MSE-Forschungspraktikum

Aktualisierung des preCICE-Fluent Adapters

Richard Hertrich

05. Juni 2018

Betreuer: Benjamin Rüth

Technische Universität München

Lehrstuhl für wissenschaftliches Rechnen, Fakultät für Informatik

Autor: Richard Hertrich

Betreuer: Benjamin Rüth

Inhaltsverzeichnis

1 Einleitung

1.1 Praktikumsbeschreibung

Dieser Bericht behandelt mein Forschungspraktikum am Lehrstuhl für Wissenschaftliches Rechnen der Technischen Universität München. Schwerpunkt der Arbeit ist die Aktualisierung des preCICE-Fluent Adapters.

preCICE ist eine Bibliothek, die es ermöglicht, verschiedene Löser zu koppeln, um Multiphysikphänomene zu simulieren. Ein Beispiel hierfür ist die Kopplung von Compultaional Fluid Dynamics (CFD)- mit Computational Solid Dynamics (CSD)-Lösern für Fluid Struktur Interaction (FSI). Dabei werden die Löser - in diesem Fall Ansys Fluent - über einem Adapter mit preCICE verbunden. preCICE übernimmt dann verschiedene Aufgaben um die Löser zu koppeln.

Bernhard Gatzhammer hat für seine Dissertation einen Adapter für Fluent entwickelt, allerdings haben sich seitdem sowohl die preCICE API, als auch Fluent verändert, sodass er für aktuelle Versionen nicht mehr funktionstüchtig ist.

In meinem Forschungspraktikum war es meine Aufgabe, die relevanten Veränderungen zu finden und die Funktionstüchtigkeit für die Kopplung von preCICE (Version 1.1.1) und Fluent (Version 18.2) wiederherzustellen.

1.2 Allgemeines zu preCICE

preCICE ist eine an der TU München und Universität Stuttgart entwickelte Kopplungsbibliothek für partitionierte Multi-Physik Simulationen wie Conjugate Heat Transfer [1] oder Fluid-Structure-Interaction [2]. Dabei sind die einzelnen Löser in der Lage ein Teilsystem zu simulieren. Um die Simulation eines Mehrphysikphänomens zu ermöglichen, werden unabhängige Löser über preCICE gekoppelt. preCICE verfolgt einen Black-Box Ansatz. Dabei werden die Löser lediglich minimalinvasiv über den Austausch von Randbedingungen gekoppelt. preCICE ermöglicht unter anderem die Kopplung beliebig vieler Subsysteme, unterschiedliche Gitter an der Grenzfläche und verschiedene Zeitschritte.

Dabei bildet ein Adapter - in diesem Fall der Fluent-Adapter - die Schnittstelle für die Kommunikation der Randbedingungen zu preCICE, preCICE selbst ermöglicht dann mithilfe von Data-Mapping, Zeitinterpolation und verschiedenen Kopplungsmethoden eine robuste Kopplung. [3]

2 Einarbeitung in Fluent

2.1 Allgemeines zu Fluent

ANSYS Fluent ist eine kommerzielle Software für Computational Fluid Dynamics. Für eine Simulation kann dabei entweder direkt in Fluent eine Geometrie importiert werden, oder man benutzt die von Ansys mitgelieferte Workbench, die für die Erstellung von Geometrien, Meshing und Postprocessing verwendet werden kann.

Fluent ermöglicht eine gute Anpassungsfähigkeit für individuelle Anforderungen über User-Defined-Functions (UDFs). Über diese User-Defined-Functions können von der Spezifizierung des Geschwindigkeitsprofils einer einfließenden Strömung bei einer Strömungssimulation bis zu Deformationen des Gitters viele Randbedingungen oder Geometrieparameter individuell verändert werden.

2.2 Kompatibilitätsprobleme

Zuerst musste ich feststellen, dass es nicht einfach ist, Precice und Fluent auf einem Betriebssystem zu installieren. Fluent und vor allem die Workbench benutzt auf Linux viele Bibliotheken, die auf den meisten Populären Linux Distributionen wie Ubuntu oder Fedora nicht vorinstalliert sind, und zum Teil auch nicht direkt über die Packagemanager installiert werden können.

