FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachbereich Elektrotechnik
und Informationstechnik

Prof. Georg Hoever

SS 2017

1. Praktikum zur Höhere Mathematik 2 für (Wirtschafts-)Informatik

Ziel dieses Praktikums ist eine Implementierung des Gradientenverfahrens mit Schrittweitensteuerung.

1. Aufgabe

Um bequem mit Vektoren $\vec{x} \in \mathbb{R}^n$ arbeiten zu können, soll eine Klasse `CMyVektor` implementiert werden:

- Überlegen Sie sich, durch welche(n) Datentyp(en) Sie die Informationen speichern, z.B. die Dimension als Integer und die Werte in einem `double`-Array; Sie können auch die Standardklasse `vector<double>` der C++-Standard-Template-Library nutzen.

  Implementieren Sie die Informationen als private Attribute.

- Implementieren Sie (public-)Methoden, um
  - einen Vektor einer bestimmten Dimension anzulegen,
  - die Dimension eines Vektors auszugeben,
  - eine bestimmte Komponente des Vektors zu setzen,
  - eine bestimme Komponente des Vektors auszugeben.

  Tipp: Sie können beispielsweise elegant den Indexoperator `[]` oder Klammeroperator `()` überladen.

- Implementieren Sie eine (public-)Methode, die die Länge des Vektors zurückgibt.

Implementieren Sie ferner zwei überladene Operator-Funktionen

```
CMyVektor operator+(CMyVektor a, CMyVektor b)
CMyVektor operator*(double lambda, CMyVektor a),
```

die eine Vektor-Addition und eine skalare Multiplikation realisieren.

2. Aufgabe

Zu einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ soll der Gradient an einer Stelle $\vec{x} \in \mathbb{R}^n$ berechnet werden:

- Implementieren Sie eine Funktionen

  `CMyVektor gradient(CMyVektor x, double (*funktion)(CMyVektor x))`,

  der man im ersten Parameter die Stelle $\vec{x}$ und im zweiten Parameter die Funktion f als Funktionspointer übergibt, und die den Gradienten $\vec{g} = \operatorname{grad} f(\vec{x})$ numerisch durch

  $$g_i = \frac{f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1} \ldots, x_n) - f(x_1, \ldots, x_n)}{h}$$

  zu festem $h = 10^{-8}$ berechnet.

3. Aufgabe

Zu einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ soll ausgehend von einer Stelle $\vec{x} \in \mathbb{R}^n$ das Gradientenverfahren mit folgender Schrittweitensteuerung zur Maximierung von f durchgeführt werden:

Sei $\vec{x}_{\text{neu}} = \vec{x} + \lambda \cdot \operatorname{grad} f(\vec{x})$.

Falls $f(\vec{x}_{\text{neu}}) > f(\vec{x})$:

Teste eine doppelte Schrittweite. Dazu sei $\vec{x}_{\text{test}} = \vec{x} + 2 \cdot \lambda \cdot \operatorname{grad} f(\vec{x})$.

Falls $f(\vec{x}_{\text{test}}) > f(\vec{x}_{\text{neu}})$:

Nimm $\vec{x}_{\text{test}}$ als neues $\vec{x}$ und verdopple die Schrittweite λ.

Ansonsten wird $\vec{x}_{\text{neu}}$ als neues $\vec{x}$ genommen und die Schrittweite beibehalten.

Falls $f(\vec{x}_{\text{neu}}) \leq f(\vec{x})$:

Halbiere die Schrittweite solange, bis für das entsprechende $\vec{x}_{\text{neu}}$ gilt: $f(\vec{x}_{\text{neu}}) > f(\vec{x})$.

Dieses $\vec{x}_{\text{neu}}$ wird dann als neues $\vec{x}$ genommen und die Schrittweite entsprechend übernommen.

Dieses Verfahren soll solange durchgführt werden, bis $\|\operatorname{grad} f(\vec{x})\| < 10^{-5}$ ist, oder bis 50 Schritte gemacht wurden.

- Implementieren Sie das entsprechende Verfahren.

  Nutzen Sie wieder einen Funktions-Pointer zur Angabe der zu maximierenden Funktion. Neben der Startstelle $\vec{x}$ soll die Schrittweite λ optionales Argument mit default-Wert `1.0` sein.

- Testen Sie das Verfahren an den folgenden Beispielen:
  - $f : \mathbb{R}^2 \to \mathbb{R}$, $f(x, y) = \sin(x + y^2) + y^3 - 6y^2 + 9y$, Startstelle $\vec{x} = \binom{3}{2}$, default-Schrittweite,
  - $g : \mathbb{R}^3 \to \mathbb{R}$, $g(x_1, x_2, x_3) = -(2x_1^2 - 2x_1x_2 + x_2^2 + x_3^2 - 2x_1 - 4x_3)$, Startstelle $\vec{x} = (0, 0, 0)^T$, Start-Schrittweite $\lambda = 0.1$.