

CFD-Programmier-Seminar

Projekt 2: Raumoperator

In dieser Aufgaben soll das Herzstück des Finite-Volumen-Verfahrens umgesetzt werden, nämlich die Flussberechnung, die aus der Rotation der Zustände in die lokalen Koordinatensysteme an den Kanten, der Berechnung des numerischen Flusses, der Rückrotation der erhaltenen Ergebnisse an jeder Kante besteht.

Theoretischer Hintergrund

Der Raumoperator des Finite-Volumen-Verfahrens ist

$$R(u) = \frac{1}{V} \oint_{\partial V} \vec{f}(u) \cdot \vec{n} dS.$$

Wir nehmen nun an, dass die zu betrachtenden Kontrollvolumina n-Ecke mit geraden Kanten sind und können so das Oberflächenintegral durch die Summe der Integrale über die M Kanten des jeweiligen Kontrollvolumens ersetzen:

$$R(u) = \frac{1}{V} \sum_{k=1}^M \int_k \vec{f}(u) \cdot \vec{n} dL$$

Das Integral über die Kante approximieren wir numerisch durch die Mittelpunktsregel $\int_a^b f(x) dx \approx (b-a)f\left(\frac{b+a}{2}\right)$ und erhalten so

$$R(u) = \frac{1}{V} \sum_{k=1}^M L_k \vec{f}(u) \cdot \vec{n}$$

Der Fluss $\vec{f}(u)$ ist

$$\vec{f} = \begin{pmatrix} f_1(u) \\ f_2(u) \end{pmatrix} \text{ mit } \vec{f}_1(u) = \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ v_1(E + p) \end{pmatrix} \text{ und } \vec{f}_2(u) = \begin{pmatrix} \rho v_2 \\ \rho v_2 v_1 \\ \rho v_2^2 + p \\ v_2(E + p) \end{pmatrix}$$

Wir können $R(u)$ also weiter ausschreiben und erhalten

$$R(u) = \frac{1}{V} \sum_{k=1}^M L_k \left(n_1 \vec{f}_1(u) + n_2 \vec{f}_2(u) \right).$$

Unter Ausnutzung der Rotationsinvarianz der Eulergleichungen können wir $R(u)$ in die effiziente Form

$$R(u) = \frac{1}{V} \sum_{k=1}^M L_k T^{-1} \vec{f}_1(Tu)$$

bringen, bei der wir nur noch den Fluss $\vec{f}_1(u)$ durch die numerische Flussfunktion $g(u_L, u_R)$ approximieren müssen (siehe Aufgabe 1). Die Matrizen T und T^{-1} sind durch

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & n_1 & n_2 & 0 \\ 0 & -n_2 & n_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ und } T^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & n_1 & -n_2 & 0 \\ 0 & n_2 & n_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

gegeben.

Implementierung in CFDFV

Da der Code CFDFV für eine Shared-Memory-Parallelisierung vorbereitet ist, muss die Implementierung der Flussberechnung anders erfolgen als es an sich logisch erscheinen würde. Der Grund dafür ist, dass wenn das Flussintegral über jede Seite direkt in eine elementbasierte Variable ($R(u)$) gespeichert würde, es u. U. zu einer sog. Race Condition käme, bei der zwei Prozesse auf einmal versuchen, die gleiche Variable zu ändern. Da die Flussberechnung seitenweise abläuft, ist es absolut möglich, dass zwei unterschiedliche Prozesse gleichzeitig den Fluss über Seiten berechnen, die an einem gemeinsamen Element liegen.

Die Implementierung erfordert daher, dass der Fluss $L_k T_L^{-1} g(T_L u_L, T_L u_R)$ zunächst auf der Seite gespeichert wird. Hierfür steht die Variable `side%flux` zur Verfügung. Die Summe wird aus technischen Gründen erst im Rahmen der Zeitdiskretisierung gebildet und soll im Rahmen dieser Aufgabe nicht implementiert werden. Beachten Sie, dass das Integral auf beiden Seiten gespeichert werden muss!

Programmierarbeiten

Die folgenden Arbeitspunkte sind im Code CFDFV umzusetzen:

1. Rotation des Zustands ins lokale Koordinatensystem
2. Flussberechnung (hierfür soll die Subroutine `ConvectiveFlux` verwendet werden)
3. Rückrotation des Flusses ins globale Koordinatensystem
4. Speichern des Flussintegrals auf den Seiten

In der Datei `FluxCalculation.f90` findet die Rotation der Zustände und Flüsse, die Quadratur, sowie das Speichern auf den Rändern statt.

Hinweis: Die Funktion `MATMUL` in Fortran wäre zwar geeignet, um die Matrix-Vektor-Produkte Tu und $T^{-1}f_1$ auszuführen, jedoch wäre dies im vorliegenden Fall äußerst ineffizient, da `MATMUL` die volle Multiplikation ohne Rücksicht auf die Struktur der Matrizen ausführt. In diesem Fall ist es die effizienteste Lösung, die beiden Produkte auf Papier auszuschreiben und das erhaltene Ergebnis direkt zu programmieren.

Validierung

Wenden Sie alle Flussfunktionen auf folgende Probleme an:

- Riemannprobleme `Toro3` und `Toro4`: Diese Probleme sind identisch, wobei die Laufrichtung entgegengesetzt ist. Die Ergebnisse sollten symmetrisch sein.
- 2D-Gausspuls im Druck mit 100 x 100 Gitterzellen (kartesisch). Dieses Beispiel sollte ein rotationssymmetrisches Ergebnis liefern.