NACHHALTIGE NUMERISCHE PARAMETERSTUDIEN MIT DEM SNAKEMAKE-WORKFLOW NTRFLOWS

Nyhuis, M., Seume, J.
Institut für Turbomaschinen und Fluid-Dynamik, Leibniz Universität Hannover
An der Universität 1, Garbsen, Deutschland

Zusammenfassung

Der Ablauf einer jeden numerischen Simulation beinhaltet Arbeitsschritte im Pre-Processing, Processing und Post-Processing. Es werden Befehlszeilenanwendungen, Scriptsprachen und Funktionsbibliotheken angewendet, um Simulationen durchzuführen, Ergebnisse auszuwerten und diese darzustellen. Jeder dieser Arbeitsschritte ist komplex. Das heißt, dass jeder Arbeitsschritt in vielfältiger Weise durchgeführt werden kann und in jedem Arbeitsschritt eine große Anzahl an Interaktionen zwischen Programmen vorliegt, die Anwender_innen nicht ohne eine Systematik kontrollieren können. Durch die kollektive Dynamik aller Arbeitsschritte können schwer interpretierbare Abweichungen von Simulationsergebnissen in Wiederholungsversuchen entstehen. Reproduzierbarkeit ist aber eine grundsätzliche Anforderung an wissenschaftliche Arbeiten. Nur reproduzierbare Studien können als effiziente Grundlage für nachfolgende Arbeitsschritte, für die gesicherte Anwendung der Simulationsergebnisse und für nachfolgende Projekte dienen und sind eine notwendige Bedingung für kostengünstige Forschungs. Der Workflow 'NTRFlows' für Parameterstudien in der Numerik verwendet die Workflow-Umgebung 'snakemake', um Reproduzierbarkeit, Transparenz und Adaptierbarkeit zu gewährleisten. Das quelloffen entwickelte NTRFlows eignet sich als Blaupause für moderne Arbeitsprozesse zur Simulation hochkomplexer Strömungsphänomene.

Keywords

workflow; reproducibility; transparency; portability; snakemake

1. EINLEITUNG

Reproduzierbarkeit von Forschungsergebnissen ist für den Fortschritt der Wissenschaft eine unerlässliche Anforderung. Erst Reproduzierbarkeit ermöglicht eine technische Validierung von Versuchsergebnissen anhand ursprünglicher oder neuer Datensätze. In einem Artikel in der Zeitschrift 'Nature' von 2016 werden die Missstände in der Wissenschaft bezüglich einer Reproduzierbarkeitskrise öffentlichswirksam benannt [1]. Die Umfrage in der Nature ergab, dass mehr als 70% der Befragten bei dem Versuch, veröffentlichte Ergebnisse zu reproduzieren, scheiterten [1]. Nur 3% der befragten Wissenschaftler_innen dementierten eine Reproduzierbarkeitskrise. Die Relevanz dieser Herausforderung wird wegen immer komplexer werdender Projekte und dem Wettbewerb unter Forschungsinstitutionen weiter zunehmen. Die Deutsche Forschungsgemeinschaft (DFG) schärft in der neuen 'Checkliste zum Umgang mit Forschungsdaten' ihre Forderung nach Konzepten zur Wiederverwertbarkeit von Datenerhebungen bei der Antragsstellung [2].

> Reproducibility is like brushing your teeth. It is good for you, but it takes time and effort. Once you learn it, it becomes a habit.
> - Irakli Loladze [1]

Während sich 'Nature' mit der Reproduzierbarkeitskrise in der Wissenschaft beschäftigt, widmet sich die vorliegende Arbeit konkreten Reproduzierbarkeitsmaßnahmen für numerische Studien in der Strömungsmechanik. Hierzu werden mögliche Mängel und Ursachen in Abschnitt 2 festgestellt. Zu den Mängeln werden bereits etablierte Gegenmaßnahmen diskutiert und die Umsetzung des quelloffenen Workflows 'NTRFlows' präsentiert.
Der Workflow 'NTRFlows' basiert auf der quelloffenen Workflow-Umgebung 'snakemake'. In der Entwicklung von snakemake wird die Philosophie vertreten, dass Reproduzierbarkeit alleine nicht ausreicht, um nachhaltige Datenanalysen durchzuführen. Damit -Forscher_innen effizient auf durchgeführten Studien aufbauen können, muss zudem die Transparenz und die Adaptierbarkeit der Studien sichergestellt werden. Transparente Studien sind jene, deren Datenstruktur ein Nachvollziehen der Methodik und der technischen Umsetzung ermöglicht, um deren Qualität beurteilen zu können. Die Sicherstellung der Adaptivität ist die Bedingung für die effziente Anwendung von Studien auf neue Datensätzen oder Forschungsfragen [3].
Mit NTRFlows werden diese Stärken von 'snakemake' für numerische Anwendungen umgesetzt und erprobt. Es wird dabei postuliert, dass immer vier Arbeitsschritte notwendig sind, um eine Simu-

doi: 10.25967/570465

lation durchzufüren. Der Workflow verbindet die Definition eines Simulationsfalles (’create’) mit dem Preprocessing (’prep’), dem Ausführen der Simulation (’execute’) und dem Auswerten der Simulation (’post’). Der Workflow ist entwickelt anhand einer Kaskadensimulationen in OpenFOAM als Anwendungsbeispiel. Die Reproduzierbarkeit wird sichergestellt durch die Verwendung von Softwarecontainern und Versionskontrollsystemen. Der Workflow ist nicht beschränkt auf den im Anwendungsbeispiel verwendeten Strömungslöser OpenFOAM. Jeder Löser kann eingesetzt werden, bei dem ein Simulationsfall durch textbasierte Konfigurationsfiles definiert wird. Den Autoren sind keine numerischen Löser bekannt, bei dem dies nicht der Fall ist. Die Beschreibung der Workflowdurchführung und Adaptierbarkeit des Workflows ist Teil dieser Publikation. Um die Weiterentwicklung des Workflows sicherzustellen und -Forscher_innen einen Einblick in das Arbeiten mit dem Workflow zu ermöglichen, ist dieser quelloffen mit dem Versionskontrollsystem ’git’ entwickelt worden.

2. STAND DER TECHNIK

Nicht viele Studien widmen sich der Reproduzierbarkeit numerischer Simulationen. Einzelne Arbeiten betrachten die Reproduzierbarkeit im Sinne der erreichbaren Wiederholungsgenauigkeit wie beispielsweise in [4], [5] und [6]. Seit dem SARS-Ausbruch von 2002 werden CFD-Analysen vielfach eingesetzt, um mögliche Infektionswege zu untersuchen [7]. So wurden auch während des Beginns der COVID-19-Krise zahlreiche numerische Strömungssimulationen zu möglichen Infektionswegen veröffentlicht. Quelle [7] stuft lediglich 13% der von ihm untersuchten COVID-19 Studien als reproduzierbar. Dabei werden fehlende Informationen zu den Simulationen bemängelt.

