{ext},T}} \langle \mathbf{C}_{\boldsymbol{u}}(H_{\text{bdr}}, \boldsymbol{U}_{\text{bdr}}) \, \boldsymbol{n}, \boldsymbol{\varphi}_u \rangle_\gamma \\ & + \sum_{\gamma \in \mathcal{I}^{\text{v}}_{\text{int},T}} \left\langle \widehat{\boldsymbol{C}}_{\boldsymbol{u}}(H, \boldsymbol{U}), \boldsymbol{\varphi}_u \right\rangle_\gamma + \sum_{\gamma \in \mathcal{I}^{\text{h}}_{\text{int},T}} \left\langle \mathbf{C}_{\boldsymbol{u}}(H, \{\!|\boldsymbol{U}_{xy}|\!\}, W^{\downarrow}) \, \boldsymbol{n}, \boldsymbol{\varphi}_u \right\rangle_\gamma \,, \end{aligned}$$

$$A_{W,T}(\boldsymbol{U}, \delta_w) \coloneqq \sum_{\gamma \in \mathcal{I}^{\text{h}}_T} \left\langle \{\!|\boldsymbol{U}_{xy}|\!\} \cdot \boldsymbol{n}_{xy} + W^{\downarrow} \, n_z, \delta_w \right\rangle_\gamma \,,$$

$$
\begin{aligned}
A_{R,T}(\boldsymbol{U}, R, \delta_R) \coloneqq & -(\boldsymbol{U}\, R \cdot \boldsymbol{\nabla}, \delta_R)_T + \sum_{\gamma \in \mathcal{I}_{\mathrm{ext},T}} \langle \boldsymbol{U}_{\mathrm{bdr}}\, R_{\mathrm{bdr}} \cdot \boldsymbol{n}, \delta_R \rangle_\gamma + \sum_{\gamma \in \mathcal{I}^{\mathrm{v}}_{\mathrm{int},T}} \left\langle \widehat{C}_R(\boldsymbol{U}, R) \cdot \boldsymbol{n}, \delta_R \right\rangle_\gamma \\
& + \sum_{\gamma \in \mathcal{I}^{\mathrm{h}}_{\mathrm{int},T}} \left\langle (\{\!|\boldsymbol{U}_{xy}|\!\} \cdot \boldsymbol{n}_{xy} + W^{\downarrow} n_z)\, R^{\mathrm{up}}, \delta_R \right\rangle_\gamma , \\
E_{\boldsymbol{U},T}(\mathbf{Q}, \boldsymbol{\varphi}_u) \coloneqq & -(\mathbf{Q}\, \boldsymbol{\nabla}, \boldsymbol{\varphi}_u)_T + \sum_{\gamma \in \mathcal{I}_{\mathrm{ext},T}} \langle \mathbf{Q}_{\mathrm{bdr}}\, \boldsymbol{n}, \boldsymbol{\varphi}_u \rangle_\gamma + \sum_{\gamma \in \mathcal{I}_{\mathrm{int},T}} \langle \{\!|\mathbf{Q}|\!\}\, \boldsymbol{n}, \boldsymbol{\varphi}_u \rangle_\gamma , \\
E_{\mathbf{Q},T}(\boldsymbol{U}_{xy}, \boldsymbol{\Phi}) \coloneqq & -(\boldsymbol{U}_{xy} \otimes \boldsymbol{\nabla}, \boldsymbol{\Phi})_T + \sum_{\gamma \in \mathcal{I}_{\mathrm{ext},T}} \langle (\boldsymbol{U}_{xy})_{\mathrm{bdr}} \otimes \boldsymbol{n}, \boldsymbol{\Phi} \rangle_\gamma + \sum_{\gamma \in \mathcal{I}_{\mathrm{int},T}} \langle \{\!|\boldsymbol{U}_{xy}|\!\} \otimes \boldsymbol{n}, \boldsymbol{\Phi} \rangle_\gamma , \\
E_{R,T}(\boldsymbol{Q}_R, \delta_R) \coloneqq & -(\boldsymbol{Q}_R \cdot \boldsymbol{\nabla}, \delta_R)_T + \sum_{\gamma \in \mathcal{I}_{\mathrm{ext},T}} \langle (\boldsymbol{Q}_R)_{\mathrm{bdr}} \cdot \boldsymbol{n}, \delta_R \rangle_\gamma + \sum_{\gamma \in \mathcal{I}_{\mathrm{int},T}} \langle \{\!|\boldsymbol{Q}_R|\!\} \cdot \boldsymbol{n}, \delta_R \rangle_\gamma , \\
E_{\boldsymbol{q},T}(R, \boldsymbol{\varphi}_R) \coloneqq & -(R\, \boldsymbol{\nabla}, \boldsymbol{\varphi}_R)_T + \sum_{\gamma \in \mathcal{I}_{\mathrm{ext},T}} \langle R_{\mathrm{bdr}}\, \boldsymbol{n}, \boldsymbol{\varphi}_R \rangle_\gamma + \sum_{\gamma \in \mathcal{I}_{\mathrm{int},T}} \langle \{\!|R|\!\}\, \boldsymbol{n}, \boldsymbol{\varphi}_R \rangle_\gamma .
\end{aligned}
$$

Remark 1 *(Mesh smoothing)***:** With all state variables approximated by functions from a discontinuous space, including water height $H = \Xi - z_{\mathrm{b}}$, the free surface elevation Ξ may have jumps across element boundaries. However, the computational mesh uses a continuous representation of the top boundary using a smoothing algorithm that updates z-coordinates of top boundary nodes every small (chosen) number of time steps (cf. Figure 3.2). This piecewise linear free surface representation $\xi_{\mathrm{s}}(t, x, y)$ is computed by taking the weighted mean of the free surface elevation approximation $\Xi(t, x, y)$ from the surrounding elements [1, 51]. The smoothed mesh height is also incorporated into fluxes over lateral faces $A_{H,T}$ of the semi-discrete primitive continuity equation (3.5a), in which boundary terms of the weak formulation (3.4) are expressed using the strictly positive water height h

$$
\sum_{T \in \mathcal{I}_{T,T_{xy}}} \langle \boldsymbol{u}_{xy} \cdot \boldsymbol{n}_{xy}, \delta_h \rangle_{\partial T_{\mathrm{lat}}} = \sum_{T \in \mathcal{I}_{T,T_{xy}}} \left\langle \frac{\boldsymbol{u}_{xy}\, h}{h} \cdot \boldsymbol{n}_{xy}, \delta_h \right\rangle_{\partial T_{\mathrm{lat}}} ,
$$

with the water height in the denominator replaced by the smoothed mesh height $H_s(t, x, y) = \xi_{\mathrm{s}}(t, x, y) - z_{\mathrm{b}}(x, y)$ after discretization. This yields the flux function $\widehat{C}_h(H, H_s, \boldsymbol{U}_{xy})$ as an approximation of the non-linear boundary flux $(\boldsymbol{U}_{xy}\, H) \cdot \boldsymbol{n}_{xy} / H_s$, which can then be treated using approximate Riemann solvers (see Remark 4). To preserve local mass conservation the same formulation is also applied to boundary fluxes in continuity equation (3.5d) [1].

Remark 2 *(Diffusive fluxes)***:** Linear fluxes in forms $E_{*,T}$ stem from the diffusion operators in the equations for momentum (2.2) and species transport (2.8), which can be approximated by central fluxes in agreement with the isotropic nature of the operator [1]. On horizontal faces, these fluxes can also be discretized by an asymmetric approach to reduce matrix bandwidths in a semi-implicit formulation by taking the values of primary unknowns in forms $E_{\mathbf{Q},T}$, $E_{\boldsymbol{q},T}$ from the element below and values of auxiliary flux variables in forms $E_{\boldsymbol{U},T}$, $E_{R,T}$ from the element above [9]. This is also done for the fluxes in the one-dimensional turbulence equations (3.5g) and (3.5h) denoting the subjacent values (i. e., from element below) $Q_M^{\downarrow}$ and upper values (i. e., from element above) $M^{\uparrow}$. For brevity, we assume boundary conditions (see Section 2.1) are incorporated in this definition.

Remark 3 *(Vertical velocity)***:** The discrete continuity equation (3.5d) serves as means to compute the vertical velocity component W diagnostically from given horizontal velocities $\boldsymbol{U}_{xy}$ and thus to maintain a divergence-free velocity field. This can be interpreted as solving an ordinary differential equation for W upward within each column of prismatic

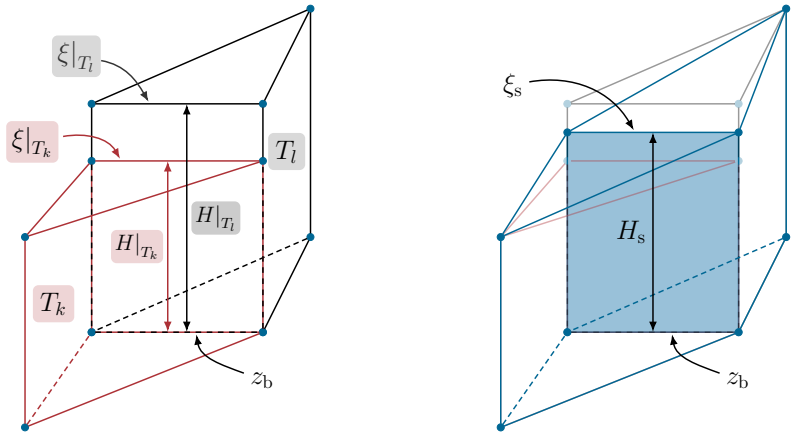

Figure 3.2: Discontinuous representation of water height and mesh smoothing

elements. For that, W is used from given initial conditions (2.6a) at the bottom boundary or the element below on other horizontal faces, denoted by $W^{\downarrow}$ in form $A_{W,T}$, and the average for horizontal velocity components. For consistency, the same is done for the advective fluxes over horizontal faces in forms $A_{\boldsymbol{U},T}$ and $A_{R,T}$, whereas upwinding is used for the transported species denoted by R^{up} [1].

Remark 4 *(Riemann solvers)***:** Fluxes over lateral faces are much more critical for the stability of the discretization than those over horizontal faces. For that reason, normal fluxes $\widehat{C}_h$, $\widehat{\boldsymbol{C}}_{\boldsymbol{u}}$, and $\widehat{C}_R$ must be computed by solving a Riemann problem in a coupled way [1, 51]. A number of different Riemann solvers are available for that with different characteristics in terms of computational cost, numerical diffusion, etc. The simplest choice that guarantees stability of the method is the *Lax–Friedrichs Riemann solver* that approximates a normal flux $\boldsymbol{C}(c)\cdot\boldsymbol{n}$ by

$$\widehat{C}(c) = \{\!|\boldsymbol{C}(c)\cdot\boldsymbol{n}|\!\} + \tfrac{|\hat{\lambda}|}{2}\,[\![c]\!]\cdot\boldsymbol{n}\,,$$

where $\hat{\lambda}$ is the largest (absolute) eigenvalue of the Jacobian of the primitive numerical fluxes (3.1). Other choices include the Riemann solver of Roe [129] or the HLLC solver [151].

Remark 5 *(Boundary values)***:** On domain boundaries, values for unknowns H_{bdr}, $\boldsymbol{U}_{\mathrm{bdr}}$, R_{bdr} are defined as given boundary data, where available (cf. (2.6)), or taken from the interior otherwise. This way, given boundary conditions are imposed weakly. On lateral boundaries, one can also apply the Riemann solver formulation using normal fluxes $\widehat{C}_h$, $\widehat{\boldsymbol{C}}_{\boldsymbol{u}}$, and $\widehat{C}_R$ (see Remark 4) with values from the interior and the specified boundary data, thus enforcing boundary conditions somewhat similarly to the penalty method [9].

The remaining step in the spatial discretization procedure is to choose bases for the dG spaces $\mathcal{V}^d$, $\mathcal{X}$, $\mathcal{Z}$ and to represent all variables in terms of these bases. For the barotropic system restricted to a vertical slice the entire discretization procedure is described in full detail in [Pp1]. A proof of discrete stability for the three-dimensional barotropic system was done by Aizinger [1] and Aizinger and Dawson [7] and was extended to a coupled setting (see Section 3.5) in [P8]. Numerical experiments (see e. g., [P8, 1, 6, 7, 51]) indicated for $k > 0$ order of convergence $k+1$ for the water height and horizontal velocity components and order k for the vertical velocity component. However, to our knowledge, no *a priori* error analysis for that system exists to substantiate this.

For the discretization in time, a semi-implicit time-stepping algorithm is employed. It is based on time-explicit *strong stability-preserving* (SSP) Runge–Kutta methods up to third order [42, 72, 73] that preserve the *total variation diminishing*[13] (TVD) property of the space discretization with a time-implicit treatment of vertical diffusion terms to overcome the quadratic time-step limitation of vertical eddy viscosity/diffusivity [9]. The linear system for vertical diffusion terms is solved once at the end of each time-step, thus applying the diffusion operator as a kind of correction for the advective solution.

3.4 LDG discretization of saturated groundwater flow

The area of numerical modeling for groundwater flow enjoys a great deal of attention, with a multitude of studies on development and analysis of new discretization techniques, including discontinuous Galerkin methods. For an overview of the wide range of activities refer to the review by Di Pietro and Vohralik [57].

A very detailed description of our discretization procedure for time-dependent diffusion equations using the local discontinuous Galerkin method in two spatial dimensions all the way to the linear system of equations is given in [P1], and a more rigorous presentation in the context of a coupled three-dimensional model can be found in [P8]. The most important parts of this procedure are reproduced in the following.

The first step in an LDG discretization is to rewrite the second order operator by introducing specific discharge $\widetilde{\boldsymbol{q}}$ as an auxiliary unknown and transforming problem (2.13) into a mixed formulation with corresponding changes to boundary conditions

$$\begin{aligned}
\partial_t \widetilde{h} + \boldsymbol{\nabla} \cdot \widetilde{\boldsymbol{q}} &= \widetilde{F} && \text{in } (t_0, t_{\mathrm{end}}) \times \widetilde{\Omega}\,, \\
\widetilde{\mathbf{D}}^{-1}\, \widetilde{\boldsymbol{q}} + \boldsymbol{\nabla} \widetilde{h} &= 0 && \text{in } (t_0, t_{\mathrm{end}}) \times \widetilde{\Omega}\,, \\
\widetilde{h} &= \widetilde{h}_{\mathrm{D}} && \text{on } (t_0, t_{\mathrm{end}}) \times \partial\widetilde{\Omega}_{\mathrm{D}}\,, \\
\widetilde{\boldsymbol{q}} \cdot \widetilde{\boldsymbol{n}} &= \widetilde{q}_{\mathrm{N}} && \text{on } (t_0, t_{\mathrm{end}}) \times \partial\widetilde{\Omega}_{\mathrm{N}}\,, \\
\widetilde{h} &= \widetilde{h}_0 && \text{in } \{t_0\} \times \widetilde{\Omega}\,,
\end{aligned}$$

where we have divided by the specific storage S_0 and replaced hydraulic conductivity and right hand side by $\widetilde{\mathbf{D}} \coloneqq \widetilde{\mathbf{K}}/S_0$ and $\widetilde{F} \coloneqq \widetilde{W}_0/S_0$, for simplicity.[14]

Multiplying with smooth test functions $\widetilde{\delta}$, $\widetilde{\boldsymbol{\varphi}}$ and integrating by parts over element $T \in \mathcal{T}_\Delta(\widetilde{\Omega})$ yields the *weak formulation*

$$\begin{aligned}
\left(\partial_t \widetilde{h}, \widetilde{\delta}\right)_T - \left(\widetilde{\boldsymbol{q}} \cdot \boldsymbol{\nabla}, \widetilde{\delta}\right)_T + \left\langle \widetilde{\boldsymbol{q}} \cdot \widetilde{\boldsymbol{n}}, \widetilde{\delta} \right\rangle_{\partial T} &= \left(\widetilde{F}, \widetilde{\delta}\right)_T\,, \\
\left(\widetilde{\mathbf{D}}^{-1}\, \widetilde{\boldsymbol{q}}, \widetilde{\boldsymbol{\varphi}}\right)_T - \left(\widetilde{h}\, \boldsymbol{\nabla}, \widetilde{\boldsymbol{\varphi}}\right)_T + \left\langle \widetilde{h}\, \widetilde{\boldsymbol{n}}, \widetilde{\boldsymbol{\varphi}} \right\rangle_{\partial T} &= 0\,,
\end{aligned}$$

which is well-defined for $\widetilde{h}, \widetilde{\delta} \in H^1(\widetilde{\Omega})$, $\widetilde{\boldsymbol{q}}, \widetilde{\boldsymbol{\varphi}} \in [H^1(\widetilde{\Omega})]^3$, and for a. e. $t \in [t_0, t_{\mathrm{end}}]$. Next, we approximate the solution to the weak problem by functions from finite-dimensional

[13] Sometimes also termed *total variation bounded* (TVB).