Ansys unterstützt offiziell die Linux Distributionen RedHat Enterprise und das freien Bestriebssystem Centos 7.[4] In Centos 7 sind die benötigten Bibliotheken schon vorhanden und Ansys Produkte können einfach installiert werden. Allerdings hat Centos den großen Nachteil, dass der Packagemanager "yum" oft nur sehr alte Versionen an Software bereitstellt. Das hat zur Folge, dass viele preCICE Abhängikeiten (z.B. Boost, GCC) manuell installiert werden müssen. Insgesamt ist es jedoch deutlich einfacher Centos für preCICE zu benutzen als Fluent auf beispielsweise Ubuntu zu installieren.

Zusammenfassend kann man also sagen, dass für die Benutzung von Fluent auf Linux erheblichen Aufwand oder eine große Einschränkung bei der Wahl des Betriebssystems bedeutet.

2.3 Erste CFD Simulationen mit Fluent

Nach einer erfolgreichen Fluent Installation, ist es vergleichsweise einfach eine erste Strömungssimulation aufzusetzen und zu rechnen. Im Internet findet man dazu viele Video-Tutorials, die teilweise sogar die CAD-Modelle bereitstellen, sodass man den Workflow für simple Simulationem leicht lernen kann.

In meinem Fall habe ich eine umströmte Sonde simuliert. Abbildung 1 zeigt dabei die Konturen des Drucks der stationären Strömung. [5]

2.4 User Defined Functions in Fluent

Über UDFs kann komplexes dynamisches Verhalten in Fluent eingebaut werden. Der preCICE-Fluent Adapter nutzt besipielsweise UDFs um dynamische Randbedingungen aufzubringen, die von preCICE vorgegeben werden. Umgekehrt werden UDFs auch verwendet, um preCICE das lesen von Simulationsrgebnissen aus Fluent zu erlauben. Da der Adapter mehrere UDFs kombiniert und UDFs am Anfang nicht besonders intuitiv erscheinen ist es hilfreich sich zuerst das Konzept anhand von einfachen Beispielen

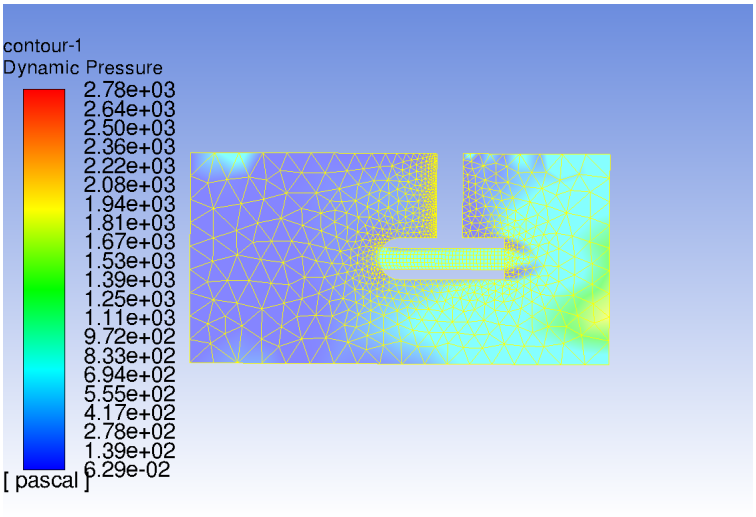

Abbildung 1: Druck in der Luft beim Umfließen einer Sonde

zu veranschaulichen. So versteht man wie auf verschiedene Daten zugreift und wie man diese verändern kann.

UDFs bestehen aus Code in der Sprache C und benutzen von Fluent vorgefertigte Macros. Um UDFs in eine Simulation einzubiinden gibt es 2 Möglichkeiten: Den Code zu kompilieren und die erstellte Bibliothek laden oder den Code während der Laufzeit interpretieren. Interpretierte UDFs haben den Vorteil, dass sie portabel zwischen verschiedenen Fluent Versionen sind, während kompilierte UDFs neu kompiliert werden müssen. Allerdings sind kompilierte UDFs schneller bei der Berechnng und manche UDF-Makros nur für kompilierte UDFs nutzbar.