Der Reproduzierbarkeitsbegriff
In der Literatur werden die Begriffe reproduzierbare Forschung, Reproduzierbarkeit, Replizierbarkeit, Wiederholbarkeit usw. häufig nicht eindeutig verwendet. Der Begriff ’reproduzierbare Forschung’ wurde in den 1990er Jahren eingeführt und bezieht sich auf computergestützte Studien, die von anderen Wissenschaftler_innen reproduziert werden können. Diese Fähigkeit erfordert offene Daten und quelloffene Software, weshalb die Reproduzierbarkeitsbewegung eng mit der Open-Science-Bewegung verbunden ist. In den folgenden Jahren wurde der Begriff Replikation übernommen, um sich auf eine unabhängige Studie zu beziehen, bei der Experimente erneut durchgeführt werden, um neue Daten zu generieren, die bei der Analyse zu denselben Ergebnissen führen.
Reproduzierbarkeit: Die Autor_innen einer Studie stellen ihren Code und ihre Daten zur Verfügung, so dass die Leser die Analyse überprüfen und wiederholen können, um die Zahlen in der Veröffentlichung zu reproduzieren. Reproduzierbare Forschung erleichtert die Replikation.
Replikation: Eine unabhängige Studie, bei der neue Daten mit ähnlichen oder anderen Methoden erhoben und analysiert werden, um zu denselben wissenschaftlichen Ergebnissen zu gelangen wie bei der ursprünglichen Studie. [8] [9]

Die Herausforderungen der Wiederholbarkeit von CFD-Studien korreliert naturgemäß mit ihrer Komplexität. Ein besonders herausfordernder Fall können instationäre Simulationen turbulenter Strömungen bei niedrigen Reynolds-Zahlen sein. Eine ausführliche Untersuchung hierzu bietet [8]. Darin werden Publikationen mit einem entsprechenden Qualitätskennzeichen versehen. Teil der Praxis ist eine durchgehende Versionierung von Codes, eine vollständige Veröffentlichung der genutzten und erzeugten Datensätzen und einer offenen (MIT-)Lizenzierung von Daten und Methoden. In Anbetracht dieser bereits umfassenden Reproduzierbarkeitsmaßnahmen ist es ernüchternd und überraschend, dass ein Replikationsversuch von 2019 zu einer zweidimensionalen DNS-Studie aus [8] scheitert.
In Abb. 1 werden die transienten Verläufe der Auftriebs- und Widerstandsbeiwerte eines umströmten Profils bei einer sehnenlängenbezogenen Reynoldszahl von Re=2000 der Reproduzierbarkeitsstudie gezeigt. Zunächst minimale Abweichungen im Simulationsverlauf führen zu einer signifikanten Abweichung der Kennwerte nach einer längeren Simulationszeit.
Die Abweichungen sind Betriebspunktabhängig, wie Abb. 2 zeigt. Grundlage der Studie ist wohlbemerkt die Verwendung desselben Datensatzes und derselben Quelltexte für das Durchführen und Auswerten der Simulationen. Trotzdem weichen die zeitlich gemittelten Ergebnisse einer Profilumströmung der DNS betriebspunktabhängig um bis zu 15% ab. Begründet wird dies durch eine Änderungen in der verwendeten Arbeitsumgebung. Eine Arbeitsumgebung in der Informationstechnik ist die Gesamtheit von verfügbaren Hardware, Softwarenwendungen, Hintergrunddiensten und Daten an einem Arbeitsplatz. Während die ausgeführten Anwendungen in der Replikationsstudie gleichwertig waren, werden andere Versionen von Software-Abhängigkeiten im Hintergrund ausgeführt. Diese Problematik ist nicht neu und sie begründet fortlaufende Weiterentwicklungen von Programmen zur Verwaltung von Arbeitsumgebungen, die ein reproduzierbares Arbeiten erleichtern sollen. Weiterhin sind Softwarecontainervirtualisierungen entwickelt worden, mit denen sich Anwendungen portierbar isolieren lassen und mit denen reproduzierbare Arbeitsumgebungen weitgehend sichergestellt werden können.

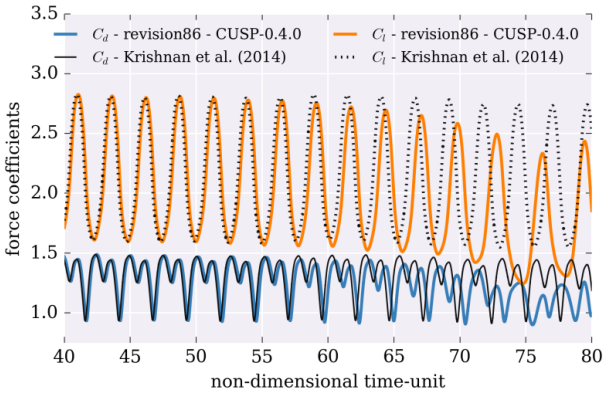

BILD 1. Abweichungen in den zeitabhängigen Auftriebsbeiwerten c_p und Widerstandsbeiwerten c_d in 2D-DNS-Simulationen in der Reproduzierbarkeitsstudie [8].

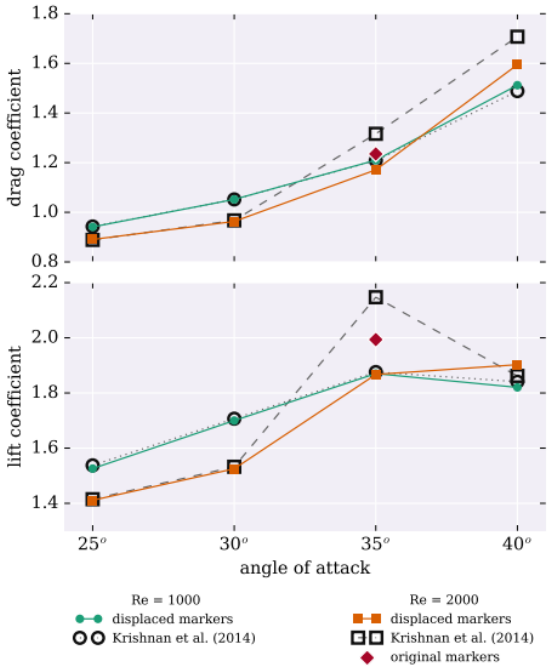

BILD 2. Abweichungen in den zeitlich gemittelten Ergebnissen der Reproduzierbarkeitsstudie [8] bei unterschiedlichen Reynolds-Zahlen und Anstellwinkeln

In der Folgearbeit [10] wird ein Workflow für CFD-Anwendungen mittels Softwarecontainer-Management vorgestellt, der durch die nachhaltige Speicherung von Datensätzen auf Cloud-Servern und die Arbeitsumgebungsverwaltung durch Softwarecontainer auf den Erfahrungen aus [8] aufbaut. Aus den Erfahrungen in [8], [10] und den Entwicklungen in der digitalen Arbeit ergibt sich eine Reihe an essenziellen Werkzeugen, die transparente und reproduzierbare digitale Arbeiten ermöglichen.
Die bisher erläuterten Erkenntnisse sind in Abb. 3 zusammengefasst. Die vorliegende Arbeit erhebt den Anspruch einer vollständigen Reproduzierbarkeit. Deshalb werden zentrale Werkzeuge wie die Versionskontrolle, Arbeitsumgebungsverwaltungen und Workflows als verknüpfende Elemente im Folgenden erläutert.