[14] There is some flexibility in placing the diffusion tensor $\widetilde{\mathbf{D}}$: Including it in the flux equation, as done here, makes it easier to prove stability of the method in a coupled setting (see [P8]), while including it in the first equation can simplify the implementation (cf. [P1]). One can also symmetrically split the tensor between both equations to obtain a symmetric formulation.

dG spaces, choose suitable numerical fluxes across element boundaries, and incorporate boundary conditions (2.13b)–(2.13c) to obtain the *semi-discrete problem*:

Seek $(\widetilde{H}(t), \widetilde{\boldsymbol{Q}}(t)) \in \mathbb{P}_k^1(\mathcal{T}_\Delta(\widetilde{\Omega})) \times \mathbb{P}_k^3(\mathcal{T}_\Delta(\widetilde{\Omega}))$ such that, for all $T \in \mathcal{T}_\Delta(\widetilde{\Omega})$, the following holds for a. e. $t \in [t_0, t_{\text{end}}]$ and for all $(\widetilde{\delta}, \widetilde{\boldsymbol{\varphi}}) \in \mathbb{P}_k^1(\mathcal{T}_\Delta(\widetilde{\Omega})) \times \mathbb{P}_k^3(\mathcal{T}_\Delta(\widetilde{\Omega}))$:

$$\left(\partial_t \widetilde{H}, \widetilde{\delta}\right)_T + \widetilde{E}_{\widetilde{H},T}(\widetilde{\boldsymbol{Q}}, \widetilde{\delta}) + \widetilde{\Lambda}_{\widetilde{H},T}(\widetilde{H}, \widetilde{\delta}) = \left(\widetilde{F}, \widetilde{\delta}\right)_T , \tag{3.6a}$$

$$\left(\widetilde{\mathbf{D}}^{-1}\widetilde{\boldsymbol{Q}}, \widetilde{\boldsymbol{\varphi}}\right)_T + \widetilde{E}_{\widetilde{\boldsymbol{Q}},T}(\widetilde{H}, \widetilde{\boldsymbol{\varphi}}) = 0 \tag{3.6b}$$

with

$$\begin{aligned}
\widetilde{E}_{\widetilde{H},T}(\widetilde{\boldsymbol{Q}}, \widetilde{\delta}) &:= -\left(\widetilde{\boldsymbol{Q}} \cdot \boldsymbol{\nabla}, \widetilde{\delta}\right)_T + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{int},T}} \left\langle \left\{\left|\widetilde{\boldsymbol{Q}}\right|\right\} \cdot \widetilde{\boldsymbol{n}}, \widetilde{\delta}\right\rangle_\gamma + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{D},T}} \left\langle \widetilde{\boldsymbol{Q}} \cdot \widetilde{\boldsymbol{n}}, \widetilde{\delta}\right\rangle_\gamma + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{N},T}} \left\langle \widetilde{q}_{\text{N}}, \widetilde{\delta}\right\rangle_\gamma , \\
\widetilde{E}_{\widetilde{\boldsymbol{Q}},T}(\widetilde{H}, \widetilde{\boldsymbol{\varphi}}) &:= -\left(\widetilde{H}\, \boldsymbol{\nabla}, \widetilde{\boldsymbol{\varphi}}\right)_T + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{int},T}} \left\langle \left\{\left|\widetilde{H}\right|\right\}\, \widetilde{\boldsymbol{n}}, \widetilde{\boldsymbol{\varphi}}\right\rangle_\gamma + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{D},T}} \left\langle \widetilde{h}_{\text{D}}\, \widetilde{\boldsymbol{n}}, \widetilde{\boldsymbol{\varphi}}\right\rangle_\gamma + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{N},T}} \left\langle \widetilde{H}\, \widetilde{\boldsymbol{n}}, \widetilde{\boldsymbol{\varphi}}\right\rangle_\gamma , \\
\widetilde{\Lambda}_{\widetilde{H},T}(\widetilde{H}, \widetilde{\delta}) &:= \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{int},T}} \frac{\eta}{\Delta_\gamma} \left\langle \left[\!\left[\widetilde{H}\right]\!\right] \cdot \widetilde{\boldsymbol{n}}, \widetilde{\delta}\right\rangle_\gamma + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\text{D},T}} \frac{\eta}{\Delta_\gamma} \left\langle \widetilde{H} - \widetilde{h}_{\text{D}}, \widetilde{\delta}\right\rangle_\gamma .
\end{aligned}$$

Here, the penalty terms in $\widetilde{\Lambda}_{\widetilde{H}}$ weakly enforce continuity of the hydraulic head across element boundaries and are required to ensure the full rank of the system in the absence of the time derivative [128, Lemma 2.15] with penalty parameter $\eta > 0$ and Δ_γ the diameter of face γ. Due to the isotropic nature of the diffusion operator, averaging of hydraulic head and specific discharge on interior faces is the natural choice for the approximation of numerical fluxes.

This element-local formulation is often easier to read and better suited for implementation. For analyis purposes, it is usually written as a global formulation by summing over all elements $T \in \mathcal{T}_\Delta(\widetilde{\Omega})$ introducing only few formal changes to the terms above: A sum over all elements in front of element-integration terms, sums over element-local index sets $\widetilde{\mathcal{I}}_{*,T}$ replaced by their global counterparts $\widetilde{\mathcal{I}}_*$, and jumps of the test functions introduced in the integration terms over interior faces.

The remaining step in the spatial discretization procedure is to introduce a local basis $\{\boldsymbol{\phi}_{i,T}\}$ for the ansatz space with support only on one element $T \in \mathcal{T}_\Delta(\widetilde{\Omega})$ such that

$$\mathbb{P}_k^d(\mathcal{T}_\Delta(\widetilde{\Omega})) = \bigcup_{T \in \mathcal{T}_\Delta(\widetilde{\Omega})} \operatorname{span}\{\boldsymbol{\phi}_{i,T}\}_{i \in \{1,\dots,N\}}$$

with N the number of local basis functions. Representing $\widetilde{H}$, $\widetilde{\boldsymbol{Q}}$ in terms of the local basis,

$$\widetilde{H}(t,x,y,z)|_T =: \sum_{i=1}^N \widetilde{H}_{i,T}(t)\, \phi_{i,T}(x,y,z)\,, \quad \widetilde{\boldsymbol{Q}}(t,x,y,z)|_T =: \sum_{i,j,k=1}^N \begin{pmatrix} \widetilde{Q}^1(t)\, \phi_{i,T}(x,y,z) \\ \widetilde{Q}^2(t)\, \phi_{j,T}(x,y,z) \\ \widetilde{Q}^3(t)\, \phi_{k,T}(x,y,z) \end{pmatrix}$$

and testing (3.6) with all basis functions yields a linear system of equations of the form

$$\underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M} \end{pmatrix}}_{=:\widetilde{\mathbf{W}}} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \partial\widetilde{\boldsymbol{H}}(t) \end{pmatrix} + \underbrace{\begin{pmatrix} \widetilde{\mathbf{M}} & \mathbf{0} & \mathbf{0} & \widetilde{\mathbf{A}}^1_{\widetilde{Q}}(t) \\ \mathbf{0} & \widetilde{\mathbf{M}} & \mathbf{0} & \widetilde{\mathbf{A}}^2_{\widetilde{Q}}(t) \\ \mathbf{0} & \mathbf{0} & \widetilde{\mathbf{M}} & \widetilde{\mathbf{A}}^3_{\widetilde{Q}}(t) \\ \widetilde{\mathbf{A}}^1_{\widetilde{H}} & \widetilde{\mathbf{A}}^2_{\widetilde{H}} & \widetilde{\mathbf{A}}^3_{\widetilde{H}} & \widetilde{\mathbf{A}}_\eta \end{pmatrix}}_{=:\widetilde{\mathbf{A}}(t)} \underbrace{\begin{pmatrix} \widetilde{\boldsymbol{Q}}^1(t) \\ \widetilde{\boldsymbol{Q}}^2(t) \\ \widetilde{\boldsymbol{Q}}^3(t) \\ \widetilde{\boldsymbol{H}}(t) \end{pmatrix}}_{=:\widetilde{\boldsymbol{Y}}(t)} = \widetilde{\boldsymbol{F}}(t) \tag{3.7}$$

with representation vectors (for $d \in \{1, 2, 3\}$)

$$\widetilde{\boldsymbol{Q}}^d(t) \coloneqq \left(\widetilde{Q}^d_{T_1,1}, \ldots, \widetilde{Q}^d_{T_1,N}, \widetilde{Q}^d_{T_2,1}, \ldots\right), \qquad \widetilde{\boldsymbol{H}} \coloneqq \left(\widetilde{H}_{T_1,1}, \ldots, \widetilde{H}_{T_1,N}, \widetilde{H}_{T_2,1}, \ldots\right).$$

Here, $\widetilde{\mathbf{M}}$ is the block-diagonal[15] mass-matrix, $\widetilde{\mathbf{A}}^d_*$ are sparse block-matrices, $\widetilde{\mathbf{A}}_\eta$ is a block-matrix containing the penalty terms from $\widetilde{\Lambda}_{\widetilde{H}}$, and $\widetilde{\boldsymbol{F}}$ contains the right hand side and boundary data. This is presented in more detail with extensive description of all matrix entries for a two-dimensional domain in [P1].

A priori error analysis yields $\min\{\hat{k}, k+1\}$ as convergence order estimates for the semi-discrete scheme with $\hat{k}$ and k representing the approximation orders for $\widetilde{H}$ and $\widetilde{\boldsymbol{Q}}$, respectively [10, 131, 132]. In numerical experiments, orders of convergence were obtained as $k+1$ for $\widetilde{H}$ and k for $\widetilde{\boldsymbol{Q}}$ when $k \geq 1$, and the problem is linear [P1, P8].

To complete the discretization procedure, we choose an implicit Euler method to discretize system (3.7) in time. Let $t_0 = \widetilde{t}^0 < \widetilde{t}^1 < \cdots < t_{\text{end}}$ be a not necessarily equidistant decomposition of the time interval $[t_0, t_{\text{end}}]$, and let $\Delta\widetilde{t}^n \coloneqq \widetilde{t}^{n+1} - \widetilde{t}^n$ denote the time step size, then one step of the time discretization is formulated as

$$\left(\widetilde{\mathbf{W}} + \Delta\widetilde{t}^n\, \widetilde{\mathbf{A}}\left(\widetilde{t}^{n+1}\right)\right) \widetilde{\boldsymbol{Y}}\left(\widetilde{t}^{n+1}\right) = \widetilde{\mathbf{W}}\, \widetilde{\boldsymbol{Y}}\left(\widetilde{t}^n\right) + \Delta\widetilde{t}^n\, \widetilde{\boldsymbol{F}}\left(\widetilde{t}^{n+1}\right).$$

To make efficient use of high order dG approximations, higher order time discretizations must be employed instead such as *diagonally implicit Runge–Kutta* (DIRK) methods that are presented and used together with a hybridized method in [P6].

3.5 Coupled model

When coupling the models for free-surface and subsurface flows one of the biggest challenges is the difference in time scales. At the surface, the water velocity is in the range of meters per second with surface waves often traveling at substantially higher speeds while the subsurface flow velocity is in the range of decimeters per day. Additionally, the subsurface flow model is diffusion-dominated and must be discretized implicitly in time (cf. Section 3.4) to avoid extreme restrictions on the time step size. In contrast to that, the free-surface flow model uses a semi-implicit time-stepping scheme (cf. Section 3.3), and, consequently, the free-surface flow time step Δt must be significantly smaller than the one for the subsurface problem $\Delta\widetilde{t}$. This difference in time-stepping methods must be accounted for to ensure the coupling is mass conservative.

For that, a so-called *non-simple* or *complex* boundary is used, which memorizes the flux from $\widetilde{\Omega}$ into $\Omega(t)$ across the interior boundary and vice versa. Since $\Delta t / \Delta\widetilde{t} < 1$, the flux from $\Omega(t)$ to $\widetilde{\Omega}$ has to be time averaged to filter out fast pressure changes in the interior boundary condition (2.14b), yielding

$$\widetilde{H}|_{\partial\widetilde{\Omega}_{\text{top}}}\left(\widetilde{t}^{n+1}\right) = \frac{1}{\Delta\widetilde{t}} \int_{\widetilde{t}^n}^{\widetilde{t}^{n+1}} \left(\Xi(t) + \frac{\boldsymbol{U}_{xy}(t) \cdot \boldsymbol{U}_{xy}(t)}{2\,g}\right)\Big|_{\partial\Omega_{\text{bot}}} \mathrm{d}t\,. \tag{3.8}$$

This time-averaging can be approximated by a summated trapezoidal rule. For the normal flux from $\widetilde{\Omega}$ to $\Omega(t)$ (cf. equation (2.14a)), we rely on the values from the latest subsurface time step.

[15] When using an orthogonal basis, the mass-matrix is simply a diagonal matrix.

The necessary changes to the discrete models of Sections 3.3 and 3.4 are minimal: In the free surface model, an additional term is introduced in forms $A_{\boldsymbol{U},T}$, $A_{W,T}$, and $E_{\mathbf{Q},T}$ (for baroclinic simulations also in $A_{R,T}$) of system (3.5) that replaces the usual zero normal flow boundary condition specified in equation (2.6a). For the groundwater flow model (cf. system (3.6)), a top-boundary term in the same form as for Dirichlet boundary data $\widetilde{h}_{\mathrm{D}}$ is added in forms $\widetilde{E}_{\widetilde{\boldsymbol{Q}}}$ and $\widetilde{\Lambda}_{\widetilde{H},T}$ with the averaged head from equation (3.8).

For discrete stability, consider the barotropic free surface system (i. e., (3.5a)–(3.5d) without baroclinic pressure correction p), omit the Coriolis term on the right-hand side, and rescale the system so that $g = 1$. Denote by $\partial\Omega_i$ lateral inflow boundaries, and let $\hat{\xi}, \hat{\boldsymbol{u}}_{xy}$ be prescribed boundary values. Using a Lax-Friedrichs-type Riemann solver (see Remark 4 in Section 3.3) and introducing an additional mesh penalty term $\frac{1}{2}\partial_t(\Xi_{\mathrm{s}} - \Xi)\boldsymbol{U}_{xy}$ in equation (3.5b) that penalizes the difference between the computed (discontinuous) free surface elevation field and the smoothed (continuous) free surface (see Remark 1 in Section 3.3), the following theorem holds [P8]:

Theorem 3.1 *(Discrete Stability).* Let the free surface elevation of the smoothed mesh satisfy $\Xi_{\mathrm{s}}|_{\Pi(\partial\Omega_i)} = \xi_i$, and let $\Xi, \delta_H \in \mathbb{P}^1_{2k}(\mathcal{T}_\Delta(\Omega_{xy}))$, $\boldsymbol{U}_{xy}, \boldsymbol{\varphi}_u \in \mathbb{P}^2_k(\mathcal{T}_\Delta(\Omega(t)))$, $W, \delta_w \in \mathbb{P}^1_{2k}(\mathcal{T}_\Delta(\Omega(t)))$, $\mathbf{Q}, \boldsymbol{\Phi} \in [\mathbb{P}^3_k(\mathcal{T}_\Delta(\Omega(t)))]^2$, $\widetilde{H}, \widetilde{\delta} \in \mathbb{P}^1_{\hat{k}}(\mathcal{T}_\Delta(\widetilde{\Omega}))$, and $\widetilde{\boldsymbol{Q}}, \widetilde{\boldsymbol{\varphi}} \in \mathbb{P}^3_{\bar{k}}(\mathcal{T}_\Delta(\widetilde{\Omega}))$ for some $k, \bar{k}, \hat{k} \geq 0$, a. e. $t \in [t_0, t_{\mathrm{end}}]$, and all $T \in \mathcal{T}_\Delta(\Omega(t)), \tilde{T} \in \mathcal{T}_\Delta(\widetilde{\Omega})$. Then scheme (3.5a)–(3.5d), (3.6) is stable in the following sense:

$$\partial_t \left\{ \|\Xi\|^2_{\Omega_{xy}} + \|\boldsymbol{U}_{xy}\|^2_{\Omega(t)} + \left\|\widetilde{H}\right\|^2_{\widetilde{\Omega}} \right\} + \left\| \sqrt{\mathbf{D}^{-1}}\mathbf{Q} \right\|^2_{\Omega(t)} + \sum_{\gamma \in \mathcal{I}_{\mathrm{int}}} \|[\![\boldsymbol{U}_{xy}]\!]\|^2_\gamma + \left\| \sqrt{\widetilde{\mathbf{D}}^{-1}}\widetilde{\boldsymbol{Q}} \right\|^2_{\widetilde{\Omega}}$$
$$+ \sum_{\gamma \in \widetilde{\mathcal{I}}_{\mathrm{int}}} \frac{\eta}{\Delta_\gamma} \left\| \left[\!\left[\widetilde{H}\right]\!\right] \right\|^2_\gamma + \sum_{\gamma \in \widetilde{\mathcal{I}}_{\mathrm{D}}} \frac{\eta}{\Delta_\gamma} \left\| \widetilde{H} \right\|^2_\gamma \leq C(C_{\mathrm{t}}, \Omega(t), \widetilde{\Omega}, \mathbf{D}, \widetilde{\mathbf{D}}, \eta, \boldsymbol{F}_u, \widetilde{F}, z_{\mathrm{b}}, \hat{\xi}, \hat{\boldsymbol{u}_{xy}}, \widetilde{q}_{\mathrm{N}}, \Delta),$$

where C_{t} is the constant from the discrete trace inequality. ■

The corresponding discretization procedure and proof of the theorem is presented in [P8] including numerical experiments and numerical convergence studies carried out in the framework of our toolbox FESTUNG (see Section 5.1). Implementation aspects of the coupled model are discussed in [Pp1].

