Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Christian Engwer, MSc. Marcel Koch

16.11.2017

Übung zur Vorlesung

Wissenschaftliches Rechnen

WS 2017/18 — Blatt 6

Abgabe: 23.11.2017, 10:00 Uhr, Briefkasten 111
Code zusätzlich per e-mail an `marcel.koch@uni-muenster.de`

Aufgabe 1 (SIMD Instruktionen) (4 Punkte)

Auf der Vorlesungshomepage finden Sie ein Programm, welches zwei zufällig generierte Vektoren vom Typ `std::vector<std::array<Real,d>` komponentenweise miteinander multipliziert, indem die inneren d-dimensionalen Vektoren komponentenweise miteinander multipliziert werden. Diese innere komponentenweise Multiplikation ist in eine Prozedur `calc_i` ausgelagert.

(a) Übersetzen Sie die Datei `calc_i.cc` mit dem `g++-5.4` Compiler in Assembler (beispielsweise auf den Arbeitsplatzrechnern im SRA); einmal für den Fall $d = 3$ und einmal für den Fall $d = 4$. Benutzen Sie dabei die Compileroptionen aus dem mitgelieferten `Makefile`.

(b) Erklären Sie den zur Prozedur `calc_i` gehörenden Assemblercode in beiden Fällen zeilenweise (beispielsweise durch Kommentare im Assemblercode).

(c) Worin liegt der Grund für die Benutzung unterschiedlicher Assemblerbefehle für die beiden Fälle?

(d) Was können Sie daraus für unseren Code zum n-Körper Problem schließen?

Hinweis 1: Sie können zum kompilieren am einfachsten das mitgelieferte `Makefile` benutzen. Dieses erzeugt den Assemblercode der Datei `calc_i.cc` automatisch und legt ihn in der Datei `calc_i.s` ab. Führen Sie dazu den Befehl `make` im Programmverzeichnis aus.

Hinweis 2: Falls Sie Informationen bezüglich der Assemblerbefehle benötigen, hilft Ihnen das *Intel 64 and IA-32 Architectures Software Developer's Manual* weiter. Dieses ist unter `http://www.intel.com` zum Download verfügbar.

Definition 1 (Stabilität stationärer Lösungen)

Eine stationäre[1] Lösung x^* der Differentialgleichung $x' = f(x)$ heißt *stabil*, falls zu jeder Umgebung U von x^* eine Umgebung V von x^* existiert, so dass für jede Lösung der Anfangswertprobleme

$$x' = f(x), \quad x(0) = x_0 \in V$$

gilt $x(t) \in U$ für alle $t > 0$. Andernfalls heißt die Lösung *instabil*.

[1] Stationäre Lösungen sind zeitunabhängig, d.h. $x' = 0$.

Aufgabe 2 (Stabilität linearer Systeme von GDL) (5 Punkte)

(a) Betrachten Sie folgendes System linearer gewöhnlicher Differentialgleichungen

$$w' = Aw \quad mit\ w(0) = 0, \tag{1}$$

wobei $A \in \mathbb{R}^{n\times n}$ diagonalisierbar ist. Unter welchen Bedingungen an A ist die stationäre Lösung $w^* \equiv 0$ stabil, unter welchen instabil?

(b) Zeigen Sie, dass für $A \in \mathbb{R}^{2\times 2}$ die stationäre Lösung $w^* \equiv 0$ von (1) stabil ist, wenn

$$\text{spur}(A) < 0, \quad \det(A) > 0.$$

(c) Betrachten Sie nun die zwei linearen linearer gewöhnlicher Differentialgleichungen

$$w' = Aw, \quad w' = Bw \quad \text{mit } A, B \in \mathbb{R}^{2\times 2}. \tag{2}$$

Geben Sie ein Beispiel für A und B an, so dass $w^* \equiv 0$ eine stabile Lösung beider Differentialgleichungen ist, aber instabil ist bezüglich der Differentialgleichung

$$w' = (A + B)w.$$

Bemerkung 1 (Gleichungslöser)

