Datenauswertung in Grundlagenpraktika zur Physik

Günter Quast, Juli 2022 (letzte Änderung Juli 2023)

Inhaltsverzeichnis

Hinweise zur Datenauswertung in den physikalischen Praktika

In den Praktika zur Physik führen Sie selbständig Experimente zur Überprüfung einfacher physikalischer Modelle durch. Dabei untersuchen Sie physikalische Phänomene mit unterschiedlichen Messgeräten und Messmethoden, werten die resultierenden Daten aus und dokumentieren die Ergebnisse so, dass sie aus den Aufzeichnungen reproduzierbar sind.

Die Qualifikationsziele der Anfängerpraktika Physik sind im Modulhandbuch beschrieben:

- Beherrschung unterschiedlicher Messgeräte und Messmethoden,
- Erfassung und Darstellung experimenteller Daten,
- Analyse der Daten und Durchführung der Fehlerrechnung,
- Erstellung eines Messprotokolls.

Diese Anleitung gibt eine Übersicht über Grundlagen, empfohlene Software-Werkzeuge und Vorgehensweisen zur **Datenauswertung in der physikalischen Praktika**.

Allgemein ist das Ziel der Datenanalyse in der Physik zu überprüfen, ob ein angenommenes Modell durch die experimentellen Daten im Rahmen von deren Unsicherheiten adäquat beschrieben wird. Dazu müssen zunächst die Unsicherheiten der Ausgangsmessungen quantifiziert werden. Anschließend werden unter Annahme der Gültigkeit des Modells freie Modellparameter und deren aus den Unsicherheiten der Messdaten resultierende Unsicherheiten bestimmt. Früher nannte man diesen letzten, meist mit analytischen Näherungsrechnungen durchgeführten Schritt der Bestimmung der Parameterunsicherheiten "Fehlerrechnung". Mit der allgemeinen Verfügbarkeit von Computern mit ausreichend Rechenleistung werden in der modernen Datenauswertung numerische Verfahren zu Anpassung von Modellfunktionen an die Messdaten verwendet, die sowohl die Überprüfung der Gültigkeit der Modellhypothese als auch die Bestimmung der besten Parameterwerte und deren Unsicherheiten erlauben.

Inhalt:

1 Übersicht: Die einzelnen Schritte der Datenauswertung

1. Quantifizierung der Unsicherheiten aller eingehenden Messgrößen

Die Unsicherheiten der relevanten Messgrößen entnehmen Sie entweder den Datenblättern der verwendeten Messgeräte, oder Sie führen die gleiche Messung mehrmals aus und bestimmen die Unsicherheiten aus der Streuung der erhaltenen Ergebnisse als Standardabweichung σ der resultierenden Häufigkeitsverteilung. Mit einiger Erfahrung ist es auch möglich, die Unsicherheiten für typische Messvorgänge zuverlässig zu schätzen.

2. Durchführung der Messung

Als nächstes wird eine Messreihe, üblicherweise als Wertepaare (x_i, y_i), aufgenommen. Diese sollten in einer Datei im Textformat abgelegt werden, damit sie sowohl von Menschen als auf von Computerprogrammen gelesen werden können. Wichtig ist die sorgfältige Protokollierung auch der sogenannten "Metadaten", die den Ursprung und die Bedeutung der Messdaten dokumentieren. Es ist ratsam, schon in diesem ersten Schritt eine grafische Darstellung der Daten zur visuellen Kontrolle des Messprozesses zu erstellen. Einfache Fehler bei der Aufnahme der Messreihe können Sie dabei unmittelbar entdecken.

3. Anpassung eines Modells an die Messdaten

Die Daten und eine oder mehrere Modellhypothese(n) werden verglichen. In der Regel enthält die Modellhypothese noch freie Parameter, und außerdem ist das Modell in der Regel "überbestimmt", d.h. es gibt mehr Datenpunkte, als zur Bestimmung der freien Parameter notwendig sind. In solchen Fällen werden in einem Anpassungsschritt die optimalen Werte der freien Parameter bestimmt, die den kleinsten "Abstand" zwischen den Werten der Modellfunkton $f_i = f(x_i, a_1, ..., a_{n_p})$ mit n_p Parametern a_i und den N Messwerten y_i ergeben. Als Abstandsmaß wird dabei üblicherweise die von Carl Friedrich Gauß vorgeschlagene Summe S der Residuenquadrate ("Summe der kleinsten Fehlerquadrate") verwendet:

$$S = \sum_i^N \left(\frac{f_i - y_i}{\sigma_i} \right)^2 .$$

4. Überprüfung der Gültigkeit der Modellhypothese

Als Ergebnisse des Anpassungsalgorithmus erhalten Sie den Wert S_0 der Summe der Residuenquadrate am Optimum der Parameterwerte, $S_0(\hat{a}_1, ..., \hat{a}_{n_p})$. Wenn das Modell passt, sollten Abweichungen rein statistischer Natur sein. Auf Grund der Definition der Standardabweichung ist der Erwartungswert der quadrierten Abweichung zwischen einer Messung und dem wahren Wert gleich Eins; für S_0 am Optimum erwartet man also einen Wert von N, der Anzahl der Messungen. Es muss allerdings berücksichtigt werden, dass die Daten im Anpassungsprozess bereits genutzt wurden, um die n_p Parameterwerte zu optimieren. In diesem Fall ist der erwartete Wert von $< S_0 >= N - n_p = N_f$ (Zahl der Freiheitsgrade).

5. Extraktion der Modellparameter und von deren Unsicherheiten

Wenn die Modellhypothese nicht verworfen wurde, werden im letzten Schritt die Modellparameter bestimmt. Die optimalen Parameterwerte wurden bereits im vorangegangenen Schritt vorausgesetzt - bestimmt werden müssen noch die Parameterunsicherheiten. Diese ergeben sich aus der Verlauf des Wertes von S in der Nähe des Minimums S_0. Das Verfahren wird weiter unten ausführlich erläutert.

2 Grundsätzliches zum Messprozess

Im **Messprozess** werden Daten in einer sehr genau kontrollierten physikalischen Umgebung bestimmt; dies wird als Versuch oder auch Experiment bezeichnet. Der Messprozess liefert Zahlenwerte, die mit Werten aus Vorhersagen verglichen werden, um Aussagen über die Gültigkeit theoretischer Modelle machen zu können.

Messen heißt dabei grundsätzlich, eine physikalische Größe mit einer Referenzgröße, einer Maßeinheit, zu **vergleichen**. Dabei kann ein direkter Vergleich durchgeführt werden - z.B. die Länge eines Objekts mit einem Maßstab oder eine elektrische Spannung mit einer Spannungsreferenz. Häufig werden Messungen auch

indirekt durchgeführt, indem eine präzise bekannte und gut überprüfte Annahme verwendet wird, z.B. eine Längenmessung über die Laufzeit von Licht oder eine Strommessung mit Hilfe des ohmschen Gesetzes.