3.6 Slope limiting

Slope and flux limiters are widely used techniques to enforce the discrete maximum principle in finite volume and discontinuous Galerkin methods for conservation laws. In the context of dG approximations in space and explicit TVD time stepping schemes, slope limiters exploit the fact that the lowest-order (piecewise constant) part of a dG solution is guaranteed to preserve the monotonicity of the solution and produce no spurious extrema, thus yielding admissible maximum and minimum values for the higher-order solution. While flux limiters change numerical fluxes, slope limiters modify the solution to meet these requirements in chosen control points [108]. The challenge in formulating a limiter lies in effectively detecting and eliminating violations of these extrema while preserving the approximation order in smooth regions and keeping the solution free of spurious side effects such as excessive numerical diffusion.

With the classical *minmod* limiter of Cockburn and Shu [42, 45] as a starting point, a number of different limiting approaches for linear dG solutions were introduced and improved

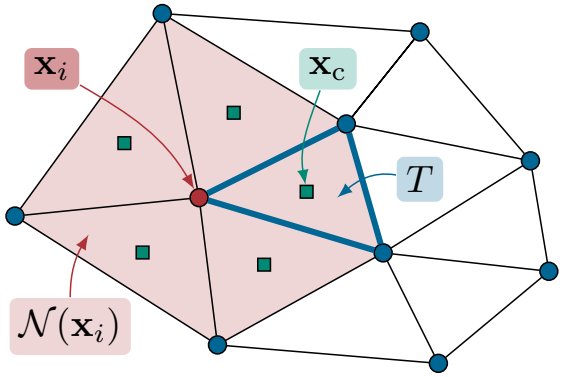

Figure 3.3: Elements in the neighbourhood $\mathcal{N}(\boldsymbol{x}_i) \coloneqq \{T \in \mathcal{T}_\Delta(\Omega_{xy}) : \boldsymbol{x}_i \in T\}$ of a vertex $\boldsymbol{x}_i \in T$ with corresponding centroids from which admissible minimum and maximum values are determined

over the course of the years [87, 104, 105]. The underlying idea is to locally modify the solution on an element so that the maximum principle is fulfilled in certain control points without affecting local mass conservation. For that, the class of *vertex-based slope limiters*, where vertices of the computational mesh are used as control points, imposes less restrictive constraints and thus proved to be superior compared to algorithms where edge / face midpoints are employed. Most notable is the modification of the established Barth-Jespersen (edge midpoint) limiter [21] to a vertex based formulation for dG methods by Kuzmin [107], and an equivalent formulation as a local constrained optimization problem with application to three-dimensional shallow-water flow by Aizinger [2]. These vertex-based limiters can be extended to work on higher order (polynomial degree 2 or higher) dG solutions in a hierarchical manner [107, 109, 110] yielding a method that preserves as much information as possible in contrast to the classical approach of applying linear limiters where needed and discarding higher-order parts of the solution [119].

The following illustrates the limiting procedure for a solution $c_\Delta(x, y) \in \mathbb{P}_k^1(T)$ defined on an element T from the partition of a two-dimensional domain $\mathcal{T}_\Delta(\Omega_{xy})$ with a comprehensive description given in [P3]. For three-dimensional domains, the algorithm is the same.

The goal of linear vertex-based limiters is to determine the maximum admissible slope on all elements $T \in \mathcal{T}_\Delta(\Omega_{xy})$ for a linear reconstruction of the form

$$c_\Delta(\boldsymbol{x}) = c_\mathrm{c} + \alpha_T \, (\boldsymbol{\nabla} c)_\mathrm{c} \, (\boldsymbol{x} - \boldsymbol{x}_\mathrm{c}) \,, \tag{3.9}$$

where c_c and $(\boldsymbol{\nabla} c)_\mathrm{c}$ are the values of the unconstrained solution and its gradient in the centroid $\boldsymbol{x}_\mathrm{c} \coloneqq \frac{1}{|T|} \int_T \boldsymbol{x} \, \mathrm{d}\boldsymbol{x}$, respectively. The correction factor $\alpha_T \in [0, 1]$ is chosen such that the above reconstruction is bounded in all vertices $\boldsymbol{x}_i \in T$ by the minimum and maximum centroid values of all elements containing $\boldsymbol{x}_i$, i. e.,

$$\begin{aligned} &c_i^\mathrm{min} \leq c_\Delta(\boldsymbol{x}_i) \leq c_i^\mathrm{max} \qquad \text{for all vertices } \boldsymbol{x}_i \in T \,, \\ &\text{where} \qquad c_i^\mathrm{min} \coloneqq \min_{T' \in \mathcal{N}(\boldsymbol{x}_i)} c_\mathrm{c}|_{T'} \qquad \text{and} \qquad c_i^\mathrm{max} \coloneqq \max_{T' \in \mathcal{N}(\boldsymbol{x}_i)} c_\mathrm{c}|_{T'} \end{aligned} \tag{3.10}$$

are the minimum and maximum centroid values of all elements in the neighbourhood $\mathcal{N}(\boldsymbol{x}_i)$ of vertex $\boldsymbol{x}_i \in T$ (see Figure 3.3 for an illustration). The correction factor α_T is defined

as [107]

$$\alpha_T \coloneqq \min_{\boldsymbol{x}_i \in T} \begin{Bmatrix} (c_i^{\max} - c_\mathrm{c})/(c_i - c_\mathrm{c}) & \text{if} & c_i > c_i^{\max} \\ 1 & \text{if} & c_i^{\min} \leq c_i \leq c_i^{\max} \\ (c_i^{\min} - c_\mathrm{c})/(c_i - c_\mathrm{c}) & \text{if} & c_i < c_i^{\min} \end{Bmatrix}, \tag{3.11}$$

where c_i is the unconstrained linear reconstruction in vertex $\boldsymbol{x}_i$.

For higher polynomial approximation orders this means that the solution is reduced to a linear reconstruction on elements violating the maximum principle. Yang and Wang [167, 168] described an edge/face midpoint limiting procedure that keeps higher order terms and multiplies all derivatives of order l by a common correction factor $\alpha_T^{(l)}$. Kuzmin [107, 109, 110] incorporated this idea into the framework of vertex-based limiters and described the resulting algorithm for quadratic approximations. The first closed-form expression of this limiting procedure for arbitrary-order discretizations was given in [P3] and is briefly outlined in the following.

Let $\mathcal{A}_l \coloneqq \{\boldsymbol{a} \in \mathbb{N}_0^2 : |\boldsymbol{a}| = l\}$ be the set of all two-dimensional multi-indices $\boldsymbol{a} \coloneqq (a_1, a_2)^\mathrm{T}$ of order l, and let k be the polynomial approximation order of the dG discretization. Using some standard multi-index notation

$$|\boldsymbol{a}| = a_1 + a_2\,, \qquad \boldsymbol{a}! = a_1!\,a_2!\,, \qquad \boldsymbol{x}^{\boldsymbol{a}} = x^{a_1}\,y^{a_2}\,, \qquad \partial_{\boldsymbol{a}} = \partial_x^{a_1}\,\partial_y^{a_2}\,,$$

we consider the Taylor series expansion of c_Δ around centroid $\boldsymbol{x}_\mathrm{c}$ on element $T \in \mathcal{T}_\Delta(\Omega_{xy})$

$$c_\Delta(\boldsymbol{x}) = \sum_{0 \leq |\boldsymbol{a}| \leq k} (\partial_{\boldsymbol{a}} c)_\mathrm{c}\, \frac{(\boldsymbol{x} - \boldsymbol{x}_\mathrm{c})^{\boldsymbol{a}}}{\boldsymbol{a}!}\,.$$

Then, correction factors $\alpha_T^{(l)}$ for each order $0 < l \leq k$ are determined using linear reconstructions for all partial derivatives of order $l-1$ in vertices $\boldsymbol{x}_i \coloneqq (x_i, y_i)^\mathrm{T} \in T$

$$c_{\boldsymbol{a},i} \coloneqq c_{\boldsymbol{a}} + c_{\boldsymbol{a}+(1,0)}\,(x_i - x_\mathrm{c}) + c_{\boldsymbol{a}+(0,1)}\,(y_i - y_\mathrm{c}) \qquad \forall \boldsymbol{a} \in \mathcal{A}_{l-1}\,, \tag{3.12}$$

where we abbreviated the partial derivatives in the centroid by $c_{\boldsymbol{a}} \coloneqq (\partial_{\boldsymbol{a}} c)_\mathrm{c}$. Computing $\alpha_{\boldsymbol{a},T}^{(l)}$ for each derivative using the expression in equation (3.11) yields the correction factor

$$\alpha_T^{(l)} = \min_{\boldsymbol{a} \in \mathcal{A}_{l-1}} \alpha_{\boldsymbol{a},T}^{(l)}\,, \quad \text{with} \quad \alpha_{\boldsymbol{a},T}^{(l)} \coloneqq \min_{\boldsymbol{x}_i \in T} \begin{Bmatrix} (c_{\boldsymbol{a},i}^{\max} - c_{\boldsymbol{a}})/(c_{\boldsymbol{a},i} - c_{\boldsymbol{a}}) & \text{if} & c_{\boldsymbol{a},i} > c_{\boldsymbol{a},i}^{\max} \\ 1 & \text{if} & c_{\boldsymbol{a},i}^{\min} \leq c_{\boldsymbol{a},i} \leq c_{\boldsymbol{a},i}^{\max} \\ (c_{\boldsymbol{a},i}^{\min} - c_{\boldsymbol{a}})/(c_{\boldsymbol{a},i} - c_{\boldsymbol{a}}) & \text{if} & c_{\boldsymbol{a},i} < c_{\boldsymbol{a},i}^{\min} \end{Bmatrix},$$

where $c_{\boldsymbol{a},i}^{\max}$, $c_{\boldsymbol{a},i}^{\min}$ are defined as in (3.10). Recognizing that lower-order derivatives are typically smoother than higher-order derivatives, lower-order terms should not be limited stronger than any higher orders. Beginning with the highest-order derivatives, the limited solution becomes

$$c_\Delta(\boldsymbol{x}) = c_\mathrm{c} + \sum_{1 \leq |\boldsymbol{a}| \leq k} \alpha_T^{(|\boldsymbol{a}|)}\, c_{\boldsymbol{a}}\, \frac{(\boldsymbol{x} - \boldsymbol{x}_\mathrm{c})^{\boldsymbol{a}}}{\boldsymbol{a}!} \qquad \text{with} \quad \alpha_T^{(l)} \coloneqq \max_{l \leq j \leq k} \alpha_T^{(j)} \qquad \forall\, 0 < l \leq k\,.$$

However, this approach does not produce solutions that satisfy the maximum principle in all cases. In particular, for jumps in the solution, the assumption about the smoothness of lower-order derivatives does not always hold possibly causing problems in applications where compliance to these bounds is mandatory. For that reason, [P3] suggests a stricter version of the hierarchical vertex-based limiter that modifies two key components of the procedure:

1. Replace the linear reconstruction in (3.12) by a full reconstruction of the derivatives

$$c_{\boldsymbol{a},i} \coloneqq \sum_{0 \leq |\boldsymbol{b}| \leq k-l} c_{\boldsymbol{a}+\boldsymbol{b}} \frac{(\boldsymbol{x} - \boldsymbol{x}_\mathrm{c})^{\boldsymbol{b}}}{\boldsymbol{b}!} .$$

2. Instead of the hierarchical restriction of correction factors, begin with the highest-order derivatives and apply coefficients immediately to all terms involved yielding

$$c_\Delta(\boldsymbol{x}) = c_\mathrm{c} + \sum_{1 \leq |\boldsymbol{a}| \leq k} \left(\alpha_T^{(1)} \cdots \alpha_T^{(|\boldsymbol{a}|)} \right) c_{\boldsymbol{a}} \frac{(\boldsymbol{x} - \boldsymbol{x}_\mathrm{c})^{\boldsymbol{a}}}{\boldsymbol{a}!} .$$

For the implementation of slope limiters, c_Δ is usually represented in a Taylor basis where degrees of freedom coincide with partial derivatives, and thus correction factors can be directly applied (see [P3] for details). In the case of a non-orthogonal Taylor basis (e. g., on triangular meshes), the mass matrix is non-diagonal and introduces an implicit coupling of second or higher-order degrees of freedom in time-dependent problems [109]. This is also the case when using an orthogonal modal basis for computations, where the necessary change to a Taylor-basis representation before applying the slope limiter introduces these dependencies. As a countermeasure, Kuzmin [109] suggested to apply the slope limiter not only to the new solution at a time step but also to the vector of time derivatives (i. e., the right hand side in the explicit time-stepping algorithm). Combined with a lumping of the mass matrix in the time discretization and a balancing correction term added to the right hand side, spurious dependencies are filtered out, and the results greatly improve [109]. This eliminates the deficiencies of higher-order limiting procedures previously observed [119]. A description of this procedure for arbitrary basis representations and numerical experiments can be found in [P3].

For anisotropic solutions with strong directional derivatives, the uniform choice of correction factors for all derivatives of the same order is not optimal. Instead, a limiting procedure should discern these features and incorporate them into the changes applied to the solution. That is, considering the limited linear solution in equation (3.9), the correction factor α_T is replaced by factors for each directional derivative

$$c_\Delta \coloneqq c_\mathrm{c} + \alpha_x \, c_{(1,0)} \, (x - x_\mathrm{c}) + \alpha_y \, c_{(0,1)} \, (y - y_\mathrm{c}) . \tag{3.13}$$

May and Berger [118] described a way to choose $\alpha_x, \alpha_y \in [0, 1]$ based on solving small linear programming problems that minimize changes to the solution subject to the constraints in equation (3.10). To overcome the computational cost associated with this optimization-based algorithm, [P5] proposed a closed-form expression for anisotropic slope limiting. The underlying idea is the following: Without loss of generality, start with the x-direction and add the prelimited x-variation to the mean value to obtain

$$\hat{c}_\Delta(\boldsymbol{x}) = c_\mathrm{c} + \alpha_x \, c_{(1,0)} \, (x - x_\mathrm{c}) ,$$

where correction factor α_x is chosen such that inequality (3.10) is fulfilled for $\hat{c}_\Delta(\boldsymbol{x})$. Thus, a modified version of (3.11) is used to compute α_x from the x-variation

$$\alpha_x \coloneqq \min_{\boldsymbol{x}_i \in T} \begin{Bmatrix} \frac{c_i^{\max} - c_\mathrm{c}}{c_{(1,0)}(x_i - x_\mathrm{c})} & \text{if} & c_{(1,0)} > \frac{c_i^{\max} - c_\mathrm{c}}{x_i - x_\mathrm{c}} \\ 1 & \text{if} & \frac{c_i^{\min} - c_\mathrm{c}}{x_i - x_\mathrm{c}} \leq c_{(1,0)} \leq \frac{c_i^{\max} - c_\mathrm{c}}{x_i - x_\mathrm{c}} \\ \frac{c_i^{\min} - c_\mathrm{c}}{c_{(1,0)}(x_i - x_\mathrm{c})} & \text{if} & c_{(1,0)} < \frac{c_i^{\min} - c_\mathrm{c}}{x_i - x_\mathrm{c}} \end{Bmatrix} .$$

Adding the y-variation to $\hat{c}_\Delta$ multiplied by the corresponding correction factor α_y enforcing (3.10) yields the limited solution in equation (3.13) with α_y computed as

$$\alpha_y \coloneqq \min_{\boldsymbol{x}_i \in T} \left\{ \begin{array}{lc} \frac{c_i^{\max} - \hat{c}_\Delta(\boldsymbol{x}_\mathrm{c})}{c_{(0,1)}(y_i - y_\mathrm{c})} & \text{if} \quad c_{(0,1)} > \frac{c_i^{\max} - \hat{c}_\Delta(\boldsymbol{x}_\mathrm{c})}{y_i - y_\mathrm{c}} \\ 1 & \text{if} \quad \frac{c_i^{\min} - \hat{c}_\Delta(\boldsymbol{x}_\mathrm{c})}{y_i - y_\mathrm{c}} \leq c_{(0,1)} \leq \frac{c_i^{\max} - \hat{c}_\Delta(\boldsymbol{x}_\mathrm{c})}{y_i - y_\mathrm{c}} \\ \frac{c_i^{\min} - \hat{c}_\Delta(\boldsymbol{x}_\mathrm{c})}{c_{(0,1)}(y_i - y_\mathrm{c})} & \text{if} \quad c_{(0,1)} < \frac{c_i^{\min} - \hat{c}_\Delta(\boldsymbol{x}_\mathrm{c})}{y_i - y_\mathrm{c}} \end{array} \right\} .$$

Since anisotropies of the solution are not necessarily aligned with the axes of the Cartesian coordinate system, and the order of the sequential limiting procedure potentially affects the outcome of the algorithm, this approach has been generalized to an arbitrary frame of reference: Utilizing a coordinate transform, the local coordinate system is rotated around the centroid $\boldsymbol{x}_\mathrm{c}$ by an angle ϑ to align it with a pair of orthonormal direction vectors $\boldsymbol{\xi} \coloneqq (\cos\vartheta, -\sin\vartheta)^\mathrm{T}$ and $\boldsymbol{\eta} \coloneqq (\sin\vartheta, \cos\vartheta)^\mathrm{T}$.