Nachdem die UDFs kompiliert bzw. interpretiert wurden, findet man diese je nach Macro in verschiedenen Menüs zu Randbedingungen oder unter User-Defined-Function-Hooks. Dort können die UDFs ausgewählt werden, um an bestimmten Gebieten oder Zeitpunkten ausgeführt zu werden. [6]

2.4.1 Beispiel: Parabelförmiges Geschwindigkeitsprofil

Eine der häufigsten Anwendungen von UDFs ist ein parabelförmiges Geschwindigkeitsprofil. Ein solches Geschwindigkeitsprofil findet beispielsweise bei Rohrströmungen als Dirichlet-Randbedingung häufig Anwendung. Über das DEFINE_PROFILE Makro kann dafür eine UDF erstellt werden. UDFs dieser Art können sowohl kompiliert, als auch interpretiert werden. [6]

Der folgende Code kann genutzt werden, um wie in Abbildung 2 und 3 ein parabelförmiges Geschwindigkeitsprofil als Randbedingung festzulegen.

```
#include "udf.h"
DEFINE_PROFILE(inlet_y_velocity, thread, position)
{
        real x[ND_ND];
        real w;
        face_t f;
        w=0.05;
        begin_f_loop(f,thread) /*Loop over all nodes of the zone*/
        {
                F_CENTROID(x,f, thread); /*get coordinates*/

                F_PROFILE(f, thread, position)= 1000*(-x[1]*x[1]+w*x[1]); /*wri
        }
        end_f_loop(f,thread)
}
```

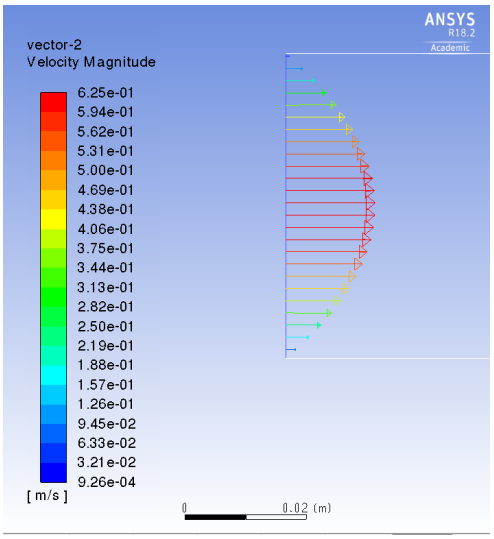

Abbildung 2: Per UDF erzeugtes parabelförmiges Geschwindigkeitsprofil

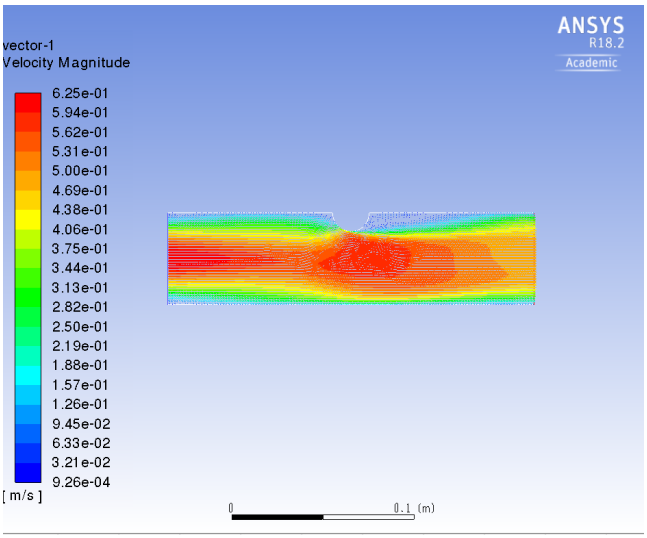

Abbildung 3: Gesamte Simulation mit Parabelförmigen Geschwindigkeitsprofil

2.4.2 Beispiel: Geometrieverformung

Eine weitere UDF, bei der die Funktionsweise deutlich wird, kann mit dem DEFINE_GEOM Makro erstellt werden. Dabei werden die Gitterpunkte der gewählten Bereiche nach der in der UDF beschriebenen Geometrie verschoben. Da das eine größere Veränderung der Simulation darstellen kann, können Geometrie-UDFs nur als kompilierte UDF verwendet werden. Normalerweise werden die Quelldateien direkt in der Fluent GUI ausgewählt und der Buildprozess gestartet.

