Diff:

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 ../kurs03/03a-2d-visualisierung/uebung/code/plotten.py ../kurs03/03a-2d-visualisierung/uebung/loesung/plotten.py
```

../kurs03/03a-2d-visualisierung/uebung/code/plotten.py	../kurs03/03a-2d-visualisierung/uebung/loesung/plotten.py
⋮	⋮
3	3
4 `"""`	4 `"""`
5 Dieses Skript besteht im wesentlichen aus der Lösung der Aufgabe 2 von Kurs02b	5 Dieses Skript besteht im wesentlichen aus der Lösung der Aufgabe 2 von Kurs02b
6 Es soll um Plotfunktionen erweitert werden.	6 Es ist um Plotfunktionen erweitert.
7 `"""`	7 `"""`
8	8
9 `from scipy.integrate import odeint`	9 `from scipy.integrate import odeint`
⋮	⋮
26	26
27	27
28 `# Leere Liste zum Sammeln der Zwischenergebnisse der Optimierung anlegen:`	28 `# Leere Liste zum Sammeln der Zwischenergebnisse der Optimierung anlegen:`
29 `resList = XXX`	29 `resList = []`
30	30
31	31
32 `def min_target(p):`	32 `def min_target(p):`
⋮	⋮
45 `    res = odeint(rhs, x0, t)`	45 `    res = odeint(rhs, x0, t)`
46	46
47 `    # Zwischenergebnisse merken (an Liste anhängen)`	47 `    # Zwischenergebnisse merken (an Liste anhängen)`
48 `    resList.XXX(XXX)`	48 `    resList.append(res)`
49	49
50 `    # Differenz der x-Positionen bilden (jeweils erste Spalte) und quadrieren`	50 `    # Differenz der x-Positionen bilden (jeweils erste Spalte) und quadrieren`
51	51
⋮	⋮
55	55
56 `    return err`	56 `    return err`
57	57
58 `# Startwerte fuer Optimierung`	58 `## Um bei wiederholter Ausführung (z.B. zur Anpassung der Plots)`
59 `p0 = [.5, .7]`	59 `## Zeit zu sparen, sollte man das Ergebnis der Simulation speichern.`
60	60 `## Hier wird das Serialisierungsmodul pickle benutzt.`
61 `# Durchführen der Optimierung`	61 `## Damit können (fast) beliebige Pythonobjekte persistent gespeichert`
62 `p_res = fmin(min_target, p0)`	62 `## werden. Siehe https://docs.python.org/3/library/pickle.html`
	63
	64 `## Alternativ bieten sich für numerische Daten natürlich auch die`
	65 `## Funktionen numpy.save und numpy.load an.`
	66
	67 `## Generell ist eine Trennung von (aufwendiger) Datenerzeugung und`
	68 `## Visualisierung oftmals sehr sinnvoll.`
	69
	70 `import pickle # Serialisierungsmodul`
	71 `pfname = "res_list.pcl" # Dateiname beliebig`
	72
	73
	74 `## Nach dem erfolgreichen speichern hier eine 0 einsetzten:`
	75 `if 0:`
	76 `    # Optimierung - Ergebnis sind die zwei Parameter m2 und l`
	77
	78 `    # Startwerte fuer Optimierung`
	79 `    p0 = [.5, .7]`
	80
	81 `    # Durchführen der Optimierung`
	82 `    p_res = fmin(min_target, p0)`
	83
	84 `    myfile = open(pfname, 'wb') # Datei zum Schreiben öffnen (Binär-Modus)`
	85 `    pickle.dump(resList, myfile) # resList wird serialisiert u. gespeichert`

```
                                                                                    myfile.close()
                                                                                    print("Simulationsdaten erfolgreich gespeichert:", pfname)
                                                                                    print("Programm-Ende")
                                                                                    import sys
                                                                                    sys.exit()
                                                                                else:
                                                                                    # Berechnungsergebnisse laden
                                                                                    myfile = open(pfname, 'rb') # Datei zum Lesen öffnen (Binär-Modus)
                                                                                    resList = pickle.load(myfile)
                                                                                    myfile.close()
                                                                                
                                                                                
#-------------------------------------------------------------------------------  #-------------------------------------------------------------------------------
⋮                                                                               ⋮
                                                                                
