

Vorversuch_Musterprotokoll

May 22, 2023

1 Fakultät für Physik

1.1 Physikalisches Praktikum P1 für Studierende der Physik

Praktikumsvorversuch

Raum: zu Hause

Name: _____ Vorname: _____ E-Mail: _____

Name: _____ Vorname: _____ E-Mail: _____

Gruppennummer: _____

Betreuer: _____

Versuch durchgeführt am: _____

Beanstandungen:

M U S T E R P R O T O K O L L (WS 2022/23)

Testiert am: _____ Vermerk: _____

2 Datenverarbeitung am Beispiel des Pendels

2.1 Motivation

Im Mittelpunkt jedes physikalischen Experiments steht die **Messung, d.h. die nachvollziehbare Erfassung und Weiterverarbeitung von Daten unter Laborbedingungen**. In diesem Praktikumsvorversuch möchten wir Sie anhand eines einfachen physikalischen Vorgangs mit den wichtigsten Methoden der (heutzutage meist computergestützten) Datenverarbeitung in der modernen Physik vertraut machen. Sie werden sehen, dass Messen schnell zur Messkunst avancieren kann.

Als Aufgabe wählen wir die Bestimmung der Erdbeschleunigung (g) mit Hilfe eines Pendels. Wir haben mit Hilfe der kostenfreien *app* [phyphox](#) der RWTH Aachen einen Datensatz für Sie vorbereitet den wir mit Ihnen gemeinsam weiterverarbeiten möchten. Sie können das hier vorgestellte Experiment bei sich zu Hause noch einmal von Grund auf neu durchführen. Beachten Sie dabei (hier wie im gesamten P1): *experimentieren* bedeutet *versuchen* – nicht jeder Versuch muss zu einem erfolgreichen Ausgang führen. Wie im gesamten P1, geht es uns mit diesem Vorversuch nicht darum, dass Sie uns die beste Messung von g präsentieren. Uns ist wichtig, dass Sie sich mit den zur Verfügung stehenden Daten vertraut machen, sich mit den vorgestellten Möglichkeiten zur weiteren Datenverarbeitung ernsthaft auseinandersetzen und vielleicht auch neue, eigene Ideen entwickeln.

2.2 Lernziele

Wir listen im Folgenden die wichtigsten Lernziele auf, die wir Ihnen mit dem **Vorversuch Datenverarbeitung** des P1 vermitteln möchten:

- Sie erhalten einen ersten Einblick darin, welche Stärken und Schwächen eine Messung haben kann?
- Sie lernen, dass Sie mit Hilfe ausgedehnter Messreihen Messunsicherheiten deutlich verringern können?

- Sie gehen mit einem mächtigen Werkzeugkasten aus diesem Versuch hervor, mit dem Sie auch größere Datenmengen mit den gehobenen Ansprüchen eines Physikers analysieren können.
- Sie lernen, wie Sie mit Hilfe dieses Werkzeugkastens physikalisch bedeutungsvolle Parameter aus der Anpassung von Modellen an die Daten bestimmen können?
- Sie lernen was der Unterschied zwischen einer *Messunsicherheit* und einem *Messfehler* ist?

Machen Sie sich zur Vorbereitung auf das Praktikum den folgenden Umstand bewusst: **Jeder Messung der modernen Physik liegt ein Modell zugrunde.** Die Messung hat i.a. die Bestimmung eines Parameters innerhalb eines solchen Modells zum Ziel.

2.3 Versuchsaufbau

Wir haben die *app* [phyphox](#) auf ein Smartphone geladen und das Smartphone mit Klebestreifen auf eines der [Reversionspendel](#) aus dem Versuch **P1-20, 21, 22 Pendel** montiert. Für die Messung haben wir die Anwendung “Beschleunigung ohne g ” verwendet. Der Versuchsaufbau ist im folgenden Bild skizziert:

Wir haben das Pendel in Schwingung versetzt, die Bewegung mit Hilfe der im Smartphone verbauten Beschleunigungssensoren ausgelesen und uns die resultierenden Datensätze im [csv-Format](#) per Mail zugeschickt. Außerdem haben wir alle für unsere Messung relevanten äußeren Parameter festgehalten. Die Datensätze, die wir aufgenommen haben finden Sie im Verzeichnis `params`, in diesem Repository hinterlegt.

2.4 Wichtige Hinweise

- Beim csv-Format handelt es sich um ein einfaches sowohl von Menschen als auch Maschinen lesbares Format, um Daten in Spalten und Zeilen organisiert abzulegen.
- Alle wichtigen Informationen zu Pendel und Smartphone haben wir aus der Anleitung des Versuchs **P1-20, 21, 22 Pendel** und den im Internet verfügbaren Datenblättern des Smartphones bezogen. Zum Teil haben wir die Abmessungen des Smartphones noch einmal mit einfachen Mitteln nachvollzogen.

2.5 Durchführung

2.5.1 Aufgabe 1: Aufsetzen der Arbeitsumgebung

Wie sehen die Daten aus? Untersuchen Sie die im Verzeichnis `Vorversuch` hinterlegten Dateien `RawData.csv` und `RawData_down_sampled_500_2200_10.csv` durch Doppelklick und bestimmen Sie die entsprechenden Dateigrößen. Diese Dateien sind die Grundlage für jedes weitere Vorgehen im weiteren Verlauf dieses Vorversuchs.

Anmerkungen:

- Bei so großen Datenmengen ist eine manuelle Weiterverarbeitung der Daten unmöglich. Wir werden dazu die Programmiersprache [python](#) verwenden. Wir möchten Ihnen weiterhin die Modulsammlung [PhyPraKit](#) vorstellen.
- Die *app* `phyphox` hat die Beschleunigungssensoren des Smartphones mit einer festen [Abtastrate](#) (engl. *sampling rate*) von 100 Hz ausgelesen. Um die Eigenschaften der Schwingung für uns ausreichend gut erfassen zu können haben wir 5 min lang gemessen. Sie können sich schnell klar machen mit welcher Datenmenge Sie zunächst zu tun haben werden. Die Periode der Schwingung haben wir während des Versuchs zu 1–2 s abgeschätzt. Wir haben

daher als ersten Schritt der Weiterverarbeitung die Signalarate um den Faktor 10 reduziert, indem wir nur jede 10. Zeile der csv-Datei ausgelesen haben. Man bezeichnet diese Vorgehensweise als *down sampling*. Diesen nachprozessierten Datensatz finden Sie in der Datei `RawData_down_sampled_500_2200_10.csv` die wir für alle weiteren Versuchsteile nutzen werden.

2.5.2 Lösung:

Die *app* exportiert die Daten in fünf Spalten mit den eindeutigen Bezeichnungen:

- Time (s)
- Linear Acceleration x (m/s^2)
- Linear Acceleration y (m/s^2)
- Linear Acceleration z (m/s^2)
- Absolute acceleration (m/s^2)

Die Datei `RawData.csv` hat eine Größe von 2.4 MB und 32304 Zeilen (ohne Überschriften); die Datei `RawData_down_sampled_500_2200_10.csv` hat eine Größe von 13 kB und 171 Zeilen. Es handelt sich um eine Reduktion der Datenmenge um einen Faktor von 190. Die Reduktion über den Faktor 10 hinaus ergibt sich daraus, dass nur ein Zeitfenster von sample 500 bis sample 2200 (5–22 s nach Beginn der Datennahme) verwendet wurde.

Eine Inspektion der ausgewählten Daten aus der Datei `RawData_down_sampled_500_2200_10.csv` ist im folgenden Bild gezeigt:

Abbildung 1.1

```
[26]: import pandas as pd
import numpy as np
from os import path
import matplotlib.pyplot as plt

def subplot(ax, df, column):
    """
    Create a single subplot per column vs. time from a dataframe.
    """
    ax.plot("Time (s)", column, data=df, color="darkblue")
    ax.set_title("N=%d"%df.shape[0]) # Number of points in plot
    ax.set_xlabel(column)
    ax.set_ylabel("AU")

def plot(df, **kwargs):
    """
    Create a figure for quick inspection of the csv-file.

    Creates a 4x2 figure for all columns that are provided by the original csv-
    file. The upper row shows the raw data; for the second row the raw data are
    smoothed through a running mean with a user-defined window size of 10. The
```

```

figure is saved in pdf format.
"""

# Set up the Axes for this example
fig, ax = plt.subplots(2, 4, figsize=(12., 6.), constrained_layout=True)
# Columns of the original csv
subplot(ax[0][0], df, "Linear Acceleration x (m/s^2)")
subplot(ax[0][1], df, "Linear Acceleration y (m/s^2)")
subplot(ax[0][2], df, "Linear Acceleration z (m/s^2)")
subplot(ax[0][3], df, "Absolute acceleration (m/s^2)")
# Columns after smoothing
subplot(ax[1][0], df, "Convolutd x")
subplot(ax[1][1], df, "Convolutd y")
subplot(ax[1][2], df, "Convolutd z")
subplot(ax[1][3], df, "Convolutd abs")
print("Saving figure as ", kwargs["filename"].replace(".csv", ".pdf"))
plt.savefig("./"+kwargs["filename"].replace(".csv", ".pdf"))
plt.show()

def main(**kwargs):
    if not path.isfile(kwargs["filename"]):
        print("Could not find file %s. Please check filename and_
↳location"%kwargs["filename"])
        exit()
    print("Reading from file:", kwargs["filename"])
    # Read csv as pandas dataframe
    df = pd.read_csv(kwargs["filename"])
    # Apply a running mean smoothing
    df["Convolutd x"] = np.convolve(
        # This is the column name
        df["Linear Acceleration x (m/s^2)"],
        np.ones(kwargs["w"])/kwargs["w"],
        mode="same"
    )
    df["Convolutd y"] = np.convolve(
        df["Linear Acceleration y (m/s^2)"],
        np.ones(kwargs["w"])/kwargs["w"],
        mode="same"
    )
    df["Convolutd z"] = np.convolve(
        df["Linear Acceleration z (m/s^2)"],
        np.ones(kwargs["w"])/kwargs["w"],
        mode="same"
    )
    df["Convolutd abs"] = np.convolve(
        df["Absolute acceleration (m/s^2)"],
        np.ones(kwargs["w"])/kwargs["w"],
        mode="same"

```

```

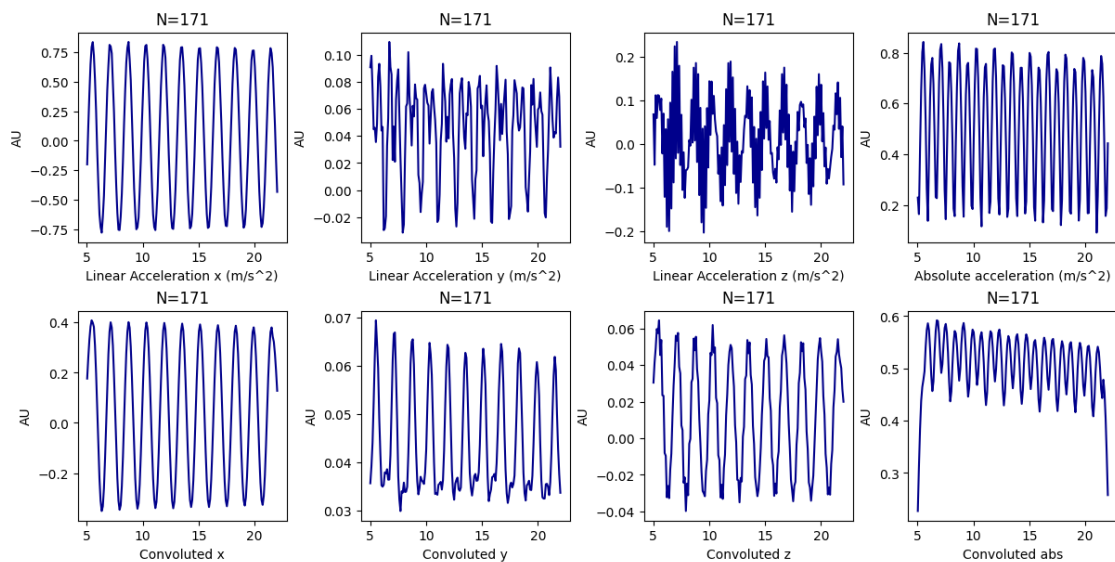
    )
    plot(df, **kwargs)

main(
    # File name
    filename="RawData_down_sampled_500_2200_10.csv",
    # Window size for running mean
    w=10
)

