S. Kurtz, T. Schwabe,
G. Gonnella, D. Willrodt

Universität Hamburg
Zentrum für Bioinformatik

Programmierung für Naturwissenschaften
Sommersemester 2015
Übungen zur Vorlesung: Ausgabe am 18.06.2015

Punkteverteilung: Aufgabe 10.1: 5 Punkte, Aufgabe 10.2: 5 Punkte

Abgabe bis zum 24.06.2015, 10:00 Uhr.

Aufgabe 10.1 Plotten von Daten

Implementieren Sie in Python eine Klasse `daten_analyse`, welche folgende Eigenschaften hat:

- Es gibt eine Initialisierungsfunktion, welche als Argument einen Dateinamen benötig; aus dieser Datei werden alle Werte zu weiteren Verarbeitung eingelesen. Gültige Dateien sollen wie die Datei `Messwerte.txt` (STiNE) aufgebaut sein
- `.mean_<x>`: Mittelwert für <x> = T (Temperatur), v (Fließgewschwindigkeit), n (Anzahl notwendiger Tests), w (Messwert)
- `.std_<x>`: Standardabweichung (<x> wie oben)
- `.range_<x>`: Wertebereich (<x> wie oben)
- `.__str__()`: die String-Repräsentation der Klasse, die Ausgabe soll wie folgt aussehen:

```
=== Analyse der Ergebnisse in <Datei> ===
 - Temperatur: <Mittelwert> +/- <Standardabweichung>
   Wertebereich: <Min> - <Max>
 - Fliessgeschwindigkeit: <Mittelwert> +/- <Standardabweichung>
   Wertebereich: <Min> - <Max>
 - notwendige Tests: <Mittelwert> +/- <Standardabweichung>
   Wertebereich: <Min> - <Max>
 - Messwert: <Mittelwert> +/- <Standardabweichung>
   Wertebereich: <Min> - <Max>
```

  Achten Sie auf eine sinnvolle Anzahl von Nachkommastellen

- `.plot_results()`: ein Methode, die die Ergebnisse graphisch aufbereitet. Nutzen Sie dazu das Modul `matplotlib`, um einen Plot mit den Unterplots für Temperatur, Fließgeschwindigkeit und Messwert zu erzeugen. Jeder Unterplot soll die folgenden Spezifikationen erfüllen:
  - auf der x-Achse wird der Index der Messung aufgetragen
  - auf der y-Achse wird der gesuchte Wert und der aktuelle Mittelwert mit dem aktuelle Standardfehler des arithmetischen Mittels als vertikale Fehlerbalken aufgetragen
  - die Werte sollen in schwarz mit Kreissymbolen (Temperatur), Quadraten (Fließgeschwindigkeit) oder Dreiecken (Messwert) gezeichnet werden
  - der aktuelle Mittelwert soll in rot mit einem Pentagon, die Fehlerbalken ebenfalls in rot dargestellt werden

  Für die Anzahl notwendiger Tests soll ein Histogram für die Werte 0 bis 6 (Kastenbreite = 1) erzeugt werden. Die Balken sollen grau gefüllt sein.

  Beide Grafiken sollen als pdf-Datei `Abb1.pdf` bzw. `Abb2.pdf` gespeichert werden.

Schreiben Sie dann ein Skript, welche den Nutzer nach einer zu analysierenden Datei fragt, ein Objekt der Klasse `daten_analyse` mit dieser Information erzeugt, dem Nutzer das Ergebnis der Analys druch `print(<Objekt>)` ausgibt und schließlich die beiden Abbildungen erzeugt und den Nutzer darauf hinweist.

Hinweise:

Das Modul `matplotlib` gehört *nicht* zur Python-Standardbibliothek. Zur Bearbeitung dieser Aufgabe benötigen Sie also ggf. eine gülte Installation des Moduls. Mehr dazu unter:

Achten Sie bei den Grafiken auf eine aussagekräftige Achsenbeschriftung.

Der aktuelle Mittel bzw. Standardfehler des arithmetischen Mittels für einen Wert x_i bezieht sich auf die statistischen Werte über alle Werte $i = 0$ bis i.

Der Standardfehler des arithmetischen Mittels errechnet sich als:

$$\sigma(\bar{x}) \quad = \quad \frac{\sigma_x}{\sqrt{N}}, \tag{1}$$

mit der Standardabweich σ_x und der Anzahl der Messwerte N.