By that, the anisotropic limiting procedure can be carried out for correction factors α_ξ and α_η corresponding to the directional derivatives. Aligning the local coordinate system with the reconstructed gradient of the solution, thus limiting in the direction of and perpendicular to the solution gradient separately, proved to be a sensible choice. For a detailed description of this generalization and numerical examples, including a comparison to the optimization-based anisotropic limiter, refer to [P5].

3.7 Locally filtered transport

A common application of circulation models, e. g., based on the two-dimensional or three-dimensional shallow water equations, is to use the computed hydrodynamics as input for multi-component transport–reaction models to simulate ecosystem evolution or pollutant dispersion. In such systems, many of the physical, chemical, and biological processes involved take place on highly-localized spatial and temporal scales, while outside of these regions of interest transported constituents remain at quasi-uniform concentration levels or are completely absent. However, in standard transport models the treatment of advective transport terms induces the same computational cost everywhere irrespective of the fact that this does not change the values of unknowns in large parts of the computational domain.

The *Locally Filtered Transport* (LFT) scheme proposed in [P7] offers a way to adaptively turn on / off the computation of discrete transport terms, thus decreasing computational work significantly while remaining locally mass conservative and maintaining a similar level of accuracy. It is based on the work by Hodges [85] for structured finite volume methods but avoids its unconventional and computationally expensive mass transport algorithm. Instead, in the context of dG methods, it comes at almost no additional cost as it can make use of the minimum and maximum values computed for vertex based slope-limiters (see Section 3.6). This allows to derive lists of active and disabled elements depending on the difference of these maximum and minimum values being larger than a chosen threshold in each element's neighborhood.

In [P7], the performance of the method is demonstrated for the NPZ ecosystem model of nitrogen, phytoplankton, and zooplankton dynamics within our FESTUNG framework (see Section 5.1). However, the method can easily be extended to other numerical methods and applications.

Chapter 4

High performance computing aspects

4.1 Computing architectures

While a detailed discussion of modern processors is clearly out of scope, this section is intended to give a rough overview of the features and characteristics of current computing architectures most relevant to the applications in this thesis. More details on architectures and modern processors with a focus on scientific computing applications are presented intelligibly in the book by Hager and Wellein [75].

Today's processors still adhere to the original principles of the *stored-program computer architecture*, which became colloquially known as the *von Neumann architecture* following John von Neumann's design proposal in his *First Draft of a Report on the EDVAC* [121] from 1945. The logical design (cf. Figure 4.1) is divided into a control unit, arithmetic logic units (ALU), memory, and input/output (I/O) parts with the distinguishing property, compared to earlier designs, that instructions are data that are stored in memory. The combination of ALU and control unit (responsible for reading and executing instructions from memory) together with interfaces to I/O and memory (usually hierarchically structured, with caches) are referred to as the *Central Processing Unit* (CPU) [75, 148].

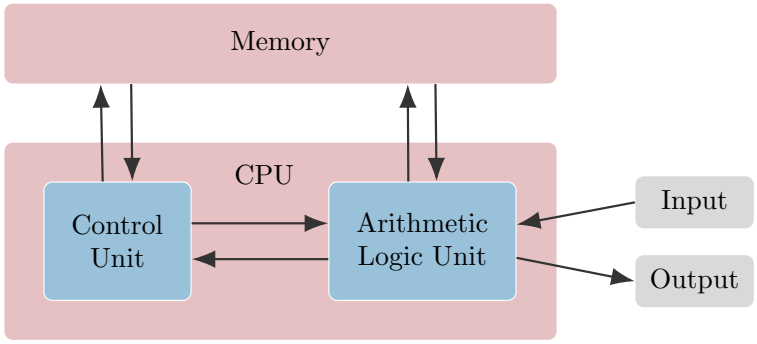

Figure 4.1: Schematic of the stored-program computer architecture (adapted from [75, Fig. 1.1], [148, Abb. 1.4])

Modern CPUs are microprocessors with sophisticated implementations of these original components: Control units are capable of superscalar[1], pipelined[2], and out-of-order[3] execution of instructions; highly optimized integer and floating point arithmetic units offer support for vectorized *SIMD* (Single Instruction, Multiple Data) instructions; a hierarchy

[1] Dispatch multiple instructions to different execution units at the same time
[2] Split instructions in multiple phases to allow partial overlapping of instructions in the same execution unit
[3] Rearrange the order of instructions to avoid idle cycles

of memory levels is used starting from a large but comparatively slow main memory via (typically three) levels of increasingly faster and smaller caches up to few very fast registers, where operands of instructions have to reside during computation; advanced memory management units are responsible for coherence of these caches and capable of (speculative) prefetching of data from lower levels. Moreover, all computers today possess some form of parallelism, which appears on multiple levels:

- On the arithmetic level, in the form of SIMD instructions allowing to apply the same operation to multiple operands at the same time;
- on the instruction level, independent instructions (with regard to their inputs and outputs) may be executed in parallel by dispatching them to different execution units (e. g., multiple ALUs within one core);
- on the chip level, as multicore processors, where several processors ("cores") are placed on a single chip often sharing some components, e. g., cache levels or memory and I/O interfaces;
- on the node level, where multiple (multicore) processors share resources such as memory[4] and network connectivity, potentially accompanied by accelerator cards to offload computations;
- on the system level with multiple nodes connected by a high-speed interconnect.

At the time of writing, processors by Intel and AMD using the *AMD64 instruction set*[5] established themselves as the quasi-standard for desktop computers and servers. The same is largely true in the world of high performance computing (HPC); however, microprocessors based on other *Instruction Set Architectures* (ISA) time and again feature prominently in the list of the fastest supercomputers (e. g., IBM Power, SPARC64) [147]. Relatively recent is the appearance of architectures from energy-constrained scenarios (mobile) such as ARM processors. Often, systems using these architectures are characterized by a larger number of light-weight (and often energy-efficient) processors combined into a single chip. For example, the Post-K supercomputer will use ARM-based CPUs [115], and the Astra supercomputer was the first ARM supercomputer to enter the Top500 [144]. A number of different computing platforms are compared with respect to the performance of our regional ocean model in [P2], and evaluations of the same model on low-power ARM processors are the subject of [P4] and [C1].

The use of co-processors (also called accelerators, mostly in the form of graphics cards) is a popular choice in HPC systems to offload compute-intensive parts while the outer algorithm and I/O operations are executed on regular CPUs. Such *Graphics Processing Units* (GPUs) feature a many-core architecture designed to handle data-parallel and throughput-intensive workloads, which is why they provide a large number of arithmetic units along with relatively few control units and very fast memory. This makes them particularly well-suited to applications that make heavy use of linear algebra operations on large data sets [143], but algorithms with unstructured data access patterns (e. g., due to unstructured meshes) or frequent branching hardly benefit from such co-processors [68, 148]. Moreover, GPUs

[4] In some multiprocessor computers or sometimes even multicore processors, all available memory is only logically shared but is, in fact distributed among the processors dividing it into multiple *NUMA* (Non-Uniform Memory Access) domains [75]. In NUMA architectures, access to data in remote sections of the memory must pass through a dedicated interconnect or bus, which reduces memory bandwidth and increases latency.

[5] Often named *x86_64* due to the *x86 instruction set* of its predecessor

require the programmer to employ a completely different programming model[6] and thus make it necessary to rewrite large code parts of existing applications.

A similar architecture is used in Intel's Xeon Phi processors that were originally offered as co-processors and evolved to stand-alone manycore processors in their latest generation before their discontinuation. In contrast to GPUs, they consist of a large number of x86-compatible cores and as such are capable of running codes using traditional parallel programming models and languages [62, 82]. However, to make optimal use of Xeon Phi processors, the algorithm must be suitable for massive parallelism and vectorized instructions, which once again has to be provided for by the programmer, thus defeating one of the original advantages of Xeon Phi processors over GPUs [62]. This need for a specific adaptation was also observed when evaluating the performance of our regional ocean model UTBEST3D (see Section 5.2) on a broad range of different architectures including Intel Xeon Phi co-processors in [P2].

4.2 Node-level performance

A program or implementation of an algorithm is considered to be fast or efficient when it delivers an acceptable level of performance according to some metric. The established metric for performance

$$P = \frac{W}{T} \tag{4.1}$$

is the ratio between the amount of conducted work W and the required time T [75, 76]. While time typically refers to execution or wall-clock time,[7] a greater variety of choices exists to quantify the work conducted: In numerical simulations, this is typically related to the number of *floating-point operations* (FLOP), but one could also be interested in the count of iterations of a loop or the number of higher-level update operations executed.[8] In the context of this work, we are mainly interested in *time-to-solution* or *energy-to-solution* measurements, either for a certain part of the total algorithm or the total program run (up to a selected end time), which means W is equivalent to one pass through the corresponding part of the program. Nonetheless, FLOP are a useful metric when evaluating the efficiency of a program in regard to the capability of an architecture.

The maximum number of floating-point operations that can be carried out per second is a hardware-specific value and referred to as *peak performance* P_{peak} [FLOP/s], determined by multiplying the number of processing units (number of cores) with the clock frequency (number of clock cycles per second) and the number of floating point operations that can be carried out per clock cycle by a single core.[9] A program that reaches a significant fraction of this value is considered to be efficient – providing that the operations executed actually comprise useful work. However, performance is often limited by other factors: On the one hand, FLOP require data to operate on, which has to be loaded from and stored to memory.

[6] For example, NVIDIA's *CUDA toolkit* (`https://developer.nvidia.com/cuda-toolkit`) or standards such as *OpenCL* (`https://www.khronos.org/opencl/`) or *OpenACC* (`https://www.openacc.org/`).
[7] In some applications or investigations, other units such as number of clock cycles could be useful as well.
[8] For example, in climate simulations, a typical metric is the number of simulated years per day.
[9] Typically, more than one floating-point operation can be executed per clock cycle when using SIMD or *fused multiply-add* (FMA) instructions. One also has to distinguish between *single precision* and *double precision* operations, of which the latter is the more relevant for most scientific computing applications.

On the other hand, there is always some overhead involved due to branching, integer arithmetic (e. g., indexing of memory locations), or I/O operations, all of which should be kept at the minimum amount possible. Each load/store-operation involves some latency, and the memory interface has a hardware-specific *memory bandwidth* b [Byte/s] that gives the maximum amount of data that can be transfered per second. For linear memory access patterns, latency can often be hidden by overlapping data transfer with useful instructions [75]. This is accomplished using prefetchers and speculative execution built into modern processors. Thus, an algorithm that executes a relatively small amount of floating-point operations on a large data set will usually saturate the memory bandwidth before reaching the theoretical peak performance. It is useful to determine the ratios between required data transfer volume V [Byte] and arithmetic instructions W [FLOP] on the one side and between memory bandwidth and peak performance on the other yielding

$$\textit{code balance}\quad B_\mathrm{c} = \frac{V}{W}\ [\mathrm{Byte/FLOP}] \quad \text{and } \textit{machine balance}\quad B_\mathrm{m} = \frac{b}{P_\mathrm{peak}}\ [\mathrm{Byte/FLOP}]\,.$$

If $B_\mathrm{c} \geq B_\mathrm{m}$, performance is expected to be limited by the memory bandwidth of the hardware instead of the peak performance [75]. With increasing number of cores in modern processors and the moderate grow of the memory bandwidth shared by those, machine balance is presumed to decrease further and more applications will be limited by memory bandwidth [75].

Using these machine and code characteristics, the *Roofline performance model* [162] even allows to make a prediction of the (theoretically) achievable maximum performance

$$P_\mathrm{max} = \min\left\{P_\mathrm{peak}, I \cdot b\right\}\,,$$

where $I = 1/B_\mathrm{c}$ is called *computational intensity* [76]. Better predictions are possible when more sophisticated performance models are used such as the *Execution-Cache-Memory* (ECM) model that takes the cache hierarchy into account [146]. However, applying such a model is a time-consuming task only feasible for few compute-intensive loops in a program.

Generally, shorter time-to-solution can be achieved in two ways: By improving the resource utilization until either memory bandwidth or peak performance are the bottleneck, or by adding more processors and sharing the work among them. While the latter comes with its own set of challenges (this is the topic of the next section), a moderate level of parallelization must also be taken into account when looking at resource utilization. In today's processors, the available memory bandwidth is typically shared by some or all cores of a multicore processor, and clock frequency may be adjusted depending on the type of instructions. Thus, it is useful to look at the performance of an application on a per-socket or (more general) per-node level and aim at achieving a code balance that matches the machine characteristics. This results in an iterative process, called *node-level performance engineering* that includes measuring performance for a suitable benchmark, identifying bottlenecks, attempting to overcome these, and repeating from the beginning.

An analysis of the runtime distribution for our regional ocean model UTBEST3D (cf. Section 5.2) is included in [P2] and node-level performance was looked at in the contex of a KONWIHR-III[10] project. It resulted in a significant reduction of branching overhead while maintaining the flexibility of the implemented hierarchical model as described in [R1] and presented in [C2].

[10] https://konwihr.fau.de

4.3 Parallelization and scalability

For decades, the performance of processors and thus applications was improved on the hardware side by shrinking transistor sizes and increasing clock frequencies, which required the use of higher voltages. This, in turn, increased the amount of required power and waste heat, which depend quadratically on the voltage. Providing the power and dissipating the heat became more and more of a problem, which led to hitting the peak clock frequency at close to 4 GHz in the early 2000s [148]. Since then, manufacturers switched to increasing the number of cores in multicore processors to further improve the speed of their processors requiring performance-aware software developers to learn how to make use of those, i. e., how to identify and exploit *parallelism* in their programs.

Parallelization had already played an important role in scientific computing applications for decades at that point due to the complexity of the problems considered: either time-to-solution demands, i. e., obtaining results for a fixed problem size in a given amount of time, or to overcome memory limitations when the problem size of interest exceeds the available memory made it necessary to employ parallel programming techniques and execute programs on multiple processors. In the first case, the work for a fixed problem size is distributed among multiple processors, which is called *strong scaling*; the second case involves increasing the problem size at the same time as more processors are used and is refered to as *weak scaling* [75].

While parallelism is present at many levels in today's computers (see Section 4.1), here we restrict ourselves to gains and challenges when making use of multiple processing units, i. e., multiple cores within a node or multiple nodes. In general, a distinction is drawn between two different parallelism concepts: Splitting different tasks of a larger problem into heterogeneous subtasks is called *functional parallelism* [75], which is common in coupled problems such as climate models, where the different components (e. g., sea ice, land ice, ocean, atmosphere, etc.) are distributed to different processors and executed in parallel [71]. This type of parallelism is connected to the programming paradigm *MPMD* (Multiple Program, Multiple Data), as code and data are different in each parallel instance. In contrast to that, when large amounts of data are processed in parallel using the same program with each processor being responsible for different parts of that data, this is termed *data parallelism* and represents the most common scenario in the context of scientific computing. This goes hand in hand with the *SPMD* (Single Program, Multiple Data) programming paradigm, where the same program is executed on all processors [75]. Both concepts can of course be used in conjunction: functional subtasks themselves could be further split up using data parallelism. Here, we concern ourselves only with SPMD parallelization.