2.1. Versionskontrollsysteme

Versionsverwaltungssysteme werden typischerweise in der Softwareentwicklung eingesetzt, um Quelltexte zu verwalten. Das wohl geläufigste System zur Versionsverwaltung ist die freie Software git, die im Jahr 2005 zur Entwicklung des Betriebssystems Linux veröffentlicht wurde.
Versionskontrollsysteme erfassen Änderungen an Dateien. Diese Änderungen werden in einem Repositorium mit Zeitstempeln, der Benutzerkennung der bearbeitenden Person und einer erklärenden Mitteilung zu der Änderung festgehalten. Versionskontrollsysteme wie 'git' arbeiten mit unterschiedlichen logischen Ebenen. In der sichtbaren Arbeitsebene liegen die aktuell aus dem Versionierungssystem erzeugten und für den Anwender sichtbaren Dateien vor. Aus den gespeicherten Änderungen kann ein gewünschter Zustand des Repositoriums wiederhergestellt und ausgeführt werden [11].

2.2. Arbeitsumgebungsverwaltungen

Wie [8] gezeigt hat, ist es nicht ausreichend, lediglich darauf zu achten, dass dieselben Datensätze und dieselben Programmversionen genutzt werden, um Simulationsergebnisse zuverlässig zu reproduzieren. Es ist darauf zu achten, dass dieselbe Arbeitsumgebung zur Simulationsdurchführung zur Verfügung steht. Eine Arbeitsumgebung ist die Gesamtheit der Bedingungen, unter dem ein Computer betrieben wird. Dies bezieht sich also auf die Hardware, das Betriebssystem und alle zur Verfügung stehenden Programme. Da Anwender_innen nicht dazu in der Lage sind, diese Komplexität zu beherrschen, wurden Arbeitsumgebungsverwaltungen entwickelt [8].
Für die in der Wissenschaft weitverbreitete Scriptsprache 'Python' existiert das Paketmanagement 'pip'. Da die meisten Anwendungen auf externe Funktionsbibliotheken angewiesen sind, ist es erforderlich, auch die Programmversionen dieser externen Abhängigkeiten zu verwalten. 'pip' kann dies nicht leisten. Eine bessere Reproduzierbarkeit kann mit dem Paketmanager 'conda' erhalten werden, mit dem Anwender_innen ohne Administrationsrechte auch Betriebssystemsabhängigkeiten verwalten können. 'conda' beschränkt sich nicht auf Python-Anwendungen und ist insbesondere für die Einrichtung von Umgebungen auf High-Performance-Computers (HPC) geeignet und hat deshalb in der Wissenschaft einen hohen Stellenwert [12].
conda-Umgebungen sind nicht portabel und die Reproduzierbarkeit ist weiterhin eingeschränkt. Mit Softwarecontainern können Anwendungen und alle benötigten Abhängigkeiten portierbar gespeichert werden, um diese reproduzierbar und performant in unterschiedlichen Arbeitsumgebungen anzuwenden. Zwei freie Softwareanwendungen stehen zur

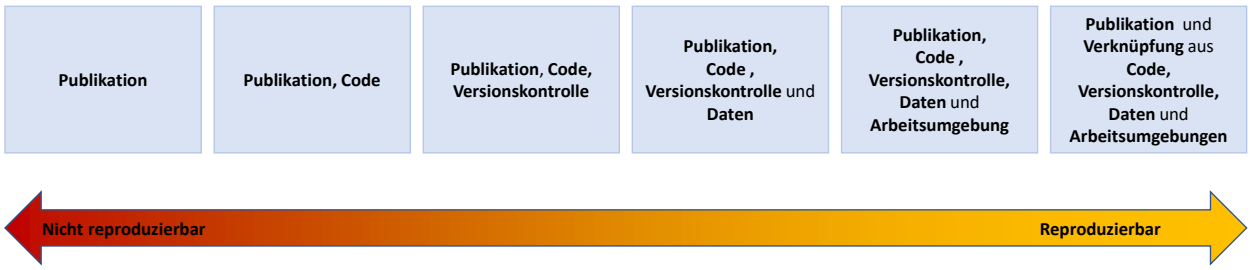

BILD 3. Reproduzierbarkeits-Hierarchie wissenschaftlicher Arbeiten nach dem Stand der Technik [3] [8] [10]

Verfügung: 'Docker' und 'Singularity'. 'Singularity' ist 'Docker' in Sachen Systemsicherheit und Performanz in HPC-Anwendungen überlegen. Durch die weite Verbreitung von Docker in der Unternehmenskultur und Wissenschaft wird 'Docker' jedoch noch immer eine Existenzberechtigung zugeschrieben. Singularity ist wegen der großen Verbreitung von Docker dazu in der Lage, Docker-Container auszuführen. [13].

2.3. Workflow

Ein Workflow besteht aus einem orchestrierten und wiederholbaren Muster der Programmausführungen und der systematischen Organisation von Ressourcen und Prozessen. Erläutert wurden bereits die Notwendigkeiten für wiederholbare digitale Arbeitsabläufe. Workflow-Umgebungen ermöglichen Anwender_innen die Handhabung der beschriebenen Komplexitäten und ermöglichen so die Definition von wiederholbaren Arbeitsabläufen. Seit der ersten Veröffentlichung einer Workflow-Umgebung kam es zu einer 'kambrischen Explosion' an Entwicklungen, die bis heute anhält [3]. Insbesondere für Machine-Learning-Projekte ist aufgrund der besonderen Herausforderungen in Sachen Reproduzierbarkeit die Verwendung von Workflow-Umgebungen ein anerkannter Standard in der Wissenschaft.
Um Anwendungsfehler eines Workflows zu vermeiden, erlaubt 'snakemake' eine Validierung der Ausführungskonfiguration. Hierzu sind Konfigurationsdateien zu erzeugen, mit denen erlaubte Parameterräume beschränkt werden.