```

Reading from file: RawData_down_sampled_500_2200_10.csv

Saving figure as RawData_down_sampled_500_2200_10.pdf



Diskussion:

Der Datensatz umfasst **10 Perioden der Schwingung**. Für uns von Relevanz ist die Beschleunigung in x -Richtung, die auch bei weitem die größten Auslenkungen mit dem regelmäßigen Verhalten aufweist. Die Beschleunigung in y -Richtung zeigt eine Grundschiwingung, die mit der Bewegung in x -Richtung korreliert ist. Das kann ein Hinweis darauf sein, dass das Smartphone nicht ganz plan auf das Pendel montiert war. Die Oberschwingungen rühren vermutlich eher nicht von einem schrägen Antoß des Pendels sondern von der Beschaffenheit und Auflage des Keils her, der in der Keilpfanne der Aufhängung des Pendelaufbaus liegt. Ein ähnliches Verhalten beobachten wir für die Bewegung in z -Richtung, die zunächst überraschend unregelmäßig und in der Amplitude nicht viel größer als die Bewegung in y -Richtung ist. Die Oberschwingungen in dieser Bewegung würden wir ebenfalls auf die Beschaffenheit des Keils und der Keilpfanne zurückführen.

2.5.3 Aufgabe 2: Erste Schritte

Aufgabe 2.1: Visualisierung der Daten mit der Funktion `plotData.py` Der erste Schritt, um sich mit großen Datensätzen zurechtzufinden ist sie besser sichtbar zu machen. Wählen Sie ein für Sie gut geeignetes Werkzeug aus, um die aufgezeichneten Daten graphisch darzustellen. Wir schlagen das Skript `plotData.py` vor. Dieses Skript erlaubt es Ihnen Daten auf der x -Achse gegen Daten auf der y -Achse graphisch darzustellen.

Aufgabe 2.2: Anpassung eines Modells an die Daten mit der Funktion `run_phyFit.py` Der nächste Schritt ist die Anpassung eines einfachen Modells an die beobachteten Daten. Dies können Sie erreichen, indem Sie Ihre Eingabedatei um die Definition einer python-Funktion erweitern, die der mathematischen Abbildung des Modells entspricht. Ein solcher *Funktionsblock* in Ihrer *yaml*-Datei könnte z.B. so aussehen:

```
model_label: "Hamonic oscillation"
model_function: |
    def my_model(x, A=0.8, omega=4., phi=0.):
        return A*np.cos(omega*x+phi)
```

Der Funktionsname `my_model` ist in diesem Fall frei gewählt. Beachten Sie, dass dieses Modell beim Aufruf des Skripts `plotData.py` mit den Daten zusammen dargestellt wird. Die hierzu verwendeten Funktionsparameter entsprechen den Defaultwerten der Funktionsargumente. Wenn Sie das Modell nicht nur mit fest vorgegebenen Parametern darstellen, sondern an die Daten anpassen möchten führen Sie das Skript `run_phyFit.py` mit der gleichen Eingabedatei aus.

Stellen Sie die Daten geeignet dar und fügen Sie Ihrer Auswertung eine graphische Darstellung zu. Verwenden Sie für diese Aufgabe den Datensatz `RawData_down_sampled_500_2200_10.csv`

Anmerkungen:

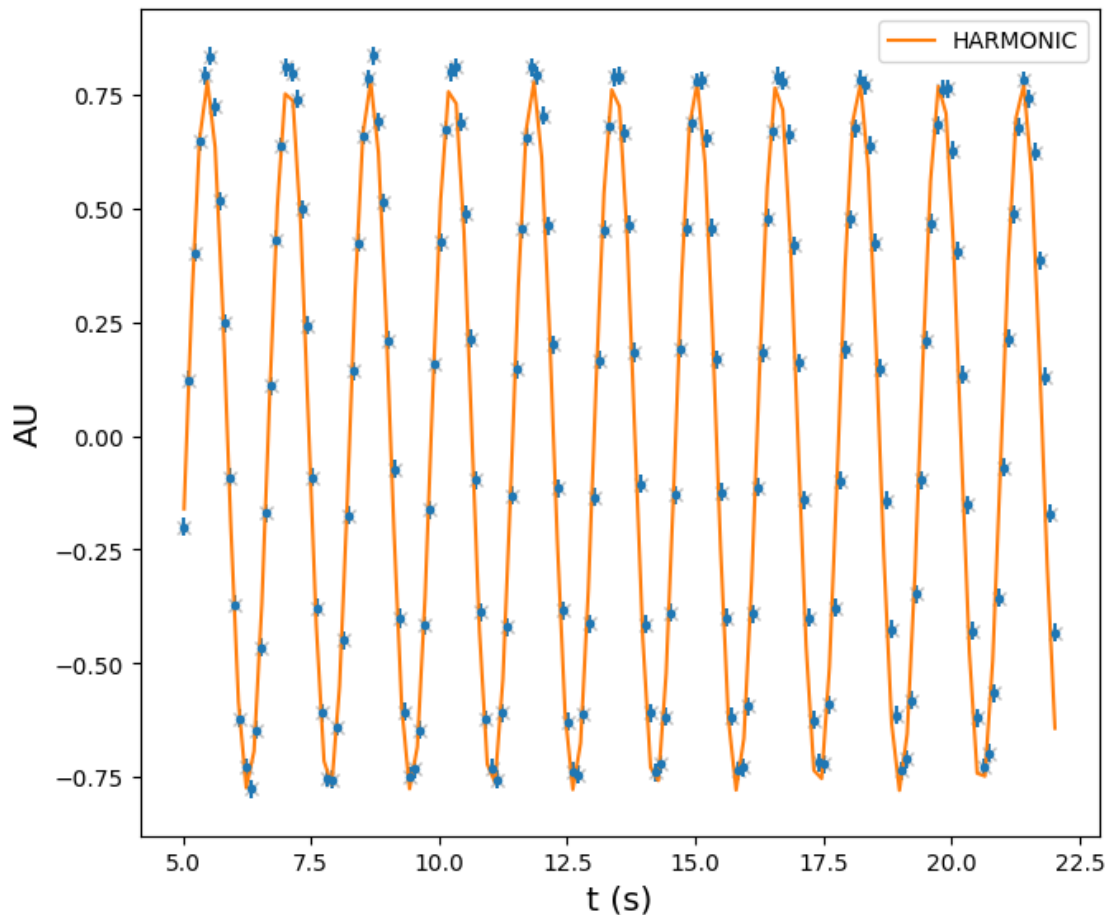
- Beachten Sie, dass die Skripte `plotData.py` und `run_phyFit.py` ihre Eingangs- und Konfigurationsdaten nicht im csv-Format sondern in der Struktursprache `yaml` erwarten. Die Notwendigkeit Daten umzuformatieren ist ein häufiges Problem in der Datenverarbeitung. Sie können zur Umformatierung das Skript `cvs2yaml.py` verwenden.
- Beachten Sie, dass Sie für eine Anpassung mit Hilfe der Funktion `xyFit` des Skripts `run_phyFit.py` Unsicherheiten für die Daten in x - und y -Richtung angeben müssen. Wir haben Abschätzungen für diese Unsicherheiten für Sie in der Datei `params/uncertainties_data.py` hinterlegt.
- Wenn Sie Aufgabe 2 zu Ihrer Zufriedenheit abgeschlossen haben sind Sie mit allen Werkzeugen ausgestattet, die Sie benötigen, um die weitere Analyse der Daten fortsetzen zu können. Mehr als die Skripte `plotData.py` und `run_phyFit.py` und ein Grundverständnis, wie man die beiden Skripte mit Hilfe der Struktursprache `yaml` ansteuert werden Sie für diesen Vorversuch nicht benötigen.

2.5.4 Lösung:

Aufgabe 2.1: Visualisierung der Daten mit der Funktion plotData.py Eine mögliche Ausgabe des Skripts plotData.py ist im folgenden gezeigt. Die Konfigurationsdatei haben wir unter dem Namen RawData_down_sampled_500_2200_10_HARMONIC.py hinterlegt.

Abbildung 2.1

```
[27]: %run /opt/conda/bin/plotData.py yaml/RawData_down_sampled_500_2200_10_HARMONIC.  
      ↪yaml
```



Diskussion:

Der relevante Abschnitt der *yaml*-Datei, die wir uns dafür erstellt haben sieht wie folgt aus:

```
# -----  
# Input data:  
# input file : Master/RawData_down_sampled_500_2200_10.csv  
# x_data      : Time (s)  
# y_data      : Linear Acceleration x (m/s2)  
# -----  
model_label: "HARMONIC"
```



```

model_function: |
    def model(x, x0=0.78, T=1.59, phi0=3.5):
        return x0*np.cos(2*np.pi/T*x+phi0)

y_label: "AU"
y_errors: 0.02

x_errors: 0.0125
x_label: "t (s)"

# Data:
x_data:
- 5.024349958
...

```

Die in der *yaml*-Datei angegebenen Unsicherheiten für die Datenpunkte in *x*- und *y*-Richtung haben wir aus der Datei `params/uncertainties_data.py` entnommen. Die Modellfunktion mit dem Label "HARMONIC" wird bei der Verwendung mit dem Skript `plotData.py` lediglich gemeinsamen mit den Datenpunkten dargestellt. Die verwendeten Parameter entsprechen den Defaultwerten in der Definition der python-Funktion. Wir haben diese Defaultwerte durch Ausprobieren ermittelt.

Aufgabe 2.2: Anpassung eines Modells an die Daten mit der Funktion `run_phyFit.py`
Mit der gleichen Konfigurationsdatei haben wir das Skript `run_phyFit.py` aufgerufen. In diesem Fall wird die Funktion nicht mit den Defaultwerten angezeigt. Stattdessen werden die Parameter an die Daten angepasst. Das Ergebnis sieht wie folgt aus:

Abbildung 2.2

```

[28]: %run /opt/conda/bin/run_phyFit.py yaml/
      ↪RawData_down_sampled_500_2200_10_HARMONIC.yaml

```

```

*** /opt/conda/bin/run_phyFit.py received valid yaml data for fit:
** Type of Fit: xy
model_label: HARMONIC
model_function: def model(x, x0=0.78, T=1.59, phi0=3.5):
                  return x0*np.cos(2*np.pi/T*x+phi0)

y_label: AU
y_errors: 0.02
x_errors: 0.0125
x_label: t (s)
x_data: [5.024349958, 5.124306958, 5.224263958, 5.324220958, 5.424177958,
5.524134958, 5.624092958, 5.72404975, 5.82400675, 5.92396375, 6.02392075,
6.12387775, 6.22383475, 6.32379175, 6.42374975, 6.52370675, 6.62366375,
6.723620583, 6.823577583, 6.923534583, 7.023491583, 7.123448583, 7.223405583,
7.323363583, 7.423320583, 7.523277583, 7.623234583, 7.723192, 7.823149,