Beim **Messprozess** werden Eingangsgrößen in Messsignale gewandelt, aus denen die Messgrößen gewonnen wird. Diese Messgrößen werden schließlich mit einer Modellannahme verglichen, um die Ergebnisgröße(n) zu erhalten. Ein Beispiel ist die Bestimmung der Erdbeschleunigung als Ergebnisgröße aus der Ausdehnung einer Feder mit bekannter Federkonstanten als Messgröße bei Anhängen einer Referenzmasse. Heute werden Messgrößen fasst ausnahmslos als elektrische Signale erfasst, die in Analog-Digital-Wandlern in Zahlenwerte umgesetzt und dann als digitale Daten weiter verarbeitet werden.

Alle Messgrößen sind von unvermeidbaren **Unsicherheiten** betroffen, die die wahren Werte der Größen überlagern und diese damit zufällig verändern. Bei einem Messergebnis handelt es sich also um eine "Schätzung" des wahren Wertes im Sinne der Statistik, deren Unsicherheit immer zusätzlich angegeben werden muss:

$m = (\hat{m} \pm \sigma_m)[m]\,.$

Dabei ist $\hat{m}$ der Schätzwert, σ_m die Unsicherheit auf den Schätzwert, und $[m]$ ist die Einheit der Messgröße. Die Angabe eines Messergebnisses ohne Unsicherheit und Einheit ist, wie unten genauer erläutert wird, praktisch wertlos!

Häufig werden Unsicherheiten auch als dimensionslose **relative** Unsicherheiten angegeben,

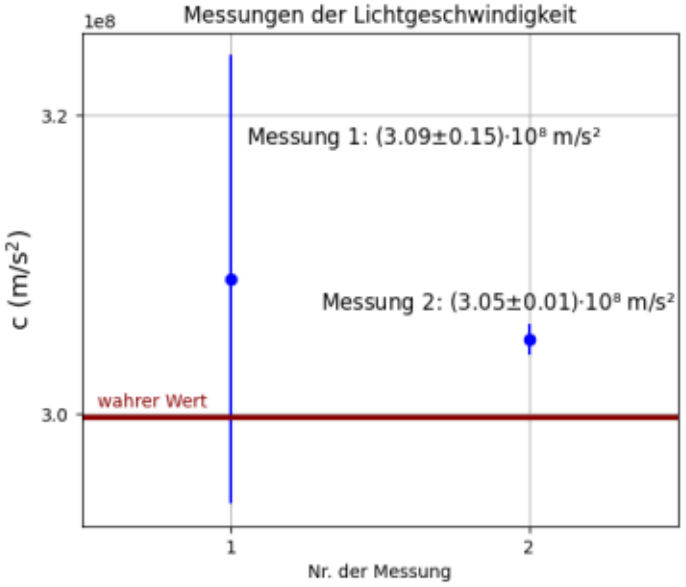
$\sigma_m^{(R)} = \frac{\sigma_m}{m}\,.$

Werden mehrere Messgrößen aufgenommen, so sind viele der dabei auftretenden Unsicherheiten (statistisch) unabhängig voneinander, d.h. die Änderung einer der Messgröße im Rahmen ihrer Unsicherheiten ändert die Werte anderer Messgrößen nicht. Solche Unsicherheiten werden als "statistische Unsicherheiten" bezeichnet. Sie können durch **mehrfache Messung** bestimmt und reduziert werden.

Es gibt bei Messungen aber fast immer auch Effekte, die alle Messwerte gleichermaßen betreffen, also nicht unabhängig sondern **korreliert** sind. Beispiele sind die Kalibrationsunsicherheiten der verwendeten Messgeräte, z.B. ein auf Grund von Temperatureffekten zu langer oder zu kurzer Maßstab oder eine nicht ganz korrekte Spannungsreferenz. Andere Quellen solcher Unsicherheiten sind als klein angenommene, vernachlässigte Umgebungseffekte wie Reibung oder Luftwiderstand bei mechanischen Versuchen, oder oft nicht vermeidbare Näherungen in den verwendeten theoretischen Modellen. Solche korrelierten Unsicherheiten werden als "systematische Unsicherheiten" bezeichnet. Systematische Unsicherheiten können durch Mehrfachmessung nicht reduziert werden. Häufig gibt es zur Bestimmung systematischer Unsicherheiten lediglich Konventionen, aber keine allgemein gültigen oder anerkannten Verfahren.

Die Angabe eines Messergebnisses ohne Unsicherheit ist praktisch wertlos, denn die Überprüfung der Übereinstimmung einer Messung mit einer theoretischen Erwartung ist nur im Rahmen der Unsicherheiten möglich. Bei dieser Bewertung handelt es sich aus Sicht der Statistik um einen **Hypothesentest**. Ein Messergebnis wird grafisch dargestellt als Messpunkt mit "Fehlerbalken", der die Größe der Unsicherheit symbolisiert. Wir nehmen dazu an, dass die Messwerte aus einer Messreihe mit unabhängigen Messungen unter identischen Bedingungen gaußförmig um ihren Mittelwert verteilt sind ("gaußförmige Unsicherheit"), eine Begründung dazu wird in Kapitel 3 gegeben. In der Physik verwenden wir die Konvention, dass die Länge des Fehlerbalkens zwei Standardabweichungen σ dieser Gaußverteilung entspricht. Innerhalb des Intervalls [m-σ, m+σ] liegen damit 68,3% der Messwerte.

> Ein **Beispiel** zeigt die Grafik unten. Bei der ersten Messung überlappt der Fehlerbalken mit dem wahren Wert, im "Slang" der Physik würden wir sagen: "Die Messung stimmt innerhalb von einem σ (d.h. einer Standardabweichung) mit dem wahren Wert überein". Die zweite Messung ist nicht mit dem wahren Wert verträglich: sie liegt zwar näher am wahren Wert, besitzt aber eine viel kleinere Unsicherheit als die erste Messung, wir sagen: "sie weicht um 5 σ ab".



3 Wichtige Grundlagen der Statistik

Auf Grund des **zentralen Grenzwertsatzes der Wahrscheinlichkeitstheorie** folgen die aus vielen Einzelbeiträgen bestehenden zufälligen Abweichungen u der Messwerte vom wahren Wert einer Gaußverteilung mit Mittelwert Null und Standardabweichung σ,

$$f_u(u) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2}\right).$$

Ein Messwert m ergibt sich aus einem angenommenen wahren Wert m^w, zu dem die zufällige Abweichung u addiert wird:

$$m = m^w + u\,.$$

Damit ergibt sich die Verteilungsdichte der Messwerte zu

$$f_m(m) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(m-m^w)^2}{2\sigma^2}\right).$$

3.1 Schätzung der Unsicherheit durch wiederholte Messung

Durch N-mal wiederholte Messung kann die Unsicherheit, definiert als die auf Verzerrung korrigierte **empirische Standardabweichung** (auch: Stichprobenstandardabweichung) der Messwerte, bestimmt werden:

$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(m_i - \bar{m})^2$, wobei

$\bar{m} = \frac{1}{N}\sum_{i=1}^{N} m_i$ der Mittelwert der Messungen ist.