Aufgabe 10.2 Gauss-Elemenierung

Schreiben Sie ein Python-Skript zur Lösung von linearen Gleichungssystemen (LGS) implementiert werden. Dabei soll ein LGS ein Objekt der Klasse `lgs` sein. Die Klasse ist wie folgt definiert:

- Eigenschaften:
  - `.koeffmat`: die Koeffizientenmatrix, eine Liste von Listen mit Koeffizienten (jede Liste entspricht einer Zeile)
  - `.lsgvektor`: Lösungsvektor; leer, falls nicht eindeutig lösbar
- Methoden:
  - `__init__`: übernimmt eine beliebig lange Liste von Zeilen (Strings) und erzeugt daraus die Koeffizientenmatrix, die `.koeffmat` zugewiesen wird; falls eindeutig lösbar, wird `.lsgvektor` bestimmt
  - `.istgueltig`: gibt `True` zurück, wenn es in jeder der n Zeilen genau $n + 1$ Einträge gibt und mindestens ein Eintrag der ersten n Einträge pro Zeile ungleich Null ist
  - `.isthomogen`: gibt `True` zurück, wenn in jeder Zeile das letzte Element gleich Null ist und das LGS gültig ist
  - `.isteindeutig`: gibt `True` zurück, wenn keine linearen Abhängigkeiten von Zeilen gefunden wird und das LGS gültig ist
  - `.bestimme_loesungsvektor`: gibt einen Vektor mit den Unbekannten zurück
  - `__str__`: Ausgabe der Informationen zu Gültigkeit, Eindeutigkeit, Typ (homogen/inhomogen), Koeffizientenmatrix und, wenn lösbar, des Lösungsvektors

Das Skript soll zeilenweise die Eingabe der Koeffizientenmatrix (s.unten) abfragen und mit diesen Strings ein Objekt der Klasse `lsg` erzeugen. Anschließen wird die Ausgabe zu dem Objekt erzeugt. Testen sie ihre Skript mit folgenden Koeffizientenmatrizen:

$$\left(\begin{array}{rrr|r} 4 & 2 & 1 & 11 \\ 7 & 1 & -5 & -6 \\ -8 & 0 & 6 & 10 \end{array}\right) \qquad \left(\begin{array}{rrrr|r} 3 & -2 & 1 & 2 & 6 \\ 0 & 4 & -4 & 1 & 6 \\ 6 & -4 & 3 & 2 & 7 \\ 1 & 0 & 0 & -1 & -1 \end{array}\right)$$

Hinweise:

Ein Standardverfahren zur Bestimmung des Lösungsvektors ist die Gauss-Eleminierung. Die Idee dabei ist, dass sich ein Gleichungssystem wie:

$$\begin{aligned} 4x_1 + 2x_2 + x_3 &= 11 \\ 7x_1 + x_2 - 5x_3 &= -6 \\ -8x_1 + 6x_3 &= 10 \end{aligned}$$

umschreiben lässt zu:

$$\begin{pmatrix} 4 & 2 & 1 \\ 7 & 1 & -5 \\ -8 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ -6 \\ 10 \end{pmatrix}$$

oder einfacher:

$$\left(\begin{array}{rrr|r} 4 & 2 & 1 & 11 \\ 7 & 1 & -5 & -6 \\ -8 & 0 & 6 & 10 \end{array}\right)$$

Diese Darstellung bezeichnet man als Koeffizientenmatrix. Durch Addition von Zeilen (und Zeilentausch) wird sie in eine blockdiagonale Form gebracht:

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a_{22} & a_{23} & b_2 \\ 0 & 0 & a_{33} & b_3 \end{array}\right)$$

Nun lassen sich, von unten beginnend, die Unbekannten bestimmen. In Pseudo-Code sieht das Verfahren wie folgt aus:

```
for i = 1 bis Anzahl Unbekannte:
  for j = i+1 bis Anzahl Unbekannte:
    faktor = -1 * A_ji / A_ii
    for k = i bis Anzahl Elemente pro Zeile:
      A_jk = A_jk + faktor * A_ik

for i = Anzahl Unbekannte bis 1:
  x_i = A_i(Elemente pro Zeile)
  for j = i+1 bis Anzahl Unbekannte
    x_i = x_i - (A_ij * x_j)
  x_i = x_i / A_ii
```

Beim zweiten Schritt wird der Lösungsvektor vom letzten zum ersten Element aufgebaut. In Python können Sie hierzu entweder `liste.insert(Wert, 0)` nutzen, um einen `Wert` an erster Stelle einzufügen, oder Sie hängen alle Werte an eine Liste an und nutzen zum Schluss `liste.reverse()`.

Tipp: Wer mag, kann die Rückwertssubstitution im zweiten Lösungsschritt als rekursive Funktion implementieren.

Das Verfahren funktioniert so nur, solange die erste Zahl der gerade betrachteten Zeile ungleich Null ist. Sollte dies nicht der Fall sein, muss diese Zeile mit der nächsten Zeile, in der das erste Elemente ungleich Null ist, vertauscht werden (für Interessierte: siehe Stichwort *Pivotisierung*).

Eine lineare Abhängigkeit zwischen Zeilen einer Matrix $\mathbf{A}$ besteht, wenn für die Zeilenvektoren folgende Beziehung gilt: $\vec{a_i} = \sum_{j \neq i} C_j \vec{a_j}$ (C_j sind beliebige Konstanten). Dies kann am einfachsten geprüft werden, indem die blockdiagonale Form der Koeffizientenmatrix aufgebaut wird und dann das Produkt $\Pi_i a_{ii}$ gebildet wird. Ist es gleich Null, so liegt eine lineare Abhängigkeit vor.

Die Lösungen zu diesen Aufgaben werden am 25.06.2015 besprochen.