

Diff: Übung 07a

Unterschiede zwischen Vorgabe und Lösung

Um die Musterlösung leichter nachvollziehbar zu gestalten wurden die Unterschiede zur Quelltext-Vorgabe mit Hilfe des Programms `diff` grafisch aufbereitet.

Legende:

- Grau: unveränderter Text (nur auszugsweise)
- Grün: neue Zeilen
- Gelb: veränderte Zeilen
- Rot: gelöschte Zeilen (kommt hier nicht vor)

Hinweis: nicht aufgeführte Dateien wurden nicht verändert.

Dieses Dokument wurde mit Hilfe von [diff2html](#) erstellt.

diff -u ../kurs07a_Datenauswertung/uebung/code/aufgabe1.py ../kurs07a_Datenauswertung/uebung/loesung/aufgabe1.py

../kurs07a_Datenauswertung/uebung/code/aufgabe1.py	../kurs07a_Datenauswertung/uebung/loesung/aufgabe1.py
:	:
9 import pylab as pl # das wichtigste von Matplotlib	9 import pylab as pl # das wichtigste von Matplotlib
10	10
11	11
12 # XYZ ist jeweils durch sinnvolle Angaben zu ersetzen.	
13	
14 # Um Laufzeitfehler zu vermeiden: sys.exit() verwenden	
15 # ("weiterrutschen" nicht vergessen)	
16	
17	
18	12
19 ##### 1.	13 ##### 1.
20	14
21 daten = np.loadtxt('data/XYZ')	15 daten = np.loadtxt('data/messdaten.dat')
22 XYZ, XYZ, XYZ, XYZ = daten.T # 2d-Array entpacken (4 Spalten)	16 tt, x1, u, I = daten.T # 2d-Array entpacken
23	17
24 # Daten anschauen	18 # Daten anschauen
25 if 0: # (0 -> nicht ausführen, 1 -> ausführen)	19 if 0: # (0 -> nicht ausführen)
26 pl.plot(tt, XYZ)	20 pl.plot(tt, x1)
27 pl.figure()	21 pl.figure()
28 #...	22 pl.plot(tt, u)
	23 pl.figure()
	24 pl.plot(tt, I)
	25
29 pl.show()	26 pl.show()
30 sys.exit()	27 sys.exit()
31	28
:	:
33 ##### 2.	30 ##### 2.
34	31
35 # Vorzeichen vom Strom anpassen:	32 # Vorzeichen vom Strom anpassen:
	33 I = I*np.sign(u)
36	34
37 I = XYZ	35 # Hinweis: Wenn Spannung 0 -> VZ = 0 -> Strom = 0
38	36
	37 if 0:
	38
	39 pl.plot(tt, I)
	40
	41 pl.show()
	42 sys.exit()
	43
39	44 ##### 3.
40 ##### 3.	45
41	46 # Indizes der Spannungsimpulse herausfinden
42 # Indizes der Spannungsimpulse herausfinden	47
43	48
...	...

