Institut für Informatik
Dr. Martin Buchholz
Benjamin Peherstorfer

Technische Universität München
WS 2010/2011

Einführung in die wissenschaftliche Programmierung

Übungsblatt 10

1.) Minimierung der Rosenbrock Funktion

Die Rosenbrock Funktion $f : \mathbb{R}^2 \to \mathbb{R}$ ist gegeben durch

$$f(x, y) = (1 - x)^2 + 100 \cdot \left(y - x^2\right)^2 .$$

Bestimmen Sie mit `scipy.optimize.fmin` das Minimum und speichern Sie die Iterationsschritte mit Hilfe einer *Callback Funktion* f. Übergeben Sie f an die Funktion `fmin` mit dem Keyword Argument `callback = f`. Es wird dann f nach jedem Iterationsschritt aufgerufen. Als Parameter bekommt die Callback Funktion die aktuelle Koordinate übergeben.

Stellen Sie die Funktion und die Iterationsschritte mit Gnuplot dar.

2.) Interpolation und Regression

Gegeben sind die Daten

x	1	2	3	4	5	6	7
y	1	2	1.5	2	3	2.5	4

.

a.) Polynominterpolation

Finden Sie die Koeffizienten des Interpolationspolynoms p

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

mit $p(x_i) = y_i$ für die Daten in der Tabelle.

i) Berechnen Sie die Koeffizienten mit der Vandermonde Matrix, d.h. stellen Sie das folgende Gleichungssystem auf und lösen Sie es mit einer geeigneten `scipy` Funktion.

$$\begin{pmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

ii) Benutzen Sie die Funktion `numpy.polyfit`.

iii) Stellen Sie die Lösungen mit Gnuplot dar. Was passiert weit außerhalb des Bereichs $[1, 7]$?

b.) Lineare Regression

Bestimmen Sie die lineare Ausgleichsgerade der Daten, d.h. bestimmen Sie Koeffizienten m und t der Funktion $f(x) = mx + t$, dass

$$\sum_{i=1}^{7} \left(y_i - f(x_i)\right)^2,$$

minimal ist.

i) Stellen Sie dazu das Gleichungssystem $Bz = b$ mit $B = A^T A$ und $b = A^T y$ auf, wobei

$$A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

ii) Verwenden Sie zum Lösen die Funktion `scipy.linalg.lstsq`.

iii) Stellen Sie wieder die Lösungen graphisch dar.

3.) Parameter fitten

Auf der Webseite zur Vorlesung finden Sie eine Datei mit zwei Datensätze. Die erste Spalte enthält die Werte t_i, die zweite den ersten Datensatz y_i und die dritte

den zweiten Datensatz z_i. Für jeden der zwei Datensätze sollen nun Koeffizienten $p^* = (p_0, p_1, p_2, p_3) \in \mathbb{R}^4$ der Funktion

$$f(t) = p_0 \cos\left(\frac{2\pi}{p_1} t - p_2\right) + p_3 t.$$

so bestimmt werden, dass die Summe der Fehlerquadrate minimal ist, d.h. für den ersten Datensatz muss gelten

$$p^* = \arg\min_{p \in \mathbb{R}^4} \sum_{i=0}^{n} \left(f(t_i) - y_i\right)^2.$$

Verwenden Sie die Funktion `scipy.optimize.leastsq` und stellen Sie Ihr Ergebnis mit Gnuplot dar.

4.) Masse-Feder-System

Wie betrachten ein Masse-Feder-System wie in der Abbildung unten dargestellt. Zwei Körper mit Masse m_1 und m_2 sind durch zwei Federn mit Federkonstanten k_1 und k_2 miteinander verbunden. Die Länge der Federn ist L_1 und L_2. Die Massen gleiten auf einer Oberfläche mit Reibungskoeffizienten b_1 und b_2.

Beschrieben wird dies durch ein System von gewöhnlichen Differentialgleichungen zweiter Ordnung

$$\begin{aligned} m_1 x_1'' + b_1 x_1' + k_1 (x_1 - L_1) - k_2 (x_2 - x_1 - L_2) &= 0, \\ m_2 x_2'' + b_2 x_2' + k_2 (x_2 - x_1 - L_2) &= 0. \end{aligned}$$

Dieses System kann in ein System von Gleichungen erster Ordnung umgewandelt werden

$$x_1' = y_1, \tag{1}$$

$$y_1' = \frac{1}{m_1} \cdot \left(-b_1 y_1 - k_1 (x_1 - L_1) + k_2 (x_2 - x_1 - L_2)\right), \tag{2}$$

$$x_2' = y_2, \tag{3}$$

$$y_2' = \frac{1}{m_2} \cdot \left(-b_2 y_2 - k_2 (x_2 - x_1 - L_1)\right). \tag{4}$$

Lösen Sie dieses System mit den Parameter

	m	k	L	b
1	1	8	0.5	0.8
2	1.5	40	1.0	0.5

und den Anfangsbedingungen

	x	y
1	0.5	0
2	2.25	0

mit der Funktion `scipy.integrate.odeint`. Sie benötigen hierfür eine Funktion f welche die Rechte Seite des Systems (1)-(4) berechnet. Übergeben Sie dann diese Funktion, Anfangsbedingungen und eine Liste von Zeitschritte an `odeint`. Sie bekommen die Lösung an allen Zeitschritten als Rückgabewert. Stellen Sie die Lösung wieder graphisch dar.