Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. M. Ohlberger, M. Drohmann

16.4.2009

Aufgaben zum Praktikum

Numerik Partieller Differentialgleichungen II

SS 2009 — Blatt 1

Abgabe: 30.4.2009 per Email

Finite Differenzen Verfahren in 1D

Im ersten Aufgabenblatt wollen wir uns mit Finite Differenzen Verfahren zur Approximation von skalaren Erhaltungsgleichungen in einer Raumdimension beschäftigen. Ziel der Aufgabe ist es, die ersten drei Aufgaben aus dem Praktikum im WS 2008/2009 zu wiederholen, damit alle Teilnehmer auf den gleichen Stand gebracht werden. Für Teilnehmer, die diese Aufgaben im WS 2008/2009 nicht bearbeitet haben, wird eine Beispielimplementierung zu den Aufgaben 1 und 2 zur Verfügung gestellt. Die Aufgabe 3 soll eigenständig bearbeitet werden.

Aufgabe 1 (Klassenkonzept)
Wir betrachten eine skalaren Erhaltungsgleichung in einer Raum-Dimension

$$\begin{aligned} \partial_t u(x,t) + \partial_x f(u(x,t)) &= 0 \quad \text{in } \Omega \times [0,T] \\ u(x,0) &= u_0(x) \text{ in } \Omega \\ u(x,t) &= u_{dir}(x,t) \text{ in } \Gamma_{dir}. \end{aligned}$$

Hierbei bezeichnet $T > 0$ die Endzeit, $\Omega = [a,b] \subset \mathbb{R}$ ein beschränktes Intervall, $f \in C^1(\mathbb{R})$ die Flussfunktion, u_0 stückweise stetige, beschränkte Anfangsdaten und u_{dir} beschränkte Dirichlet-Randdaten, die auf dem Einfluss-Rand $\Gamma_{dir}(t) := \{x \in \partial\Omega | f'(u)n < 0\}$ mit äußerer Einheitsnormalen n vorgeschrieben werden. Der übrige Rand $\partial\Omega \backslash \Gamma_{dir}$ ist Ausfluss-Rand, auf dem keine Bedingungen vorgegeben werden.
Die Diskretisierung dieser Gleichung soll mittels expliziten Finite Differenzen Verfahren realisiert werden. Hierzu sei K die Anzahl der Zeitintervalle, $\Delta t := T/K$ die Zeitschrittweite und es bezeichnen $t^k := k\Delta t, k = 0, \ldots, K$ die diskreten Zeitpunkte. Es seien die N Gitterpunkte im Ort gegeben als $G := \{x_n\}_{n=1}^N$ mit $a = x_1 < \ldots < x_N = b$. Das Gitter G definiert den Raum der reellwertigen Gitterfunktionen $V_h := \{v_h : \{x_n\}_{n=1}^N \to \mathbb{R}\}$. Durch das finite Differenzen Verfahren wird eine Sequenz von diskreten Funktionen $u_h^k \in V_h$ generiert, indem eine Anfangsprojektion $u_h^0 := P_h(u_0)$ durchgeführt wird und anschließend sequenziell $u_h^{k+1} := u_h^k + \Delta t L_h(u_h^k)$ mit dem Ortsdiskretisierungsoperator $L_h : V_h \to V_h$ berechnet wird. Der Projektionsoperator und Ortsdiskretisierungsoperator sind punktweise definiert durch $(P_h(u_0))(x_n) := u_0(x_n)$ und $(L(v_h))(x_n) :=$

$-\left(\frac{1}{\Delta x_n}[g(v_n, v_{n+1}) - g(v_{n-1}, v_n)]\right)$ mit numerischem Fluss g, Ortsschrittweite Δx_n und je nach Randtyp sinnvoll definierten Werten v_0 und v_{N+1}.

a) Entwerfen Sie ein objektorientiertes Klassenkonzept, d.h. geben Sie für die untenstehenden (und eventuell weiteren) notwendigen Klassen die Schnittstelle an, d.h. Methoden mit Argumenten und kurzer Funktionsbeschreibung. Wie hängen die Klassen zusammen und wie kann das durch Konstruktoren und Membervariablen berücksichtigt werden? Welche Methoden und Argumente können als const deklariert werden?

b) Schreiben Sie in wenigen Zeilen Pseudo-Code ein Hauptprogramm, das mit Hilfe der Klassen die Finite Differenzen Simulation durchführt.

Einige wichtigen Strukturen sind die folgenden:

Model: Eine Sammlung der Datenfunktionen der Differentialgleichung, d.h. Klasse, welche Punktauswertung der Flussfunktion, Anfangs- und Randwerte erlaubt.

Grid: Eine Klasse, die die Geometrieinformationen verwaltet, d.h. Iteration über die Punkte, Zugriff auf Punkte und Punktabstände, Information darüber, ob ein Punkt Randpunkt ist, äußere Normalen zu Randpunkten.

DiscreteFunction: Eine Klasse, die eine diskrete Funktion $v_h \in V_h$ darstellt, d.h. einen Freiheitsgrad für jeden Gitterknoten besitzt, Setzen und Auslesen von Funktionswerten erlaubt, und die Ausgabe der Funktionswerte auf den Bildschirm oder in eine Textdatei ermöglicht.

Projection: Eine Klasse, die eine Projektion der Anfangsdaten $v_h := P_h[u_0]$ durchführt.