Falls Sie diese Gleichung selbst in Programmcode implementieren möchten: durch folgende Umformung erhalten wir die numerisch effizienter zu berechnende Darstellung

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N} {m_i}^2 - \frac{N}{N-1}\bar{x}^2 \underset{N\to\infty}{\simeq} \frac{1}{N}\sum_{i=1}^{N} {m_i}^2 - \bar{x}^2\,.$$

Wenn mehrere Größen gleichzeitig gemessen werden, stellt sich die Frage, ob sie unabhängig voneinander sind, oder ob es einen Zusammenhang zwischen den statistischen Effekten gibt, beide Größen also z.B. bevorzugt in die gleiche Richtung abweichen. Als Maß dafür eignet sich die **Kovarianz** zweier Größen x und y,

$$\mathrm{cov}(m_x, m_y) = \tfrac{1}{N} \sum_{i=1}^{N} (m_{xi} - \bar{m}_x)\,(m_{y_i} - \bar{m}_y)\,.$$

Die Kovarianz der Größen m_x und m_y ist Null, wenn sie unabhängig sind. Wenn sie vollständig korreliert sind, nimmt sie den Wert $\sigma_{m_x}\sigma_{m_y}$ an. Deshalb wird der **Korrelationskoeffizient** wie folgt definiert:

$$\rho = \frac{\mathrm{cov}(m_x, m_y)}{\sigma_{m_x}\sigma_{m_y}} \text{ mit } -1 \le \rho \le 1\,.$$

Zwei Größen mit $\rho > 0$ werden als **korreliert**, Größen mir $\rho < 0$ als **antikorreliert** bezeichnet.

Wenn es mehr als zwei Messgrößen m_i gibt, fassen wir die Kovarianzen von Paaren von Messgrößen zur Kovarianzmatrix $\mathbf{V}$ zusammen, deren Elemente gegeben sind durch:

$$(\mathbf{V})_{ij} = V_{i,j} = \mathrm{cov}(m_i, m_j).$$

Die Diagonalelemente $V_{i,i}$ entsprechen den quadrierten Standardabweichungen der Messgrößen, $V_{i,i} = \mathrm{cov}(m_i, m_i) = {\sigma_{m_i}}^2$. Für unkorrelierte Messgrößen mit verschwindenden Kovarianzen ist $\mathbf{V}$ eine Diagonalmatrix. Die Elemente der Kovarianzmatrix lassen sich durch Mehrfachmessungen von Paaren $(m_i, m_j)_k$ durch Anwenden der Definitionsgleichung für die Kovarianz auch empirisch aus den Daten bestimmen.

Das Resultat einer gleichzeitigen Messung von zwei oder mehr Messgrößen sollte also immer als **Messwerte, Einheiten und Kovarianzmatrix** angegeben werden.

3.2 Fortpflanzung der Unsicherheiten

Werden mehrere Messgrößen m_i, $i = 1, \ldots, n$ bestimmt, von denen eine gesuchte Ergebnisgröße $g(m_i)$ abhängt, so wird die Standardabweichung σ_g von $g(m_i)$ näherungsweise durch die **Fehlerfortpflanzungsformel** bestimmt:

$${\sigma_g}^2 = \sum_{1}^{n} \left(\frac{\partial g}{\partial m_i}\right)^2 {\sigma_i}^2\,.$$

Anmerkung 1: Zur Herleitung wird eine Taylorentwicklung von $g(m_i)$ um die Messwerte m_i durchgeführt; die Formel ist also nur anwendbar, wenn die Funktion im Bereich der Unsicherheiten σ_i gut durch eine Gerade beschrieben wird.

Anmerkung 2: Diese Gleichung gilt nur für unabhängige, d.h. unkorrelierte Messungen. Wenn die Messungen korreliert sind, muss deren Kovarianzmatrix berücksichtigt werden. Für den Spezialfall von zwei Messungen lautet der Ausdruck für σ_g:

$${\sigma_g}^2 = \left(\tfrac{\partial g}{\partial m_1}\right)^2 {\sigma_1}^2 + \left(\tfrac{\partial g}{\partial m_2}\right)^2 {\sigma_2}^2 + 2\tfrac{\partial g}{\partial m_1}\tfrac{\partial g}{\partial m_2}\mathrm{cov}(m_1, m_2) \text{ mit der Kovarianz } \mathrm{cov}(m_1, m_2) \text{ von } m_1 \text{ und } m_2\,.$$

3.2.1 Standardabweichung des Mittelwerts Wird die Fehlerfortpflanzungsformel auf den Mittelwert von N unabhängigen Messungen angewendet, so ergibt sich für die Standardabweichung des Mittelwerts das wichtige Ergebnis

$$\sigma_{\bar{m}} = \tfrac{\sigma}{\sqrt{N}}\,;$$

d.h. **die Unsicherheit auf die Messgröße reduziert sich durch unabhängige, wiederholte Messungen**. Die Reduktion der Unsicherheit skaliert mit der Wurzel der Zahl der Messungen, z.B. halbiert sich die Unsicherheit mit der vierfachen Zahl von Messungen.

3.3 Zusammenhang der Summe der Residuenquadrate mit der χ^2-Verteilung

Das Ziel einer Messung ist es, ein physikalisches Modell durch dem Vergleich mit der Messreihe zu überprüfen (Hypothesentest). Wenn das Modell geeignet ist, können die Modellparameter durch die Messwerte (im statistischen Sinne) "geschätzt" werden. Im physikalischen Praktikum kommt dazu häufig die **χ^2-Methode** zum Einsatz, die auf einem Zusammenhang der in Kapitel 1 eingeführten Summe der Residuenquadrate S mit der χ^2-Verteilung beruht. Werden die Messwerte in die Gleichung für S eingesetzt, kann S als Funktion der Modellparameter aufgefasst werden und deren Werte am Minimum von S, hier als S_0 bezeichnet, liefern Schätzwerte für diese Parameter. Bei gaußförmigen Unsicherheiten der Messwerte entspricht S_0 der Summe von Quadraten von n gaußverteilten Zufallszahlen z_i, $\chi^2 = \sum_{i=1}^{n} {z_i}^2$, deren Verteilungsdichte durch die χ^2-Verteilung gegeben ist. Die χ^2-Verteilung besitzt einen Parameter N_f, der als "Anzahl der Freiheitsgrade" bezeichnet wird. Werden aus N Messwerten n_p Modellparameter bestimmt, so folgt S_0 einer χ^2-Verteilung mit $N_f = N - n_p$ Freiheitsgraden, $\chi 2(S_0, N_f)$.

Die χ^2-Verteilung ist in allen gängigen Programmpaketen zur Statistik implementiert, so dass leicht Werte der Dichteverteilung oder der kumulativen Verteilung und damit Quantile der Verteilung berechnen werden können. Die χ^2-Verteilung $\chi^2(x; N_f)$ hat einen Erwartungswert von $\mathrm{E}[\chi^2(N_f)] = N_f$ und eine Varianz $\sigma^2 = \mathrm{Var}[\chi^2(N_f)] = 2N_f$.