```

7.923106, 8.023063, 8.12302, 8.222977, 8.322935, 8.422892, 8.522849, 8.622806,
8.722762958, 8.822719958, 8.922676958, 9.022633958, 9.122590958, 9.222547958,
9.322504958, 9.422461958, 9.522419958, 9.622376958, 9.722333708, 9.822290708,
9.922247708, 10.02220471, 10.12216171, 10.22211871, 10.32207571, 10.42203271,
10.52198971, 10.62194671, 10.72190371, 10.82186071, 10.92181771, 11.02177571,
11.12173271, 11.22168971, 11.32164671, 11.42160371, 11.52156071, 11.62151771,
11.72147462, 11.82143162, 11.92138862, 12.02134562, 12.12130262, 12.22125962,
12.32121662, 12.42117362, 12.52113062, 12.62108762, 12.72104546, 12.82100246,
12.92095946, 13.02091746, 13.12087446, 13.22083146, 13.32078846, 13.42074546,
13.52070246, 13.62065946, 13.72061571, 13.82057271, 13.92052971, 14.02048671,
14.12044371, 14.22040071, 14.32035771, 14.42031471, 14.52027171, 14.62022871,
14.72018637, 14.82014337, 14.92010037, 15.02005737, 15.12001437, 15.21997137,
15.31992837, 15.41988537, 15.51984237, 15.61979937, 15.71975646, 15.81971346,
15.91967046, 16.01962746, 16.11958446, 16.21954146, 16.31949846, 16.41945546,
16.51941246, 16.61936946, 16.71932558, 16.81928258, 16.91923958, 17.01919658,
17.11915458, 17.21911158, 17.31906858, 17.41902558, 17.51898258, 17.61893958,
17.71889671, 17.81885371, 17.91881071, 18.01876771, 18.11872471, 18.21868171,
18.31863871, 18.41859571, 18.51855271, 18.61850971, 18.71846671, 18.81842371,
18.91838071, 19.01833771, 19.11829471, 19.21825171, 19.31820871, 19.41816571,
19.51812271, 19.61807971, 19.71803733, 19.81799433, 19.91795133, 20.01790833,
20.11786533, 20.21782233, 20.31777933, 20.41773633, 20.51769333, 20.61765033,
20.71760708, 20.81756408, 20.91752108, 21.01747708, 21.11743408, 21.21739208,
21.31734808, 21.41730508, 21.51726208, 21.61721908, 21.71717608, 21.81713308,
21.91709008, 22.01704708]

y_data: [-0.1997231989, 0.1225266681, 0.4020419037, 0.6469383518, 0.7932787375,
0.8341633464, 0.7236726053, 0.5166886264, 0.2480305557, -0.09172321628,
-0.3712864995, -0.6222052874, -0.728521482, -0.7767518748, -0.6462890916,
-0.4653517906, -0.1693552458, 0.1100302364, 0.4309675679, 0.6369901914,
0.8109249035, 0.7987021031, 0.7397423619, 0.4993210072, 0.242197081,
-0.09183958501, -0.3778064739, -0.6088913966, -0.7521712035, -0.7551054094,
-0.6405515112, -0.4490290688, -0.1754098915, 0.1433068137, 0.4238335993,
0.6607164384, 0.7861605565, 0.8358004206, 0.6911543387, 0.5139052787,
0.2101174652, -0.07397222986, -0.4009491321, -0.6076723618, -0.7486201894,
-0.7316711578, -0.6475558908, -0.41658762, -0.1626306555, 0.158359301,
0.4260544449, 0.6725575617, 0.8006421362, 0.8104562491, 0.6894230513,
0.4883145739, 0.2144678302, -0.09728084858, -0.3876322081, -0.6239515583,
-0.7300468013, -0.7555174188, -0.6094878857, -0.4202097143, -0.1315586977,
0.145720654, 0.4559011187, 0.6575324102, 0.8117785238, 0.7923597014,
0.7034183619, 0.4620813332, 0.2018080601, -0.1154284224, -0.3831218487,
-0.6291048557, -0.7379914863, -0.7454776034, -0.6127069606, -0.4129662932,
-0.1345960338, 0.166161387, 0.4537871309, 0.6826964875, 0.7889552332,
0.791376419, 0.6656307308, 0.4643466177, 0.1837737122, -0.1054909332,
-0.4143776649, -0.609118232, -0.7404253167, -0.7217743067, -0.6179310458,
-0.3895465134, -0.1298084798, 0.1914881265, 0.4578757234, 0.6872261792,
0.7793857531, 0.7828821688, 0.6544036382, 0.4566309974, 0.1680467851,
-0.1247703727, -0.3992372136, -0.6183909293, -0.73671577, -0.7281120307,
-0.5945850141, -0.3906517103, -0.1125802436, 0.1833649187, 0.4783845399,
0.6695181059, 0.7909719379, 0.7806059939, 0.6631938305, 0.4194446792,

```

0.1616605658, -0.1409690604, -0.4006429939, -0.6256357026, -0.717948839,
-0.7221022625, -0.5911024124, -0.3780585985, -0.09839636028, 0.1903169292,
0.4779136198, 0.6773630228, 0.7846111905, 0.7712306415, 0.6392149839,
0.4244822017, 0.1476163213, -0.1418761097, -0.4254862417, -0.6149222879,
-0.735367329, -0.709930406, -0.5818266819, -0.3479699596, -0.09709096938,
0.2109081915, 0.4670812143, 0.6837134644, 0.7614358243, 0.7637974416,
0.6275016953, 0.4065399474, 0.1342015295, -0.1505459225, -0.4289698666,
-0.6198666935, -0.7288435174, -0.6980264251, -0.5651871872, -0.3557519093,
-0.0691921402, 0.2131352132, 0.4881150377, 0.6783859932, 0.7830247678,
0.7413724194, 0.6234380997, 0.3864223827, 0.1303110115, -0.1704625623,
-0.4323219292]

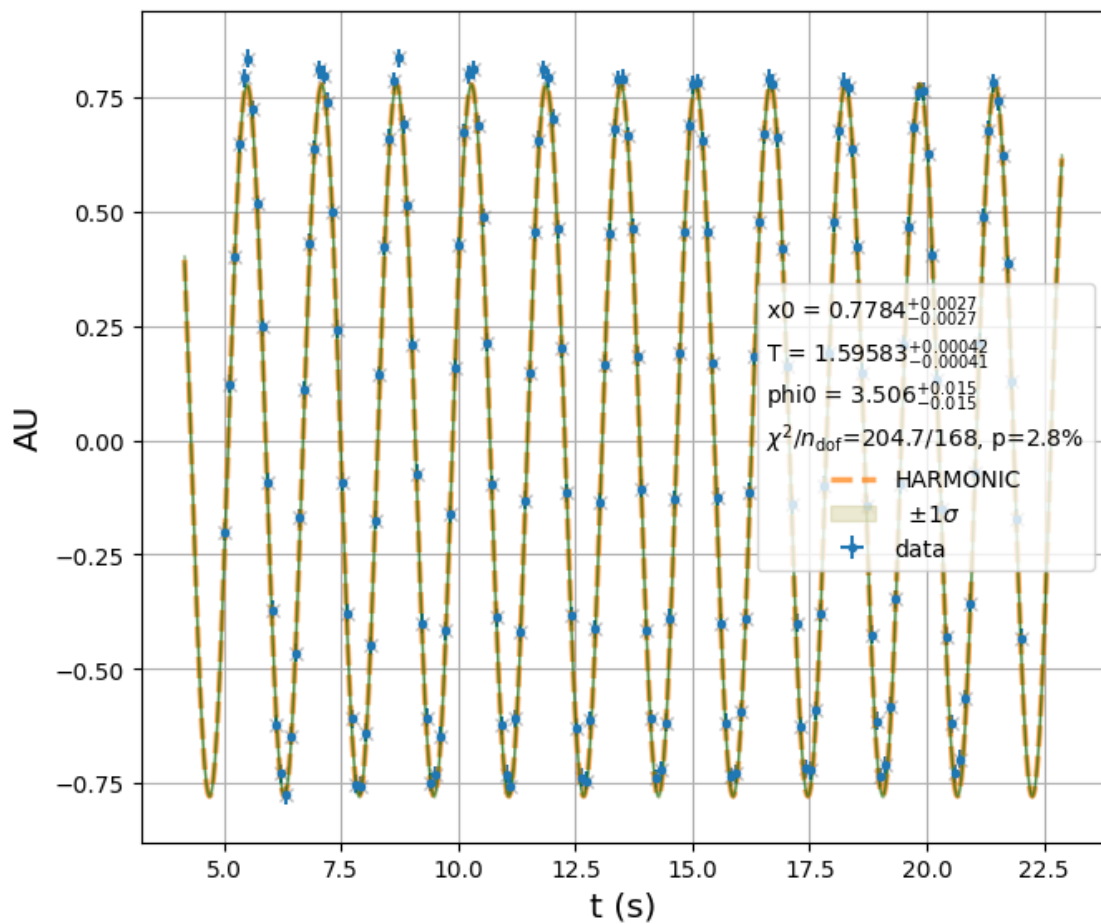
```

== Fit Result:

```

chi2: 205
parameter names:      ('x0', 'T', 'phi0')
parameter values:     [0.778 1.596 3.506]
neg. parameter errors: [-0.003 -0.    -0.015]
pos. parameter errors: [0.003 0.    0.015]
correlation matrix :
[[ 1.    -0.017 -0.016]
 [-0.017  1.    0.806]
 [-0.016  0.806  1.    ]]

```



Diskussion:

Die angepassten Parameter und ihre Unsicherheiten werden vom Skript zurückgegeben. Sie können aber auch aus dem Bild abgelesen werden. Wir geben sie hier als wichtige Zwischenergebnisse für **Aufgabe 3** tabellarisch an:

Tabelle 2.1

| Parameter | Wert | Unsicherheit Δ |
|-----------|-----------|-----------------------|
| x_0 | 0.778 | ± 0.003 |
| ϕ_0 | 3.51 | ± 0.02 |
| T | 1.59583 s | ± 0.00042 s |

Für die Werte, die wir in Formeln weiterverwenden wollen notieren wir 1–2 signifikante Stellen mehr. Dies ist in unserem Fall für T und ΔT der Fall.

2.5.5 Aufgabe 3: Ein einfaches Modell zur Bestimmung von g

Wir legen unserer ersten Messung das einfache Modell eines [mathematischen Pendels](#) zugrunde.

$$m l^2 \ddot{\phi} + m g l \phi = 0,$$

wobei m einer Punktmasse und l dem Abstand der Punktmasse vom Angelpunkt des Pendels entsprechen. Diese Differentialgleichung wird von harmonischen Schwingungen der Form

$$\phi(t) = \phi_0 \cos(\omega t + \delta) \quad (1)$$

gelöst, wobei ϕ_0 die Amplitude, ω die Kreisfrequenz und δ eine freie Phase der Schwingung sind. Mit Hilfe dieses Modells können Sie bei gegebenem l die Größe von g mit Hilfe der Gleichung

$$g = \frac{4\pi^2}{T^2} l \quad (2)$$

aus der Periode T der Schwingung bestimmen. Zusätzliche Information, die Sie zur Lösung dieser Aufgabe benötigen, finden Sie in der Datei `params/parameters_task_3.py`.

Aufgabe 3.1: Referenzmessung Bestimmen Sie einen Wert für T , aus einer einzelnen Periode, die Sie aus den Werten der angegebenen *csv*-Datei ermitteln. Überlegen Sie sich eine sinnvolle Unsicherheit ΔT und ermitteln Sie einen ersten Wert für g mit entsprechender Unsicherheit Δg mit Hilfe linearer Fehlerfortpflanzung. Vergleichen Sie das Ergebnis mit Ihrer Erwartung und setzen Sie es in Bezug zu Δg .

Aufgabe 3.2: Erste Verbesserung der Messung Erstellen Sie eine *yaml*-Datei für die Anpassung einer harmonischen Schwingung, wie in Gl. (1) angegeben, an die Daten aus der angegebenen *csv*-Datei, so dass Sie T aus einer Anpassung an die Daten bestimmen können. Notieren Sie sich die Qualität dieser Anpassung (quantifiziert durch die Größe χ^2/n_{dof}) und die ermittelten Werte mit entsprechenden Unsicherheiten für alle Parameter. Fügen Sie Ihrer Auswertung eine graphische Darstellung zu. Berechnen Sie aus den bestimmten Werten für T und ΔT verbesserte Abschätzungen für g und Δg . Vergleichen Sie das Ergebnis mit Ihrer Erwartung und setzen Sie es in Bezug zu Δg .