On *shared memory* architectures (such as multicore CPUs or Intel Xeon Phi accelerators), where multiple processors share the same address space and can access the same data, the most common technique to write parallel programs is *OpenMP* [31], a compiler extension for C/C++ and Fortran that provides directives to introduce parallel regions, e. g., to distribute the iterations of a loop among multiple threads. In *distributed memory* computers, there is no common address space, and communication between different processors must be done explicitly, e. g., using message passing. The most popular choice for that is the *Message Passing Interface* (MPI) [66], which equips programmers with a maximum amount of flexibility to exchange data, synchronize the program flow, or to use I/O mechanisms, while hiding the details of the underlying interconnect. As an alternative, *Partitioned Global Ad-*

dress Space (PGAS) languages[11] are an attempt to introduce a common global address space for distributed memory computers, which gained some momentum with recent advances of interconnect technology [92]. They provide a different parallel programming model than MPI formulated around data dependencies and one-sided communication to hide communication overhead when obeying to an asynchronous data flow formulation [142].

Quantifying the benefit from parallelizing a program and executing it on more processors requires suitable scalability metrics, for which we rely once again on the definitions given by Hager and Wellein [75]. Consider a fixed problem size W with normalized serial runtime $T_1 = 1$ and assume it can be broken down into a fraction p that can be (perfectly) parallelized and an inherently serial fraction s (e. g., due to data dependencies). Then serial and parallel runtimes are given as

$$T_1 = s + p = 1 \qquad \text{and} \qquad T_N = s + \frac{p}{N}\,,$$

where T_N is the runtime for solving the problem on N processors. When executing the program on a given number of processors $N > 1$, one of the most interesting results is the performance gain compared to the serial program run, i. e., the *speedup*

$$S_N = \frac{P_N}{P_1}\,.$$

Using the performance metric in Equation (4.1) and above assumptions about serial and parallel runtimes, we obtain

$$S_N = \frac{T_1}{T_N} = \frac{s+p}{s+\frac{p}{N}} = \frac{1}{s+\frac{1-s}{N}}\,,$$

which is known as *Amdahl's Law* [14]. Given a serial fraction s, it yields $S_\infty = 1/s$ as the upper limit for the maximum speedup for $N \to \infty$.

Another common metric is *parallel efficiency*

$$\varepsilon = \frac{P_N}{N \cdot P_1} = \frac{S_N}{N}\,,$$

as a measure for the achieved fraction of an optimal (linear) speedup.

In practice, there is a number of factors besides inherently serial portions in an algorithm, including communication overhead, load imbalance, bottlenecks (such as I/O bandwidth), etc. that can limit scalability [75], some of which (e. g., communication overhead) can become more pronounced for large processor counts. On the other hand, in certain cases, there is even superlinear speedup for simple loops, e. g., when reducing the local problem size on each processor leads to better cache usage and thus improves the node-level performance [166]. However, more common are cases where using all cores of a multicore processor leads to saturation of the memory bandwidth of the socket, and thus reducing the number of used processors (per socket) can actually improve performance [12]. Consequently, *parallel performance engineering* is an iterative process similar to node-level performance engineering that requires precise measurements and instrumentation of the performance for different processor counts to obtain a good understanding of the resource utilization and bottlenecks to eventually achieve optimal performance.

[11] For example: *Unified Parallel C* (UPC) [47], *GASPI* [65], or *Coarrays* in the Fortran 2008 standard [126].

In the context of this work, i. e., solving partial differential equations numerically, parallelism is introduced in the form of *domain decomposition*, i. e., by splitting the computational domain into parts that can be processed in parallel. When using mesh-based numerical methods, such as dG methods, one can exploit the partitioning of the domain into mesh elements and assign (connected) sets of elements to individual processes. This decomposition can be done using different strategies, of which graph-based partitioning algorithms are the most popular. For that, the computational mesh is transformed into its dual graph[12], which is then partitioned, e. g., using one of the algorithms provided in the widely-used library *METIS* [99].

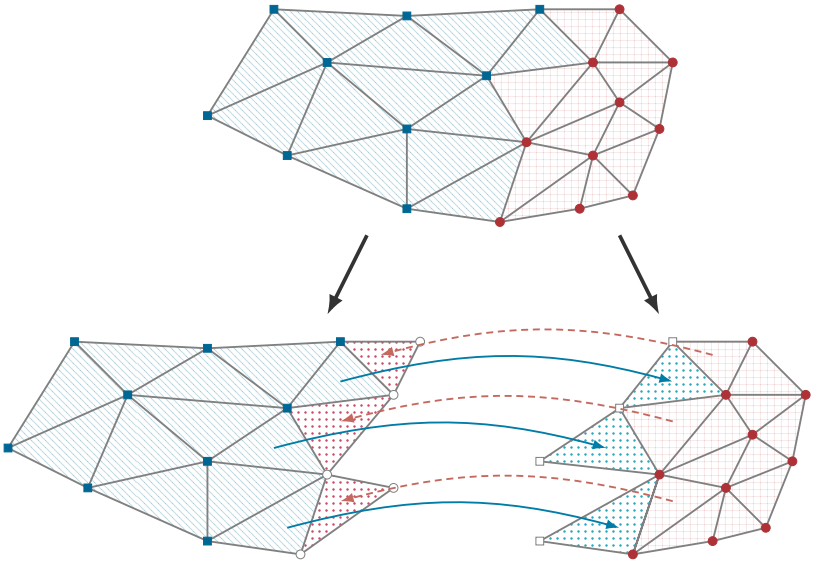

Figure 4.2: Schematic of domain decomposition for a two-dimensional mesh and two processes: Every element and node belongs to exactly one process (elements with blue lines and squared nodes belong to one process, and elements with red grid and circle nodes belong to the other), and each process computes (and thus, stores) only the data for its own mesh entities. Additionally, each process has a layer of ghost elements (dotted) and ghost nodes from the other process that are required to compute the numerical fluxes over edges. Arrows indicate the element data that has to be communicated from one process to the other.

Discontinuous Galerkin methods are particularly well-suited for parallel computing that relies on domain decomposition, as discussed in Section 3.1: For the spatial discretization, the solution on a mesh element depends only on adjacent elements with a shared edge (2D) or face (3D) via the numerical fluxes that connect both. This is implemented by introducing a layer of copies of the adjacent elements from neighboring processes around each partition – called *ghost layer* – making all required data available locally (see Figure 4.2). Consequently, each processor has to communicate only with its direct neighbors, e. g., once per time step, to exchange the values in the ghost layers. The number of neighbours per processor is typically small and even stays roughly the same when increasing the total amount of processors. For

[12] A graph in which each mesh element is represented by a vertex, and each edge (in 2D) or face (in 3D) between elements is translated to an edge in the graph connecting the two vertices that correspond to these elements.

explicit time discretizations, only the block-diagonal mass matrix (cf. Section 3.4) has to be inverted, which can be done locally on each processor. However, if implicit time integration schemes are used, it is necessary to solve a global system of equations for which HSS or hybridized methods (cf. Section 3.1) are promising techniques to reduce the performance penalties.

For the three-dimensional ocean model described in Section 3.3 and implemented in UTBEST3D (cf. Section 5.2), the computational domain is decomposed with respect to the two-dimensional rather than the three-dimensional mesh [9]. This means columns of the prismatic mesh are not allowed to be split over multiple processors permitting to solve linear systems locally when computing the vertical velocity component and the time-implicit vertical diffusion terms. This avoids the need for global communication completely, and consequently, the code displays excellent scalability properties on distributed memory computers demonstrated by Aizinger et. al. [9] and in [P4]. Only in extreme scale-out situations, where each processors is responsible only for few (less than ten) columns, it becomes challenging to find an equal distribution of two- and three-dimensional elements due to the large variation in the number of prisms per two-dimensional element in computational domains that comprise large bathymetry variations. As a consequence, this can cause load imbalance and make it impractical to increase the number of processors even further.

When using shared memory parallelization, there is no explicit decomposition of the computational domain, but iterations of loops[13] are executed in parallel on an on-demand basis as described in [P2]. This avoids load balancing issues because it lifts the column restrictions when distributing the work. However, this comes at a price: To avoid data races due to concurrent write operations (e. g., when assembling the contributions from face integrals for each element), it requires to introduce temporary vectors and an additional reduction step. Additionally, the shared memory parallelization in UTBEST3D does not take into account memory locality, and thus, scalability significantly deteriorates when using more than one NUMA domain (see Footnote 4 on page 32). In practice, the OpenMP-based shared memory parallelization of UTBEST3D is only used within one NUMA domain and combined with the MPI-based distributed memory parallelization inbetween. This delivered the best performance for UTBEST3D due to the superior scalability of the MPI parallelization [P2].

4.4 Energy efficiency

Historically, research in the field of high performance computing was almost exclusively centered around improving the computational performance of hardware and software. Only in recent years, when power became one of the driving factors towards multicore processors and in the light of the dawning exascale era, attention was also directed to power and energy considerations [88]. There are essentially two motivations for energy-efficient hardware and software:

First and foremost, energy is expensive, and lowering power drain and energy-to-solution reduces costs for HPC site operators and users running simulations, as energy expenses over the lifetime of an HPC system are in the same order of magnitude as initial acquisition costs [137]. On the other hand, there might not be enough energy available, which can be of importance when computations are part of hazard prediction systems such as

[13] For example, over elements or faces when assembling the right hand side vector for explicit time integration.

tsunami warning systems. For example, after extreme geophysical events or in developing countries, communication and power infrastructures may not be reliable enough, and thus running simulations battery-powered and on-site could provide a viable solution. Similar constraints apply even to the spearhead of high performance computing, when aiming to enhance the performance of the fastest supercomputers further, which is why the US Department of Energy defined a power drain of 20 MW as the upper limit for their own exascale systems [140].

Wilde, Auweter, and Shoukourian [161] categorized the range of measures that can be taken to address energy efficiency of HPC computations in a *4 pillar framework*:

1. Improve the building infrastructure, e. g., reduce losses or reuse waste heat;
2. Buy or develop better hardware;
3. Make better use of the available system, e. g., improve workload management or exploit power saving features of the hardware;
4. Optimize the performance of user applications.

While significant gains can be expected in either of those, hardware–software co-design approaches are necessary to properly account for mutual influence between them. But even without applying any changes to hardware or software, users can already make a contribution that falls into the third pillar above: Memory-bound applications often saturate the memory bandwidth without making use of all available cores (as discussed in Section 4.3), thus the overall runtime does not improve when employing more cores per multicore processor. However, the additional cores usually dissipate the same power as fully saturated ones, while keeping them idle is extremely energy-efficient due to the power saving features of modern CPUs. Thus, determining the optimum number of cores per node for an application and running simulations with this configuration could already result in non-negligible energy savings without any drawbacks in terms of time-to-solution as detailed in [P4].

As a tool to predict energy consumption in addition to runtime and resource utilization, energy metrics have beeen integrated into performance models (cf. Section 4.2). While such models have proven to deliver accurate results [86], applying them is a time-consuming task and mostly done only for single loops or the most compute intensive kernels in an application. For applications that exhibit a flat performance profile with a number of complex code parts that have similar runtime shares, this is often not feasible. In such cases, hardware virtualization could evolve into a promising approach for evaluating or predicting the efficiency and utilization of an entire application on existing or future hardware generations. [C1] demonstrates good agreement of runtimes between simulated and real ARM hardware for our three-dimensional ocean model UTBEST3D (see Section 5.2).

To further investigate the capabilities of low-power processors and compare them to established architectures with respect to time-to-solution and energy-to-solution for a full application code, we undertook a comparison study using UTBEST3D on ARM and Intel Haswell processors in [P4]. As a result, we found that good parallel scalability of the application may indeed yield an advantage for low-power architectures in both metrics at the same time.

Chapter 5

Implementation aspects and software packages

5.1 FESTUNG

The *Finite Element Simulation Toolbox for Unstructured Grids* (FESTUNG)[1] is a free and open source MATLAB / GNU Octave toolbox started by Balthasar Reuter, Florian Frank, and Vadym Aizinger, which is available under the conditions of the GNU GPLv3.[2] It is designed as a fast and flexible prototyping platform and provides building blocks for discontinuous Galerkin methods on unstructured grids in two space dimensions. The compact and user-friendly programming interface and comprehensive documentation[3] make the toolbox easy to use while maintaining full compatibility with GNU Octave to support users of open source software. In an attempt to deliver optimal computational performance it employs vectorized operations throughout the entire code and was the first fully-vectorized MATLAB code for time-implicit dG methods at the time of its initial release in 2015. Since then it was used in several bachelor and master theses [78, 84, 120, 130, 145], publications [P1, Pp1, P3, C3, P6, P7, P8, 4, 33, 67, 79, 80, 124, 131], and continues to be used in teaching and research at FAU and TU Dortmund.

The toolbox consists of a core set of library functions that streamline the implementation of dG discretizations for differential operators and a generic solver framework that simplifies implementing mathematical models in a structured and well-organized fashion. This framework is built around the perception that the overall algorithm used in numerical models can typically be subdivided into three major steps:

1. The *setup phase* in which parameters are defined, input files are read, or initial computations (e. g., runtime-constant data structures such as the computational mesh) are performed.
2. The *solver phase* where the majority of the work is done, e. g., assembling and solving a linear system or applying an iterative method for nonlinear problems, both potentially many times when iterating over time steps.
3. The *postprocessing phase*, e. g., to write results to output files, evaluate approximation errors, etc.

FESTUNG splits these phases further up into more fine-grained steps, of which the user implements the relevant ones as MATLAB functions and gathers them together with any required additional routines in a folder. In the following, we refer to the set of functions describing a numerical model as *problem*. A common driver routine takes care of execut-

[1] https://github.com/FESTUNG
[2] https://www.gnu.org/licenses/gpl-3.0.en.html
[3] https://www1.am.uni-erlangen.de/FESTUNG

ing the problem steps in the correct order and repeating them in the solver phase until termination is indicated via a parameter.

Next to the advantage of having a well-organized and comprehensible code structure, this allows to realize coupled solvers in an efficient and flexible way: Each of the subproblems is implemented such that they can be used individually (e. g., shallow-water and subsurface model as described in Chapter 2), and an additional problem formulation describes the coupling of both yielding well-separated, maintainable, and reusable code. This approach offers the flexibility to accommodate different coupling strategies, for example when coupling separate computational domains at a common interface (as described in Section 2.3) or when using the results of one model as input for another in the same computational domain (e. g., shallow-water solver and transport model in Section 3.7). The details of this solver structure are explained in [Pp1], summarized in [C3], and were used for advection [P3] and diffusion operators in [P1, 4, 33], Darcy flow in [124, 131], a shallow-water model in a vertical slice coupled to Darcy flow in [P8], two-dimensional shallow-water flow [78, 79] coupled to multi-component reactive transport in [P7], and Cahn–Hilliard equations in [67]. Many of those problem implementations are available in the FESTUNG repository.

The core library offers functions for defining computational domains, generating triangular or quadrilateral meshes, reading / writing parameter or visualization files (currently, VTK and Tecplot file formats are supported), checkpointing and resuming simulations, and other useful helper functions. In addition, it provides quadrature rules of various orders, arbitrary order slope limiters, different types of basis functions (Taylor, tensor product, or modal basis), projection operators between the corresponding function spaces, and routines for the integration and assembly of many different types of element and edge integrals. Such integrals appear in a similar manner for different flavors of dG methods and can often be formulated on an element-by-element (or edge-by-edge) basis as sums of products between some coefficients (e. g., the degrees of freedom of a function represented in the dG basis) and corresponding integrals of products of two or more basis functions.

In FESTUNG, such contributions can be assembled efficiently into a system matrix or right hand side vector using a tensor of integral values computed once on a reference element and some vectorized operations (such as the Kronecker product `kron` or our own, more general variant `kronVec` introduced in [P3]). These assembly routines are organized by the type of integral (e. g., element or edge) and the form of the integrand (e. g., two basis functions and a discrete function) allowing to compute the global contributions of an integral using only two instructions. Variants of such functions allow for more complicated or even nonlinear expressions, where numerical quadrature and vectorized assembly are combined in an efficient manner. The details of the required algebraic transformations and efficient implementation are presented in [P1, Pp1, P3, P6].

Currently, the range of supported variants of dG methods includes LDG, IP-dG, and HDG (cf. Section 3.1) complemented by total variation diminishing (TVD) time discretization schemes of orders 1 to 3 from the class of strong stability preserving (SSP) Runge–Kutta methods [73] and diagonally implicit Runge–Kutta (DIRK) methods [13, 77] of orders 1 to 4.

5.2 UTBEST3D

The coastal and regional ocean model UTBEST3D (University of Texas Bays and Estuaries Simulator – 3D) was started by Vadym Aizinger in his PhD-project [1] and since then steadily enhanced with additional features, verified against analytical testcases and experimental data, and used in a number of publications [C1, P2, P4, 6, 7, 9].

