

CFD-Programmier-Seminar

Projekt 4: Rekonstruktion 2. Ordnung im Raum

Finite-Volumen-Verfahren haben ihre Daten prinzipbedingt nur als integrale Zellmittelwerte, d.h. als Polynome vom Grad null innerhalb der einzelnen Zellen vorliegen. Die Werte, die in die Flussberechnung eingehen, sind also die integralen Zellmittelwerte – dies entspricht einer Genauigkeit 1. Ordnung. Um die Ordnung und damit die Genauigkeit des Verfahrens zu erhöhen, muss der Zustand innerhalb der Zelle, der i.d.R. nicht konstant sein wird, “rekonstruiert” werden. Da innerhalb der Zelle keine ausreichenden Informationen dafür vorliegen, müssen Informationen aus den Nachbarzellen hinzugezogen werden

Theoretische Grundlagen der Raumrekonstruktion auf unstrukturierten Gittern

Die gesamte Idee, die hinter der Rekonstruktion steht, basiert auf der Idee, dass man den Zustand an jedem beliebigen Punkt *innerhalb* einer Zelle als Taylorentwicklung um den Zustand im Baryzentrum ausdrücken kann:

$$U(x, y) \approx U(x, y)_{Bary} + \nabla U(x, y)_{Bary} \cdot \Delta \vec{x} + O((\Delta \vec{x})^2)$$

An dem quadratischen Abbruchfehler kann man sofort erkennen, dass der auf diese Art und Weise berechnete Wert zweiter Ordnung genau ist. Unser Problem ist nun, dass wir in jeder Zelle den integralen Zellmittelwert kennen, den wir auch als den Wert im Baryzentrum ansetzen, jedoch ist uns der Gradient nicht bekannt. Dieser muss rekonstruiert werden, was unter Zuhilfenahme der direkten Seitennachbarn bewerkstelligt wird. Hierfür macht man den Ansatz, dass die Änderung des Zustands über eine Zellkante über den Gradienten des Zustands mal dem Vektor von Baryzentrum zu Baryzentrum ausgedrückt werden kann:

$$(\nabla U_i) \cdot \vec{r}_{ij} = U_j - U_i$$

Diese Beziehung stellt man über jede der N Zellkanten einer Zelle i auf, womit man ein überbestimmtes Gleichungssystem für den Gradienten kommt:

$$\begin{pmatrix} \Delta x_{i1} & \Delta y_{i1} \\ \Delta x_{i2} & \Delta y_{i2} \\ \vdots & \vdots \\ \Delta x_{iN} & \Delta y_{iN} \end{pmatrix} \begin{pmatrix} U_x \\ U_y \end{pmatrix} = \begin{pmatrix} U_1 - U_i \\ U_2 - U_i \\ \vdots \\ U_N - U_i \end{pmatrix}$$

Dieses überbestimmte Gleichungssystem wird mit Hilfe der Methode der kleinsten Quadrate gelöst:

Hat man ein überbestimmtes Gleichungssystem der Form

$$Cx = b,$$

so multipliziert man zunächst die Gleichung mit der Transponierten von C :

$$C^T Cx = C^T b$$

Somit kann man direkt nach x lösen:

$$x = (C^T C)^{-1} C^T b = Wb$$

Die Matrix W ist in einer besonderen Form im Code gespeichert, und zwar so, dass man den Gradienten einer Zustandsvariablen folgendermaßen ausdrücken kann:

$$\nabla U_i = \sum_{j=1}^N \vec{w}_j (U_j - U_i)$$

Die Vektoren \vec{w}_j sind dabei alles, was von W übrig bleibt. Diese Summe muss für jede Zelle und für jede Variable durchgeführt werden. Aus Stabilitäts- und Effizienzgründen wird diese Prozedur in primitiven Variablen durchgeführt.

Die benötigte Limitierung der Gradienten wird in einer anderen Aufgabe (s. Projekt 5) durchgeführt.

Programmierarbeiten

Die folgenden Arbeitspunkte sind im Code CFDFV umzusetzen:

1. Bereitstellung der Zustandswerte auf der Zellkante der Ghostzelle: `SetBCatSides()` in der `Boundary.f90`. Siehe Randbedingungs-Aufgabe (Projekt 3).
2. Bereitstellung der Zellmittelwerte in den Ghostzellen:
 - `SetBCatBarys(CONST, MESH)` in der `Boundary.f90`:
Diese Routine setzt die Randbedingung am Baryzentrum der Ghostzelle (`aElem%bary`). Hierfür ist eine Schleife über alle Rand-Ghost-Seiten (analog zur `SetBCatSides()`) zu implementieren: `MESH%BCSides(:)`
3. Raumrekonstruktion:
Die Raumrekonstruktion findet in der Datei `SpatialReconstruction.f90` statt.
 - Zunächst müssen wir die Raumrekonstruktion für die erste Ordnung machen. In diesem Fall besteht die Raumrekonstruktion darin, dass die Seitenzustandswerte denen im Zell-Bary-Zentrum entsprechen. Dafür programmieren Sie eine Schleife über alle Elemente unter Zuhilfenahme des `MESH%Elems(:)`-Arrays und setzen Sie `aSide%pVar(:) = aElem%pVar(:)`. Der `aSide`-Pointer kann dafür mit dem `aElem%firstSide` auf die erste Seite des Elements gesetzt werden. Mit dem `aSide%nextElemSide`-Pointer können Sie dann über alle Seiten des Elements laufen. Für Dreiecke haben Sie nach drei Schleifendurchläufen alle Seitenzustände `aSide%pVar(:)` gesetzt. Für Vierecke benötigen Sie vier durchläufe. Allgemein können Sie diese Schleife mit einer `DO WHILE(ASSOCIATED(aSide))`-Abfrage programmieren.
 - Lösung des Least-Squares Problems:
Dazu implementieren Sie eine Schleife über alle Seiten `MESH%Sides(:)` und berechnen die Differenzen der primitiven Variablen