Für den in einem Experiment beobachteten Wert S_0 kann mit Hilfe der bekannten Quantile der χ^2-Verteilung die sogenannte χ^2-Wahrscheinlichkeit p_{χ^2} angeben werden. Dies ist die Wahrscheinlichkeit dafür, bei einer Messreihe den beobachteten Wert für S_0 oder einen größeren Wert zu erhalten, unter der Annahme, dass das Modell zur Beschreibung der Messung geeignet ist. Wenn diese Wahrscheinlichkeit zu klein ist, müssen Annahmen über die Genauigkeit der Eingangsmessgrößen oder über die Angemessenheit des Messverfahrens überdacht werden, oder die Modellhypothese sollte angepasst bzw. verbessert werden, bevor ernsthaft daran gedacht werden kann, gültige Werte für die Modellparameter zu extrahieren. In der Praxis geschieht das häufig bei einem Wert von $p_{\chi^2} < 0.05$.

> **Beispiel**: Anstatt eines ohmschen Widerstands haben Sie versehentlich eine Diode vermessen. Die Überprüfung des χ^2-Wertes, also Schritt 4 aus Kap. 1, bewahrt Sie davor, einer Halbleiterdiode einen ohmschen Widerstand zuzuweisen!

3.4 Parameteranpassung und Bestimmung der Parameterunsicherheiten

Zur Schätzung der Parameter a_k eines an den Stützstellen (= Messpunkten) x_i ausgewerteten Modells $f(x_i, a_k)$ aus den Messwerten y_i wird meist die bereits oben eingeführte Summe der Residuenquadrate S verwendet. In vielen Fällen stellt dies die optimale Methode dar, die die kleinste Varianz der Parameterunsicherheiten liefert. S entspricht für gaußförmige, von den Modellparametern nicht abhängige Unsicherheiten bis auf einen Faktor Zwei dem negativen Logarithmus der Likelihood-Funktion, die in der mathematische Schätztheorie verwendet wird. Allgemein schreibt man S mit Hilfe der Inversen der oben eingeführten Kovarianz-Matrix der Unsicherheiten der Datenpunkte,

$$S\left(y_1, \ldots, y_N; f_1, \ldots, f_N\right) = \sum_{i,j=1}^{N} \left(f_i - y_i\right)^T \left(\mathbf{V}^{-1}\right)_{ij} \left(f_j - y_j\right)$$

mit der Notation $f_i = f(x_i; a_1, \ldots, a_{n_p})$.

In dieser Form wird S auch verwendet, wenn die Messdaten korrelierte Unsicherheiten aufweisen, wenn $\mathbf{V}$ also keine Diagonalmatrix ist.

Informationen über die Unsicherheiten der Parameter werden aus dem Verlauf der Verteilung $S(y_i, x_i; a_j)$ in der Nähe des Minimums gewonnen. Das ist die Stelle im Parameterraum, an der die Parameter ihre optimalen Werte $\hat{a}_j$ annehmen, wie sie sich aus der Minimierung von S bzgl. der Parameterwerte ergeben. Dabei wird S als Funktion der Parameter a_j aufgefasst, $S = S(a_j)$. Ein scharfes Minimum, also große Krümmung von $S(\hat{a}_j)$ am Minimum oder ein schneller Anstieg in der Nähe des Minimums, entsprechen sehr genau bestimmten Parameterwerten; ist das Minimum flacher, die Krümmung am Minimum oder der Anstieg kleiner, ist die Unsicherheit größer. Der typische Verlauf von S am Minimum ist in der Grafik unten gezeigt.

Aus diesen Betrachtungen wird ersichtlich, dass die Unsicherheiten mit den Krümmungen, also den zweiten Ableitungen von S_0 nach den Parametern, zusammenhängen. Für S ergibt sich:

$$\mathrm{Var}[\hat{a}_j] = \sigma^2_{\hat{a}_j} = 2\left(\left.\frac{\partial^2 S}{\partial {a_j}^2}\right|_{a_j=\hat{a}_j}\right)^{-1}.$$

Diese Methode wird von fast allen Anpassungsprogrammen zur Bestimmung der Parameterunsicherheiten verwendet. Die Kehrwerte der gemischten zweiten Ableitungen bestimmen im Falle mehrerer anzupassender Parameter die Elemente der Kovarianzmatrix. Manche Anpassungsprogramme erlauben es, Konfidenzkonturen von Paaren von Parametern anzuzeigen, die vorhandene Korrelationen zwischen den Parameter anschaulich illustrieren.

Genauer als die Bestimmung der Unsicherheiten aus den zweiten Ableitungen am Minimum S_0 ist es, einen kompletten Scan der Umgebung des Minimums durchzuführen und die Punkte im Parameterraum zu markieren, an denen S um den Wert Eins größer ist als am Minimum. Mit dieser Methode können Konfidenzintervalle für die Parameterwerte bestimmt werden, die auch Gültigkeit haben, wenn die Verteilung der Parameterunsicherheiten nicht gaußförmig ist. Diese Methode erlaubt die Bestimmung asymmetrischer Unsicherheiten, die 68,3%-Konfidenzintervallen für die Parameterwerte entsprechen. Einige der weiter unten vorgeschlagenen Programmpakete verwenden diese bessere Methode entweder als Standard oder optional.

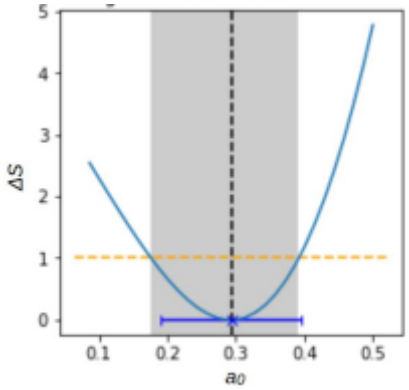


Abbildung 1: Typischer Verlauf von S am Minimum für ein physikalisches Modell mit einem Parameter. Eingezeichnet sind die Parameterunsicherheiten, die sich aus den zweiten Ableitungen (Fehlerbalken) oder aus dem Anstieg um Eins (graues Band) ergeben.

3.4.1 Besonderheiten bei der Bestimmung der Parameterunsicherheiten In vielen Fällen werden Anpassungen genutzt, um Parameter zu bestimmen, ohne einen Hypothesentest durchführen zu wollen. Die Gültigkeit des verwendeten Modells wird dann als gegeben vorausgesetzt, d.h.
$\chi^2/N_f \stackrel{!}{=} 1$ wird postuliert. Diese Bedingung lässt sich erfüllen, wenn alle nach den oben beschriebenen Verfahren bestimmten Parameterunsicherheiten mit einem Faktor $\sqrt{\chi^2/N_f}$ skaliert werden. Diese Skalierung der Parameterunsicherheiten mit der Wurzel aus dem reduzierten Chi-Quadrat χ^2/N_f ist die Standardeinstellung in den meisten Anpassungsprogrammen. In der Physik ist diese Vorgehensweise sinnvoll, wenn lediglich eine Parametrisierung benötigt wird, die gegebene Daten im Rahmen der Messunsicherheiten möglichst gut repräsentiert. Wenn Sie allerdings eine Überprüfung der Modellhypothese benötigen, müssen Sie unbedingt daran denken diese voreingestellte Option abzuschalten!