It employs the LDG discretization described in Section 3.3 for the system of three-dimensional baroclinic shallow-water equations with a hierarchical implementation of the model to allow for different levels of physical accuracy. This allows to cover the range from barotropic single-layer quasi-2D setups to the full baroclinic three-dimensional model with higher-order turbulence closure schemes (using algebraic, 1st, or 2nd order vertical eddy viscosity parameterizations). Spatial approximation orders can be chosen individually for each unknown based on polynomial approximations of orders 0 to 2. Time integration is done via a semi-implicit algorithm based on strong stability preserving Runge-Kutta methods [73] as described in Section 3.3. Numerical fluxes are approximated using full upwinding, Lax-Friedrichs [112], Roe's [129], HLL [81], HLLC [151], or FORCE [150] Riemann solvers, and vertex-based slope limiters are available to enforce local maximum principles in the solution (cf. Section 3.6). The numerical model is capable of simulating wetting/drying scenarios, supports non-conforming meshes to allow for varying vertical resolution in different parts of the domain, and provides local mesh and space refinement (hp-adaptivity). The full solution algorithm is presented in [P2] and [P4].

UTBEST3D is written in C++ and parallelized using MPI and OpenMP (see Section 4.3) with domain decomposition based on METIS [99]. The model proved good scalability and portability to a number of different hardware platforms [C1, P2, P4, 9], and its runtime distribution was analyzed in detail in [P2]. Node-level performance was improved upon in a KONWIHR-III project [R1] resulting in reasonable resource utilization and little overhead. However, it continues to suffer from a lack of vectorization in floating point operations. The reason for the latter is the need for frequent evaluation of short scalar products when computing values of unknowns in quadrature points during numerical integration of edge or element terms in system (3.5). These scalar products contribute a major share of the overall instruction count and consist of vectors of local degrees of freedom and values of the basis functions in the quadrature point with typical vector lengths of 1 (constant), 4 (linear) or 10 (quadratic) entries for three-dimensional unknowns. Due to the unstructured triangular mesh and varying number of prisms in a vertical column, these vectors are stored neither aligned nor linear in memory making it impossible for the compiler to auto-vectorize the corresponding operations. Existing approaches to overcome this disadvantage are based on structured meshes [24, 60], thus making the memory location for the degrees of freedom predictable. Solutions for unstructured dG methods are still to be developed.

Chapter

6 Summary and outlook

The present work covers mathematical models for the numerical simulation of free surface and subsurface flow systems and proposes a way of coupling those. Free surface flows are modeled using the three-dimensional shallow-water equations with a free surface and vertical eddy viscosity parameterizations. For subsurface flows, saturated flow through an aquifer is considered and modeled by the equation for groundwater flow based on Darcy's law. Coupling conditions are formulated for the barotropic free surface flow model in a mass and pressure conservative form on a sharp interface separating the two domains. Both models are discretized using the local discontinuous Galerkin (LDG) method, and discrete stability of the coupled model is proved.

Slope limiters are presented as effective tools to enforce the local maximum principle for discontinuous Galerkin (dG) solutions, and this work includes contributions related to a stricter limiter for higher-order approximations and an anisotropic limiter. In addition, a scheme is proposed allowing to utilize the information about the solution obtained in the course of the slope limiting procedure to yield a computationally efficient transport scheme.

The LDG method demonstrates excellent scalability on a wide range of computing architectures due to the absence of global coupling in the degrees of freedom. This scalability can compensate up to a certain degree for the additional computational work incurred by the large number of degrees of freedom peculiar to LDG. Furthermore, it enables the use of larger numbers of comparably slow low-power CPUs to increase energy efficiency without sacrificing time-to-solution.

In the course of this work, the MATLAB / GNU Octave framework FESTUNG has been developed and applied to a number of differential operators. It is a toolbox centered around discontinuous Galerkin methods and well-suited for use in rapid prototyping and teaching. With comprehensive documentation and consistent application of vectorized instructions, this framework is easy to use and, at the same time, fast enough for small to medium scale problems.

The regional ocean model UTBEST3D employs the LDG method for the system of three-dimensional baroclinic shallow-water equations with higher-order turbulence closure schemes. It is written in C++, parallelized using MPI and OpenMP, and was tested on a wide range of computing architectures. The shared-memory parallelization, improved node-level performance, and better parallel scalability have been contributed in the course of this work. Further additions to UTBEST3D by the author include adaptive vertical discretization techniques in the context of a DFG project[1] that modify the structure of the mesh in vertical direction according to the current flow and density fields to capture stratified or overflow situations and reduce non-physical mixing.

[1] German Research Foundation (DFG) Grant AI 117/1

Following the topics covered in this work, a number of open questions remain that offer prospects for future investigations:

- The stability analysis of the discrete shallow-water model as well as of the coupled system could be extended to the full baroclinic system. For that, it is necessary to include the additional pressure term in the interface conditions and to accommodate the baroclinic pressure correction term along with the additional transport equations in the analysis framework.
- While numerical experiments suggest convergence for the system of three-dimensional shallow-water equations, the author is not aware of any *a priori* error analysis for the shallow-water model proving its convergence and giving its rate.
- The highly anisotropic setting of shallow-water flow scenarios offers opportunities to develop anisotropic slope limiters tailored to these anisotropies, e. g., by applying different limiting strategies for horizontal and vertical velocity components. In addition to that, vectorial limiters that consider velocity vectors as a whole instead of applying the limiting procedure component-wise might also prove useful in this context.
- The large number of degrees of freedom in LDG methods results in a compute-intensive algorithm that is characterized by the evaluation of many short scalar products in each quadrature point of mesh entities. Exploiting the vectorization capabilities of modern CPUs for these types of operations on unstructured meshes is not a trivial task, thus resource utilization on current computing architectures leaves room for improvement. Quadrature-free schemes [17] might be one way of tackling this issue.
- Using accelerators in an efficient way could improve both energy efficiency and time-to-solution for the numerical models. Developing an approach to improve vectorization on current CPUs would also help in using accelerators efficiently.
- With LDG proving its suitability for the simulation of three-dimensional shallow-water systems on low-power architectures, a cluster study that verifies the efficiency of the method on larger systems remains to be done.

Bibliography

[1] V. Aizinger. "A discontinuous Galerkin Method for Two- and Three-Dimensional Shallow-Water Equations". Ph. D. thesis. The University of Texas at Austin, 2004. URL: http://hdl.handle.net/2152/1863.

[2] V. Aizinger. "A Geometry Independent Slope Limiter for the Discontinuous Galerkin Method. The 4th Russian-German Advanced Research Workshop, Freiburg, Germany, October 12 to 16, 2009". In: ed. by E. Krause, Y. Shokin, M. Resch, D. Kröner, and N. Shokina. Vol. 115. Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Springer, 2011, pp. 207–217. DOI: 10.1007/978-3-642-17770-5.

[3] V. Aizinger. "Unstructured finite element models for baroclinic ocean flows". Habilitation thesis. Friedrich-Alexander-University Erlangen-Nürnberg, 2019.

[4] V. Aizinger, L. Bungert, and M. Fried. "Comparison of two local discontinuous Galerkin formulations for the subjective surfaces problem". In: *Computing and Visualization in Science* 18.6 (2018), pp. 193–202. DOI: 10.1007/s00791-018-0291-4.

[5] V. Aizinger and C. Dawson. "A discontinuous Galerkin method for two-dimensional flow and transport in shallow water". In: *Advances in Water Resources* 25.1 (2002), pp. 67–84. DOI: 10.1016/S0309-1708(01)00019-7.

[6] V. Aizinger and C. Dawson. "A discontinuous Galerkin method for three-dimensional shallow water flows with free surface". In: *Developments in Water Science* 55.2 (2004), pp. 1691–1702. DOI: 10.1016/S0167-5648(04)80177-1.

[7] V. Aizinger and C. Dawson. "The local discontinuous Galerkin method for three-dimensional shallow water flow". In: *Computer Methods in Applied Mechanics and Engineering* 196.4–6 (2007), pp. 734–746. DOI: 10.1016/j.cma.2006.04.010.

[8] V. Aizinger, D. Kuzmin, and L. Korous. "Scale separation in fast hierarchical solvers for discontinuous Galerkin methods". In: *Applied Mathematics and Computation* 266 (2015), pp. 838–849. DOI: 10.1016/j.amc.2015.05.047.

[9] V. Aizinger, J. Proft, C. Dawson, D. Pothina, and S. Negusse. "A three-dimensional discontinuous Galerkin model applied to the baroclinic simulation of Corpus Christi Bay". In: *Ocean Dynamics* 63.1 (2013), pp. 89–113. DOI: 10.1007/s10236-012-0579-8.

[10] V. Aizinger, A. Rupp, J. Schütz, and P. Knabner. "Analysis of a mixed discontinuous Galerkin method for instationary Darcy flow". In: *Computational Geosciences* 22.1 (2018), pp. 179–194. DOI: 10.1007/s10596-017-9682-8.

[11] V. Aizinger and J. Schütz. "A hierarchical scale separation approach for the hybridized discontinuous Galerkin method". In: *Journal of Computational and Applied Mathematics* 317 (2017), pp. 500–509. DOI: 10.1016/j.cam.2016.12.018.

[12] L. Akyl et al. *Intel Guide for Developing Multithreaded Applications*. Tech. rep. Intel Corp., Jan. 2012. URL: https://www.intel.com/software/threading-guide.

[13] R. Alexander. "Diagonally Implicit Runge–Kutta Methods for Stiff O.D.E.'s". In: *SIAM Journal on Numerical Analysis* 14.6 (1977). DOI: 10.1137/0714068.

[14] G. M. Amdahl. "Validity of the single processor approach to achieving large scale computing capabilities". In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. 1967, pp. 483–485. DOI: 10.1145/1465482.1465560.

[15] D. N. Arnold. "An Interior Penalty Finite Element Method with Discontinuous Elements". In: *SIAM Journal on Numerical Analysis* 19.4 (1982), pp. 742–760. DOI: 10.1137/0719052.

[16] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. "Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 39.5 (2002), pp. 1749–1779. DOI: 10.1137/S0036142901384162.

[17] H. L. Atkins and C.-W. Shu. "Quadrature-Free Implementation of Discontinuous Galerkin Method for Hyperbolic Equations". In: *AIAA Journal* 36.5 (1998), pp. 775–782. DOI: 10.2514/2.436.

[18] I. Babuška. "The finite element method with Lagrangian multipliers". In: *Numerische Mathematik* 20.3 (1973), pp. 179–192. DOI: 10.1007/BF01436561.

[19] I. Babuška and M. Zlámal. "Nonconforming Elements in the Finite Element Method with Penalty". In: *SIAM Journal on Numerical Analysis* 10.5 (1973), pp. 863–875. DOI: 10.1137/0710071.

[20] L. Bao, R. D. Nair, and H. M. Tufo. "A mass and momentum flux-form high-order discontinuous Galerkin shallow water model on the cubed-sphere". In: *Journal of Computational Physics* 271 (2014), pp. 224–243. DOI: 10.1016/j.jcp.2013.11.033.

[21] T. Barth and D. Jespersen. "The design and application of upwind schemes on unstructured meshes". In: *AIAA 27th Aerospace Sciences Meeting, Reno*. 1989. DOI: 10.2514/6.1989-366.

[22] F. Bassi, A. Ghidoni, S. Rebay, and P. Tesini. "High-order accurate p-multigrid discontinuous Galerkin solution of the Euler equations". In: *International Journal for Numerical Methods in Fluids* 60.8 (2009), pp. 847–865. DOI: 10.1002/fld.1917.

[23] F. Bassi and S. Rebay. "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations". In: *Journal of Computational Physics* 131.2 (1997), pp. 267–279. DOI: 10.1006/jcph.1996.5572.

[24] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander. "A generic grid interface for parallel and adaptive scientific computing. Part II: implementation and tests in DUNE". In: *Computing* 82.2–3 (2008), pp. 121–138. DOI: 10.1007/s00607-008-0004-9.

[25] G. Beavers and D. Joseph. "Boundary conditions at a naturally permeable wall". In: *Journal of Fluid Mechanics* 30.1 (1967), pp. 197–207. DOI: 10.1017/S0022112067001375.

[26] N. Beisiegel and J. Behrens. “Quasi-nodal third-order Bernstein polynomials in a discontinuous Galerkin model for flooding and drying”. In: *Environmental Earth Sciences* 74.11 (2015), pp. 7275–7284. DOI: 10.1007/s12665-015-4745-4.

[27] P.-E. Bernard, N. Chevaugeon, V. Legat, E. Deleersnijder, and J.-F. Remacle. “High-order h-adaptive discontinuous Galerkin methods for ocean modelling”. In: 57.2 (2007), pp. 109–121. DOI: 10.1007/s10236-006-0093-y.

[28] C. A. Blain and T. C. Massey. “Application of a coupled discontinuous–continuous Galerkin finite element shallow water model to coastal ocean dynamics”. In: *Ocean Modelling* 10.3–4 (2005), pp. 283–315. DOI: 10.1016/j.ocemod.2004.09.002.

[29] S. Blaise, R. Comblen, V. Legat, J.-F. Remacle, E. Deleersnijder, and J. Lambrechts. “A discontinuous finite element baroclinic marine model on unstructured prismatic meshes. Part I: space discretization”. In: *Ocean Dynamics* 60.6 (2010), pp. 1371–1393. DOI: 10.1007/s10236-010-0358-3.

[30] S. Blaise, J. Lambrechts, and E. Deleersnijder. “A stabilization for three-dimensional discontinuous Galerkin discretizations applied to nonhydrostatic atmospheric simulations”. In: *International Journal for Numerical Methods in Fluids* 9 (2016), pp. 558–585. DOI: 10.1002/fld.4197.

[31] OpenMP Architecture Review Board. *OpenMP Application Programming Interface*. Tech. rep. Version 5.0. Nov. 2018. URL: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf.

[32] F. Brunner and P. Knabner. “A global implicit solver for miscible reactive multiphase multicomponent flow in porous media”. In: *Computational Geosciences* 23.1 (2019), pp. 127–148. DOI: 10.1007/s10596-018-9788-7.

[33] L. Bungert, V. Aizinger, and M. Fried. “A Discontinuous Galerkin Method for the Subjective Surfaces Problem”. In: *Journal of Mathematical Imaging and Vision* 58.1 (2017), pp. 147–161. DOI: 10.1007/s10851-016-0695-z.

[34] G. Chavent and B. Cockburn. “The local projection P^0-P^1-discontinuous-Galerkin finite element method for scalar conservation laws”. In: *Mathematical Modeling and Numerical Analysis* 23.4 (1989), pp. 565–592. DOI: 10.1051/m2an/1989230405651.

[35] S.-J. Choi and F. X. Giraldo. “Geoscientific Model Development Discussions”. In: 7 (2014), pp. 4119–4151. DOI: 10.5194/gmdd-7-4119-2014.

[36] B. Cockburn, D. A. Di Pietro, and A. Ern. “Bridging the hybrid high-order and hybridizable discontinuous Galerkin methods”. In: *ESAIM: M2AN* 50.3 (2016), pp. 635–650. DOI: 10.1051/m2an/2015051.

[37] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. “Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems”. In: *SIAM Journal on Numerical Analysis* 47.2 (2009), pp. 1319–1365. DOI: 10.1137/070706616.

[38] B. Cockburn, S. Hou, and C.-W. Shu. “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. The multidimensional case”. In: *Mathematics of Computation* 54.190 (1990), pp. 545–581. DOI: 10.1090/S0025-5718-1990-1010597-0.

[39] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds. *Discontinuous Galerkin Methods*. Vol. 11. Lecture Notes in Computational Science and Engineering. Springer, 2000. DOI: 10.1007/978-3-642-59721-3.

[40] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. "The Development of Discontinuous Galerkin Methods". In: ed. by B. Cockburn, G. E. Karniadakis, and C.-W. Shu. Vol. 11. Lecture Notes in Computational Science and Engineering. Springer, 2000, pp. 3–50. DOI: 10.1007/978-3-642-59721-3.

[41] B. Cockburn, S. Y. Lin, and C.-W. Shu. "TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. III. One-dimensional systems". In: *Journal of Computational Physics* 84.1 (1989), pp. 90–113. DOI: 10.1016/0021-9991(89)90183-6.

[42] B. Cockburn and C.-W. Shu. "TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework". In: *Mathematics of Computation* 52.186 (1989), pp. 411–435. DOI: 10.1090/S0025-5718-1989-0983311-4.

[43] B. Cockburn and C.-W. Shu. "The Runge–Kutta local projection P^1-discontinuous-Galerkin finite element method for scalar conservation laws". In: *Mathematical Modeling and Numerical Analysis* 25.3 (1991), pp. 337–361. DOI: 10.1051/m2an/1991250303371.

[44] B. Cockburn and C.-W. Shu. "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems". In: *SIAM Journal on Numerical Analysis* 35.6 (1998), pp. 2440–2463. DOI: 10.1137/S0036142997316712.

[45] B. Cockburn and C.-W. Shu. "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. V. Multidimensional systems". In: *Journal of Computational Physics* 141.2 (1998), pp. 199–224. DOI: 10.1006/jcph.1998.5892.