Anmerkungen: Als Referenzwert für Ihre Messungen können Sie den Wert

$$g_{\text{exp}} = (9.809599 \pm 0.000034) \text{ m/s}^2$$

verwenden, der aus der Global Gravity Database des Bureau Gravimetrique International (BGI) für die Stadt Mannheim (bei 49,49° nördlicher Breite und 8,53° westlicher Länge auf einer Referenzhöhe von 101 m) ausgelesen wurde.

2.5.6 Lösung:

Aufgabe 3.1: Referenzmessung Als Referenz haben wir die Periode T der Schwingung aus der einzelnen Messung eines Nulldurchgangs der Schwingung in x -Richtung bestimmt. Die Nullstellen haben wir mit dem folgenden das Skript *per Hand* bestimmt:

```
[8]: import pandas as pd
import numpy as np
from os import path
import yaml

# Proxy for a histogram fit. Values in {*} should be replaced
HEADSTRING=\
"""
# Input data:
# input file   : {file}
# raw_data     : {raw_data}
# -----
type: histogram

label: Intercepts
x_label: 'T (sec)'
y_label: 'Density p(T)'

n_bins: 30
bin_range: [{min}, {max}]

model_density_function: |
    def normal_distribution(x, mu={mean}, sigma={std}):
        return np.exp(-0.5*((x-mu)/sigma)**2)/np.sqrt(2.*np.pi*sigma**2)

# Data:
"""

def main(**kwargs):
    if not path.isfile(kwargs["filename"]):
        print("Could not find file %s. Please check filename and_
↪location"%kwargs["filename"])
        exit()
    print("Reading from file:", kwargs["filename"])
    # Read csv as pandas DataFrame. Rename coumns to x and f for convenience
    # reasons.
    df = pd.read_csv(kwargs["filename"]).rename(
        columns={
            kwargs["x_data"]:"x",
            kwargs["y_data"]:"f"
```

```

    }
    ).loc[:, "x":"f"]
    # Find intercept row; a change of sign in f is indicated by "-1" when going
    # from positive to negative and "+1" when going from negative to positive
    # values in f(x); pd.diff() returns the difference w.r.t. the previous line
    # and nan in case there is none.
    df["idx"] = ((df["f"]+df["f"].apply(np.abs))/2/df["f"]).diff()
    # Find linearly interpolated intercept in x.
    df["dx"] = df["idx"]*(df["x"]-df["f"]/df["f"].diff()*df["x"].diff())
    # Reduce df to values>0 and get differences between neighboring rows. Also
    # values<0 could be chosen. The restriction to values of the same sign
    # among other reasons is made to obtain the period T and not T/2.
    df = pd.DataFrame(df.loc[df["dx"]>0, "dx"].diff())
    # Dump to yaml in a format as expected from the PhyPraKit tool
    # "run_phyFit.py" for a histogram fit. The name of the output file is
    # the same as for the input file, only the ending is replaced from
    # ".csv" to "_intercepts.yaml".
    output_name=kwargs["filename"].replace(".csv","_intercepts.yaml")
    print("Saving file:", output_name)
    with open(output_name, "w") as f:
        f.write(
            HEADSTRING.format(
                file = kwargs["filename"],
                raw_data = kwargs["x_data"],
                min = kwargs["min"],
                max = kwargs["max"],
                mean= kwargs["mean"],
                std = kwargs["std"]
            )
        )
        yaml.dump(
            df.rename(
                columns={
                    # The name required for a histogram fit with the script
                    # "run_phyFit.py"
                    "dx":"raw_data",
                }
            ).dropna().to_dict(orient="list"),
            f
        )
main(
    # Name of the input file
    filename="RawData.csv",
    # Name of the column to dump to yaml as "x_data" (in case)
    x_data="Time (s)",
    # Name of the column to dump to yaml as "y_data" (in case)
    y_data="Linear Acceleration x (m/s^2)",

```

```

# Minimum for fitting -- potentially adjust by plotting
min=1.57,
# Maximum for fitting -- potentially adjust by plotting
max=1.63,
# Mean for fitting -- potentially adjust by plotting
mean=1.60,
# Standard deviation for fitting -- potentially adjust by plotting
std=0.004,
)

```

Reading from file: RawData.csv
Saving file: RawData_intercepts.yaml

Diskussion:

Das Skript liest die Datei RawData.csv ein und findet den Nullstellendurchgang dort, wo f sein Vorzeichen wechselt. An diesen Stellen wird jeweils x ausgelesen. Alle ermittelten Nullstellen werden in eine yaml-Datei geschrieben, mit deren Hilfe wir die Verteilung der Nulldurchgänge auch darstellen und ggf. den Mittelwert und die Standardabweichung durch Anpassung einer Gauß-Funktion bestimmen können. Wir haben ein bisschen mit den Möglichkeiten die Funktion run_phyFit.py zu konfigurieren herumgespielt und die folgende Abbildung und Anpassung zu unserer Auswertung zugefügt:

Abbildung 3.1

```
[25]: %run /opt/conda/bin/run_phyFit.py RawData_intercepts.yaml
```

```

*** /opt/conda/bin/run_phyFit.py received valid yaml data for fit:
** Type of Fit: histogram
type: histogram
label: Intercepts
x_label: T (sec)
y_label: Density p(T)
n_bins: 30
bin_range: [1.57, 1.63]
model_density_function: def normal_distribution(x, mu=1.6, sigma=0.004):
    return np.exp(-0.5*((x-mu)/sigma)**2)/np.sqrt(2.*np.pi*sigma**2)

raw_data: [0.18696261058181723, 0.10639414857535812, 0.053363951922565156,
0.2906315887751335, 0.30163355025787064, 0.33722706649904644,
0.08738367870884534, 0.2726986578324808, 0.22135853130093164,
0.20322700488908918, 0.15603121810467258, 0.1014620097141985,
0.18510880495225202, 0.10617526965210455, 0.09543041315178602,
0.036787580676103104, 0.5471304539945918, 1.6087833507842504,
1.5901592627532297, 1.5903205535582146, 1.6013088500981336, 1.597084206589651,
1.587573240466929, 1.6022055199940262, 1.5936263342556156, 1.5926460754382177,
1.5994342792404979, 1.5982654810959254, 1.5933581612750949, 1.595139838941357,
1.6012551881187136, 1.5881453750406536, 1.6002204844971502, 1.5934354746863875,
1.5928952576689248, 1.59804827782704, 1.5959103407029716, 1.5927674458636432,

```


1.5963222394228538, 1.5947564700226025, 1.5949318595662945, 1.5980703177330469,
1.5962458939658788, 1.5902062114050324, 1.5971580420183003, 1.5992804590816405,
1.593897834782382, 1.5942534641494746, 1.5967069646658558, 1.59573440401644,
1.5952845188547968, 1.595160697720857, 1.5990820441540592, 1.5924001812948418,
1.5978719869089062, 1.5948313056226624, 1.5930611520965243, 1.594899414674913,
1.5972814759821574, 1.5932072410261071, 1.5967880113659163, 1.5946035399489773,
1.5943316283381677, 1.598216213059942, 1.5927765413347572, 1.5933485946536194,
1.59813406487595, 1.5968184262007554, 1.594210681059252, 1.5958692992417696,
1.5932573462629307, 1.596804600989742, 1.5981565979206067, 1.5900946992646539,
1.5968326277373421, 1.593416686448819, 1.5946597965041178, 1.5967154154230911,
1.592501216842848, 1.5962867952465984, 1.5950243497584893, 1.594409641940544,
1.5976010527823377, 1.5924311969646112, 1.5954337923968893, 1.5898976437447772,
1.5949545449371811, 1.5981342266635465, 1.5967855411695098, 1.5931134703679817,
1.593927309963206, 1.59606927713169, 1.592014553194204, 1.5980122860816266,
1.5906641052554988, 1.5963834580982876, 1.5933031058590927, 1.5902111530146499,
1.5974092947818406, 1.596098145504527, 1.5924475023166735, 1.5897738269043202,
1.5982283673573363, 1.5937188895829024, 1.596102048995391, 1.5987675861279342,
1.5919390943075769, 1.594093651756566, 1.5918969118988855, 1.5926983380209379,
1.5989036080319465, 1.591082646288669, 1.5907801400928463, 1.5943059910980821,
1.594804539671145, 1.593925834810932, 1.5997810006291218, 1.5894495355692868,
1.5959966697239167, 1.5902004645570855, 1.5903683546484046, 1.5953745036236455,
1.5970497773897137, 1.597926463920743, 1.5908800592855528, 1.5941619701569039,
1.5879358542822217, 1.5935130748887332, 1.5974380534576653, 1.589390183412462,
1.5980650373635115, 1.5933473275948131, 1.5920410763926327, 1.5936338307898552,
1.592542699930874, 1.5973325876167905, 1.5896732273709233, 1.5946218436396578,
1.5970778501173015, 1.5882172729017157, 1.5961040397132251, 1.5950518784503345,
1.5892434829472109, 1.5977499737118421, 1.5906656957577638, 1.5958753778890866,
1.5929557608060065, 1.5960640960445573, 1.5881597973344128, 1.5912819663141988,
1.5958729763867723, 1.593799853132424, 1.5963392724033554, 1.5923912792062538,
1.5929074347982635, 1.5886472878464986, 1.5968058361521287, 1.589685873159283,
1.5947093772880123, 1.5938013870720908, 1.5971106138563016, 1.587974444922338,
1.5934514841457883, 1.5906804346296042, 1.5965931606560844, 1.593932453951595,
1.5952680880677121, 1.5899661807029588, 1.592467398267587, 1.5943748969451406,
1.5925819229450724, 1.5957848155791794, 1.5898593205507723, 1.5973132905198213,
1.5873372268420098, 1.5896419565148392, 1.5952821871809988, 1.5966370114763322,
1.589505518190606, 1.5936403899271454, 1.582821462293623, 1.6014725010986126,
1.5946620459195628, 1.5959762967170832, 1.5846536726575664, 1.6030884254934676,
1.5909383417150025, 1.5903436588513955, 1.5914543562374206, 1.5924354943271624,
1.59462239036867, 1.5968011093417545, 1.5851580096724547, 1.5938530569557088,
1.5949240806921239, 1.5892788829889355, 1.5983132858593763, 1.5871631063343443,
1.600998027395633, 1.5920503343204473, 1.591603169205655, 1.5898004786312185,
1.5928178307121925, 1.5960134614672938, 1.5857849007296068, 1.590141088792052,
1.6023674350361148, 1.5906293631812218, 1.5915747916019996, 1.5915728760789989,
1.5941071859768954, 1.588897705218585, 1.5955434502484422, 1.58957544539453,
1.5988774908137202, 1.030765114386611, 0.19752182815796004, 0.08697505147722495,
0.0644848739864301, 0.012718461284066507, 0.07732660077323317,
0.09457219822644447, 0.2838549589575905, 0.042695801838647185,
0.18922884271768226, 0.19349572585292663, 0.07209052059243959,

0.1825258506020191, 0.09556974174824973, 0.16788171933188778,
0.13270116902964446, 0.07192151371179989, 0.032174619815862116,
0.03349985434505243, 0.03599217163838375, 0.0905867154449993,
0.04036154989739771, 0.025143491284097763, 0.03840638562928689,
0.019439111103451978, 0.10857904117983708, 0.024467244072127414,
0.06998198827756141, 0.03615454506234528, 0.12853772877213032,
0.04048958721159579]

== Fit Result:

chi2: 12.6

parameter names: ('mu', 'sigma')

parameter values: [1.594 0.004]