3. WORKFLOW-UMGEBUNG SNAKEMAKE

NTRFlows verwendet die in Python geschriebene und quelloffene Workflow-Umgebung 'snakemake'. Die Entwicklung von 'snakemake' wird fortlaufend im Rolling-Paper [3] veröffentlicht und wird mehr als 300 Mal im Jahr zitiert.
Nach der in 5 dargestellte Philosophie von 'snakemake' wird Nachhaltigkeit erst durch die Gewährleistung von Reproduzierbarkeit, Adaptierbarkeit und Transparenz erreicht. Eine Workflow-Umgebung unterstützt die oberste Hierarchieebene, um diese drei Eigenschaften zu erlangen. Sind diese Eigenschaften erfüllt, ist eine wissenschaftliche Datenerhebung nachhaltig.
Zentrales Element von 'snakemake' ist die Unterteilung eines Workflows in einzelne Regeln. Jede Regel wird definiert über Funktionsaufrufe, die beispielsweise durch einen Eingangsdatensatz Ausgabedateien erstellen. Die Verkettung von Regeln wird definiert durch übereinstimmende Dateinamenmuster. Es sind keine technischen Kenntnisse notwendig, um Zusammenhänge von Regeln herzustellen.
Bei der Definition von Snakemake-Workflows sind Ordner-Strukturen einzuhalten, um Lesbarkeit und Kompatibilität des Workflows zu ermöglichen. In einem 'resources'-Ordner werden Daten aufbewahrt, die zur Ausführung des Workflows benötigt werden. Ergebnisdateien, die durch die Ausführung entstehen, sind im 'results'-Ordner abzulegen. Die Konfiguration des Workflows wird im 'config'-Ordner definiert. Die Regeln, Arbeitsumgebungen, Scripte und weitere workflow-spezifische Daten werden in entsprechenden Unterordnern des Ordners 'workflow' abgelegt. Eine ausführliche Erklärung und detaillierte Beschreibungen sind der snakemake-Dokumentation zu entnehmen.
Einer Regel kann eine Arbeitsumgebung zugeordnet werden. 'snakemake' kann 'conda'-Umgebungen, Docker-Container als auch Singularity-Container verwenden, um eine Arbeitsumgebung herzustellen. Es

```
def get_results(wildcards):
    return f"results/{wildcards.samplename}_copieddata.tsv"

rule all:
    input: get_results

rule copydata:
    input: "resources/{samplename}_originaldata.tsv"
    output: "results/{samplename}_copieddata.tsv"
    singularity: "docker://ubuntu:latest"
    log: "logs/{samplename}"
    shell:
        """
        cp {input} {output} > {log}
        """
```

BILD 4. Aufbau eines minimalistischen, funktionalen snakemake-Workflows zum Kopieren von Datensätzen.

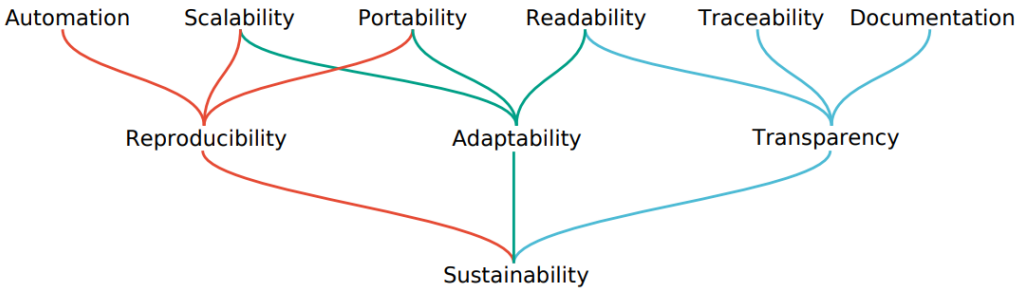

BILD 5. Aspekte nachhaltiger Datenerhebungen nach Snakemake

ist möglich, auf Arbeitsumgebungsverwaltungen zu verzichten. Es ist die Entscheidung des Anwenders, welches Niveau an Reproduzierbarkeit gewählt wird.
In Abb. 4 ist ein minimaler aber funktionaler snakemake-Workflow dargestellt. Die Regel 'all' definiert die Ergebnisdateien, die aus einem Workflow-Aufruf entstehen sollen. Der Workflow nutzt 'wildcards', um Dateien eindeutig zu identifizieren. Diese 'wildcards' werden an die Funktion get_results weitergegeben, um die zu erzeugenden Ergebnisdateien zu identifizieren und die Verkettung notwendiger Arbeitsschritte zu definieren. Die Regel 'copydata' benötigt Eingangsdateien (input), um Ausgangsdateien (output) zu erzeugen. Der Regelaufruf wird in einem eindeutig zu identifizierender Protokolldatei dokumentiert. Bei einem Regelaufruf wird ein Singularity-Container von einem Cloud-Service genutzt, um eine reproduzierbare Arbeitsumgebung herzustellen.
Die Workflow-Umgebung wird explizit entwickelt für HPC-Anwendungen. So kann 'snakemake' die Verwaltung von sogenannten 'Job-Schedulern' auf Clustersystemen übernehmen. Unterstützt werden geläufigen Systeme wie 'PBS' und 'slurm'. Angeforderte Ressourcen für die Ausführung von Regeln werden diesen Systemen automatisch weitergeben, wodurch dem Job-Scheduler eine optimale Auswahl von Computing-Nodes erlaubt wird. In Cluster-Profilen können Informationen zu den verwendeten Cluster-Systemen gespeichert werden, die zur automatisierten Definition von Job-Scripten verwendet werden können.

4. NTRFLOWS

Der Workflow 'NTRFlows' greift die in Abschnitt 2 beschriebenen und bereits in der Wissenschaft etablierte Lösungsansätze für Herausforderungen bezüglich Reproduzierbarkeit, Transparenz, Anpassungsfähigkeit, Skalierbarkeit und Portabilität auf. Grundlage hierfür ist die Workflow-Umgebung 'snakemake'. Ziel ist der Arbeit ist es, einen Workflow zu definieren der einerseits die Anforderungen zum reproduzierbaren Arbeiten nach dem Stand der Technik erfüllt und zum anderen anpassbar ist für die Verwendung etwaiger Strömungslöser und Strömungsfälle.
NTRFlows postuliert, dass mindestens vier Arbeitsschritte notwendig sind, um eine Simulation durchzuführen. Um eine Simulation zu definieren, müssen 1. Konfigurationsdateien erzeugt werden, 2. muss die Simulation durch den Aufruf von löserinternen Funktionen zur Ausführung vorbereitet werden, 3. muss die Simulation durchgeführt werden und 4. sind die Simulationsergebnisse auszuwerten.
Arbeitsumgebungen werden in der veröffentlichten Version in dem Anwendungsbeispiel ausschließlich durch Singularity-Container verwaltet. Dabei wird man bei einer Anwendung nicht zur Verwendung von Softwarecontainern gezwungen. Werden keine Container genutzt, kann die entwickelnde Person der Parameterstudie nicht weiter von einer vollständigen Reproduzierbarkeit ausgehen.
Vor der ersten Ausführung des Workflows werden sämtliche Arbeitsumgebungen von Cloud-Services heruntergeladen und eingerichtet. Es entfallen für die Anwender_innen und für die Administration komplexe Installationsprozesse.
NTRFlows soll sich ausdrücklich nicht auf die Verwendung von OpenFOAM als numerischen Löser beschränken. Neben der Beschreibung der Durchführung dieser Regeln ist außerdem eine Diskussion der Anpassung für neue numerische Probleme Teil der Veröffentlichung.