3.5 Parameter mit externen Einschränkungen

In manchen Fällen kommen in einer Modellfunktion Parameter vor, über die Kenntnisse aus externen Messungen oder aus Hilfsmessungen vorliegen, wie z.B. Naturkonstanten oder die Eigenschaften von Messgeräten (Federkonstante eines Federkraftmessers, Kalibrationsfaktoren usw.). Wir nehmen an, dass für diese Parameter p_k bereits ein Schätzwert p_k^0 und eine Unsicherheit σ_{p_k} bekannt sind. In der Parameteranpassung werden

daher Abweichungen der Parameter von p_k^0 zugelassen, indem sie mit einem zusätzlichen Summanden in der Summe der Residuenquadrate berücksichtigt werden:

$$\tilde{S} = S + \sum_{k=1}^{n_p} \left(\frac{p_k - p_k^0}{\sigma_{p_k}} \right)^2 .$$

Je weiter p_k von p_k^0 abweicht, desto größer wird der Wert von $\tilde{S}$. Der Vorteil dieser Methode besteht darin, dass die Unsicherheiten der extern festgelegten Parameter bei der Bestimmung der Unsicherheiten der interessierenden Parameter mit berücksichtigt werden. Eine händische Fortpflanzung dieser Unsicherheiten in das Endergebnis ist bei Verwendung dieser Methode also nicht notwendig.

Anhang

A1 Übersicht: Hilfsmittel für die physikalischen Praktika

Als Vorbereitung zum Einsatz rechnergestützter Methoden in den Praktika wird an der Fakultät für Physik am KIT die Vorlesung "Computergestützte Datenauswertung" mit praktischen Übungen angeboten [**Link CgDA**].

Eine Programmbibliothek mit nützlichen Funktionen und Beispielen zur Unterstützung typischer Aufgaben bei der Auslese, Visualisierung und Auswertung von Daten in den physikalischen Praktika liefert das Paket [**PhyPraKit**]. Die in diesem Paket enthaltenen Beispiele können als Vorlagen für eigene *Python*- Skripe zur Versuchsauswertung dienen, und daher empfiehlt es sich, das gesamte Paket als gepackte Datei im *.zip*-Format herunter zu laden.

Zur Einführung in die Grundlagen der statistischen Datenauswertung, die Anwendung der Skript-Sprache *Python* und fortgeschrittene Methoden stehen Tutorials zu den Themen "Grundlagen der Statistik" und "Fehlerrechnung" als Jupyter-Notebooks zur Verfügung, s. [**Link Jupyter-Tutorials**].

Das Skript [**Funktionsanpassung mit der χ^2-Methode**] erklärt die Grundlagen der Anpassung von Funktionen an Messdaten und beschreibt einige der gängigen Werkzeuge.

Für die Arbeit von PhysikerInnen ist die Entwicklung von Methoden zur Lösung neuer Problemstellungen wichtig, die in der Regel die Erstellung von eigenem Programmcode notwendig machen, weil Standard-Lösungen noch nicht existieren. Dennoch gibt es viele immer wiederkehrende Routineaufgaben, die durchaus mit Standard-Programmen gelöst werden sollten.

a) Einige der üblichen Programmpakete setzen einfache Programmierkenntnisse in der Sprache *Python* voraus. Damit wird direkt und sehr flexibel das Programmierinterface (API = "Application Programming Interface") angesprochen. Dieser Zugang empfiehlt sich für den besonders programmier-affinen Teilnehmerkreis.

b) Für die Durchführung von Datenauswertungen in den physikalischen Praktika am KIT empfiehlt sich allerdings das Paket *PhyPraKit* (siehe [`PhyPraKit Download Link`], das vereinfachte Schnittstellen, sogenannte ("Wrapper-Funtionen") bereit stellt, um den notwendigen eigenen Programmcode zu minimieren. Die Anwendung dieser Funktionen ist in zahlreichen Beispielen im Verzeichnis `PhyPraKit/examples` illustriert.

c) Weiter enthält *PhyPraKit* auch einige eigenständige Programme, die die Visualisierung von Daten und Modellen und die Anpassung von Modellen an Messdaten ermöglichen, ohne eigenen *Python*-Code zu erstellen:

- Mit dem Script *plotData.py* lassen sich eine oder mehrere Messreihen von Wertepaaren grafisch darstellen. Möglich ist auch die Darstellung von Messreihen als Häufigkeitsverteilungen ("Histogramme").
- Mit dem Script *plotCSV.py* können Daten in Dateien im *csv*-Format direkt angezeigt werden. Das zur Trennung der Felder verwendet Separator-Zeichen (voreingestellt ist ',') oder die Zahl der Kopfzeilen (Vorgabe ist 1) können als Option angegeben werden.

- Das Script *run_phyFit.py* dient dazu, Anpassungen von Modellen an Messreihen von Wertepaaren durchzuführen; die Daten und das Modell werden dabei aus einer in der einfachen Datenbeschreibungssprache *yaml* erstellen Text-Datei gelesen.
- Das Scritpt *csv2yaml.py* erlaubt die Konversion von Eingabedaten im häufig verwendeten *csv*-Format in einen gültigen Ausdruck in der Datenbeschreibungssprache *yaml*, die von den Programmen oben zur Dateneingabe verwendet wird. Auch das Windows-Programm Excel oder machen in den Praktika verwendeten Programme zur Auslese von Geräten von Lehrmittelfirmen unterstützen das *csv*-Format. Bei Verwendung einer deutschen Betriebssystemversion werden Daten allerdings als deutsche Dezimalzahl. also mit ‘,’ anstelle von ‘.’, abgespeichert, und deshalb erledigt diese Script auch die ggf. notwendige Umwandlung, um mit wissenschaftlichen Pakten zur Datenauswertung kompatibel zu sein.

- Das Script *kafe2go* aus dem Paket *kafe2* dient (wie auch *run_phyFit*) zur Anpassung von Modellen an Daten, bei denen die gesamte Anpassung basierend auf einer *yaml*Datei durchgeführt wird. *kafe2go* ist sehr viel funktionsreicher als *run_phyFit.py*, das Datenformat ist aber kompatibel.

Es gibt viele weitere Werkzeuge, mit denen sich typische Aufgaben der Darstellung und Auswertung von Daten bewerkstelligen lassen. Nicht alle davon können flexibel genug mit den typischen Unsicherheiten bei physikalischen Messungen umgehen, oder die Standardeinstellungen erlauben es nicht, die Gültigkeit der angenommenen Modellhypothese zu überprüfen. Auch die strikte Interpretation der ausgegebenen Parameterunsicherheiten als Konfidenzbereich ist nicht in jedem Fall gegeben. Es ist also erforderlich, die jeweiligen Optionen anzupassen, um nachvollziehbare und zu den üblichen Anforderungen in der Physik konforme Ergebnisse zu erhalten.

