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

8.11.2007

Aufgaben zum Praktikum

Wissenschaftliches Rechnen

WS 2007/2008 — Blatt 2

Aufgabe 1 (Objektorientiertheit in C++)
Gegeben ist das Programm `reference_elements.cc` auf der Webseite der Veranstaltung. Das Hauptprogramm erstellt ein allgemeines Polygon in 2D, ein Referenz-Dreieck und ein Referenz-Quadrat und ruft verschiedene Methoden auf den Elementen auf. Die gewünschten Ausgaben des Programms sind im entsprechenden Kommentarblock gegeben.

(a) Überlegen Sie ein auf Vererbung basiertes Klassenkonzept, welches die 3 Klassen (möglichst effizient) abbildet. Geben Sie insbesondere Vererbungs-Beziehungen an, welche Methoden als virtuell gewählt werden müssen, welche Member-Variablen in den Klassen deklariert werden und welche Funktionsargumente als const deklariert werden sollten.

(b) Implementieren Sie ihr Konzept in einer Datei `reference_elements.hh`, so dass das Programm lauffähig ist und die gewünschten Ausgaben liefert.

Aufgabe 2 (Finite Volumen Verfahren)
Gegeben ist folgende nichtlineare hyperbolische Erhaltungsgleichung auf einem Interval $\Omega = [x_{min}, x_{max}] \subset \mathbb{R}$:

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

wobei $\Omega_{dir} := \{x \in \partial\Omega | f(u) \cdot n(x) < 0\}$ den Dirichlet-Rand bezeichnet ($n(x)$ ist äussere Einheitsnormale) und der übrige Rand $\partial\Omega \backslash \partial\Omega_{dir}$ den Ausflussrand ohne weitere Randbedingung darstellt. Es soll eine objektorientierte Diskretisierung dieser Gleichung mit einem expliziten Finite Volumen Verfahren erster Ordnung realisiert werden, also $u_h^0 = P(u_0)$ geeignete Projektion der Anfangsdaten und $u_h^{k+1} = u_h^k - \Delta t L_h^k(u_h^k)$ mit entsprechendem Orts-Diskretisierungsoperator L_h^k. Es sollen folgende Programmkomponenten entstehen und miteinander interagieren:

Model: Eine Klasse, die den Zugriff auf die Datenfunktionen der Problemstellung erlaubt, d.h. die Funktionen f, u_0, u_{dir}. Als spezielles Beispiel soll die Burgers-Gleichung

mit homogenen Randbedingungen implementiert werden, also $f(u) = u^2$ und $u_{dir} = 0$. Als Anfangsbedingung soll eine Dreiecksverteilung implementiert werden $u_0(x) = -4x + 2$ in $[0.25, 0.5]$ und 0 sonst.

Grid: Eine Klasse, die das Gitter repräsentiert. Als spezielles Beispiel soll ein äquidistantes Gitter auf Ω mit n Zellen entstehen.

DiscreteFunction: Eine Klasse, die eine stückweise konstante diskrete Funktion repräsentiert, und elementweise Zugriff auf die Koeffizienten erlaubt.

NumericalFlux: Eine Klasse, die einen numerischen Fluss repräsentiert. Als Beispiel soll der Lax-Friedrichs-Fluss realisiert werden.

SpaceDiscOperator: Eine Klasse, die den Ortsdiskretisierungoperator $L_h^k(u_h^k)$ auswertet.

Zu dieser Vorgabe sollen folgende inhaltliche Aufgaben bearbeitet werden, (*) bezeichnet eine Zusatzaufgabe:

(a) Überlegen Sie, welche Abhängigkeiten zwischen den Klassen bestehen, und wie dies in den Konstruktoren/Membervariablen der Klassen berücksichtigt werden kann.

(b) Geben Sie an, welche Methoden die Klassen besitzen müssen, um das angegebene Verfahren zu realisieren. Zwecks späterer Austauschbarkeit der Komponenten sollen die verschiedenen Klassen kein Wissen über die speziellen Vereinfachungen der anderen Klassen verwenden. D.h. das **Grid** soll eine solche Schnittstelle bekommen, die auch zu allgemeineren nicht-äquidistanten Gitterpunkten passt. Ähnlich soll der **NumericalFlux** nachher auch durch andere numerische Flüsse ersetzt werden können.

(c) In `finite_volume.cc` auf der Kursseite ist eine Anfangs-Implementation vorgegeben. Implementieren Sie die übrigen Klassen und erzeugen Sie ein Programm, das $x_{min}, x_{max}, n, T, \Delta t, \lambda$ und einen Dateinamen als Kommandozeilen-Parameter einliest, Objekte initialisiert, die Anfangsdaten projeziert, die FV-Simulation in einer Schleife über die Zeit durchführt, und die Lösung zum Endzeitpunkt in gnuplot-tauglichem Textformat speichert.

(d) Bestimmen Sie durch Programmläufe für $n = 100$, $T = 0.5$ genügend kleine Parameter Δt und λ, so dass das Verfahren stabil läuft. Wiederholen Sie dies für die Gittergröße $n = 400$ und vergleichen Sie die Simulationsergebnisse. Warum sollten Δt und λ nicht beliebig klein gewählt werden?

(*) Implementieren Sie den Enquist-Osher-Fluss als numerischen Fluss, so dass dieser dieselbe Schnittstelle wie der bisherige numerische Fluss besitzt. Implementieren Sie hierzu die Klasse **NumericalFlux** als abstrakte Basisklasse, und leiten Sie entsprechend die Klassen **LFNumericalFlux** und **EONumericalFlux** hiervon ab. Vergleichen Sie die numerischen Ergebnisse mit denen aus der vorigen Teilaufgabe.