44		48	udiff = np.diff(u)
45	udiff = XYZ		
46	change_indices = np.arange(len(u))[udiff != 0]	49	change_indices = np.arange(len(u))[udiff != 0]
47	# Indizes der Werte True, wo es eine Änderung in U gibt	50	# Indizes der Werte True, wo es eine Änderung in U gibt
48	# (BOOL-Indizierung wurde benutzt)	51	# (BOOL-Indizierung wurde benutzt)
:	:	:	:
50	# Verhindern, dass ein evtl "halber Impuls am Ende mit erkannt wird"	53	# Verhindern, dass ein evtl "halber Impuls am Ende mit erkannt wird"
51		54	
52	if len(change_indices) % 2 == 1:	55	if len(change_indices) % 2 == 1:
53	change_indices = change_indices[XYZ] # letzten Wert weglassen	56	change_indices = change_indices[:-1]
54		57	
55		58	
56	# Bis jetzt sind die Indizes hintereinander	59	# Bis jetzt sind die Indizes hintereinander
57	# Wir wollen immer zwei in einer Zeile	60	# Wir wollen immer zwei in einer Zeile
58	#1: Impuls-Start-Index, 2: Impuls-End-Index)	61	#1: Impuls-Start-Index, 2: Impuls-End-Index)
59		62	
60	XYZ = XYZ.reshape(-1, 2) # -1 sagt: "so, dass es passt"	63	change_indices = change_indices.reshape(-1, 2) # -1 sagt: "so, dass es passt"
61		64	
62	# erste Spalte noch um eins erhöhen, weil sich der Index auf den letzten	65	# erste Spalte noch um eins erhöhen, weil sich der Index auf den letzten
63	# Wert vor dem Sprung bezieht.	66	# Wert vor dem Sprung bezieht.
64		67	
65	XYZ # = ...	68	change_indices[:,0] += 1
66			
67	print XYZ		
68		69	
		70	print change_indices
69		71	
70	##### 4.	72	##### 4.
71		73	
:	:	:	:
73	if 0:	75	if 0:
74	i1, i2 = change_indices[3,:]	76	i1, i2 = change_indices[3,:]
75		77	
76	pl.hist(XYZ[XYZ])	78	pl.hist(I[i1:i2])
77	pl.show()	79	pl.show()
78	sys.exit()	80	sys.exit()
79		81	
80			
81	##### 5.	82	##### 5.
82		83	
83			
84	# Strom mitteln:	84	# Strom mitteln:
85		85	
86	I_mean = 0*I # neues ('leeres') array anlegen	86	I_mean = 0*I # neues ('leeres') array anlegen
87		87	
88	# Zeilenweise über change_indices iterieren	88	# Zeilenweise über change_indices iterieren
89	for XYZ, XYZ in change_indices:	89	for i1, i2 in change_indices:
90	XYZ[XYZ] = np.mean(XYZ) # Mittelwert bilden und speichern	90	I_mean[i1:i2] = np.mean(I[i1:i2]) # mittelwert bilden und speichern
91		91	

92	92
93 if 0:	93 if 0:
94	94
95 pl.plot(tt, XYZ) # verwechselt,	95 pl.plot(tt, I)
96 pl.plot(tt, XYZ) # gemittelt	96 pl.plot(tt, I_mean)
97	97
98 pl.show()	98 pl.show()
99 sys.exit()	99 sys.exit()
:	:
103 if 0:	103 if 0:
104 pl.figure()	104 pl.figure()
105	105
106 start_idcs = XYZ[:,0] # erste Spalte: Indizes, wo es los geht	106 start_idcs = change_indices[:,0] # erste Spalte: Indizes, wo es los geht
107	107
108 # Für jeden Strom-Block ein Spannungs-Strom-Werte-Paar bestimmen:	108 # Für jeden Strom-Block ein Spannungs-Strom-Werte-Paar bestimmen:
109 ii = I_mean[XYZ] # die Werte sind konstant (für einen Block)	109 ii = I_mean[start_idcs]
110 uu = u[XYZ]	110 uu = u[start_idcs]
111	111
112 pl.plot(uu, ii, 'bx', ms = 7) # große blaue Kreuze (x)	112 pl.plot(uu, ii, 'bx', ms = 7) # große blaue Kreuze (x)
113	113
114 a1, a0 = sc.polyfit(XYZ, XYZ, XYZ) # lineare regression	114 a1, a0 = sc.polyfit(uu, ii, 1) # lineare regression
115	115
116 pl.plot(XYZ, XYZ, 'g-') # Polynom (Geradengleichung) auswerten und plotten	116 pl.plot(uu, a1*uu+a0, 'g-') # Polynom (Geradengleichung) auswerten und plotten
	117 # alternativ: sc.polyval [a1, a0]
117	118
118 print "Leitwert:", XYZ	119 print "Leitwert:", a1
119 print "Strom-Offset", XYZ	120 print "Strom-Offset", a0
120	121
121 pl.show()	122 pl.show()
122 sys.exit()	123 sys.exit()
123	124
124	125
125	
126 ##### 7.	126 ##### 7.
127	127
128 dt = t[1]	128 dt = tt[1]
129 xd = XYZ	129 xd = np.diff(x1)/dt
130 xdd = XYZ	130 xdd = np.diff(x1,2)/dt**2
131	
132	131
133 ##### 8.	132 ##### 8.
134 if 0:	133 if 0:
:	:
138 # wenn u < 0 ist: mit nächstem Schleifendurchlauf weitermachen	137 # wenn u < 0 ist: mit nächstem Schleifendurchlauf weitermachen
139 if u[i1] < 0: continue	138 if u[i1] < 0: continue
140	139
141 pl.plot(XYZ[i1:i2-1], XYZ[XYZ])	140 pl.plot(xd[i1:i2-1], xdd[i1:i2-1])
142	141
143 pl.show()	142 pl.show()