A2 Empfehlungen zur praktischen Vorgehensweise

Zur Durchführung der mit Computerunterstütung vergesehenen Arbeiten in den Praktika wird Ihnen im Normalfall eine über einen beliebigen Web-Browser zugängliche Arbeitsumgebung bereit gestellt. Dazu werden üblicherweise sogenannte Jupyter-Notebooks (s. weiter unten) genutzt, die die Programmiersprache *Python* voraussetzen.

Es ist allerdings auch kein Problem, eine entsprechende Umgebung auf Ihren eigenen Computer aufzusetzen. Ob Sie einen Jupyter-Server auf Ihrem eigenen System oder auf einem Server des entsprechenden Kurses nutzen, macht zwar kaum einen Unterschied; auf Ihrem eigenen System sind Sie aber flexibler, arbeiten stets in der gleichen gewohnten Umgebung und können von Ihnen erstellten Programmcode leichter auf andere Projekte übertragen.

A2.1 Arbeitsumgebung auf dem Computer

Es gibt für die verschiedenen Computer-Plattformen (Microsoft Windows, Apple OS, Linux oder Chromebook) eine schier unüberschaubare Anzahl von verschiedenen Applikationen zur Datenvisualisierung und -analyse, die mit einer ebenso kaum überschaubaren Anzahl an Optionen und mit bisweilen sehr komplexen Nutzeroberflächen eine Vielzahl von Anwendungsfällen abzudecken versuchen.

In der wissenschaftlichen Datenanalyse haben sich frei zugängliche und kostenlos verfügbare Lösungen etabliert. Insbesondere das neue Gebiet “Data Science” setzt dabei auf die Sprache *Python*, die als interpretierte Sprache einerseits einen schnellen Entwicklungszyklus von Code zur spezialisierten Datenanalyse bietet, andererseits aber auch mächtige, hochperformante Bibliotheken zur numerischen Datenverarbeitung mitbringt.

Da es die Programmiersprache *Python* auf allen verbreiteten Plattformen und sogar auf Einplatinencomputern gibt, ist es sinnvoll, auch für die Datenauswertung in den Praktika zur Physik auf Lösungen in *Python* zu setzen. Auf Windows- und neueren Apple-Systemen ist *Python* leider nicht standardmäßig vorhanden und muss erst noch installiert werden. Dazu werden in den AppStores entsprechende Pakete angeboten. Wenn *Python* installiert und die Hilfsprogramme im Pfad der ausführbaren Programme aufgeführt sind, ist die Vorgehensweise aber immer gleich.

In *Python* geschriebene Programme werden üblicherweise über die Kommandozeile gestartet; auch die Installation und Konfiguration von Programmpaketen erfolgt über die Kommandozeile. Nach der Installation

von *Python* sollten Sie zunächst einige der wichtigsten Programme und Pakete installieren. Dies erledigen Sie über den Package Installer `pip` (bzw. `pip3` bei Systemen, auf denen auch noch die ältere Version 2.7 von Python installiert ist).

Geben Sie zur Installation der zusätzlich benötigten Pakete nacheinander folgende Befehle ein:

```
> pip3 install kafe2
> pip3 install PhyPraKit
```

- `kafe2` ist ein flexibles Paket zur Anpassung von Modellen an Messdaten.
- `PhyPraKit` ist das oben schon erwähnte Paket mit Beispielen für konkrete Anwendungen in den Praktika.

Bei Installation von *kafe2* werden automatisch wichtige Pakete mitinstalliert.

- `numpy` ist eine mächtige Bibliothek zur Durchführung effizienter Berechnungen von mathematischen Funktionen auf großen Datenmengen.
- `matplotlib` ist eine Bibliothek zur Erstellung ansprechender Grafiken zur Visualisierung von Daten.
- `iminuit` ist ein Paket zur numerischen Minimierung einer Kostenfunktion und zur Analyse der Unsicherheiten, das von den Anpassungen in `kafe2` und `PhyPraKit` verwendet wird.

Jupyter Notebooks

Wenn Sie als Umgebung zur Ausführung von *Python*-Code Jupyter Notebooks ausprobieren möchten, installieren sie noch:

```
> pip3 install jupyterlab
```

Die Jupyter-Umgebung wird auf allen Systemen gestartet durch Eingabe von

```
> jupyerlab
```

beziehungsweise über *Python* mit dem Befehl

```
> python -m jupyerlab .
```

Dies startet einen lokalen Web-Server, mit dem sich normalerweise ein automatisch startender Web-Browser verbindet. Sie können nun Beispiele mit der Dateiendung *ipynb* ausführen und modifizieren oder auch eigenen Python-Code eingeben und ausführen. Die Browser-Oberfläche fungiert dabei als komplette Entwicklungsumgebung, in der Code, Dokumentation und Programmausgaben in Textform oder auch grafisch ausgegeben werden.

Weitere nützliche Programme sind das Textsatz-Programm *LaTeX* und das Paket *pandoc*, mit denen sich Dokumentation und auch Praktikumsprotokolle erstellen lassen. Auf vielen Systemen ist die *LaTeX*-Version *TexLife* installiert; für MS Windows empfiehlt sich wegen der einfacheren Installation die Variante *MiKTeX*, die neben einer ganzen Anzahl an nützlichen Hilfsprogrammen und einem Kommandozeilen-Interface auch eine grafische Oberfläche *texworks* zur Erzeugung und Darstellung von Dokumenten mitbringt.

Das Paket *pandoc* ermöglicht es, Dokumente in der leichtgewichtigen und von Menschen gut lesbaren "Markdown"-Sprache zur verfassen; Ausdrücke in *LaTeX* werden dann nur für Formeln benötigt.
Anmerkung: Dieses Dokument wurde in *Markdown* erstellt und mit Hilfe von *pandoc* ins *pdf*-Format umgewandelt.

Beispiele und Programme aus PhyPraKit

Sie sollten zusätzlich noch die Beispiele und Programme aus dem Paket *PhyPraKit* herunter laden. Diese finden Sie unter dem Link: [`PhyPraKit`]. Verwenden Sie das Download-Symbol und kopieren Sie alle Dateien im *gitlab*-Repository als *zip*-Datei herunter. Entpacken sie alles in ein Verzeichnis mit einem aussagekräftigen Namen, z.B. `Praktikum/PhyPraKit`. Im Verzeichnis `PhyPraKit/docs` finden Sie die Datei `PhyPraKit.pdf` mit einer ausführlichen Dokumentation aller enthaltenen Module und der Beispiele im Verzeichnis `PhyPraKit/examples` bzw. der Jupyter-Notebooks in `PhyPraKit/ipynb_examples/`.