$$pDiff = aSide\%connection\%Elem\%pVar() - aSide\%Elem\%pVar().$$
 Lösen Sie nun mit Hilfe der Koeffizienten aus der W -Matrix `aSide%w(1)` das Least-Square Problem, indem Sie die Gradienten `aElem%u_x` und `aElem%u_y` aufsummieren. Die Gradienten berechnet sich dabei aus

$$aElem\%u_x = aElem\%u_x + aSide\%w(1) * pDiff \text{ und}$$

$$aElem\%u_y = aElem\%u_y + aSide\%w(2) * pDiff.$$
 Für das Nachbar Element `aSide%connection%Elem` werden die Gradienten entsprechend subtrahiert

$$aElem\%u_x = aElem\%u_x - aSide\%connection\%w(1) * pDiff \text{ und}$$

$$aElem\%u_y = aElem\%u_y - aSide\%connection\%w(2) * pDiff.$$
 Achten Sie bitte darauf, dass die Vorzeichen stimmen. An der Zellkante ändern sich die Vorzeichen der Gradienten! Da der Gradient immer die Änderung des Zustandes in die Raumrichtung angibt, die vom Baryzentrum wegführt, wechselt an der Zellkante das Vorzeichen der Steigung.
 - Limitieren der Gradienten. Dies ist Teil der Limiter-Aufgabe (Projekt 5).
 - Rekonstruktion der Zustände auf den Seiten des Elements: Programmieren Sie eine Schleife über das `MESH%Elems(:)`-Array und in jedem Element eine Schleife über alle Seiten des Elements mit dem `aElem%FirstSide`-Pointer. Berechnen Sie den Wert der Zustände mittels der Vektoren vom Bary-Zentrum zum Mittelpunkt der jeweiligen Seite, $dx = aSide\%GP(X_DIR)$ und $dy = aSide\%GP(Y_DIR)$.

Mathematisch ergibt sich der Zustand dann aus den Gradienten zu

$$aSide\%pVar = aElem\%pVar + dx \frac{du}{dx} + dy \frac{du}{dy}.$$

Implementieren Sie dies wie folgt:

```
aSide%pvar(:) = aElem%pvar(:) + dx * aElem%u_x(:) + dy * aElem%u_y(:)
```

Validierung

- Untersuchen Sie, welchen Einfluss die Raumdiskretisierung auf das Ergebnis hat. Vergleichen Sie dabei jeweils das Verfahren 1. Ordnung im Raum mit dem Verfahren 2. Ordnung im Raum. Zunächst soll dabei untersucht werden, wie sich die Verfahren auf gleichem Gitter verhalten. Im Anschluss jedoch soll mit dem Verfahren 1. Ordnung durch Verfeinerung des Gitters die Genauigkeit 2. Ordnung erreicht werden. Ein Vergleich der jeweiligen Rechenzeiten dieser Rechnungen gibt im Endeffekt an, was man durch Verfahren höherer Ordnung gewinnt.
- Untersuchen Sie bei den stationären Testfällen, welchen Einfluss die 2. Ordnung auf das Konvergenzverhalten hat (Residuen-Verlauf). Untersuchen Sie bei der Profilmströmung auch den C_l und C_d -Verlauf. Welche Unterschiede lassen sich hier beobachten?
- Führen Sie eine Konvergenzanalyse für das Sinuswellen-Problem durch. Berechnen Sie dabei das Problem mit unterschiedlicher Gitterauflösung (verfeinern Sie ausgehend von der Ausgangsauflösung die räumliche Diskretisierung in x- und y-Richtung mit dem Faktor 2). Berechnen Sie dann die Konvergenzordnung für die Sinuswelle nach der folgenden Formel mit dem L_1 - oder L_2 -Fehler:

$$n = \frac{\log\left(\frac{E_1}{E_2}\right)}{\log\left(\frac{h_1}{h_2}\right)}$$

wobei E der Fehler und h der gemittelte Gitterabstand ist.

Hinweis: Achten Sie hier bitte darauf, dass Sie für die Konvergenzanalyse die Zeitdiskretisierung zweiter Ordnung benutzen.

- Warum funktioniert die Konvergenzanalyse für den SOD-Testfall nicht mehr?

Verwenden Sie hierzu die folgenden Testfälle:

- 1D-Riemannprobleme
- Sinuswelle
- Richtmyer-Meshkov-Instabilität
- Profilmströmung