144 sys.exit()	143 sys.exit()
145	144
146	
147 ##### 9.	145 ##### 9.
148	146
149 # siehe	147 # siehe
:	:
156	154
157 # wir arbeiten erstmal mit Listen und wandeln am Ende in arrays um	155 # wir arbeiten erstmal mit Listen und wandeln am Ende in arrays um
158	156
159	
160 points_vel = []	157 points_vel = []
161 points_acc = []	158 points_acc = []
162 voltage = []	159 voltage = []
:	:
166 # negative Werte ignorieren	163 # negative Werte ignorieren
167 if u[i1] < 0: continue	164 if u[i1] < 0: continue
168	165
169 points_vel += list(XYZ)	166 points_vel += list(xd[i1:i2-1])
170 points_acc += list(XYZ)	167 points_acc += list(xdd[i1:i2-1])
171	168
172 # Liste der entsprechenden Länge, in der alle Elemente den gleichen Wert haben	169 # Liste der entsprechenden Länge, in der alle Elemente den gleichen Wert haben
173 # nämlich den passenden Spannungswert	170 # nämlich den passenden Spannungswert
174 voltage += [XYZ]*(i2-1-i1)	171 voltage += [u[i1]]*(i2-1-i1)
175	172
176	173
177 # pseudo-Werte hinzufügen um nan zu vermeiden	174 # pseudo-Werte am Rand hinzufügen um nan zu vermeiden
178 # Annahme: bei 3V bewegt sich noch (fast) nichts	175 # Annahme: bei 3V bewegt sich noch (fast) nichts
179 points_vel=[0, 0, 0, 7, 7] + points_vel	176 points_vel=[0, 0, 0, 7, 7] + points_vel
180 points_acc=[0, 3, 14, 0, 14] + points_acc	177 points_acc=[0, 3, 14, 0, 14] + points_acc
181 voltage = [3, 3, 12, 12, 12] + voltage	178 voltage = [3, 3, 12, 12, 12] + voltage
182	179
183	
184 # Listen als Arrays zusammenpacken:	180 # Listen als Arrays zusammenpacken:
185 points = np.array([points_vel, points_acc]).T	181 points = np.array([points_vel, points_acc]).T
186 voltage = np.array(voltage)	182 voltage = np.array(voltage)
:	:
206 if 0:	202 if 0:
207 # Datenfeld grafisch darstellen	203 # Datenfeld grafisch darstellen
208 pl.figure()	204 pl.figure()
209 pl.imshow(interp_voltage.T, extent=(0,xd_max,0,xdd_max), origin='lower', interpolation='nearest')	205 pl.imshow(interp_voltage, extent=(0,xd_max,0,xdd_max), origin='lower', interpolation='nearest')
210 pl.colorbar()	206 pl.colorbar()
211	207
212	208
:	:
228 a = np.abs(a)	224 a = np.abs(a)
229 v = np.abs(v)	225 v = np.abs(v)
230	226

	227
231 i1 = int(v/xd_max*N_grid)	228 i1 = int(v/xd_max*N_grid)
232 i2 = int(XYZ)	229 i2 = int(a/xdd_max*N_grid)
233	230
234 return interp_voltage[XYZ, XYZ]*s	231 return interp_voltage[i1, i2]*s
235	232
236 ##### 11.	233 ##### 11.
237	234
238 data = np.load('XYZ')	235 data = np.load('data/aufschwingen.npy')
239	236
240 tt, x1, x2, x3, x4, acc = XYZ	237 tt, x1, x2, x3, x4, acc = data.T
241	238
242 u = acc*0	239 u = acc*0
243	240
:	:
245 a = acc[i]	242 a = acc[i]
246 v = x2[i]	243 v = x2[i]
247	244
248 u[i] = calc_voltage(XYZ, XYZ)	245 u[i] = calc_voltage(v, a)
249	
250 if 0:	
251	246
	247 if 1:
252 pl.figure()	248 pl.figure()
253 pl.plot(tt, acc)	249 pl.plot(tt, acc)
254 pl.plot(tt, u)	250 pl.plot(tt, u)
255	251
256	
257 pl.show()	252 pl.show()
258 sys.exit()	253 sys.exit()
259	254

Common subdirectories: ../kurs07a_Datenauswertung/uebung/code/data and ../kurs07a_Datenauswertung/uebung/loesung/data