Vor der Verformung lag ein quadratisches Hierbei ist zu beachten, dass bei alleiniger Verformung des Bodens das innere Gitter nicht verändert wird (Abbildung 4). Um auch nach der Geometrieverformung ein gleichmäßiges Gitter zu erhalten ist es also sinnvoll das gesamte innere Gebiet so zu verformen, dass am Rand die gewünschte Verformung auftritt (Abbildung 5), jedoch das gesamte Gitter mitverzerrt wird. Abbildung 6 und 7 zeigen Pfadlinien der Simulation einer lid-driven-cavity für beide Fälle.

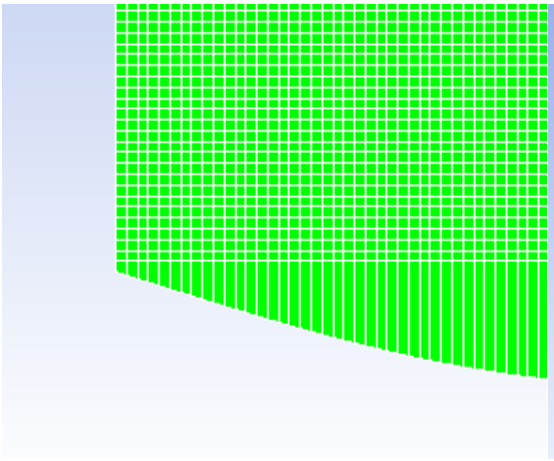

Abbildung 4: Schlecht konditionierte Zellen nach Verformung des Bodens

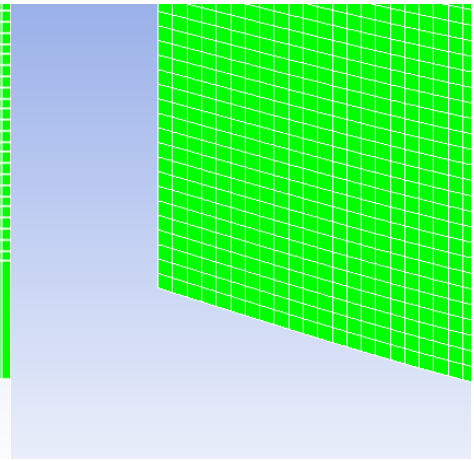

Abbildung 5: Gleichmäßiges Gitter nach Verformung des gesamten Körpers

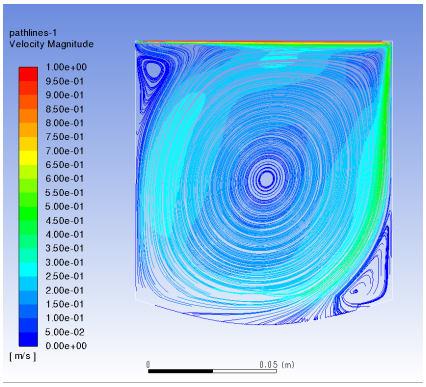

Abbildung 6: Pfadlinien in einer Lid driven cavity, bei der lediglich der Boden verformt wurde

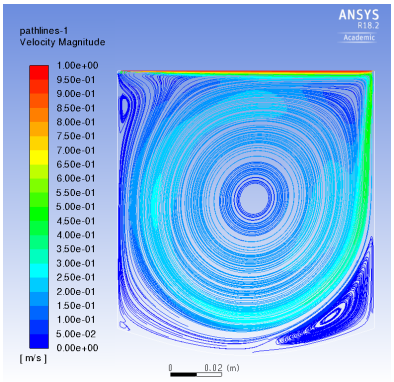

Abbildung 7: Pfadlinien nach Verformung des gesamten inneren Gitters

3 Precice Fluent Adapter

3.1 Allgemeine Funktionsweise

Der Precice Fluent Adapter für FSI besteht aus mehreren UDFs, die auf die preCICE API zugreifen. Da man externe Bibliotheken nicht direkt mit dem Compiler in der Fluent GUI verlinken kann, muss der Adapter extern kompiliert werden. Dazu kann man die ansonsten automatisch generierten Makefiles einer anderen UDF anpassen und den Adapter von der Kommandozeile mit GCC kompilieren. Dabei ist zu beachten, dass die Makefiles von der Fluent Version abhängen. Außerdem ist es dabei wichtig die Ordnerstruktur zu beachten, da man in Fluent nur eine UDF laden kann, die relativ zum Case-File (die Datei, in der die Simulation gespeichert ist) im richtigen Verzeichnis liegt. [7]

3.2 Veränderungen

Zuerst musste ich feststellen, dass die Fluent-internen Header-Dateien vom Compiler nicht gefunden werden. Das lag daran, dass Fluent die Ordnerstruktur verändert hat und die mit der Ursprünglichen Version des Adapters gelieferten Makefiles gehörten zu einer bedeutend älteren Fluent-Version. Außerdem wurden in der preCICE API Funktionen umbenannt, verändert und automatisiert, die der Adapter somit nicht mehr benutzen kann.