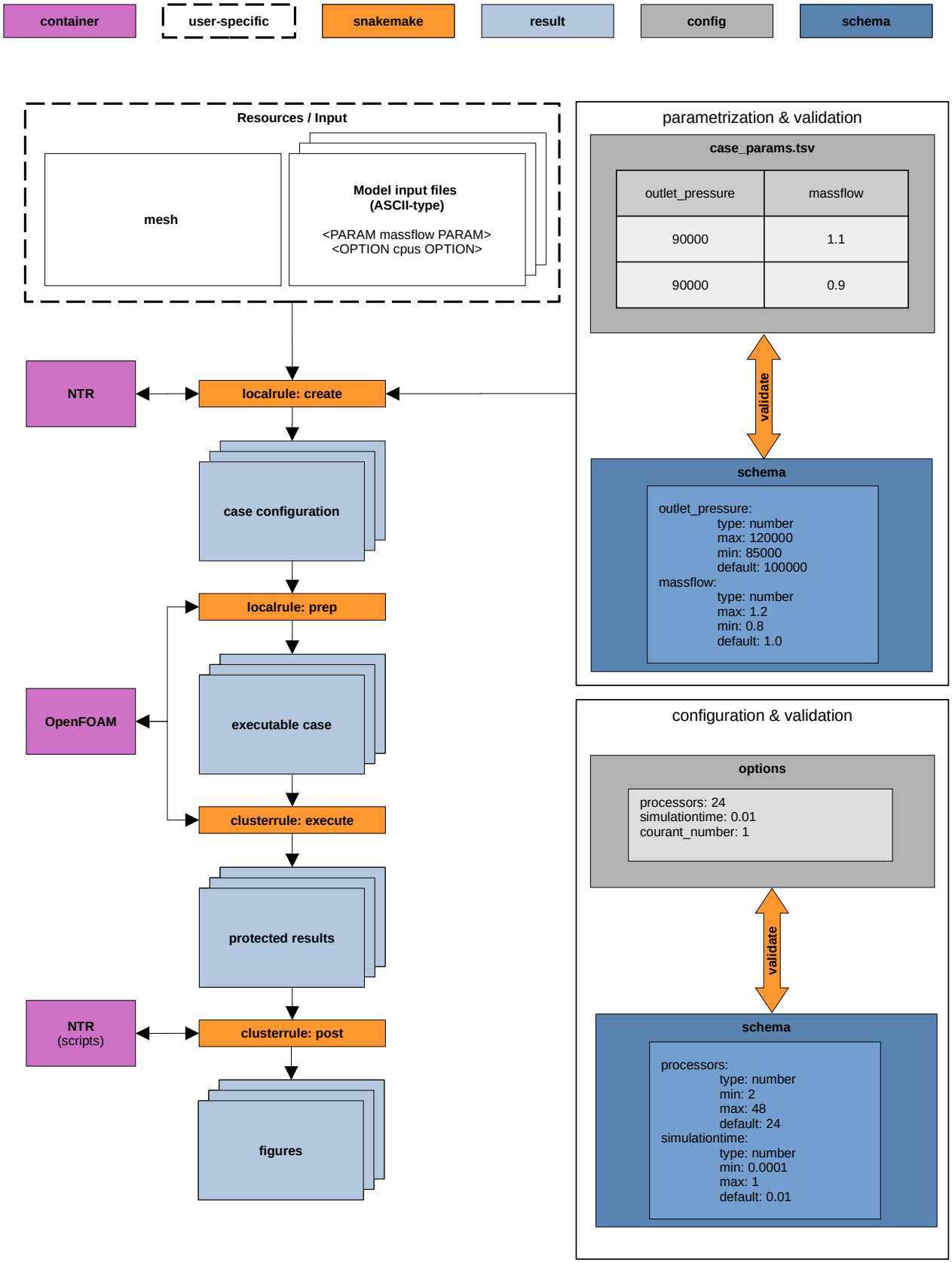

BILD 6. Vereinfachte NTRFlows Struktur

4.1. Struktur

Die Struktur (siehe Abb. 6) des Workflows folgt der Hypothese dieser Arbeit und es sind vier Regeln zur Erzeugung (create), zum Preprocessing (prep), zur Durchführung (execute) und zum Postprocessing (post) von Simulationen definiert worden.
Eine andere logische Ebene ist die Dateistruktur von NTRFlows. Für das Verständnis dieser Arbeit soll die Logik der Parametrisierung und der Simulationsspezifischen Dateistrukturen erklärt werden. Der Parametrisierungsvorgang wird in Abschn. 4.2 erläutert.
Die Dateistruktur ist in Abb. 7 gezeigt. Hier noch ein überleitender und beschreibenden Satz.
In dem 'config'-Verzeichnis wird der Ablauf der Parameterstudie eingestellt. In der Tabelle 'case_params.tsv' wird der Parameterraum der Studie definiert. Der Workflow selbst und Simulationsparameter, die nicht teil des Parameterraums der Studie sind, werden in der Konfigurationsdatei 'workflowsettings.yaml' definiert. Alle simulationsspezifischen Konfigurationen sind in dem Vorlagenverzeichnis 'resources' begründet. Hier sind nichtkonfigurierbare Vorlagen wie Rechennetze ('mesh') und Konfigurationsdateien für Simulationen ('casefiles') abzulegen.
Um eine Qualitätssicherung von wiederholt eingesetzten Funktionen zu ermöglichen, wird das Python-Paket 'NTR' (Numerical Test Rig) verwendet. Das Paket wird in einem quelloffenem git-Repositorium gepflegt. Wesentlicher Bestandteil von 'NTR' ist die freie und leicht anzuwendene Visualisierungsoftware 'PyVista', mit der auch komplexe Prototypen von Auswertungsalgorithmen zügig definiert werden können. Die Verwendung von Test-Funktionen bedingt einen Qualitätsstandard auch bei eiligen Weiterentwicklungen. Eine virtualisierte Grafikeinheit ermöglicht das Erzeugen von Abbildungen auf Rechenknoten ohne Grafikkarten.

```
NTRFlows
├── config
│   ├── case_params.tsv
│   ├── README.md
│   └── workflowsettings.yaml
├── LICENSE
├── logs
├── profiles
├── README.md
├── requirements.txt
├── resources
│   ├── mesh
│   └── casefiles
├── results
└── workflow
    ├── rules
    │   ├── common.smk
    │   ├── create.smk
    │   ├── execute.smk
    │   ├── post.smk
    │   └── prep.smk
    ├── schemas
    │   ├── config.schema.yaml
    │   └── param.schema.yaml
    ├── scripts
    │   ├── ntr_bladeloading.py
    │   ├── ntr_contours.py
    │   └── ntr_createcase.py
    └── Snakefile
```