                                                                                
# Subplot fuer Anfangswerte                                                     # Subplot fuer Anfangswerte
ax1 = fig.add_subplot(XXX, XXX, XXX)                                            ax1 = fig.add_subplot(3, 1, 1)
ax1.plot(XXX, XXX[:, XXX], label='Messung')                                     ax1.plot(t, target[:, 0], label='Messung')
ax1.plot(XXX, XXX[XXX][XXX, XXX], label='Modell (Startschätzung)')              ax1.plot(t, resList[0][:, 0], label='Modell (Startschätzung)')
ax1.legend()                                                                    ax1.legend()
ax1.grid()                                                                      ax1.grid()
ax1.set_ylabel('Weg x [m]')                                                     ax1.set_ylabel('Weg x [m]')
                                                                                
                                                                                
# Subplot fuer Optimierungsergebnise                                            # Subplot fuer Optimierungsergebnise
XXX = fig.add_subplot(XXX, XXX, XXX)                                            ax2 = fig.add_subplot(3, 1, 2)
XXX.plot(XXX, XXX, lw=3, label=XXX)                                             ax2.plot(t, target[:, 0], lw=3, label='Messung')
# ...                                                                           ax2.plot(t, resList[-1][:, 0], label='Modell optimiert')
                                                                                ax2.legend()
                                                                                ax2.grid()
                                                                                ax2.set_ylabel('Weg x [m]')
                                                                                
                                                                                
# Subplot fuer Restfehler der Position x                                        # Subplot fuer Restfehler der Position x
# ...                                                                           ax3 = fig.add_subplot(3, 1, 3)
                                                                                ax3.plot(t, target[:, 0]-resList[-1][:, 0])
                                                                                ax3.grid()
                                                                                ax3.set_xlabel('Zeit [s]')
                                                                                ax3.set_ylabel('Fehler $\epsilon$ [m]')
                                                                                #plt.show()
                                                                                
#-------------------------------------------------------------------------------  #-------------------------------------------------------------------------------
# Aufgabe 2: Verlauf der Optimierung (2D)                                       # Aufgabe 2: Verlauf der Optimierung (2D)
⋮                                                                               ⋮
    colStep = 0.9 / len(resList)                                                    colStep = 0.9 / len(resList)
    gray_value = str(0.01 + (0.9 - i*colStep))                                      gray_value = str(0.01 + (0.9 - i*colStep))
                                                                                
    plt.plot(XXX, XXX, color=gray_value )                                           plt.plot(t, resList[i][:, 0], color=gray_value )
                                                                                
# Plotten der Messung und der Simulation                                        # Plotten der Messung und der Simulation
plt.plot(XXX, XXX, color='#3366FF', lw=4, label='Messung')                      plt.plot(t, target[:,0], color='#3366FF', lw=4, label='Messung')
plt.plot(XXX, XXX, color='#FF9900', ls='--', lw=2, label='Modell optimiert')    plt.plot(t, resList[-1][:,0], color='#FF9900', ls='--', lw=2, label='Modell optimiert')
# Gitter, Legende ...                                                           plt.grid()
                                                                                plt.legend()
plt.xlabel(XXX)                                                                 plt.xlabel('Zeit [s]')
XXX.ylabel(XXX)                                                                 plt.ylabel('Weg x [m]')
                                                                                
```

```
112                                                 153
113 #plt.show()                                     154 #plt.show()
⋮                                                   ⋮
130 X, Y = meshgrid(range(len(resList)), t)         171 X, Y = meshgrid(range(len(resList)), t)
131                                                 172
132                                                 173
133 # Übersicht über color-Maps:                    174 ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.viridis)
134 # https://matplotlib.org/examples/color/colormaps_reference.html
135 ax.plot_surface(XXX, rstride=1, cstride=1, cmap=cm.XXX)
136 ax.set_xlabel('x')                              175 ax.set_xlabel('x')
137 ax.set_ylabel('y')                              176 ax.set_ylabel('y')
138 ax.set_zlabel('z')                              177 ax.set_zlabel('z')
Nur in ../kurs03/03a-2d-visualisierung/uebung/loesung: __pycache__.
Nur in ../kurs03/03a-2d-visualisierung/uebung/loesung: res_list.pcl.
```