[46] C. J. Conroy and E. J. Kubatko. "*hp* discontinuous Galerkin methods for the vertical extent of the water column in coastal settings part I: Barotropic forcing". In: *Journal of Computational Physics* 305 (2016), pp. 1147–1171. DOI: 10.1016/j.jcp.2015.10.038.

[47] UPC Consortium. *UPC Language Specifications*. Tech. rep. Version 1.3. Nov. 2013. URL: https://upc-lang.org/assets/Uploads/spec/upc-lang-spec-1.3.pdf.

[48] S. Danilov. "Ocean modeling on unstructured meshes". In: *Ocean Modeling* 69 (2013), pp. 195–210. DOI: j.ocemod.2013.05.005.

[49] C. Dawson. "A continuous/discontinuous Galerkin framework for modeling coupled subsurface and surface water flow". In: *Computational Geosciences* 12.4 (2008), pp. 451–472. DOI: 10.1007/s10596-008-9085-y.

[50] C. Dawson. "A Local Timestepping Runge–Kutta Discontinuous Galerkin Method for Hurricane Storm Surge Modeling. 2012 John H Barrett Memorial Lectures". In: ed. by X. Feng, O. Karakashian, and Y. Xing. Vol. 157. The IMA Volumes in Mathematics and its Applications. 2013, pp. 133–148. DOI: 10.1007/978-3-319-01818-8.

[51] C. Dawson and V. Aizinger. “A Discontinuous Galerkin Method for Three-Dimensional Shallow Water Equations”. In: *Journal of Scientific Computing* 22.1–3 (2005), pp. 245–267. DOI: 10.1007/s10915-004-4139-3.

[52] C. Dawson and J. Proft. “Discontinuous and coupled continuous/discontinuous Galerkin methods for the shallow water equations”. In: *Computer Methods in Applied Mechanics and Engineering* 91.41–42 (2002), pp. 4721–4746. DOI: 10.1016/S0045-7825(02)00402-4.

[53] T. De Maet, F. Cornaton, and E. Hanert. “A scalable coupled surface–subsurface flow model”. In: *Computers & Fluids* 116 (2015), pp. 74–87. DOI: 10.1016/j.compfluid.2015.03.028.

[54] D. A. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Vol. 69. Mathématiques et Applications. Springer, 2012. DOI: 10.1007/978-3-642-22980-0.

[55] D. A. Di Pietro and A. Ern. “A hybrid high-order locking-free method for linear elasticity on general meshes”. In: *Computer Methods in Applied Mechanics and Engineering* 283 (2015), pp. 1–21. DOI: 10.1016/j.cma.2014.09.009.

[56] D. A. Di Pietro, A. Ern, and S. Lemaire. “An Arbitrary-Order and Compact-Stencil Discretization of Diffusion on General Meshes Based on Local Reconstruction Operators”. In: *Computational Methods in Applied Mathematics* 14.4 (2014), pp. 461–472. DOI: 10.1515/cmam-2014-0018.

[57] D. A. Di Pietro and M. Vohralik. “A Review of Recent Advances in Discretization Methods, a Posteriori Error Analysis, and Adaptive Algorithms for Numerical Modeling in Geosciences”. In: *Oil & Gas Science and Technology – Rev. IFP Energies nouvelles* 69.4 (2014), pp. 701–729. DOI: 10.2516/ogst/2013158.

[58] P. A. Domenico and F. W. Schwartz. *Physical and chemical hydrogeology*. New York: Wiley, 1990. ISBN: 0-471-50744-X.

[59] M. Dumbser and M. Facchini. “A space-time discontinuous Galerkin method for Boussinesq-type equations”. In: *Applied Mathematics and Computation* 272.2 (2016), pp. 336–346. DOI: 10.1016/j.amc.2015.06.052.

[60] M. Dumbser, F. Fambri, M. Tavelli, M. Bader, and T. Weinzierl. “Efficient Implementation of ADER Discontinuous Galerkin Schemes for a Scalable Hyperbolic PDE Engine”. In: *Axioms* 7.3 (2018). DOI: 10.3390/axioms7030063.

[61] A. Duran and F. Marche. “Recent advances on the discontinuous Galerkin method for shallow water equations with topography source terms”. In: *Computers & Fluids* 101 (2014), pp. 88–104. DOI: 10.1016/j.compfluid.2014.05.031.

[62] J. Fang, H. Sips, L. Zhang, C. Xu, Y. Che, and A. L. Varbanescu. “Test-driving Intel Xeon Phi”. In: *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*. 2014, pp. 137–148. DOI: 10.1145/2568088.2576799.

[63] T. Fetzer, K. M. Smits, and R. Helmig. “Effect of Turbulence and Roughness on Coupled Porous-Medium/Free-Flow Exchange Processes”. In: *Transport in Porous Media* 114.2 (2016), pp. 395–424. DOI: 10.1007/s11242-016-0654-6.

[64] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal. "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations". In: *Journal of Computational Physics* 207.1 (2005), pp. 92–113. DOI: 10.1016/j.jcp.2005.01.005.

[65] GASPI Forum. *GASPI: Global Address Space Programming Interface. Specification of a PGAS API for communication*. Tech. rep. Version 17.1. Feb. 2017. URL: https://raw.githubusercontent.com/GASPI-Forum/GASPI-Forum.github.io/master/standards/GASPI-17.1.pdf.

[66] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*. Tech. rep. Version 3.1. University of Knoxville, Tennessee, June 2015. URL: https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.

[67] F. Frank, A. Rupp, and D. Kuzmin. *Bound-preserving flux limiting schemes for DG discretizations of conservation laws with applications to the Cahn-Hilliard equation*. Preprint. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2019. DOI: 10.13140/RG.2.2.13496.60160.

[68] B. Gaster, L. Howes, D. Kaeli, P. Mistry, and D. Schaa. *Heterogeneous Computing with OpenCL*. 1st ed. Morgan Kaufmann, 2011. DOI: 10.1016/C2011-0-69669-3.

[69] A. E. Gill. *Atmosphere-Ocean Dynamics*. Vol. 30. International Geophysics. Academic Press, 1982. DOI: 10.1016/s0074-6142(08)x6002-4.

[70] F. X. Giraldo, J. S. Hesthaven, and T. Warburton. "Nodal High-Order Discontinuous Galerkin Methods for the Spherical Shallow Water Equations". In: *Journal of Computational Physics* 181.2 (2002), pp. 499–525. DOI: 10.1006/jcph.2002.7139.

[71] J.-C. Golaz et al. "The DOE E3SM coupled model version 1: Overview and evaluation at standard resolution". In: *Journal of Advances in Modeling Earth Systems* 11 (2019). DOI: 10.1029/2018MS001603.

[72] S. Gottlieb and C.-W. Shu. "Total variation diminishing Runge-Kutta schemes". In: *Mathematics of Computation* 67 (1998), pp. 73–85. DOI: 10.1090/S0025-5718-98-00913-2.

[73] S. Gottlieb, C.-W. Shu, and E. Tadmore. "Strong Stability-Preserving High-Order Time Discretization Methods". In: *SIAM Review* 43.1 (2001), pp. 89–112. DOI: 10.1137/S003614450036757X.

[74] S. Griffies and Co-Authors. "Problems and Prospects in Large-Scale Ocean Circulation Models". In: *Proceedings of OceanObs'09: Sustained Ocean Observations and Information for Society*. Vol. 2. 2010. DOI: 10.5270/OceanObs09.cwp.38.

[75] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. Computational Science Series. CRC Press, 2011. DOI: 10.1201/EBK1439811924.

[76] G. Hager and G. Wellein. "Performance Engineering". In: *Informatik-Spektrum* 41.5 (2018), pp. 323–327. DOI: 10.1007/s00287-018-1122-1.

[77] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. Vol. 14. Springer Series in Computational Mathematics. Springer, 1996. DOI: 10.1007/978-3-642-05221-7.

[78] H. Hajduk. “A numerical investigation of direct bathymetry reconstruction based on a modified shallow-water model”. M. Sc. thesis. Friedrich-Alexander-University Erlangen-Nürnberg, 2017.

[79] H. Hajduk, D. Kuzmin, and V. Aizinger. *Bathymetry reconstruction using inverse shallow water models: Finite element discretization and regularization*. Tech. rep. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 585. Fakultät für Mathematik, TU Dortmund, 2018.

[80] H. Hajduk, D. Kuzmin, and V. Aizinger. “New directional vector limiters for discontinuous Galerkin methods”. In: *Journal of Computational Physics* 384 (2019), pp. 308–325. DOI: 10.1016/j.jcp.2019.01.032.

[81] A. Harten, P. D. Lax, and B. van Leer. “On upstream differencing and Godunov-type schemes for hyperbolic conservation laws”. In: *SIAM Review* 25.1 (1983), pp. 35–61. DOI: 10.1137/1025002.

[82] A. Heinecke, M. Klemm, and H.-J. Bungartz. “From GPGPU to Many-Core: Nvidia Fermi and Intel Many Integrated Core Architecture”. In: *Computing in Science & Engineering* 14.78 (2012), pp. 78–83. DOI: 10.1109/MCSE.2012.23.

[83] R. Helmig and H. Class. *Fluidmechanik II*. Lecture notes. Department of Hydromechanics and Modelling of Hydrosystems, University Stuttgart. 2014. URL: https://ilias3.uni-stuttgart.de/ilias.php?baseClass=ilrepositorygui&reloadpublic=1&cmd=frameset&ref_id=1.

[84] J. Hodai. “Problem of Open Boundaries in the Two-dimensional Shallow Water Equations”. B. Sc. thesis. Friedrich-Alexander-University Erlangen-Nürnberg, 2016.

[85] B. R. Hodges. “A new approach to the local time stepping problem for scalar transport”. In: *Ocean Modelling* 77 (2014), pp. 1–19. DOI: 10.1016/j.ocemod.2014.02.007.

[86] J. Hofmann, G. Hager, and D. Fey. “On the Accuracy and Usefulness of Analytic Energy Models for Contemporary Multicore Processors”. In: *ISC High Performance 2018: High Performance Computing*. 2018, pp. 22–43. DOI: 10.1007/978-3-319-92040-5_2.

[87] M. E. Hubbard. “Multidimensional Slope Limiters for MUSCL-Type Finite Volume Schemes on Unstructured Grids”. In: *Journal of Computational Physics* 155.1 (1999), pp. 54–74. DOI: 10.1006/jcph.1999.6329.

[88] J. H. Laros III, K. Pedretti, S. M. Kelly, W. Shu, K. Ferreira, J. Van Dyke, and C. Vaughan. *Energy-Efficient High Performance Computing*. Springer Briefs in Computer Science. Springer, 2012. DOI: 10.1007/978-1-4471-4492-2.

[89] IOC, SCOR, and IAPSO. *The international thermodynamic equation of seawater – 2010: Calculation and use of thermodynamic properties*. Intergovernmental Oceanographic Commission, Manuals and Guides 56. UNESCO, 2010. URL: http://www.teos-10.org/pubs/TEOS-10_Manual.pdf.

[90] N. Izem, M. Seaid, and M. Wakrim. “A discontinuous Galerkin method for two-layer shallow water equations”. In: *Mathematics and Computers in Simulation* 120 (2015), pp. 12–23. DOI: 10.1016/j.matcom.2015.04.009.

[91] H. Rui J. Zang. "A stabilized Crouzeix-Raviart element method for coupling Stokes and Darcy-Forchheimer flows". In: *Numerical Methods for Partial Differential Equations* 33.4 (2017), pp. 1070–1094. DOI: 10.1002/num.22129.

[92] N. Jansson. "High Performance Adaptive Finite Element Methods: With Applications in Aerodynamics". Ph. D. thesis. Stockholm: KTH Royal Institute of Technology, 2013. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-125742.

[93] A. Jaust, J. Schütz, and V. Aizinger. "An efficient linear solver for the hybridized discontinuous Galerkin method". In: *Proceedings in Applied Mathematics and Mechanics* 16.1 (2016), pp. 845–846. DOI: 10.1002/pamm.201610411.

[94] A. Jeschke, S. Vater, and J. Behrens. "A Discontinuous Galerkin Method for Non-hydrostatic Shallow Water Flows". In: *Finite Volumes for Complex Applications VIII – Hyperbolic, Elliptic and Parabolic Problems. FVCA 2017*. Ed. by C. Cancès and P. Omnes. Vol. 200. Springer Proceedings in Mathematics & Statistics. 2017, pp. 247–255. DOI: 10.1007/978-3-319-57394-6_27.

[95] L. H. Kantha and C. A. Clayson. *Numerical Models of Oceans and Oceanic Processes*. Vol. 66. International Geophysics. Academic Press, 2000. DOI: 10.1016/s0074-6142(00)x8001-1.

[96] T. Kärnä, B. Brye, O. Gourgue, J. Lambrechts, R. Comblen, V. Legat, and E. Deleersnijder. "A fully implicit wetting–drying method for DG-FEM shallow water models, with an application to the Scheldt Estuary". In: *computer Methods in Applied Mechanics and Engineering* 200.5–8 (2011), pp. 509–524. DOI: 10.1016/j.cma.2010.07.001.

[97] T. Kärnä, S. C. Kramer, L. Mitchell, D. A. Ham, M. D. Piggot, and A. M. Baptista. "Thetis coastal ocean model: discontinuous Galerkin discretization for the three-dimensional hydrostatic equations". In: *Geoscientific Model Development* 11 (2018), pp. 4359–4382. DOI: 10.5194/gmd-11-4359-2018.

[98] T. Kärnä, V. Legat, and E. Deleersnijder. "A baroclinic discontinuous Galerkin finite element model for coastal flows". In: *Ocean Modelling* 61 (2013), pp. 1–20. DOI: 10.1016/j.ocemod.2012.09.009.

[99] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM Journal on Scientific Computing* 20.1 (1999). DOI: 10.1137/S1064827595287997.

[100] G. Kesserwani and Q. Liang. "Dynamically adaptive grid based discontinuous Galerkin shallow water model". In: *Advances in Water Resources* 37 (2012), pp. 23–39. DOI: 10.1016/j.advwatres.2011.11.006.

[101] A. A. Khan and W. Lai. *Modeling Shallow Water Flows Using the Discontinuous Galerkin Method*. CRC Press, 2014. DOI: 10.1201/b16579.

[102] K. Klingbeil, F. Lemarié, L. Debreu, and H. Burchard. "The numerics of hydrostatic structured-grid coastal ocean models: State of the art and future perspectives". In: *Ocean Modeling* 125 (2018), pp. 80–105. DOI: 10.1016/j.ocemod.2018.01.007.

[103] B. A. Klinger. *Density of Seawater*. Accessed on Jan 4, 2019. 2002. URL: https://mason.gmu.edu/~bklinger/seawater.pdf.

[104] L. Krivodonova. "Limiters for high-order discontinuous Galerkin methods". In: *Journal of Computational Physics* 226.1 (2007), pp. 879–896. DOI: 10.1016/j.jcp.2007.05.011.

[105] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, and J. E. Flaherty. "Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws". In: *Applied Numerical Mathematics* 48.3–4 (2004), pp. 323–338. DOI: 10.1016/j.apnum.2003.11.002.

[106] E. J. Kubatko, Westerink, and C. Dawson. "*hp* Discontinuous Galerkin methods for advection dominated problems in shallow water flow". In: *Computer Methods in Applied Mechanics and Engineering* 196.1–3 (2006), pp. 437–451. DOI: 10.1016/j.cma.2006.05.002.

[107] D. Kuzmin. "A vertex-based hierarchical slope limiter for *p*-adaptive discontinuous Galerkin methods". In: *Journal of Computational and Applied Mathematics* 233.12 (2010), pp. 3077–3085. DOI: 10.1016/j.cam.2009.05.028.

[108] D. Kuzmin. "Algebraic Flux Correction I". In: *Flux-Corrected Transport: Principles, Algorithms, and Applications*. Ed. by D. Kuzmin, R. Löhner, and S. Turek. 2012, pp. 145–192. DOI: 10.1007/978-94-007-4038-9_6.

[109] D. Kuzmin. "Slope limiting for discontinuous Galerkin approximations with a possibly non-orthogonal Taylor basis". In: *International Journal for Numerical Methods in Fluids* 71.9 (2013), pp. 1178–1190. DOI: 10.1002/fld.3707.

[110] D. Kuzmin. "Hierarchical slope limiting in explicit and implicit discontinuous Galerkin methods". In: *Journal of Computational Physics* 257.B (2014), pp. 1140–1162. DOI: 10.1016/j.jcp.2013.04.032.

[111] W. Lai and A. A. Khan. "A discontinuous Galerkin method for two-dimensional shallow water flows". In: *Numerical Methods in Fluids* 70.8 (2012), pp. 939–960. DOI: 10.1002/fld.2721.