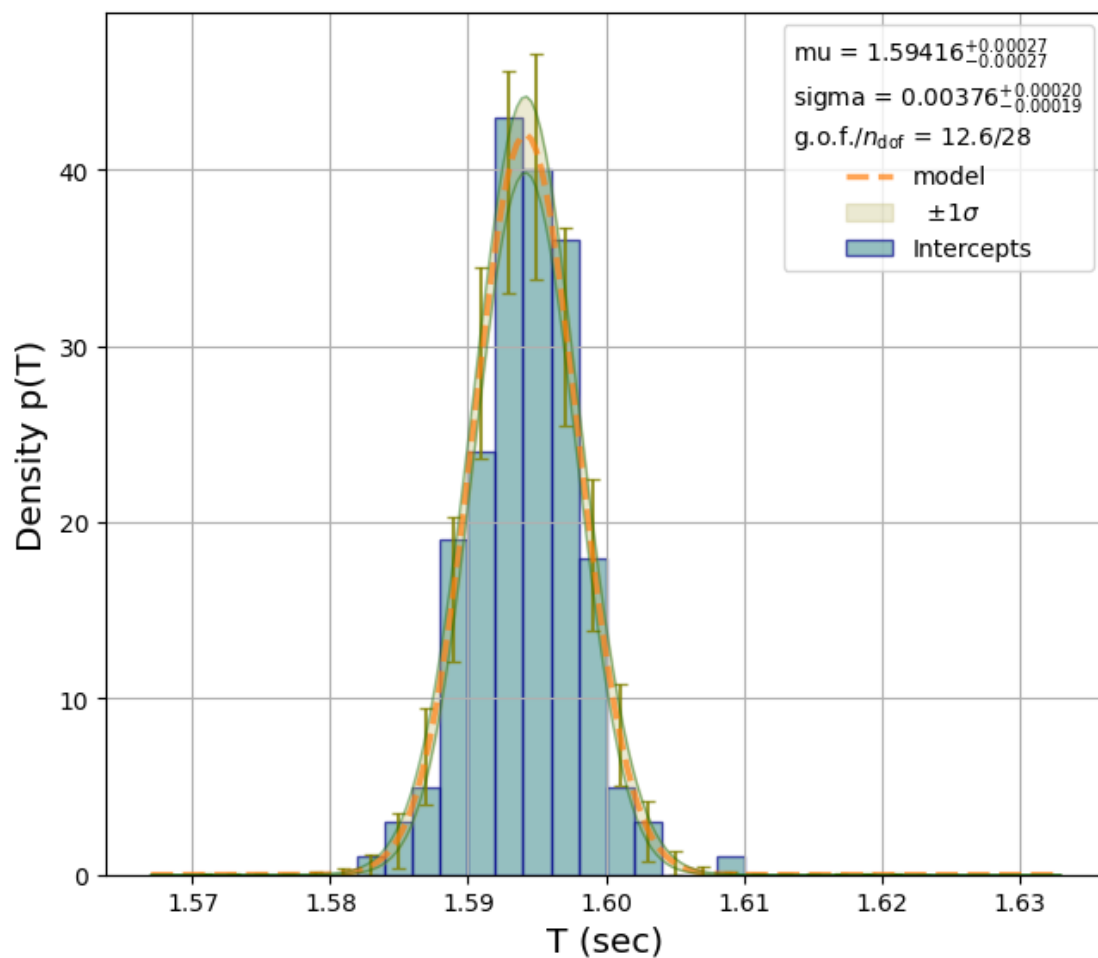
neg. parameter errors: [-0. -0.]

pos. parameter errors: [0. 0.]

correlation matrix :

[[1.000e+00 3.235e-04]

[3.235e-04 1.000e+00]]



Diskussion:

Als einzelne Referenzmessung haben wir uns den Wert aus **Zeile 49 in der *yaml*-Datei** ausgewählt:

$$T = (1.5951 \pm 0.0178) \text{ s.}$$

Wir haben ΔT mit $\sqrt{2} \cdot 0.0125 \text{ s}$ abgeschätzt. Das entspricht der Unsicherheit Δx auf einen Wert in x , aus der Datei `params/uncertainties_data.py`, mit einem zusätzlichen Faktor $\sqrt{2}$ versehen, weil wir zwei Werte in x voneinander abziehen (klassische lineare Fehlerfortpflanzung). Die Werte für l und Δl erhalten wir aus der Datei `params/parameters_task_3.py`. Wir verwenden l als den Abstand des Schwerpunkts des Smartphones vom Aufhängepunkt des Pendels.

Wir bestimmen den Wert g aus dem folgenden Abschnitt:

```
[47]: import numpy as np
from params.parameters_task_3 import l, l_UPPER, l_LOWER
dl=(l_UPPER-l_LOWER)/2

def gmath(T,dT):
    """
    g for the mathematical pendulum in m/s**2.

    T    : Measured value,
    dT   : Uncertainty in T.
    """
    g = 4*np.pi**2/T/T*l
    dg= np.sqrt(
        (g/l*dl)**2           # Uncertainty on l
        + (-2*g/T*dT)**2      # Uncertainty on T
    )
    return (g,dg)

print("Reference result (3.1):\n")
g31 = gmath(1.59583, 0.0178)
print("g31=", g31[0], "+/-", g31[1])
gexp=9.809599
print("gexp=", gexp)
print("g31/gexp=", g31[0]/gexp)
dgexp=0.000034
print("pull=", (g31[0]-gexp)/np.sqrt(g31[1]**2+dgexp**2))
```

Reference result (3.1):

```
g31= 9.74297904571733 +/- 0.21748590871929
gexp= 9.809599
g31/gexp= 0.9932086974928669
pull= -0.30631848224507885
```

Diskussion:

Den ermittelten Messwert von g bezeichnen wir im Folgenden mit $g_{3.1}$. Ein Vergleich von $g_{3.1}$ mit dem in der Anleitung angegebenen Referenzwert ist im folgenden zusammengefasst:

- **Messung von g :** $g_{3.1} = (9.75 \pm 0.218) \text{ m/s}^2$
- **Erwartung für g :** $g_{\text{exp}} = (9.809599 \pm 0.000034) \text{ m/s}^2$
- **Relativer Unterschied:** $g_{3.1}/g_{\text{exp}} = 0.994$
- **Pull:** $\delta = (g_{3.1} - g_{\text{exp}})/\sqrt{\Delta g_{3.1}^2 + \Delta g_{\text{exp}}^2} = -0.26494$

Ein relativer Unterschied von unter 1% beeindruckt. Aussagekräftiger ist jedoch der sogenannte *Pull*. Dabei handelt es sich um die Differenz aus Messung und Erwartung geteilt durch die Unsicherheit auf die Differenz, die wir wiederum durch lineare Fehlerfortpflanzung aus den Unsicherheiten der Messung und der Erwartung bestimmt haben. Für eine Erwartung, die mit der Messung kompatibel ist erwarten wir einen *Pull* $\lesssim 1$. Ein Wert von $\delta = -0.26494$ ist zufriedenstellend.

Zusammenfassend stellen wir eine **gute Übereinstimmung der Messung mit der Erwartung** fest!

Aufgabe 3.2: Erste Verbesserung der Messung Für diese Aufgabe verwenden wir die Anpassung einer harmonischen Schwingung an 10 Perioden aus Aufgabe 2 (**Abbildung 2.2**). Für die Güte der Anpassung hatten wir den folgenden Wert erhalten:

$$\chi^2/n_{\text{dof}} = 204.7/168.$$

Unter Annahme einer χ^2 -Verteilung mit 168 Freiheitsgraden entspricht dies einem p -Wert von 2.8%. Dieser Wert erscheint niedrig, was ein Hinweis darauf sein kann, dass das zugrundeliegende Modell die Daten nicht gut beschreibt. Allerdings kann es bei häufiger Wiederholung einer solchen Messung in 2.8% aller Ausgänge vorkommen, dass die Anpassung des Modells einen p -Wert $\leq 2.8\%$ erhält, selbst wenn das Modell die Daten beschreiben kann.

Mit Hilfe der Werte für T und ΔT aus **Tabelle 2.1** haben wir g neu berechnet und bezeichnen diesen Wert im Folgenden als $g_{3.2}$:

```
[59]: print("Improved result (3.2):\n")
g32 = gmath(1.5951, 0.00042)
print("g32=", g32[0], "+/-", g32[1])
gexp=9.809599
print("gexp=", gexp)
print("g32/gexp=", g32[0]/gexp)
dgexp=0.000034
print("pull=", (g32[0]-gexp)/np.sqrt(g32[1]**2+dgexp**2))
```

Improved result (3.2):

```
g32= 9.751898865412716 +/- 0.009303806024100384
gexp= 9.809599
```

$g_{32}/g_{\text{exp}} = 0.9941179925308584$
 $\text{pull} = -6.20173605893686$

Diskussion:

Der Vergleich von $g_{3.2}$ mit unserer Erwartung ist im folgenden zusammengefasst:

- **Messung von g :** $g_{3.2} = (9.752 \pm 0.0093) \text{ m/s}^2$
- **Erwartung für g :** $g_{\text{exp}} = (9.809599 \pm 0.000034) \text{ m/s}^2$
- **Relativer Unterschied:** $g_{3.2}/g_{\text{exp}} = 0.994$
- **Pull:** $(g_{3.2} - g_{\text{exp}})/\sqrt{\Delta g_{3.2}^2 + \Delta g_{\text{exp}}^2} = -6.202$

Wie zuvor weicht diese Messung nur um 0.6% von der Erwartung ab. Ein *Pull* von $-6.202(!)$ kann aber nicht ignoriert werden. Wir müssen zusammenfassend feststellen, dass diese Abweichung im Rahmen der angegebenen Unsicherheiten signifikant ist.

Das zugrundeliegende Modell ist vermutlich zu einfach, um die Realität vollständig abbilden zu können. Dies ist eine Unzulänglichkeit und damit ein (vermeidbarer) **Fehler** der bisherigen Auswertung. Dieser Fehler wurde bei der Referenzmessung aus Aufgabe 3.1 durch den großen Wert von ΔT überdeckt. Durch die deutlich präzisere Bestimmung von T tritt sie jetzt offen zutage.

2.5.7 Aufgabe 4: Erweiterung des ursprünglichen Modells

Eine offensichtliche Unzulänglichkeit des vorherigen Modells besteht in der Vernachlässigung der physikalischen Ausdehnungen des Pendels. Wir werden unser Modell entsprechend erweitern. Die Formel zur Bestimmung von g nimmt dadurch die folgende Form an:

$$g = \frac{4\pi^2}{T^2} \frac{I}{m\ell},$$

wobei I und m dem Trägheitsmoment und der Masse der gesamten Pendelkonstruktion (inklusive aller Montageteile und Smartphone!) und ℓ dem Abstand des Schwerpunkts dieser Konstruktion vom Angelpunkt des Pendels entsprechen. Zusätzliche Information zu den im Folgenden diskutierten Modellen finden Sie in der Datei `params/parameters_task_4.py`

Aufgabe 4.1: Erweiterung des Modells Modifizieren Sie Ihr Modell, so dass es dem Modell eines [physikalischen Pendels](#) entspricht, berechnen Sie aus den in Aufgabe 3.2 bestimmten Werten für T und ΔT eine neue Abschätzung von g und Δg und diskutieren Sie das Ergebnis.

Aufgabe 4.2: Direkte Bestimmung von g Sie können g und Δg auch direkt aus der Anpassung bestimmen. Formulieren Sie ihre Modellfunktion entsprechend um, führen Sie die Anpassung erneut durch und vergleichen Sie die Ergebnisse mit den zuvor bestimmten Ergebnissen. Beachten Sie, dass der Wert von Δg aus der Anpassung nur die Unsicherheiten der Datenpunkte widerspiegelt. Wie würden Sie den Einfluss der Variationen in I , m , und ℓ auf Δg abschätzen?