DiscreteSpaceOperator: Eine Klasse, die die Anwendung des diskreten Ortsoperators $v_h = L_h[w_h]$ realisiert.

NumericalFlux: Eine Klasse, die einen numerischen Fluss repräsentiert.

Aufgabe 2 (Implementation)
Erstellen Sie basierend auf dem Klassenkonzept von Blatt 1 entsprechende Implementationen der Klassen in einer Datei `finite_difference.hh` und ein entsprechendes Hauptprogramm in `finite_difference.cc`. Die folgenden Spezifikationen konkretisieren weitere Anforderungen an die Programmteile:

- Das Hauptprogramm soll die Intervallgrenzen a und b, die Anzahl der Punkte N, die Endzeit T, Anzahl der Zeitintervalle K und einen Ausgabe-Dateinamen als Kommandozeilen-Argumente übergeben bekommen, die Simulation durchführen und die Lösung zur Endzeit in eine Datei ausgeben, die man dann z.B. mit gnuplot visualisieren kann.

- Als Modellgleichungen sollen die folgenden implementiert werden:

$$f(u) = \frac{1}{2}u^2 \tag{1}$$

$$u_0(x) = \begin{cases} -\frac{1}{2}x + \frac{3}{2} & \text{für } x \in [1,3] \\ 0 & \text{sonst} \end{cases} \tag{2}$$

$$u_{dir}(x,t) = 0. \tag{3}$$

  Zur Verifikation der Numerik lautet die exakte Lösung auf dem Gebiet $\Omega = [a,b] = [0,5]$ mit Endzeit $T = 4$

$$u(x,t) = \begin{cases} 0 & \text{für } x < 1 \\ \frac{1}{t}(x-1) & \text{für } 1 \le x < x_s(t) \\ \frac{1}{2-t}(x-3) & \text{für } x_s(t) \le x < 3, t \in [0,2) \\ 0 & \text{für } 3 \le x, t \in [0,2) \\ 0 & \text{für } x_s(t) \le x, 2 \le t \end{cases}$$

  mit der Kurve

$$x_s(t) = \begin{cases} 1+t & \text{für } 0 \le t < 2 \\ 1+\sqrt{2t} & \text{für } 2 \le t. \end{cases}$$

- Erstellen Sie für das Gitter eine Implementation eines äquidistanten Gitters.
- Erstellen Sie für den numerischen Fluss sowohl eine Implementation des Lax-Friedrichs-Flusses als auch des Engquist-Osher Flusses (einfache Version für f' ohne Vorzeichenwechsel). Leiten Sie hierzu diese beiden Implementations-Klassen von einer abstrakten Schnittstellen-Klasse **NumericalFlux** ab.

Bei der Programmierung der Komponenten soll insbesondere Wert auf Dokumentation zu den Klassen und Methoden gelegt werden. Fügen Sie hierzu entsprechende C++-Kommentarzeilen mit Erläuterungen ein.

Aufgabe 3 (Experimente)
Mit Hilfe Ihrer Implementation des Finite Differenzen Verfahrens sollen einige experimentelle Einsichten gewonnen werden. Gehen Sie dazu wie folgt vor.

a) Erweitern Sie ihr Programm so, dass Sie den *experimental order of convergence (EOC)* des Verfahrens für die Endposition $x(T)$ bestimmen können. Was erhalten Sie asymptotisch für kleiner werdende Δt?

b) Bestimmen Sie für den Engquist-Osher-Fluss und gegebenem $N = 100$ empirisch ein K so dass das Verfahren eine plausible Lösung produziert. Erfüllen die zugehörigen Δx und Δt die CFL Bedingung? Wie äußert sich ein Verletzen der CFL-Bedingung, d.h. zu große Zeitschrittweite in der Lösung?

c) Visualisieren Sie eine numerische Lösung zu Zeiten $t = 0$, $t = 1$, $t = 2$ und $t = 4$ und beschreiben Sie qualitativ die Bewegung des Maximums.

d) Visualisieren Sie unter Verwendung des Lax-Friedrichs-Flusses die Lösung zum Endzeitpunkt für Gitter verschiedener Feinheit und beschreiben Sie die qualitative Änderung.

e) Vergleichen Sie die Lösungen zum Endzeitpunkt unter Verwendung des Lax-Friedrichs-Flusses und des Engquist-Osher-Flusses.

f) Führen Sie Laufzeitmessungen durch, um den Effekt der verschiedenen Optimierungslevel (-O1 bis -O3) bei Verwendung des CRTP gegenüber virtuellen Funktionen zu untersuchen.

g) Eine diskrete $L^1(\Omega)$-Norm für Gitterfunktionen $v_h \in V_h$ kann definiert werden durch $\|v_h\|_h := \sum_{n=1}^{N} \Delta x_n |v_n|$. Erstellen Sie eine Tabelle der Fehler von numerischer zu Exakter Lösung $e_h := \left\| u_h^K - P_h(u(\cdot, T)) \right\|_h$ für Gitter mit zunehmender Verfeinerung, d.h. Halbierung der Δx_n . Ermitteln Sie als weitere Spalte in der Tabelle den *experimental order of convergence (EOC)* zwischen jeweils zwei aufeinanderfolgenden Verfeinerungsstufen. Seien u_h^K und $u_{h'}^{K'}$ die Lösungen für gegebenes und verfeinertes Gitter. Damit ist der EOC definiert als

$$EOC := \frac{\log(e_h / e_{h'})}{\log 2}.$$