

Diff: Übung 03a

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 ../kurs03a/uebung/code/lagrange1.py ../kurs03a/uebung/loesung/lagrange1.py

../kurs03a/uebung/code/lagrange1.py	../kurs03a/uebung/loesung/lagrange1.py
1 # -*- coding: utf-8 -*-	1 # -*- coding: utf-8 -*-
2 from __future__ import division # 1/2 -> 0.5 (anstatt 0)	2 from __future__ import division
3	
4 ## vorgegebenes Grundgerüst	
5	3
6 from sympy import sin, cos, Function	4 from sympy import sin, cos, Function
7 import sympy as sp	5 import sympy as sp
8	
9 import sys	6 import sys
10	7
11 t = sp.Symbol('t')	
12	8
13	9
14 params = sp.symbols('m1, m2') ## unvollständig (Es fehlen Parameter)	10 # Lösung von Übung 03 (wird in Übung 04 benötigt)
15 m1, m2 = params	11
16	12
17 xt = Function('x')(t) # x(t)	
18 ## ...	
19	13
	14 t = sp.Symbol('t')
20	15
21	16
22 xdt = xt.diff(t)	17 params = sp.symbols('m1, m2, l, g')
23 ## ...	18 m1, m2, l, g = params
24	19
	20 xt = Function('x')(t) # x(t)
	21 phit = Function('phi')(t) # phi(t)
25	22
26 sys.exit() # Beendet das Programm.	23 xdt = xt.diff(t)
27 #-> Schrittweise nach unten schieben	24 phidt = phit.diff(t)
	25 xddt = xt.diff(t,2)
	26 phiddt = phit.diff(t,2)
28	27
29 # -----	
30	28
31 # Hilfsrechnung	29 # Hilfsrechnung
32	30
33 x2t = xt + l*sin(phit)	31 x2t = xt + l*sin(phit)
34 ## ...	32 y2t = -l*cos(phit)
35	33
36 x2dt = x2t.diff(t)	34 x2dt = x2t.diff(t)
37 ## ...	35 y2dt = y2t.diff(t)
38	36
39 # kinetische Energie	37 # kinetische Energie
40 T = (m2*(x2dt**2 + y2dt**2))/2 ## unvollständig	38 T = (m1*xdt**2 + m2*(x2dt**2 + y2dt**2))/2
..	..

41		39	
42	# Potentielle Energie	40	# Potentielle Energie
43	## ...	41	$U = y2t * g * m2$
44			
45	#Lagrange-Funktion		
46		42	
47	## ...	43	$L = T - U$ #Lagrange-Funktion
48		44	
49	#Funktionen durch Symbole ersetzen	45	#Funktionen durch Symbole ersetzen
50		46	
51	$x_s, \phi_s, x_d_s, \phi_{id_s} = \text{sp.symbols('x_s, phi_s, xd_s, phid_s')}$	47	$x_s, \phi_s, x_d_s, \phi_{id_s} = \text{sp.symbols('x_s, phi_s, xd_s, phid_s')}$
52		48	
53		49	
54	$\text{subs_list1} = [(xdt, x_d_s),]$ ## unvollständig	50	# Reihenfolge ist hier wichtig. Höhere Ableitungen zuerst substituieren.
55	$L = L.\text{subs}(\text{subs_list1})$ # subs Funktionen -> Symbole	51	$\text{subs_list1} = [(xdt, x_d_s), (\phi_{idt}, \phi_{id_s}), (x_t, x_s), (\phi_{it}, \phi_s)]$
56		52	$L = L.\text{subs}(\text{subs_list1})$ # subs Funktionen -> Symbole
57	# Vereinfachen der Ausdrücke		
58	$L = L.\text{expand}()$	53	$L = L.\text{expand}()$
59	$L = \text{sp.trigsimp}(L)$	54	$L = \text{sp.trigsimp}(L)$
60		55	
61			
62	# --- Lagrange-Gleichungen ---	56	# --- Lagrange-Gleichungen ---
63			
64	# Hilfst Terme:	57	# Hilfst Terme:
65		58	
66		59	
67	$L_d_x = L.\text{diff}(x_s)$	60	$L_d_x = L.\text{diff}(x_s)$
68	## $L_d_phi =$	61	$L_d_phi = L.\text{diff}(\phi_s)$
69		62	
70	## $L_d_xd =$	63	$L_d_xd = L.\text{diff}(x_d_s)$
71	## $L_d_phid =$	64	$L_d_phid = L.\text{diff}(\phi_{id_s})$
72		65	
73	# Weitere Symbole einführen		
74	$xdd_s, \phi_{idd_s} = \text{sp.symbols('xdd_s, phidd_s')}$	66	$xdd_s, \phi_{idd_s} = \text{sp.symbols('xdd_s, phidd_s')}$
75		67	
76		68	
		69	
		70	
		71	# sub_slist ergänzen (wieder Reihenfolge beachten: höhere Ableitungen vorn hin)
77	$\text{subs_list1} = [(xddt, xdd_s), (\phi_{iddt}, \phi_{idd_s})] + \text{subs_list1}$	72	$\text{subs_list1} = [(xddt, xdd_s), (\phi_{iddt}, \phi_{idd_s})] + \text{subs_list1}$
78		73	
79	# Liste umkehren (subs Symbole -> Funktionen)	74	# Liste umkehren (subs Symbole -> Funktionen)
80	# Kurznotation		
81	$\text{subs_list_rev} = [(tup[1], tup[0]) \text{ for } tup \text{ in } \text{subs_list1}]$	75	$\text{subs_list_rev} = [(tup[1], tup[0]) \text{ for } tup \text{ in } \text{subs_list1}]$
82		76	
83	# Ausführliche Notation:		
84	$\text{subs_list_rev} = []$		
85	for tup in subs_list1:		