Aufgabe 4.3: Übergang zu einer [gedämpften Schwingung](#) Das Pendel erfährt in seiner Bewegung zusätzlich eine Dämpfung. Legen wir ein Modell linearer Dämpfung zugrunde verändert sich die Formel zur Bestimmung von g wie folgt:

$$g = \left(\frac{4\pi^2}{T^2} + \frac{1}{\tau^2} \right) \frac{I}{m\ell},$$

wobei τ einer Abklingzeit in Sekunden entspricht. Verändern Sie ihr Modell entsprechend und beantworten Sie die folgenden Fragen: Ist das zugrundeliegende Modell mit den Daten kompatibel? Wie könnten Sie die Hypothese, dass das zugrundeliegende Modell die Daten beschreiben kann, besser testen? Wie groß ist der Einfluss, den dieser Effekt ggf. auf die Messung von g hat?

2.5.8 Lösung:

Aufgabe 4.1: Erweiterung des Modells Mit Hilfe der Werte für T und ΔT aus **Tabelle 2.1** berechnen wir g unter Verwendung des Modells eines [physikalischen Pendels](#) neu und bezeichnen diesen neuen Wert im Folgenden mit $g_{4.1}$:

```
[48]: import numpy as np
from params.parameters_task_4 import I, I_UPPER, I_LOWER, m, m_UPPER, m_LOWER, ell, ell_UPPER, ell_LOWER
dI=(I_UPPER-I_LOWER)/2
dm=(m_UPPER-m_LOWER)/2
dell=(ell_UPPER-ell_LOWER)/2

def gphys(T,dT):
    """
    g for the physical pendulum in m/s**2.

    T      : Measured value,
    dT     : Uncertainty in T.
    """
    g = (4*np.pi**2/T/T)*I/m/ell
    dg= np.sqrt(
        (g/I*dI)**2           # Uncertainty in I
        + (-g/m*dm)**2        # Uncertainty in m
        + (-g/ell*dell)**2     # Uncertainty in ell
        + (-2*g/T*dT)**2       # Uncertainty in T
    )
    return (g,dg)

print("Extended model (4.1):\n")
g41 = gphys(1.5951, 0.00042)
print("g41=", g41[0], "+/-", g41[1])
gexp=9.809599
print("gexp=", gexp)
print("g41/gexp=", g41[0]/gexp)
dgexp=0.000034
```

```
print("pull=", (g41[0]-gexp)/np.sqrt(g41[1]**2+dgexp**2))
```

Extended model (4.1):

```
g41= 9.77999136362522 +/- 0.18873197993921953
gexp= 9.809599
g41/gexp= 0.9969817689413419
pull= -0.15687662421532525
```

Diskussion:

Dabei haben wir die folgenden Unsicherheiten in die Abschätzung von Δg zusätzlich aufgenommen:

- Eine Unsicherheit auf das Trägheitsmoment I der gesamten Pendelkonstruktion ($\Delta I = \pm 0.00257 \text{ kg/m}^2$)
- Eine Unsicherheit auf die Masse m der gesamten Pendelkonstruktion ($\Delta m = \pm 5 \text{ g}$)
- Eine Unsicherheit auf den Abstand ℓ des Schwerpunkts des Pendels von seinem Angelpunkt ($\Delta \ell = \pm 7 \text{ mm}$)

Diese Unsicherheiten haben wir aus der Datei `params/parameters_task_4.py` entnommen. Mittels linearer Fehlerfortpflanzung haben wir diese Unsicherheiten quadratisch zu ΔT (aus **Tabelle 2.1**) addiert.

Der Vergleich von $g_{4.1}$ mit unserer Erwartung ist im Folgenden zusammengefasst:

- **Messung von g :** $g_{4.1} = (9.78 \pm 0.19) \text{ m/s}^2$
- **Erwartung für g :** $g_{\text{exp}} = (9.809599 \pm 0.000034) \text{ m/s}^2$
- **Relativer Unterschied:** $g_{4.1}/g_{\text{exp}} = 0.997$
- **Pull:** $(g_{4.1} - g_{\text{exp}})/\sqrt{\Delta g_{4.1}^2 + \Delta g_{\text{exp}}^2} = -0.157$.

Mit einem *pull* von -0.157 ist das erweiterte Modell wieder kompatibel mit den Daten!

Der Zentralwert von $g_{4.1}$ unterscheidet sich nur gering von $g_{3.2}$. Stattdessen ist $\Delta g_{4.1}$ im Vergleich zu $\Delta g_{3.2}$ wieder größer geworden und hat damit fast die Größe von $\Delta g_{3.1}$ angenommen! Es ist zu beachten, dass die Änderung des Modells keinen Einfluss auf die Anpassung und damit auch nicht auf den niedrigen p -Wert der Anpassung hat. Der große *Pull* von -6.202 , aus Aufgabe 3.2 hat also nichts mit dem niedrigen p -Wert der Anpassung zu tun. Er kam dadurch zustande, dass unser Modell für die zu berücksichtigenden Unsicherheiten unvollständig war.

Aufgabe 4.2: Direkte Bestimmung von g Um g direkt aus der Anpassung an die Daten bestimmen zu können haben wir unser Modell wie folgt modifiziert (`yaml/RawData_down_sampled_500_2200_10_G_HARMONIC.yaml`):

```
# -----
# Input data:
# input file : Master/RawData_down_sampled_500_2200_10.csv
# x_data     : Time (s)
# y_data     : Linear Acceleration x (m/s^2)
# -----
model_label: "G_HARMONIC"
```

```

model_function: |
    def model(x, x0=0.75, g=9.8, phi0=0):
        return x0*np.cos(np.sqrt(0.789*0.473*g/0.23523)*x+phi0)
y_label: "AU"
y_errors: 0.02

x_errors: 0.0125
x_label: "t (s)"

# Data:
x_data:
- 5.024349958
...

```

Bei den fest ins Modell kodierten Zahlenwerten handelt es sich um die angegebenen Werte aus der Datei: `params/parameters_task_4.py`. Damit erhalten wir das folgende Ergebnis der Anpassung:

Abbildung 4.1

```

[43]: %run /opt/conda/bin/run_phyFit.py yaml/
      ↪RawData_down_sampled_500_2200_10_G_HARMONIC.yaml

```

```

*** /opt/conda/bin/run_phyFit.py received valid yaml data for fit:
** Type of Fit: xy
model_label: G_HARMONIC
model_function: def model(x, x0=0.75, g=9.8, phi0=0):
                  return x0*np.cos(np.sqrt(0.789*0.473*g/0.23523)*x+phi0)

y_label: AU
y_errors: 0.02
x_errors: 0.0125
x_label: t (s)
x_data: [5.024349958, 5.124306958, 5.224263958, 5.324220958, 5.424177958,
5.524134958, 5.624092958, 5.72404975, 5.82400675, 5.92396375, 6.02392075,
6.12387775, 6.22383475, 6.32379175, 6.42374975, 6.52370675, 6.62366375,
6.723620583, 6.823577583, 6.923534583, 7.023491583, 7.123448583, 7.223405583,
7.323363583, 7.423320583, 7.523277583, 7.623234583, 7.723192, 7.823149,
7.923106, 8.023063, 8.12302, 8.222977, 8.322935, 8.422892, 8.522849, 8.622806,
8.722762958, 8.822719958, 8.922676958, 9.022633958, 9.122590958, 9.222547958,
9.322504958, 9.422461958, 9.522419958, 9.622376958, 9.722333708, 9.822290708,
9.922247708, 10.02220471, 10.12216171, 10.22211871, 10.32207571, 10.42203271,
10.52198971, 10.62194671, 10.72190371, 10.82186071, 10.92181771, 11.02177571,
11.12173271, 11.22168971, 11.32164671, 11.42160371, 11.52156071, 11.62151771,
11.72147462, 11.82143162, 11.92138862, 12.02134562, 12.12130262, 12.22125962,
12.32121662, 12.42117362, 12.52113062, 12.62108762, 12.72104546, 12.82100246,
12.92095946, 13.02091746, 13.12087446, 13.22083146, 13.32078846, 13.42074546,
13.52070246, 13.62065946, 13.72061571, 13.82057271, 13.92052971, 14.02048671,
14.12044371, 14.22040071, 14.32035771, 14.42031471, 14.52027171, 14.62022871,

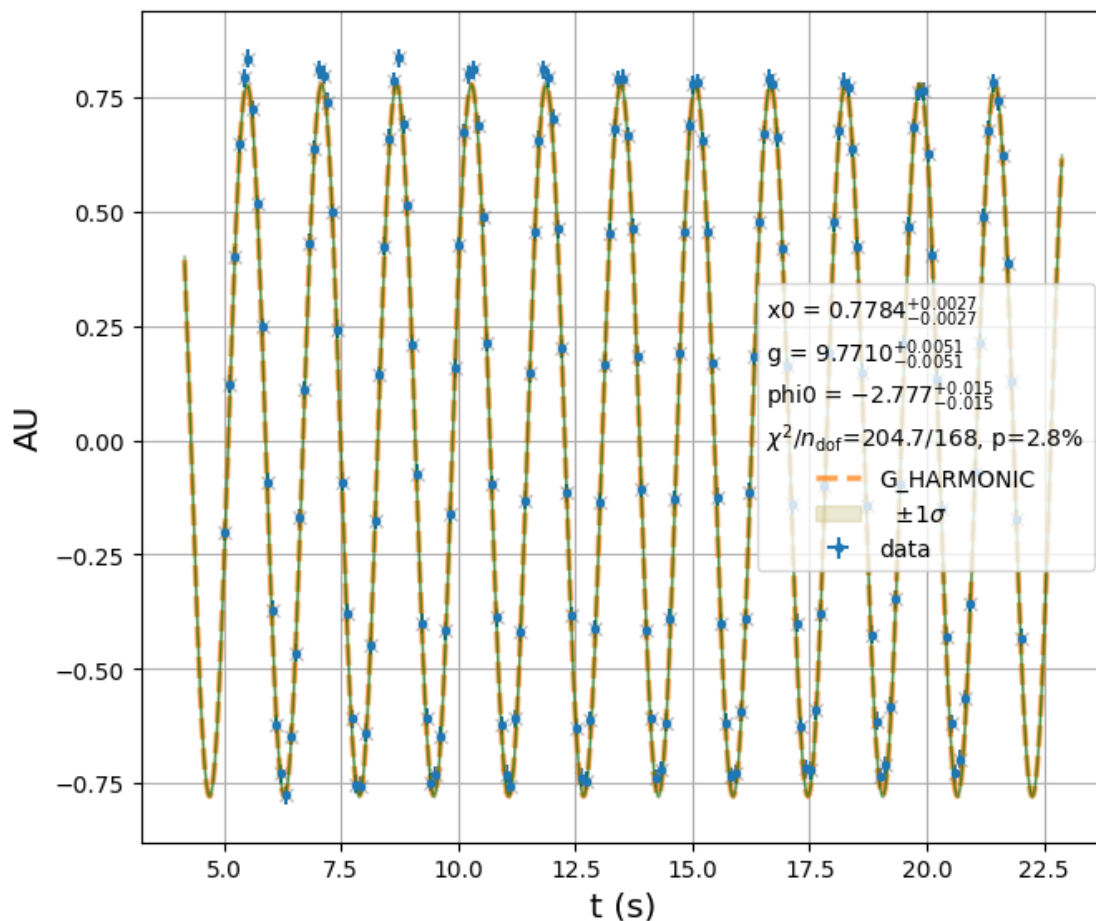
```