[112] P. Lax. "The Formation and Decay of Shock Waves". In: *The American Mathematical Monthly* 79.3 (1972), pp. 227–241. DOI: 10.1080/00029890.1972.11993023.

[113] M. N. Levy, R. D. Nair, and H. M. Tufo. "High-order Galerkin methods for scalable global atmospheric models". In: *Computational Geosciences* 33.8 (2007), pp. 1022–1035. DOI: 10.1016/j.cageo.2006.12.004.

[114] D. Liang, R. A. Falconer, and B. Lin. "Coupling surface and subsurface flows in a depth averaged flood wave model". In: *Journal of Hydrology* 337 (2007), pp. 147–158. DOI: 10.1016/j.jhydrol.2007.01.045.

[115] FUJITSU Ltd. *Post-K Supercomputer Overview*. Accessed on Apr 30, 2019. Nov. 2016. URL: https://www.fujitsu.com/global/Images/post-k-supercomputer-overview.pdf.

[116] H. Luo, J. D. Baum, and R. Löhner. "A *p*-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids". In: *Journal of Computational Physics* 211.2 (2006), pp. 767–783. DOI: 10.1016/j.jcp.2005.06.019.

[117] J. Magiera, C. Rohde, and I. Rybak. "A Hyperbolic–Elliptic Model Problem for Coupled Surface–Subsurface Flow". In: *Transport in Porous Media* 114.2 (2016), pp. 425–455. DOI: 10.1007/s11242-015-0548-z.

[118] S. May and M. Berger. "Two-Dimensional Slope Limiters for Finite Volume Schemes on Non-Coordinate-Aligned Meshes". In: *SIAM Journal on Scientific Computing* 35.5 (2013), A2163–A2187. DOI: 10.1137/120875624.

[119] C. Michoski, C. Mirabito, C. Dawson, D. Wirasaet, E. J. Kubatko, and J. J. Westerink. "Adaptive hierarchic transformations for dynamically p-enriched slope-limiting over discontinuous Galerkin systems of generalized equations". In: *Journal of Computational Physics* 230 (2011), pp. 8028–8056. DOI: 10.1016/j.jcp.2011.07.009.

[120] M. Musch. "Hierarchical scale separation as solver for linear systems resulting from discontinuous Galerkin discretizations". B. Sc. thesis. Friedrich-Alexander-University Erlangen-Nürnberg, 2016.

[121] J. von Neumann. *First Draft of a Report on the EDVAC*. Tech. rep. Moore School of Electrical Engineering, University of Pennsylvania, 1945. URL: https://sites.google.com/site/michaeldgodfrey/vonneumann/vnedvac.pdf?attredirects=0&d=1.

[122] J. Oden, I. Babuška, and C. Baumann. "A Discontinuous *hp* Finite Element Method for Diffusion Problems". In: *Journal of Computational Physics* 146.2 (1998), pp. 491–519. DOI: 10.1006/jcph.1998.6032.

[123] W. Pan, S. C. Kramer, and M. D. Piggot. "Multi-layer non-hydrostatic free surface modelling using the discontinuous Galerkin method". In: *Ocean Modelling* 134 (2019), pp. 68–83. DOI: 10.1016/j.ocemod.2019.01.003.

[124] N. Ray, A. Rupp, and A. Prechtel. "Discrete-continuum multiscale model for transport, biomass development and solid restructuring in porous media". In: *Advances in Water Resources* 107 (2017), pp. 393–404. DOI: 10.1016/j.advwatres.2017.04.001.

[125] W. H. Reed and T. R. Hill. *Triangular mesh methods for the neutron transport equation*. Tech. rep. LA-UR-73-0479. Los Alamos, NM: Los Alamos Scientific Laboratory, 1973. URL: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-73-0479.

[126] J. Reid. *Coarrays in the next Fortran Standard*. Tech. rep. ISO/IEC JTC1/SC22/WG5 N1824. Apr. 2010. URL: https://wg5-fortran.org/N1801-N1850/N1824.pdf.

[127] J.-F. Remacle, S. S. Frazao, X. Li, and M. S. Shephard. "An adaptive discretization of shallow-water equations based on discontinuous Galerkin methods". In: *International Journal for Numerical Methods in Fluids* 52.8 (2006), pp. 903–923. DOI: 10.1002/fld.1204.

[128] B. Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations*. Frontiers in Applied Mathematics. SIAM, 2008. DOI: 10.1137/1.9780898717440.

[129] P. L. Roe. "Approximate Riemann solvers, parameter vectors, and difference schemes". In: *Journal of Computational Physics* 43.2 (1981), pp. 357–372. DOI: 10.1016/0021-9991(81)90128-5.

[130] A. Rupp. "A Discontinuous Galerkin Model of Coupled Surface and Subsurface Flow in a Vertical Two-Dimensional Slice". M. Sc. thesis. Friedrich-Alexander-University Erlangen-Nürnberg, 2015.

[131] A. Rupp and P. Knabner. “Convergence order estimates of the local discontinuous Galerkin method for instationary Darcy flow”. In: *Numerical Methods for Partial Differential Equations* 33.4 (2017), pp. 1374–1394. DOI: 10.1002/num.22150.

[132] A. Rupp, P. Knabner, and C. Dawson. “A local discontinuous Galerkin scheme for Darcy flow with internal jumps”. In: *Computational Geosciences* 22.4 (2018), pp. 1149–1159. DOI: 10.1007/s10596-018-9743-7.

[133] I. Rybak. “Coupling Free Flow and Porous Medium Flow Systems Using Sharp Interface and Transition Region Concepts”. In: *Finite Volumes for Complex Applications VII – Elliptic, Parabolic and Hyperbolic Problems*. Ed. by J. Fuhrmann, M. Ohlberger, and C. Rohde. Vol. 78. Springer Proceedings in Mathematics & Statistics. 2014, pp. 703–711. DOI: 10.1007/978-3-319-05591-6_70,.

[134] I. Rybak, J. Magiera, R. Helmig, and C. Rohde. “Multirate time integration for coupled saturated/unsaturated porous medium and free flow systems”. In: *Computational Geosciences* 19.2 (2015), pp. 299–309. DOI: 10.1007/s10596-015-9469-8.

[135] P. G. Saffman. “On the Boundary Condition at the Surface of a Porous Medium”. In: *Studies in Applied Mathematics* 50 (1971). DOI: 10.1002/sapm197150293.

[136] H. Salehipour, G. R. Stuhne, and W. R. Peltier. “A higher order discontinuous Galerkin, global shallow water model: Global ocean tides and aquaplanet benchmarks”. In: *Ocean Modelling* 69 (2013), pp. 93–107. DOI: 10.1016/j.ocemod.2013.06.001.

[137] B. Schäppi, B. Przywara, F. Bellosa, T. Bogner, S. Weeren, R. Harrison, and A. Anglade. *Energy Efficient Servers in Europe*. Tech. rep. Intelligent Energy Europe Project, 2009. URL: https://ec.europa.eu/energy/intelligent/projects/sites/iee-projects/files/projects/documents/e-server_e_server_final_publishable_report_en.pdf.

[138] R. Schulz and P. Knabner. “An Effective Model for Biofilm Growth Made by Chemotactical Bacteria in Evolving Porous Media”. In: *SIAM Journal on Applied Mathematics* 77.5 (2017). DOI: 10.1137/16M108817X.

[139] R. Schulz, N. Ray, F. Frank, H. Mahato, and P. Knabner. “Strong solvability up to clogging of an effective diffusion–precipitation model in an evolving porous medium”. In: *European Journal of Applied Mathematics* 28.2 (2017). DOI: 10.1017/S0956792516000164.

[140] J. Shalf, S. Dosanjh, and J. Morrison. “Exascale Computing Technology Challenges”. In: *VECPAR 2010: High Performance Computing for Computational Science*. 2011, pp. 1–25. DOI: 10.1007/978-3-642-19328-6_1.

[141] C.-W. Shu. “High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments”. In: *Journal of Computational Physics* 316 (2016), pp. 598–613. DOI: 10.1016/j.jcp.2016.04.030.

[142] C. Simmendinger, M. Rahn, and D. Grünewald. *GASPI Tutorial*. Training course at HLRS, Stuttgart. Jan. 2014.

[143] F. Singer-Villalobos. *8 Things You Should Know About GPGPU Technology*. Accessed on Mar 14, 2019. Texas Advanced Computing Center, 2011. URL: https://www.tacc.utexas.edu/documents/13601/88790/8Things.pdf.

[144] N. Singer. *Astra supercomputer at Sandia Labs is fastest Arm-based machine on TOP500 list*. Accessed on Apr 30, 2019. Nov. 2018. URL: https://share-ng.sandia.gov/news/resources/news_releases/top_500/.

[145] N. Stahl. "Das Discontinuous-Galerkin-Verfahren für die 1D-Flachwassergleichungen". B. Sc. thesis. Friedrich-Alexander-University Erlangen-Nürnberg, 2016.

[146] H. Stengel, J. Treibig, G. Hager, and G. Wellein. "Quantifying Performance Bottlenecks of Stencil Computations Using the Execution-Cache-Memory Model". In: *Proceedings of the 29th ACM on International Conference on Supercomputing*. 2015, pp. 207–216. DOI: 10.1145/2751205.2751240.

[147] E. Strohmeier, J. Dongarra, H. Simon, and M. Meuer. *TOP500*. Accessed on Mar 13, 2019. Nov. 2018. URL: https://www.top500.org/lists/2018/11/.

[148] A. S. Tanenbaum and T. Austin. *Rechnerarchitektur*. 6th ed. Pearson, 2014. ISBN: 978-3-86894-238-5.

[149] C. Thiele, M. Araya-Polo, F. O. Alpak, B. Riviere, and F. Frank. "Inexact hierarchical scale separation: A two-scale approach for linear systems from discontinuous Galerkin discretizations". In: *Computers & Mathematics with Applications* 74.8 (2017), pp. 1769–1778. DOI: 10.1016/j.camwa.2017.06.025.

[150] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. 3rd ed. Springer, 2009. DOI: 10.1007/978-3-540-49834-6.

[151] E. F. Toro, M. Spruce, and W. Speares. "Restoration of the contact surface in the HLL-Riemann solver". In: *Shock Waves* 4.1 (1994), pp. 25–34. DOI: 10.1007/BF01414629.

[152] M. P. Ueckermann and P. F. J. Lermusiaux. "Hybridizable discontinuous Galerkin projection methods for Navier–Stokes and Boussinesq equations". In: *Journal of Computational Physics* 306 (2016), pp. 390–421. DOI: 10.1016/j.jcp.2015.11.028.

[153] L. Umlauf and H. J. Burchard. "A generic length-scale equation for geophysical turbulence models". In: *Journal of Marine Research* 61.2 (2003), pp. 235–265. DOI: 10.1357/002224003322005087.

[154] UNESCO. *Tenth report of the joint panel on oceanographic tables and standards*. UNESCO Technical Papers in Marine Science 36. Paris: UNESCO, 1981.

[155] V. Vallaeys, T. Kärnä, P. Delandmeter, J. Lambrechts, A. M. Baptista, E. Deleersnijder, and E. Hanert. "Discontinuous Galerkin modeling of the Columbia River's coupled estuary-plume dynamics". In: *Ocean Modelling* 124 (2018), pp. 111–124. DOI: 10.1016/j.ocemod.2018.02.004.

[156] S. Vater, N. Beisiegel, and J. Behrens. "A limiter-based well-balanced discontinuous Galerkin method for shallow-water flows with wetting and drying: One-dimensional case". In: *Advances in Water Resources* 85 (2015), pp. 1–13. DOI: 10.1016/j.advwatres.2015.08.008.

[157] C. B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow*. Vol. 13. Water Science and Technology Library. Springer, 1994. DOI: 10.1007/978-94-015-8354-1.

[158] J. C. Warner, C. R. Sherwood, H. G. Arango, and R. P. Signell. "Performance of four turbulence closure models implemented using a generic length scale method". In: *Ocean Modelling* 8 (2005), pp. 81–113. DOI: 10.1016/j.ocemod.2003.12.003.

[159] M. F. Wheeler. “An Elliptic Collocation-Finite Element Method with Interior Penalties”. In: *SIAM Journal on Numerical Analysis* 15.1 (1978), pp. 152–161. DOI: 10.1137/0715010.

[160] S. Whitaker. “Flow in porous media I: A theoretical derivation of Darcy’s law”. In: *Transport in Porous Media* 1.1 (1986), pp. 3–25. DOI: 10.1007/BF01036523.

[161] T. Wilde, A. Auweter, and H. Shoukourian. “The 4 Pillar Framework for energy efficient HPC data centers”. In: *Computer Science – Research and Development* 29.3–4 (2014), pp. 241–251. DOI: 10.1007/s00450-013-0244-6.

[162] S. Williams, A. Waterman, and D. Patterson. “Roofline: an insightful visual performance model for multicore architectures”. In: *Communications of the ACM* 52.4 (2009), pp. 65–76. DOI: 10.1145/1498765.1498785.

[163] D. Wirasaet, E. J. Kubatko, C. E. Michoski, S. Tanaka, J. J. Westerink, and C. Dawson. “Discontinuous Galerkin methods with nodal and hybrid modal/nodal triangular, quadrilateral, and polygonal elements for nonlinear shallow water flow”. In: *Computer Methods in Applied Mechanics and Engineering* 270 (2014), pp. 113–149. DOI: 10.1016/j.cma.2013.11.006.

[164] Y. Xing and X. Zhang. “Positivity-Preserving Well-Balanced Discontinuous Galerkin Methods for the Shallow Water Equations on Unstructured Triangular Meshes”. In: *Journal of Scientific Computing* 57.1 (2013), pp. 19–41. DOI: 10.1007/s10915-013-9695-y.

[165] Y. Xing, X. Zhang, and C.-W. Shu. “Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations”. In: *Advances in Water Resources* 33.12 (2010), pp. 1476–1493. DOI: 10.1016/j.advwatres.2010.08.005.

[166] B. Yan and R. A. Regueiro. “Superlinear Speedup Phenomenon in Parallel 3D Discrete Element Method (DEM) Simulations of Complex-shaped Particles”. In: *Parallel Computing* 75 (2018), pp. 61–87. DOI: 10.1016/j.parco.2018.03.007.

[167] M. Yang and Z.-J. Wang. “A Parameter-Free Generalized Moment Limiter for High-Order Methods on Unstructured Grids”. In: *AIAA, 47th Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Orlando*. 2009. DOI: 10.2514/6.2009-605.

[168] M. Yang and Z.-J. Wang. “A Parameter-Free Generalized Moment Limiter for High-Order Methods on Unstructured Grids”. In: *Advances in Applied Mathematics and Mechanics* 1.4 (2009), pp. 451–480. DOI: 10.4208/aamm.09-m0913.

Part B

Reprints of published journal articles

Paper 1

FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part I: Diffusion operator

F. Frank, B. Reuter, V. Aizinger, and P. Knabner
Computers and Mathematics with Applications (2015)
https://doi.org/10.1016/j.camwa.2015.04.013

Paper

2

A multi-platform scaling study for an OpenMP parallelization of a discontinuous Galerkin ocean model

B. Reuter, V. Aizinger, and H. Köstler

Computers & Fluids (2015)
https://doi.org/10.1016/j.compfluid.2015.05.020

Paper

3

FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part II: Advection operator and slope limiting

B. Reuter, V. Aizinger, M. Wieland, F. Frank, and P. Knabner

Computers and Mathematics with Applications (2016)
https://doi.org/10.1016/j.camwa.2016.08.006

Paper

4

Energy efficiency of the simulation of three-dimensional coastal ocean circulation on modern commodity and mobile processors

M. Geveler, B. Reuter, V. Aizinger, D. Göddeke, and S. Turek

Computer Science – Research and Development (2016)
https://doi.org/10.1007/s00450-016-0324-5

Paper

5 Anisotropic slope limiting for discontinuous Galerkin methods

V. Aizinger, A. Kosík, D. Kuzmin, and B. Reuter

International Journal for Numerical Methods in Fluids (2017)
https://doi.org/10.1002/fld.4360

Paper

6

FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part III: Hybridized discontinuous Galerkin (HDG) formulation

A. Jaust, B. Reuter, V. Aizinger, J. Schütz, and P. Knabner

Computers and Mathematics with Applications (2018)
https://doi.org/10.1016/j.camwa.2018.03.045

Paper

7

Locally Filtered Transport for computational efficiency in multi-component advection-reaction models

H. Hajduk, B. R. Hodges, V. Aizinger, and B. Reuter

Environmental Modelling & Software (2018)
https://doi.org/10.1016/j.envsoft.2018.01.003

Paper

8

Discontinuous Galerkin method for coupling hydrostatic free surface flows to saturated subsurface systems

B. Reuter, A. Rupp, V. Aizinger, and P. Knabner

Computers and Mathematics with Applications (2019)
https://doi.org/10.1016/j.camwa.2018.12.020