BILD 7. Dateistruktur von NTRFlows

4.2. Regel create

Diese Regel wird dazu verwendet, Konfigurationsdateien für Simulationen zu erzeugen. Diese Regel ist löserunabhängig definiert. Sie muss nicht verändert werden, egal welcher numerische Löser eingesetzt wird.
Die Konfigurationsdateien werden im Template-Verzeichnis abgelegt. Es ist essenziell, dass diese Dateien als Klartext, also in einem ASCII-Format gespeichert werden. Dabei können Parameter und Optionen definiert werden. Die verwendeten Parameter werden genutzt, um die Dateinamensmuster zu definieren. Optionen werden verwendet, um eine Konfiguration einer Parameterstudie flexibel zu halten. Beispielsweise ist in NTRFlows verwendete Option die Anzahl an verwendeten Prozessoren für eine Simulationsausführung. Diese Konfiguration kann angepasst werden, ist aber im Regelfall nicht von Interesse für die Auswertung von Simulationen und es wird kein nennenswerter Einfluss auf die Reproduzierbarkeit von Simulationen erwartet. Entsprechend entfällt die Kenntlichmachung der verwendeten Option in den Dateinamenmustern. Da Parameter als auch Optionen Eingabewerte der Anwender_innen sind, werden diese vor der Durchführung einer Parameterstudie validiert. Simulationen, bei denen Parameter 'massflow' und 'outlet_pressure' variiert werden, erhalten automatisch Dateinamen wie

results/massflow~1.1_outlet_pressure~110000

Die Namensgebung ist standardisiert und wird nicht mehr durch die bearbeitenden Person beeinflusst.
Der zu simulierende Parameterraum und damit die verwendeten Parameter zur Durchführung von 'create' werden in der Tabellendatei 'case_params.tsv' im Konfigurationsverzeichnis definiert.

- **Ausführung:**
  In NTRFlows wird zunächst die Konfiguration durch die 'snakemake'-Funktion 'validate' überprüft. Ist die Konfiguration des Workflows zulässig, sind Anwendungsfehler unwahrscheinlich und der Workflow wird durchgeführt. Um die Konfigurationsdateien einer Simulation zu erstellen, wird die NTR-Umgebung aufgerufen. Da die Template-Dateien ebenfalls Anwender-Eingaben sind und fehlerbehaftet sein können, überprüfen NTR-Funktionen die Vollständigkeit und Verwendbarkeit der Template-Dateien mit dem definiertem Parameterraum. Sind die Daten zulässig, werden die Konfigurationsdateien der Simulation vom Algorithmus ausgeschrieben.

- **Anpassung:**
  Auch proprietäre Software, dessen Verwendung häufig von grafischen Oberflächen geprägt ist, berücksichtigt das Reproduzierbarkeitsproblem. Den Autoren ist kein numerischer Löser bekannt, für den die hier aufgestellte Regel nicht unverändert verwendet werden kann.
  Die Anpassung der Regel für neue Simulationsfälle und andere numerische Löser liegt in der Definition der Daten, die im Ordner 'resources/template' abgelegt werden. Hierzu muss eine lauffähige Simulation mit einem Löser der Wahl definiert werden und die Konfiguration dieser Simulation exportiert werden. Das Verfahren hierzu ist abhängig vom verwendeten Lösungsalgorithmus und kann nicht eindeutig definiert werden. Parameter sind mit dem Muster

  <PARAM parametername PARAM>

  zu markieren. So kann in den Konfigurationsdateien ein Druck am Einströmrand parametrisiert werden, in dem dieser durch

  <PARAM pressure_inlet PARAM>

  ersetzt wird. Zu diesem Parameter ist ein Validierungsmuster in 'param.schema.yaml' zu definieren. So kann der erlaubte Parameterraum beschränkt werden, um Fehlanwendungen durch neue Anwender_innen zu vermeiden. Analog werden Optionen durch

  <OPTION optionsname OPTION>

  definiert. Erlaubte Optionen werden in 'config.schema.yaml' definiert.
  Es ist möglich, eigene NTR-Container mit singularity zu kompilieren und diese nicht der Öffentlichkeit zur Verfügung zu stellen. Wollen Institutionen ihre Algorithmen schützen, können eigene Instanzen eines Cloud-Services (namentlich Singularity Registry Server) genutzt werden, die nur im Netzwerk des Unternehmens erreichbar sind.

4.3. Regel prep

Die Regel 'prep' dient der Ausführung von löserspezifischen Funktionen, um das Preprocessing abzuschließen und um ausführbare Simulationen zu erhalten. NTRFlows geht nicht davon aus, dass sämtliche zur Simulation gehörenden Dateien konfigurierbare ASCII-Dateien sind. Beispielsweise können Netzdateien binäre Dateitypen darstellen, die nicht verändert werden sollen.

- **Ausführung:**
  In dem Anwendungsbeispiel wird die OpenFOAM-Arbeitsumgebung durch singularity bereitgestellt. Anschließend wird das Rechennetz eingelesen und eine optimale Partitionierung der Simulation zum Durchführen eines Multiprocessing-Jobs wird vorbereitet.
- **Anpassung:**
  Wird ein anderer numerischer Löser verwendet als im Anwendungsbeispiel, muss die Arbeitsumgebung umdefiniert werden. Die Regel 'prep' nutzt die Ausgabedateien der Regel 'create'. Da diese nicht verändert wird und NTRFlows die Dateistrukturmuster aus der Definition von Parameternamen und Parameterwerten erzeugt, müssen diese nicht neu definiert werden. Lediglich wenn andere Daten zum Preprocessing hinzugefügt werden sollen, sind Änderungen in der Definition der Eingangsdateien zu vollziehen. Um einen Eindruck vom Anpassungsprozess zu erhalten, ist in Abb. 8 der generisch gültige Code blau, und der anzupassende Code grün markiert.