86	subs_list_rev.append((tup[1], tup[0]))	
87		77
		78 #L_d_x = L_d_x.subs(subs_list_rev)
		79 #L_d_phi = L_d_phi.subs(subs_list_rev)
88		80
89	# Summanden der Lagrange-Gleichung berechnen	
90		
91	# Dazu erstmal wieder Zeitfunktionen für Symbole einsetzen	
92	L_d_xd = L_d_xd.subs(subs_list_rev)	81 L_d_xd = L_d_xd.subs(subs_list_rev)
93	## L_d_phid =	82 L_d_phid = L_d_phid.subs(subs_list_rev)
94		83
95	# Jetzt nach der Zeit ableiten	84 DL_d_xd = L_d_xd.diff(t).subs(subs_list1)
		85 DL_d_phid = L_d_phid.diff(t).subs(subs_list1)
96		86
97	## DL_d_xd =	87 F = sp.Symbol('F') # eingeprägte Last (translatorisch)
98	## DL_d_phid =	
99		88
100	# Wieder Funktionen durch Symbole ersetzen	89 # Gleichungen
101	# (Zweifach Ableitungen berücksichtigen)	
102		90
		91 Eq1 = DL_d_xd - L_d_x - F
		92 Eq2 = DL_d_phid - L_d_phi
103		93
104	# Lagrange-Gleichungen aus den Hilfsternen zusammensetzen	94 #sp.preview(Eq1)
105	# (bzw. jeweils eine Seite der Gleichung. Auf der anderen steht 0)	95 #sp.preview(Eq2)
106		96
107	F = sp.Symbol('F') # eingeprägte Last (translatorisch)	
108	Eq1 = DL_d_xd - L_d_x - F	
109	## Eq2 =	
110		97
		98 sol = sp.solve([Eq1, Eq2], xdd_s, phidd_s)
111		99
112	# Gleichungssystem lösen	100 # Gleichungen Umformen für Simulation
113	## siehe Doku von sp.solve	101
		102
		103
		104
		105 params_values = {m1: 0.8, m2: 0.3, l:0.5, g:9.81}
		106
		107 xdd_expr = sol[xdd_s].subs(params_values)
		108 phidd_expr = sol[phidd_s].subs(params_values)
		109
		110
		111 # Funktionen für die numerische Berechnung der Beschleunigungen
		112
		113 xdd_fnc = sp.lambdify([x_s, phi_s, xd_s, phid_s, F], xdd_expr, 'numpy')
		114 phidd_fnc = sp.lambdify([x_s, phi_s, xd_s, phid_s, F], phidd_expr, 'numpy')
		115
		116 ## xdd_fnc und phi_dd_fnc sind ausführbare bzw. aufrufbare-Objekte

	117 ## ("callable objects")
	118 ## Es sind Funktionen, die fünf Argumente bekommen und einen Wert zurrückgeben.
114	119
115 ## sol = sp.solve(...)	
116 ## sol ist ein dictionary	
117	120
118	121
119	122
:	:
129 plt.axis('off')	132 plt.axis('off')
130 plt.show()	133 plt.show()
131	134
132 # Anzeige der LSG des lin. Gleichungssystems	
133	135
134 ## preview(sol[xdd_s])	
135	136
136 ## Zur Verbesserung der Anzeige:	
137 ## (siehe Docstring von sp.latex)	
138 ## symbol_names = {phid_s : r'\dot{\varphi}', ...}	
139	137
140	138
141 # Ausdrücke in Aufrufbare-Funktion umwandeln	139 if __name__ == "__main__":
142	140
	141 # Dieser Block wird nur ausgeführt, wenn diese Datei das Haupt-Skript ist,
	142 # nicht wenn sie als Modul woanders importiert wird.
143	143
144 params_values = {m1: 0.8, m2: 0.3, l:0.5, g:9.81}	
145	144
146 xdd_expr = sol[xdd_s].subs(params_values)	145 # Anzeigen:
147 #phidd_expr = ...	
148	146
149 xdd_fnc = sp.lambdify([x_s, phi_s, xd_s, phid_s, F], xdd_expr, 'numpy')	147 ## sp.Eq -> sympy Equation object
150 #phidd_fnc = ...	148
	149 Eq1a = sp.Eq(xdd_s, sol[xdd_s])
	150 Eq2a = sp.Eq(phidd_s, sol[phidd_s])
	151
	152
	153 # Symbolbenennung in LaTeX-Notation
	154 sn_dict = {phi_s: r'\varphi', phid_s: r'\dot{\varphi}',
	155 xdd_s:r'\ddot{x}', phidd_s:r'\ddot{\varphi}'}
	156
	157 preview(Eq1a, symbol_names = sn_dict)
	158 preview(Eq2a, symbol_names = sn_dict)