Im praktischen Teil dieses Aufgabenblattes geht es um die Lösung nichtlinearer Gleichungssysteme in $\mathbb{R}^n$. Diese treten zum Beispiel bei impliziten Zeitschrittverfahren auf. Das Lösen eines nichtlinearen Gleichungssystems lässt sich formulieren als die Bestimmung der Nullstelle einer Funktion $r : \mathbb{R}^n \to \mathbb{R}^n$. Um verschiedene lineare oder nichtlineare Gleichungslöser auswählen zu können (Newton-Verfahren, CG-Verfahren, Gauß-Seidel-Verfahren, ...), führen wir folgendes Interface ein:

```
template <typename Vector, typename Matrix>
class Solver
{
public:
  using VectorType = Vector;
  using MatrixType = Matrix;
  using FunctionType = DifferentiableFunction<VectorType,VectorType,MatrixType>;

  virtual void apply (const FunctionType& r, VectorType& z) const = 0;
};
```

Zur Repräsentation von Funktionen wie r verwendet dieses wiederum folgendes Interface für differenzierbare Funktionen, die nicht von der Zeit abhängen:

```
template <typename Domain, typename Range, typename JacobianRange>
class DifferentiableFunction{
public:
  using DomainType = typename DifferentiableFunction::DomainType;
  using RangeType = typename DifferentiableFunction::RangeType;
  using JacobianRangeType = JacobianRange;

  virtual RangeType operator()(const DomainType& x) const = 0;

  virtual JacobianRangeType evaluateJacobian (const DomainType& x) const = 0;
};
```

Definition 2 (Newton-Verfahren)

Sei $r : \mathbb{R}^n \to \mathbb{R}^n$ stetig differenzierbar. Das Newton-Verfahren liefert über die Fixpunktiteration

$$z^{l+1} := z^l - \left(J^k(z^l)\right)^{-1} r(z^l), \quad l = 0, 1, 2, \ldots \tag{3}$$

zu einem geeignet gewählten Startwert $z^0 \in \mathbb{R}^n$ eine Folge $(z^l)_{l \in \mathbb{N}}$, die gegen eine Nullstelle von r konvergiert. Genauer kann man zeigen, dass es zu jeder Nullstelle $z \in \mathbb{R}^n$ eine Umgebung $D \subset \mathbb{R}^n$ gibt, so dass das Newton-Verfahren für alle Startwerte $z^0 \in D$ quadratisch gegen z konvergiert, falls r in einer Umgebung von z Lipschitz-stetig ist und in z eine invertierbare Jakobi-Matrix $J^k(z)$ besitzt.

Aufgabe 3 (Newton-Verfahren) (4 Punkte)

Implementieren Sie das Newton-Verfahren in C++. Schreiben Sie dazu ein Klassentemplate `NewtonSolver`, welches das Interface `Solver` erfüllt.

- Das Verfahren soll für verschiedene Datentypen anwendbar sein, mit denen sich Vektoren in $\mathbb{R}^n$ und Matrizen in $\mathbb{R}^{n \times n}$ repräsentieren lassen. Zu diesem Zweck soll `NewtonSolver` zwei Template-Parameter `Vector` und `Matrix` besitzen. Setzen Sie voraus, dass `Vector` eine Methode `operator[]` für den indexbasierten Zugriff auf die Elemente des Vektors und eine Methode `size` zur Verfügung stellt (wie z.B. `std::array`). Setzen Sie ferner voraus, dass auf die Elemente und Größe einer Matrix vom Typ `Matrix` durch die doppelte Anwendung dieser Methoden zugegriffen werden kann, d.h. dass die Matrix als Vektor von Zeilenvektoren repräsentiert wird.

- Abbruchkriterium: Die Fixpunktiteration (3) soll so lange durchgeführt werden, bis die Änderungsrate $\|z^{l+1} - z^l\|$ eine zu übergebende Schranke `eps` unterschreitet.

- Beschränken Sie sich auf den Fall $n \in \{1, 2, 3\}$, für den sich die Invertierung einer regulären $n \times n$-Matrix explizit hinschreiben lässt (siehe Lineare Algebra I); behandeln Sie den Fall $n > 3$ mit einer Exception oder fangen Sie ihn mit einer Assertion ab.

- Leiten Sie vom Interface `DifferentiableFunction` eine Funktion ab, die der Funktion

$$f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \frac{\cos(x_1) - \sin(x_2)}{4} - x_1 \\ \frac{\cos(x_1) - 2\sin(x_2)}{4} - x_2 \end{pmatrix}$$

  entspricht und bestimmen Sie deren Nullstelle. Als Startvektor wählen Sie $x_0 = (0, 0)^T$.