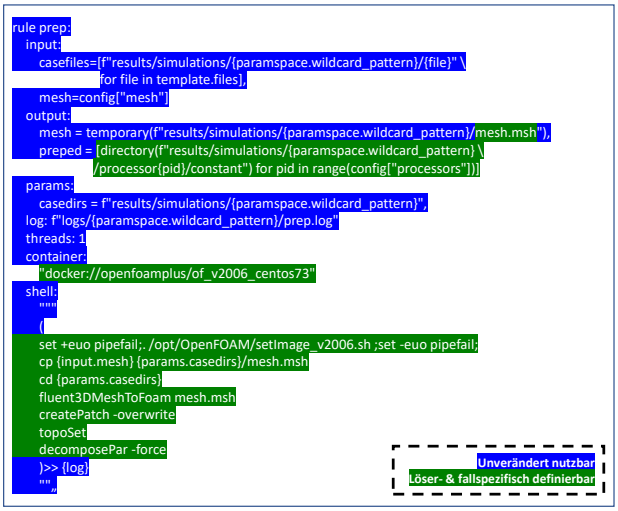

BILD 8. Quelltext der Regel 'prep' im Anwendungsbeispiel des Repositoriums. Der grün markierte Abschnitt ist Löser-& fallspezifisch und ist für eigene Zwecke anzupassen.

4.4. Regel execute

Die Regel 'execute' wird verwendet, um Simulationen durchzuführen. Dies wird im Regelfall durch die Computing-Nodes von Cluster-Systemen realisiert. Wird snakemake mit einem Profil zur Ausführung auf einem Cluster aufgerufen oder wird der Flag 'cluster' zur Ausführung hinzugefügt, erstellt snakemake für jede Simulationsdurchführung eigenständig Jobscripte und verwaltet die Simulationsdurchführung.

- **Ausführung:**
  Die Regel 'execute' kann lokal oder auf einem Cluster-System ausgeführt werden. Wird die Regel auf einem Cluster-System ausgeführt, werden die in der Regel definierten Befehlsaufrufe dem Job-Scheduler übergeben. Vor dem Ausführen der Befehlskette wird in dem Anwendungsbeispiel die OpenFOAM-Arbeitsumgebung bereit gestellt. Die Simulationsergebnisse, die eben die Ausgangsdateien der Regel darstellen, werden vor versehentlichen Manipulationen oder Löschen des Anwenders geschützt.
- **Anpassung:**
  Wird ein anderer numerischer Löser verwendet als im Anwendungsbeispiel, muss die Arbeitsumgebung umdefiniert werden. Da jeder Löser eigene Namensmuster für Ergebnisdateien definiert, sind

neben dem Aufruf des Lösungsalgorithmusses in der Regel die Ausgangsdateien der Regel anzupassen.

4.5. Regel post

Zu einer vollständigen numerischen Studie gehört auch die Auswertung der erzeugten Datensätze. Reproduzierbarkeitsprobleme werden häufig durch fehlerhaften Auswertungsroutinen verursacht. In der Regel 'post' werden deshalb gesammelt Auswerteroutinen durchgeführt.

- **Ausführung:**
  Partitionierte OpenFOAM-Simulationen bleiben auch nach der Ausführung aufgeteilt in Dateistrukturen für die einzelnen Prozessoren. Um die Simulation auszuwerten, wird zunächst erneut die OpenFOAM-Umgebung aufgerufen, um eine einfacher auszuwertende VTK-Ergebnisdatei zu erzeugen.
  In dem Anwendungsbeispiel werden Auswerteroutinen aus dem NTR-Paket aufgerufen. Diese Struktur ist gegenüber einfachen Auswertescripten attraktiv, weil das NTR-Paket eine Qualitätssicherung ebendieser ermöglicht.
- **Anpassung:**
  Es ist die Natur von Auswerteroutinen, dass sie individuell Problemen angepasst werden müssen. Die Postprocessing-Regel benötigt am meisten Anpassungen, um sie für neue Anwendungen vorzubereiten. Je nach verwendeten numerischen Löser kann Erzeugung einer Ergebnisdatei aus einem Partitioniertem Simulationsfall gänzlich verzichtet werden. Anstelle des NTR-Pakets für eine Python-basierte Auswertung können auch andere Auswerteumgebungen eingesetzt werden. Beispielsweise existieren offizielle Containerversionen zu der weitverbreiteten Postprocessing-Software 'paraview'.

5. ERGEBNISSE

Eingänglich konnten im Abschnitt 2 Werkzeuge definiert werden, die nach dem heutigem Stand der Technik bestmöglich Reproduzierbarkeit in der digitalen Datenerzeugung gewährleisten. In NTRFlows werden diese Werkzeuge mit 'snakemake' eingesetzt und ein wiederholbarer Arbeitsablauf zur Parametrisierung einer Verdichterkaskadensimulation in OpenFOAM entsteht.
Zur Einordnung von NTRFlows müssen die von snakemake versprochenen Eigenschaften (siehe Abb. 5) bewertet werden.

5.1. Automatisierung

Eine vollständige Durchführung von Parameterstudien von Pre-Processing bis zur Auswertung sämtlicher Studien ist durch einen einzigen Befehlsaufruf möglich.

5.2. Skalierbarkeit

Es ist ohne Probleme möglich, einen Parameterraum und damit einer Parameterstudie zu skalieren. Einschränkend ist die Definition der Regel 'execute', bei der eine Skalierung des Multiprocessings bei der Simulationsdurchführung derzeit beschränkt ist auf einen Rechenknoten. Im NTRFlows-Repositorium liegt ein entsprechendes Issue zur Dokumentation und Problembeseitigung vor.

5.3. Portabilität

Der Workflow ist portabel und kann auf allen Linux-Systemen ausgeführt werden, ohne dass komplexere Vorbereitungen getroffen werden müssen. Zur Ausführung des Workflows muss lediglich eine Python-Instanz und die Pakete 'snakemake' und 'pandas' vorliegen. Hierzu sind keine Administrationsrechte notwendig.
Die Portierbarkeit wurde ebenfalls in Verbindung mit dem Strömungslöser TRACE des Deutschen Zentrums für Luft- und Raumfahrt erprobt. Dieser wurde für diese Zwecke Containerisiert und Kaskadensimulationen wurden durchgeführt.

5.4. Lesbarkeit

NTRFlows hält sich an die Vorgaben zur Lesbarkeit von Snakemake-Workflows. Sogenannte 'linting'-Algorithmen können eingesetzt werden um Quelltexte auf Verstöße gegen Formatierungsstandards und Fehler zu überprüfen. Diese Überpüfung schlägt zu diesem Zeitpunkt allerdings fehl. Das 'linting' schlägt dabei für jede Regel geringfügige Refaktorierungen (also Umstrukturierungen) vor. Dazu gehört beispielsweise das Ausgliedern von Code zur Erzeugung von Dateinamen, da der kompliziert erscheinende Code die Regel unleserlich macht.