`PhyPraKit` enthält die oben bereits erwähnten Programme, die Standardaufgaben auch ohne die Erstellung von eigenem Programmcode lösen. Für die meisten Aufgaben sind diese Standard-Programme ausreichend; für den einzelnen Anwendungsfall angepasst werden müssen nur die Daten und Metadaten in der Eingabedatei im *yaml*-Format.

Für komplexere Aufgaben gibt es Beispiele in *Python*, die als Vorlage zur Anpassung für ähnliche Aufgabenstellungen gedacht sind. Zur direkten Anwendung der API der Programmpakete *kafe2* oder *PhyPraKit.phyFit* studieren Sie am besten den Code der in den Beispielen aufgerufenen Wrapper-Funktionen, oder Sie nutzen die entsprechenden API-Dokumentationen.

A2.1 Digitale Datenformate

Bei Verwendung von vorgegebenen Programmen müssen Daten in den von diesen unterstützten Formaten vorliegen. Ein sehr einfaches, weit verbreitetes Format ist dabei das `csv`-Format (**c**omma bzw. **c**haracter **s**eparated **v**alues), bei dem die Daten in Zeilen vorliegen und die einzelnen Datenwerte in einer Zeile durch Kommata oder häufig auch andere Zeichen (Tabulator, Leerzeichen, ;) getrennt werden. Damit lässt sich z.B. eine Tabelle gut abbilden:

a	b	c
1	2	3
4	5	6

die als `csv`-Datei dargestellt so aussieht:

```
a,b,c
1,2,3
4,5,6
```

Es ist möglich, Daten in diesem Format in eine Excel-Tabelle einzulesen oder auch Excel-Daten in diesem Format zu exportieren.

Leider bietet dieses einfache Format keine Möglichkeit, komplexere Datenstrukturen oder die ebenfalls benötigten Metadaten darzustellen. Für die Programmpakete in *PhyPraKit* und *kafe2* wurde deshalb als Datenformat die einfache Datenbeschreibungssprache `yaml` gewählt. Unsere Tabelle sieht in diesem Format so aus:

```
a: [1,4]
b: [2,5]
c: [3,6]
```

Der Vorteil ist, dass weitere Schlüssel hinzugefügt werden können, um zusätzliche Information anzugeben. Auch vom einlesenden Programm nicht ausgewertete, durch das Zeichen `#` eingeleitete Kommentarzeilen sind möglich:

```
# Beispiel einer Datei im yaml-Format
Datentyp: "Einfache Tabelle"
Datum:    11.11.2011
a: [1,4]
b: [2,5]
c: [3,6]
```

Eingelesen in ein Programm entspricht die Datenstruktur der eines sogenannten “Dictionaries”, d.h. eines “assoziativen Feldes”, bei dem Datenstrukturen mit einem Text oder einer Zahl als Schlüssel indiziert werden. Die vom *yaml*-Format unterstützten Kommentarzeilen werden zwar beim Einlesen ignoriert, sind aber zur Strukturierung der Daten und zur Dokumentation von Optionen für den Ersteller oder Leser der Datei extrem hilfreich.

In unserem Beispiel sind die Datenstrukturen Listen aus Zahlen, Text oder ein Datum. Es wäre auch möglich, eine ganze Tabelle mit ihren Metadaten als eigene Datenstruktur einzubinden, oder auch viele Tabellen mit ganz unterschiedlicher Bedeutung in einem Dokument zu vereinen.

Ein weiteres Beispiel für Messdaten mit Unsicherheiten und einer als Python-Code dargestellten Funktion veranschaulicht die Relevanz für typische Problemstellungen im Praktikum:

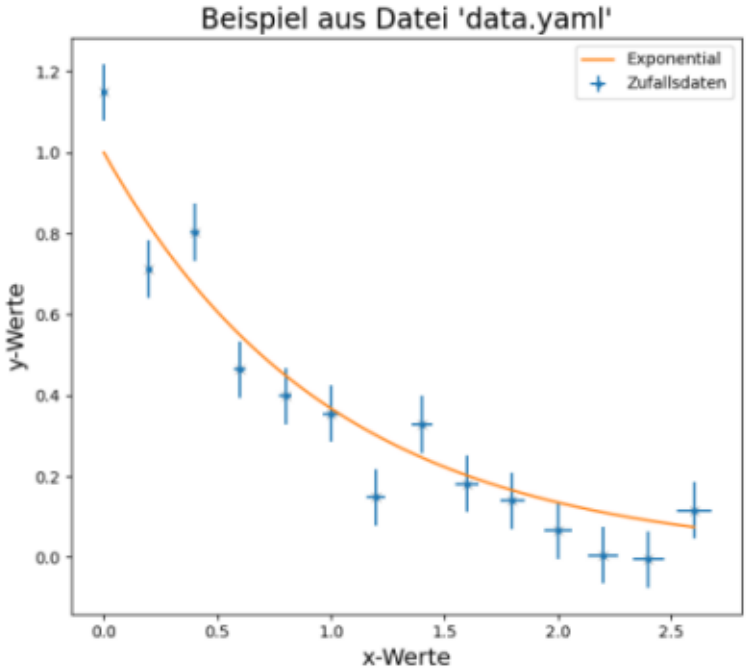
```
# data.ydat: Beispiel-Datei im yaml-Fromat
# ----------------------------------------------------------------
title: "Beispiel aus Datei 'data.yaml'"
x_label: 'x-Werte'
y_label: 'y-Werte'

label: Zufallsdaten
x_data: [0., 0.2, 0.4, 0.6, 0.8, 1., 1.2, 1.4, 1.6, 1.8, 2., 2.2, 2.4, 2.6]
x_errors: [0.01,0.015,0.02,0.025,0.03,0.035,0.04,0.045,0.05,0.055,0.06,
           0.065,0.07,0.075]
y_data: [ 1.149, 0.712, 0.803, 0.464, 0.398, 0.355, 0.148,
          0.328, 0.181, 0.140, 0.0651, 0.005, -0.005, 0.116 ]
y_errors: 0.07

# optional model specification
model_label: 'Exponential'
model_function: |
    def exp_model(x, A=1., x0=1.):
      return A*np.exp(-x/x0)
```

Mit Hilfe des Programms `plotData.py` und dem Aufruf
`> python3 plotData.py data.ydat`
wird dann daraus die unten gezeigte grafische Ausgabe erzeugt.

Das Programm `plotData.py` ist auch bei der händischen Datenerfassung sehr nützlich, weil damit ohne viel Aufwand eine grafische Überprüfung der Daten schon während der Datenaufnahme möglich ist.



Die Programme `PhyPraKit/examples/run_phyfit.py` und `kafe2go` aus dem Paket `kafe2` nutzen das gleiche Eingabeformat zur Durchführung von Anpassungen der Modellparameter an die Messdaten.

Hinweis: Daten aus üblichen Programmen wie *LibreOffice* oder *MS Office* oder auch von manchen Geräten wie Datenloggern oder Oszilloskopen werden im *csv*-Format ausgegeben. Zur Konversion in einen *yaml*-Block gibt es in PhyPraKit das Script `csv2yml.py`.