Das Problem der nicht gefundenen Header konnte durch Benutzung der passenden Makefiles gelöst werden. Die Makefiles stammen von Fluent und müssen mit jeder neuen Fluent Version aktualisiert werden. Bei einer Kompilierung über die Fluent GUI werden diese automatisch erstellt. Da die Makefiles von UDFs bis auf die Quelldateien

und verlinkten Bibliotheken gleich sind, konnten die Makefiles aus den Beispielen in 2.4 für den preCICE Fluent Adapter mit kleinen Anpassungen übernommen werden.

Im Bezug auf die Schnittstellen mit preCICE wurden die Funktionen auf die neuen Namen und Argumente angepasst. Lediglich die Anzahl der Nodes an der Grenzfläche muss noch automatisiert bestimmt werden. Das wäre eventuell über die Fluent-Makros zum Datenzugriff möglich.

Zum Ende meines Praktikums habe ich es geschafft den Adapter erfolgreich zu kompilieren. Es war auch möglich die dadurch entstandene Bibliothek in ein Fluent-Case zu laden. Allerdings waren die einzelnen Funktionen in den dafür vorgesehenen Menüs nicht zu finden.

4 Ausblick und Diskussion

Da die Installation von Fluent und die Einarbeitung mehr Zeit in Anspruch genommen haben als gedacht, war es im Rahmen dieses Forschungspraktikums für mich nicht möglich den Adapter wieder funktionstüchtig zu machen. Allerdings wurden wurden . Mithilfe des ausgearbeiteten technischen Reports sollte jemand schnell auf meinen Wissensstand kommen, um die Arbeit am Adapter weiterführen zu können.

Um zu verstehen, warum die UDFs nicht in den Menüs auftauchen, wäre es hilfreich, sich allgemein besser mit den jeweiligen UDFs zu beschäftigen. Der damit einhergehende Aufwand ist jedoch schwer abzusehen und übersteigt Rahmen dieses Praktikums.

Für die Zukunft sollten die Makefiles automatisch aus der jeweiligen Fluent Version geholt werden, damit der Adapter zwischen verschiedenen Fluent Versionen kompatibel wird. Eine deutlich schönere Variante als die Precice Bibliothek zu kopieren und den Python-Pfad explizit anzugeben wäre außerdem eine Suche über die Umgebungsvariable PRECICE_ROOT und eine automatische Suche für dynamische Bibliotheken. Die Quelldateien und verlinkten Bibliotheken werden dann mithilfe eine Konfigurationsdatei spezifiziert und automatisch eingebunden.

Für einen neuen Nutzer wäre es somit möglich den Adapter zu benutzen ohne sich mit der Funktionsweise auseinderzusetzen.

Literatur

[1] L. Cheung Yau: Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE, Master's thesis, Institut für Informatik, Technische Universität München, December 2016.

[2] FSI-tutorial, URL: https://github.com/precice/precice/wiki/FSI-tutorial, Zugriffsdatum: 05.07.2018

[3] H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann: preCICE - A Fully Parallel Library for Multi-Physics Surface Coupling. Computers and Fluids, Elsevier, 141, 250–258, 2016

[4] ANSYS Platform Support by Application Release 18.2, URL: https://www.ansys.com/-/media/ansys/corporate/files/pdf/solutions/it-professionals/platform-support/platform-support-by-application-182.pdf?la=de-de, Zugriffsdatum 03.07.2018

[5] Mike Foster: Introduction to ANSYS Fluent, URL: https://www.youtube.com/watch?v=3mJnql_ncuw, Zugriffsdatum 03.07.2018

[6] Dokumentation der Ansys Produkte, Version 18.2

[7] Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions,Bernhard Gatzhammer, 2015, PhD Thesis, Institut für Informatik, Technische Universität München, 2015