14.72018637, 14.82014337, 14.92010037, 15.02005737, 15.12001437, 15.21997137,
 15.31992837, 15.41988537, 15.51984237, 15.61979937, 15.71975646, 15.81971346,
 15.91967046, 16.01962746, 16.11958446, 16.21954146, 16.31949846, 16.41945546,
 16.51941246, 16.61936946, 16.71932558, 16.81928258, 16.91923958, 17.01919658,
 17.11915458, 17.21911158, 17.31906858, 17.41902558, 17.51898258, 17.61893958,
 17.71889671, 17.81885371, 17.91881071, 18.01876771, 18.11872471, 18.21868171,
 18.31863871, 18.41859571, 18.51855271, 18.61850971, 18.71846671, 18.81842371,
 18.91838071, 19.01833771, 19.11829471, 19.21825171, 19.31820871, 19.41816571,
 19.51812271, 19.61807971, 19.71803733, 19.81799433, 19.91795133, 20.01790833,
 20.11786533, 20.21782233, 20.31777933, 20.41773633, 20.51769333, 20.61765033,
 20.71760708, 20.81756408, 20.91752108, 21.01747708, 21.11743408, 21.21739208,
 21.31734808, 21.41730508, 21.51726208, 21.61721908, 21.71717608, 21.81713308,
 21.91709008, 22.01704708]
 y_data: [-0.1997231989, 0.1225266681, 0.4020419037, 0.6469383518, 0.7932787375,
 0.8341633464, 0.7236726053, 0.5166886264, 0.2480305557, -0.09172321628,
 -0.3712864995, -0.6222052874, -0.728521482, -0.7767518748, -0.6462890916,
 -0.4653517906, -0.1693552458, 0.1100302364, 0.4309675679, 0.6369901914,
 0.8109249035, 0.7987021031, 0.7397423619, 0.4993210072, 0.242197081,
 -0.09183958501, -0.3778064739, -0.6088913966, -0.7521712035, -0.7551054094,
 -0.6405515112, -0.4490290688, -0.1754098915, 0.1433068137, 0.4238335993,
 0.6607164384, 0.7861605565, 0.8358004206, 0.6911543387, 0.5139052787,
 0.2101174652, -0.07397222986, -0.4009491321, -0.6076723618, -0.7486201894,
 -0.7316711578, -0.6475558908, -0.41658762, -0.1626306555, 0.158359301,
 0.4260544449, 0.6725575617, 0.8006421362, 0.8104562491, 0.6894230513,
 0.4883145739, 0.2144678302, -0.09728084858, -0.3876322081, -0.6239515583,
 -0.7300468013, -0.7555174188, -0.6094878857, -0.4202097143, -0.1315586977,
 0.145720654, 0.4559011187, 0.6575324102, 0.8117785238, 0.7923597014,
 0.7034183619, 0.4620813332, 0.2018080601, -0.1154284224, -0.3831218487,
 -0.6291048557, -0.7379914863, -0.7454776034, -0.6127069606, -0.4129662932,
 -0.1345960338, 0.166161387, 0.4537871309, 0.6826964875, 0.7889552332,
 0.791376419, 0.6656307308, 0.4643466177, 0.1837737122, -0.1054909332,
 -0.4143776649, -0.609118232, -0.7404253167, -0.7217743067, -0.6179310458,
 -0.3895465134, -0.1298084798, 0.1914881265, 0.4578757234, 0.6872261792,
 0.7793857531, 0.7828821688, 0.6544036382, 0.4566309974, 0.1680467851,
 -0.1247703727, -0.3992372136, -0.6183909293, -0.73671577, -0.7281120307,
 -0.5945850141, -0.3906517103, -0.1125802436, 0.1833649187, 0.4783845399,
 0.6695181059, 0.7909719379, 0.7806059939, 0.6631938305, 0.4194446792,
 0.1616605658, -0.1409690604, -0.4006429939, -0.6256357026, -0.717948839,
 -0.7221022625, -0.5911024124, -0.3780585985, -0.09839636028, 0.1903169292,
 0.4779136198, 0.6773630228, 0.7846111905, 0.7712306415, 0.6392149839,
 0.4244822017, 0.1476163213, -0.1418761097, -0.4254862417, -0.6149222879,
 -0.735367329, -0.709930406, -0.5818266819, -0.3479699596, -0.09709096938,
 0.2109081915, 0.4670812143, 0.6837134644, 0.7614358243, 0.7637974416,
 0.6275016953, 0.4065399474, 0.1342015295, -0.1505459225, -0.4289698666,
 -0.6198666935, -0.7288435174, -0.6980264251, -0.5651871872, -0.3557519093,
 -0.0691921402, 0.2131352132, 0.4881150377, 0.6783859932, 0.7830247678,
 0.7413724194, 0.6234380997, 0.3864223827, 0.1303110115, -0.1704625623,
 -0.4323219292]

```

*** Fit Result:
chi2: 205
parameter names:      ('x0', 'g', 'phi0')
parameter values:     [ 0.778  9.771 -2.777]
neg. parameter errors: [-0.003 -0.005 -0.015]
pos. parameter errors: [0.003 0.005 0.015]
correlation matrix :
[[ 1.      0.017 -0.016]
 [ 0.017  1.     -0.805]
 [-0.016 -0.805  1.    ]]

```



Diskussion:

Den Wert von g aus der Anpassung bezeichnen wir im Weiteren mit $g_{4.2}$. Wir stellen fest, dass wir bis auf die zweite Nachkommastelle das gleiche Ergebnis für $g_{4.2}$, wie zuvor für $g_{4.1}$ erhalten. Das ist nicht verwunderlich, denn mit dem neuen Modell, aus dem wir g direkt bestimmen, integrieren wir die Rechenschritte, die wir zuvor *per Hand* ausgeführt haben in die Anpassung. Uns fällt insbesondere auf, dass der p -Wert zum p -Wert der vorherigen Anpassung identisch ist.

Wir reflektieren noch darüber, was die viel geringere Unsicherheit von $\Delta g_{4.2} = 0.005 \text{ m/s}^2$ im Vergleich zu $\Delta g_{4.1}$ bedeuten sollte und kommen zu der folgenden Antwort: Der Wert $\Delta g_{4.2}$, wie wir ihn aus der Anpassung erhalten repräsentiert nur die Unsicherheiten auf die Datenpunkte, wie wir sie aus der Datei `params/uncertainties_data.py` ausgelesen haben.

Um die Unsicherheiten aufgrund von ΔI , Δm und $\Delta \ell$ aus der Anpassung abschätzen zu können müssen wir die Parameter einzeln und Schritt für Schritt, im Rahem ihrer jeweiligen Unsicherheiten, nach oben und unten variieren und die Anpassung nach jeder Variation erneut durchführen. Diese Prozedur gilt es, bei drei Größen, für alle Parametervariationen, insgesamt sechs mal durchzuführen.

Im Experiment würde man die unzureichenden Kenntnisse von I , m und ℓ als **systematische Unsicherheiten** bezeichnen. Die Variationen der Parameter bei der Signalbestimmung als **systematische Variationen**.

Nach wiederholter Durchführung der obigen Anpassung gelangen wir zum folgenden Ergebnis für $g_{4.2}$:

```
[60]: print("Direct estimate of g (4.2):\n")
      # Unsicherheit aus wiederholter Anpassung nach Variation
      # von I, m, ell innerhalb der angegebenen Unsicherheiten
      # mit anschließender quadratischer Addition
      g42 = (9.771, 0.19)
      print("g42=", g42[0], "+/-", g42[1])
      gexp=9.809599
      print("gexp=", gexp)
      print("g42/gexp=", g42[0]/gexp)
      dgexp=0.000034
      print("pull=", (g42[0]-gexp)/np.sqrt(g42[1]**2+dgexp**2))
```

Direct estimate of g (4.2):

```
g42= 9.771 +/- 0.19
gexp= 9.809599
g42/gexp= 0.9960651806460183
pull= -0.20315262832625228
```

Diskussion:

Der Vergleich von $g_{4.2}$ (einschließlich systematischer Variationen) mit unserer Erwartung ist im Folgenden zusammengefasst:

- **Messung von g :** $g_{4.2} = (9.77 \pm 0.19) \text{ m/s}^2$
- **Erwartung für g :** $g_{\text{exp}} = (9.809599 \pm 0.000034) \text{ m/s}^2$
- **Relativer Unterschied:** $g_{4.2}/g_{\text{exp}} = 0.996$
- **Pull:** $(g_{4.2} - g_{\text{exp}})/\sqrt{\Delta g_{4.2}^2 + \Delta g_{\text{exp}}^2} = -0.203$

Wir erhalten also ein identisches Ergebnis zu Aufgabe 4.1. Der Unterschied im *pull* ist auf den leicht anderen Zentralwert für $g_{4.2}$ zurückzuführen, wie sich leicht nachprüfen lässt.

Aufgabe 4.3: Übergang zu einer gedämpften Schwingung Im nächsten Schritt erweitern wir das Modell um den Umstand, dass die Schwingung des Pendels gedämpft wird. Wir versuchen es mit einem Modell mit linearer Dämpfung. Ein solches Modell führt zu einer Schwingung der Form:

$$\phi(t) = \phi_0 e^{-t/\tau} \cos(\omega t + \delta),$$

wobei τ ein weiterer zu bestimmender Parameter der Anpassung wird. Das entsprechende Modell in unsrer *yaml*-Datei (*yaml/RawData_down_sampled_500_2200_20_G_DAMPED.yaml*) nimmt dadurch die folgende Form an:

```
# -----
# Input data:
# input file   : Master/RawData_down_sampled_500_2200_10.csv
# x_data      : Time (s)
# y_data      : Linear Acceleration x (m/s^2)
# -----
model_label: "G_DAMPED"

model_function: |
    def model(x, tau=240., x0=0.75, g=9.8, phi0=0):
        return x0*np.exp(-x/tau)*np.cos(np.sqrt(0.789*0.473*g/0.23523)*x+phi0)

y_label: "AU"
y_errors: 0.02

x_errors: 0.0125
x_label: "t (s)"

# Data:
x_data:
- 5.024349958
...
```

Damit erhalten wir das folgende Ergebnis der Anpassung:

Abbildung 4.2

```
[58]: %run /opt/conda/bin/run_phyFit.py yaml/
      ↪RawData_down_sampled_500_2200_10_G_DAMPED.yaml
```

```
*** /opt/conda/bin/run_phyFit.py received valid yaml data for fit:
** Type of Fit: xy
model_label: G_DAMPED
model_function: def model(x, tau=240., x0=0.75, g=9.8, phi0=0):
                  return x0*np.exp(-x/tau)*np.cos(np.sqrt(0.789*0.473*g/0.23523)*x+phi0)

y_label: AU
```

```

y_errors: 0.02
x_errors: 0.0125
x_label: t (s)
x_data: [5.024349958, 5.124306958, 5.224263958, 5.324220958, 5.424177958,
5.524134958, 5.624092958, 5.72404975, 5.82400675, 5.92396375, 6.02392075,
6.12387775, 6.22383475, 6.32379175, 6.42374975, 6.52370675, 6.62366375,
6.723620583, 6.823577583, 6.923534583, 7.023491583, 7.123448583, 7.223405583,
7.323363583, 7.423320583, 7.523277583, 7.623234583, 7.723192, 7.823149,
7.923106, 8.023063, 8.12302, 8.222977, 8.322935, 8.422892, 8.522849, 8.622806,
8.722762958, 8.822719958, 8.922676958, 9.022633958, 9.122590958, 9.222547958,
9.322504958, 9.422461958, 9.522419958, 9.622376958, 9.722333708, 9.822290708,
9.922247708, 10.02220471, 10.12216171, 10.22211871, 10.32207571, 10.42203271,
10.52198971, 10.62194671, 10.72190371, 10.82186071, 10.92181771, 11.02177571,
11.12173271, 11.22168971, 11.32164671, 11.42160371, 11.52156071, 11.62151771,
11.72147462, 11.82143162, 11.92138862, 12.02134562, 12.12130262, 12.22125962,
12.32121662, 12.42117362, 12.52113062, 12.62108762, 12.72104462, 12.82100162,
12.92095862, 13.02091562, 13.12087262, 13.22082962, 13.32078662, 13.42074362,
13.52070062, 13.62065762, 13.72061462, 13.82057162, 13.92052862, 14.02048562,
14.12044262, 14.22039962, 14.32035662, 14.42031362, 14.52027062, 14.62022762,
14.72018462, 14.82014162, 14.92009862, 15.02005562, 15.12001262, 15.21996962,
15.31992662, 15.41988362, 15.51984062, 15.61979762, 15.71975462, 15.81971162,
15.91966862, 16.01962562, 16.11958262, 16.21953962, 16.31949662, 16.41945362,
16.51941062, 16.61936762, 16.71932462, 16.81928162, 16.91923862, 17.01919562,
17.11915262, 17.21910962, 17.31906662, 17.41902362, 17.51898062, 17.61893762,
17.71889462, 17.81885162, 17.91880862, 18.01876562, 18.11872262, 18.21867962,
18.31863662, 18.41859362, 18.51855062, 18.61850762, 18.71846462, 18.81842162,
18.91837862, 19.01833562, 19.11829262, 19.21824962, 19.31820662, 19.41816362,
19.51812062, 19.61807762, 19.71803462, 19.81799162, 19.91794862, 20.01790562,
20.11786262, 20.21781962, 20.31777662, 20.41773362, 20.51769062, 20.61764762,
20.71760462, 20.81756162, 20.91751862, 21.01747562, 21.11743262, 21.21738962,
21.31734662, 21.41730362, 21.51726062, 21.61721762, 21.71717462, 21.81713162,
21.91708862, 22.01704562]
y_data: [-0.1997231989, 0.1225266681, 0.4020419037, 0.6469383518, 0.7932787375,
0.8341633464, 0.7236726053, 0.5166886264, 0.2480305557, -0.09172321628,
-0.3712864995, -0.6222052874, -0.728521482, -0.7767518748, -0.6462890916,
-0.4653517906, -0.1693552458, 0.1100302364, 0.4309675679, 0.6369901914,
0.8109249035, 0.7987021031, 0.7397423619, 0.4993210072, 0.242197081,
-0.09183958501, -0.3778064739, -0.6088913966, -0.7521712035, -0.7551054094,
-0.6405515112, -0.4490290688, -0.1754098915, 0.1433068137, 0.4238335993,
0.6607164384, 0.7861605565, 0.8358004206, 0.6911543387, 0.5139052787,
0.2101174652, -0.07397222986, -0.4009491321, -0.6076723618, -0.7486201894,
-0.7316711578, -0.6475558908, -0.41658762, -0.1626306555, 0.158359301,
0.4260544449, 0.6725575617, 0.8006421362, 0.8104562491, 0.6894230513,
0.4883145739, 0.2144678302, -0.09728084858, -0.3876322081, -0.6239515583,
-0.7300468013, -0.7555174188, -0.6094878857, -0.4202097143, -0.1315586977,
0.145720654, 0.4559011187, 0.6575324102, 0.8117785238, 0.7923597014,
0.7034183619, 0.4620813332, 0.2018080601, -0.1154284224, -0.3831218487,
-0.6291048557, -0.7379914863, -0.7454776034, -0.6127069606, -0.4129662932,

```

```

-0.1345960338, 0.166161387, 0.4537871309, 0.6826964875, 0.7889552332,
0.791376419, 0.6656307308, 0.4643466177, 0.1837737122, -0.1054909332,
-0.4143776649, -0.609118232, -0.7404253167, -0.7217743067, -0.6179310458,
-0.3895465134, -0.1298084798, 0.1914881265, 0.4578757234, 0.6872261792,
0.7793857531, 0.7828821688, 0.6544036382, 0.4566309974, 0.1680467851,
-0.1247703727, -0.3992372136, -0.6183909293, -0.73671577, -0.7281120307,
-0.5945850141, -0.3906517103, -0.1125802436, 0.1833649187, 0.4783845399,
0.6695181059, 0.7909719379, 0.7806059939, 0.6631938305, 0.4194446792,
0.1616605658, -0.1409690604, -0.4006429939, -0.6256357026, -0.717948839,
-0.7221022625, -0.5911024124, -0.3780585985, -0.09839636028, 0.1903169292,
0.4779136198, 0.6773630228, 0.7846111905, 0.7712306415, 0.6392149839,
0.4244822017, 0.1476163213, -0.1418761097, -0.4254862417, -0.6149222879,
-0.735367329, -0.709930406, -0.5818266819, -0.3479699596, -0.09709096938,
0.2109081915, 0.4670812143, 0.6837134644, 0.7614358243, 0.7637974416,
0.6275016953, 0.4065399474, 0.1342015295, -0.1505459225, -0.4289698666,
-0.6198666935, -0.7288435174, -0.6980264251, -0.5651871872, -0.3557519093,
-0.0691921402, 0.2131352132, 0.4881150377, 0.6783859932, 0.7830247678,
0.7413724194, 0.6234380997, 0.3864223827, 0.1303110115, -0.1704625623,
-0.4323219292]

```

```

/opt/conda/lib/python3.10/site-packages/PhyPraKit/phyFit.py:1997:

```

```

RuntimeWarning: overflow encountered in multiply

```

```

    nLL2 += np.sum(_r * _r * self.data.iErr2)

```

```

/opt/conda/lib/python3.10/site-packages/PhyPraKit/phyFit.py:1997:

```

```

RuntimeWarning: overflow encountered in multiply

```

```

    nLL2 += np.sum(_r * _r * self.data.iErr2)

```

```

*==* Fit Result:

```

```

chi2: 173

```

```

parameter names:      ('tau', 'x0', 'g', 'phi0')

```

```

parameter values:     [245.379  0.822  9.771 -2.778]

```

```

neg. parameter errors: [-3.734e+01 -8.419e-03 -5.101e-03 -1.469e-02]

```

```

pos. parameter errors: [5.309e+01 8.587e-03 5.080e-03 1.477e-02]

```

```

correlation matrix :

```

```

[[ 1.    -0.939 -0.011  0.014]

```

```

[-0.939  1.     0.014 -0.017]

```

```

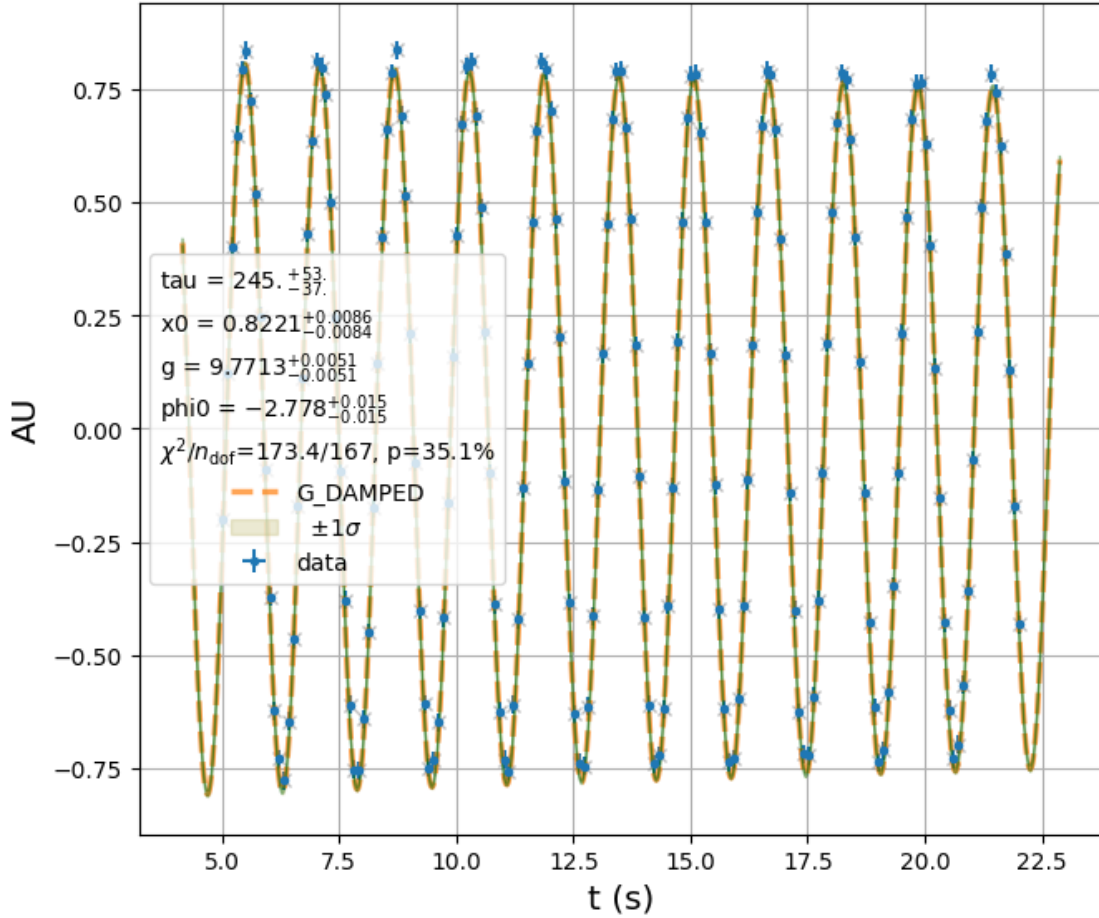
[-0.011  0.014  1.    -0.938]

```

```

[ 0.014 -0.017 -0.938  1.    ]]

```



Diskussion:

Wir bemerken einen deutlich verbesserten p -Wert der Anpassung von 35,1%, der nachträglich die Annahme eines Modells mit linearer Dämpfung legitimiert. Durch den zusätzlich in das Modell eingebrachten, zu bestimmenden Parameter τ nimmt die Anzahl der Freiheitsgrade wie zu erwarten von zuvor 168 auf 167 ab. Der ermittelte Wert für τ beträgt:

$$\tau = (245 \pm_{37}^{53}) \text{ s}$$

das entspricht einer Abklingzeit von ziemlich genau 4 min. Der Wert von g wird durch diese Änderung nicht beeinflusst. Wir vergewissern uns nochmal, ob dieser Umstand unserer Erwartung entspricht. Der Effekt der Dämpfung auf g beträgt:

$$\frac{\Delta g}{g} = \frac{T^2}{\tau^2} = \left(\frac{1.59 \text{ s}}{245 \cdot 60 \text{ s}} \right)^2$$

Es handelt sich bei dieser Größenordnung von τ also um einen Effekt jenseits der Präzision unserer Abschätzung von g .

Das Modell einer linear gedämpften Schwingung würde sich stichhaltiger testen und τ besser bestimmen lassen, wenn wir ein größeres Zeitfenster für die Anpassung wählen würden.

2.5.9 Bonusaufgabe: Vom bloßen Messen zur Kunst

Die Anmerkung zu Aufgabe 4.2 wirft eine Frage auf, die wir im Rahmen dieses Vorversuchs bisher offen gelassen haben: wie sind die Unsicherheiten auf die zusätzlichen Parameter der Messung korreliert? Wenn Sie möchten können Sie dieser Frage mit den folgenden Bonusaufgaben nachgehen. Die Bearbeitung dieser Fragen ist jedoch nicht verpflichtend.

Bonusaufgabe 1: Korrelierte Unsicherheiten: Jede Variation eines der drei Parameter m , I oder ℓ in Aufgabe 4.2 hat einen nicht-trivialen Einfluss auf die jeweils anderen Parameter. Durch quadratische Addition von ΔI , Δm , und $\Delta \ell$ unterlegen Sie die Annahme, dass alle drei Variationen unabhängig sind. Diese Annahme ist auf jeden Fall falsch. Machen Sie einen Vorschlag zur Lösung dieses Problems.

Bonusaufgabe 2: Die Kunst des Experimentierens Diskutieren Sie, wie dieser Versuch konzeptionell verbessert werden könnte, um die in Bonusaufgabe 1 diskutierten Probleme von vornherein zu vermeiden.

Die Bonusaufgaben wurden nicht weiter bearbeitet.