5.5. Nachvollziehbarkeit

Zur Durchführung des Workflows werden zu allen komplexenm Regeln Protokolldateien ausgeschrieben, um die Durchführung nachvollziehen zuu können.

5.6. Dokumentation

Der Workflow ist von Beginn an unter der Versionskontrolle 'git' erzeugt worden, um eine bestmögliche Nachvollziehbarkeit der Entwicklung zu gewährleisten.

6. AUSBLICK

Mit NTRFlows wurde ein Konzept entwickelt, durch welches nachhaltige Datenerhebungen mittels numerischer Simulationen durchgeführt werden können. Wie in den Ergebnissen festgestellt und im Repositorium dokumentiert, ist die Herausforderung von Multinode-Simulationen zu lösen, damit der Workflow

auch für Parameterstudien mit hohen Ressourcenansprüchen angewendet werden kann.
Ein weiterer verfolgter Entwicklungsstrang ist die automatisierte Abfolge voneinander abhängigen Simulationen. Numerische Simulationen werden häufig abhängig voneinander gestartet. Das heißt, dass nach dem Beenden einer Simulation das Ergebnis dieser genutzt wird, um eine weitere Simulation zu starten. Eine solche Entwicklung wird bereits im Repositorium des Projekts verfolgt.

7. REPOSITORIEN

Die vorliegende Arbeit soll dazu dienen, Erfahrungen in der Praxis der nachhaltigen Datenerhebungen zu sammeln, einzelne Aspekte dieser Arbeitsweisen zu übernehmen oder die hier vorgestellte Arbeit weiterzuentwickeln. Die Codes können den folgenden Repositorien übernommen werden.

NTRFlows

https://gitlab.uni-hannover.de/tfd_public/tools/NTRFlows

NTR

https://gitlab.uni-hannover.de/tfd_public/tools/NTRfC

8. SCHLUSSFOLGERUNGEN

NTRFlows ist eine Funktionsstruktur für reproduzierbare CFD-Analysen. Der Workflow wurde entsprechend der Erfahrungen aus Reproduzierbarkeitsstudien entwickelt. Durch die Verwendung von Softwarecontainern kann eine maximale Portabilität und Reproduzierbarkeit von Arbeiten garantiert werden. Damit sollen administrierende Mitarbeitende entlastet werden und die Einarbeitungszeiten neuer Mitarbeiter_innen verkürzt werden.
Der Workflow wird quelloffen entwickelt. -Forscher_innen sollen dazu eingeladen werden, reproduzierbare Arbeitsweisen im Kontext der numerischen Strömungsmechanik kennenzulernen und diese Arbeitsweisen zu adaptieren.
Der Workflow beweist, dass nachhaltige Arbeitsweisen in der numerischen Strömungsmechanik möglich sind. Für die erfolgreiche Durchführung eines NTRFlow-Workflows liegen außer 'snakemake', 'singularity' und 'pandas' keine Softwareabhängigkeiten vor.

9. DANKSAGUNG

Dank gilt Henning Schiebenhöfer des Robert-Koch-Instituts in Berlin. Ohne hilfreiche Kritik und ausgiebige Beratung wäre diese Arbeit nicht entstanden.

Kontaktadresse:

nyhuis@tfd.uni-hannover.de

Literatur

[1] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature News*, 533(7604):452, May 2016. DOI: 10.1038/533452a.

[2] Deutsche Forschungsgemeinschaft. Checkliste zum umgang mit forschungsdaten, 2021. https://www.dfg.de/download/pdf/foerderung/grundlagen_dfg_foerderung/forschungsdaten/forschungsdaten_checkliste_de.pdf.

[3] Felix Mölder, Kim Jablonski, Brice Letcher, Michael Hall, Christopher Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven Twardziok, Alexander Kanitz, Andreas Wilm, Manuel Holtgrewe, Sven Rahmann, Sven Nahnsen, and Johannes Köster. Sustainable data analysis with snakemake. *F1000Research*, 10:33, 04 2021. DOI: 10.12688/f1000research.29032.2.

[4] T Uchida. Reproducibility of complex turbulent flow using commercially-available cfd software: Report 1: For the case of a three-dimensional isolated-hill with steep slopes. *Repots of Reseach Institute for Applied Mechanics Kyushu University*, (150):47–59, Mar. 2016. ISSN: 1345-5664. DOI: 10.15017/1660833.

[5] T Uchida. Reproducibility of complex turbulent flow using commercially-available cfd software: Report 2: For the case of a two-dimensional ridge with steep slopes. *Repots of Reseach Institute for Applied Mechanics Kyushu University*, (150):60–70, Mar. 2016. ISSN: 1345-5664. DOI: 10.15017/1660834.

[6] T Uchida. Reproducibility of complex turbulent flow using commercially-available cfd software: Report 3: For the case of a three-dimensional cube. *Repots of Reseach Institute for Applied Mechanics Kyushu University*, (150):60–70, Mar. 2016. ISSN: 1345-5664. DOI: 10.15017/1660834.

[7] Khalid M. Saqr. Amicus plato, sed magis amica veritas: There is a reproducibility crisis in covid-19 computational fluid dynamics studies, 2021. DOI: 10.48550/ARXIV.2101.04874, https://arxiv.org/abs/2101.04874.

[8] O. Mesnard and L. A. Barba. Reproducible and replicable cfd: it's harder than you think. 2016. DOI: 10.48550/ARXIV.1605.04339.

[9] Thomas Ludwig and Beate Geyer. Reproduzierbarkeit. *Informatik Spektrum*, 2019. DOI: https://doi.org/10.1007/s00287-019-01149-2.

[10] Olivier Mesnard and Lorena Barba. Reproducible workflow on a public cloud for computational fluid dynamics. *Computing in Science & Engineering*, PP:1–1, 09 2019. DOI: 10.1109/MCSE.2019.2941702.

[11] Nazatul Nurlisa Zolkifli, Amir Ngah, and Aziz Deraman. Version control system: A review. *Procedia Computer Science*, 135:408–415, 2018. ISSN: 1877-0509. The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life. DOI: https://doi.org/10.1016/j.procs.2018.08.191.

[12] Björn Grüning, John Chilton, Johannes Köster, Ryan Dale, Nicola Soranzo, Marius van den Beek, Jeremy Goecks, Rolf Backofen, Anton Nekrutenko, and James Taylor. Practical computational reproducibility in the life sciences. *Cell Systems*, 6(6):631–635, 2018. ISSN: 2405-4712. DOI: https://doi.org/10.1016/j.cels.2018.03.014.

[13] Carlos Arango, Rémy Dernat, and John Sanabria. Performance evaluation of container-based virtualization for high performance computing environments. *CoRR*, abs/1709.10140, 2017.