A2.2 Bestimmung der Unsicherheiten von Messdaten

Wie bereits eingangs erläutert, ist es für die Interpretation von Messergebnissen in der Physik essentiell, die Messunsicherheiten zu kennen.

Durch wiederholte Messung lassen sich zufällige (also statistische) Einflüsse quantifizieren.

Abweichungen oder Unzulänglichkeiten des Messgeräts oder der Messmethode, sogenannte systematische Unsicherheiten, lassen sich auf diese Weise aber nicht bestimmen, sondern dazu muss auf Angaben des Herstellers oder des Urhebers des Messverfahrens zurück gegriffen werden.

A2.2.1 Wiederholte Messung zur Bestimmung statistischer Unsicherheiten

Um die Unsicherheiten von Messungen abzuschätzen, kann man die gleiche Größe mehrmals unabhängig messen, z.B. von mehreren Personen, oder durch möglichst unabhängige Wiederholung der gesamten Messprozedur inklusive der Neuausrichtung bzw. Neuverbindung des Messgeräts. Aus der Standardabweichung der Messwerte ergibt sich dann der Wert für die Unsicherheit einer Einzelmessung. Für diese Vorgehensweise gibt es ein vom Programm `plotData.py` unterstütztes Datenformat zur Erstellung von Häufigkeitsverteilungen und zur Berechnung der statistischen Daten, wie im folgenden Beispiel gezeigt:

```
# hData.ydat: Beispiel einer Histogramm-Darstellung
# -------------------------------------------------
type: histogram
title: "Wiederholte Messungen von Tischhöhen"

label: Beispieldaten
```

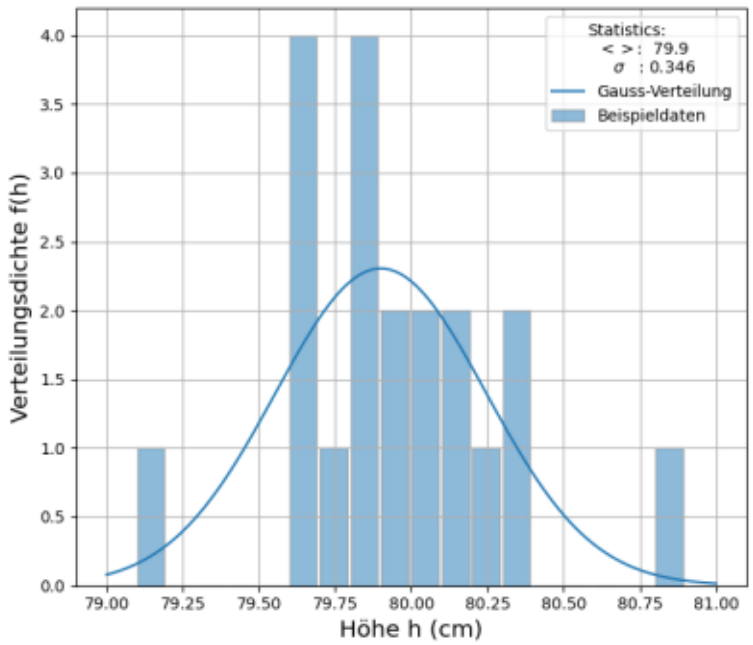
```
  x_label: 'Höhe h (cm)'
  y_label: 'Verteilungsdichte f(h)'

  # Daten:
  raw_data: [
     79.83,79.63,79.68,79.82,80.81,79.97,79.68,80.32,79.69,79.18,
     80.04,79.80,79.98,80.15,79.77,80.30,80.18,80.25,79.88,80.02 ]

  n_bins: 20
bin_range: [79., 81.]
   # alternatively an array for the bin edges can be specified
  #bin_edges: [79., 79.5, 80, 80.5, 81.]

  model_label: Gauss-Verteilung
  model_density_function: |
    def normal_distribution(x, mu=79.9, sigma=0.346):
      return np.exp(-0.5 *((x-mu)/sigma)**2)/np.sqrt(2.*np.pi*sigma**2)
```

Die Ausgabe enthält auch die Angabe von Mittelwert und Standardabweichung der eingegebenen Daten. Die eingezeichnete Funktion repräsentiert die Verteilungsdichte, die zur Berücksichtigung der Bin-Breite im Histogramm und der Anzahl der Einträge entsprechend skaliert wurde. Die in der Eingabedatei anzugebende Verteilungsdichte ist dagegen auf die Fläche von Eins normiert.



A2.2.2 Apparative Unsicherheiten

Alle Messgeräte haben eine endliche Genauigkeit, die typischerweise im jeweiligen Datenblatt angegeben ist. Solche auf das Messgerät oder die Messmethode zurück zu führende "systematische Unsicherheiten" betreffen alle aufgenommenen Messwerte in gleicher Weise und lassen sich durch Mehrfachmessungen auch nicht verringern.

Wenn mehrere Messgeräte des gleichen Typs vorhanden sind, kann man allerdings die Methode aus 5.2.1 anwenden und die Unsicherheiten aus der Streuung der mit verschiedenen Geräten der gleichen Serie erhaltenen

Messergebnisse bestimmen. Die systematische Unsicherheit ist damit auf eine statistische Unsicherheit zurückgeführt.

> **Beispiel:** Die Genauigkeit von digitalen Messgeräten wird typischerweise in der Form "± x% + y digits" angegeben. Die erste Unsicherheit ist dabei die Kalibrationsunsicherheit des Geräts; dies ist eine relative, auf den wahren Wert bezogene Unsicherheit, die alle Messungen in gleicher Weise betrifft, etwa so wie ein zu langer oder zu kurzer Maßstab bei Längenmessungen. Die zweite Größe ist die statistische Unsicherheit jeder Einzelmessung, angegeben in Abweichungen der letzten Ziffer der Anzeige. In der Praxis kommen dazu noch statistische Unsicherheiten durch Rauschen, das über die Messleitungen und den Aufbau eingestreut wird, oder Übergangswiderstände an der Kontaktstelle.

Da sich Unsicherheiten quadratisch addieren, d.h.
$\sigma_{\text{tot}}{}^2 = \sqrt{\sum_i \sigma_i^2}$ gilt, können häufig die kleinsten Unsicherheiten vernachlässigt werden. In der Praxis dominiert oft das als statistisch anzusehende Rauschen. Wenn aber die relative, auf den wahren Wert bezogene Unsicherheit dominiert, müssen Verfahren angewandt werden, die korrelierte, relative Unsicherheiten korrekt berücksichtigen können.

Die Berücksichtigung von allen oder mehreren Messwerten gemeinsamen Unsicherheiten benötigt entweder die Einführung der Kovarianzmatrix aller Messunsicherheiten, wie es in den Paketen *kafe2* und *PhyPraKit/phyFit* vorgesehen ist. Man kann das Problem in einfache Fällen aber auch durch Iteration lösen, indem man mehrere Datenauswertungen mit jeweils um die gemeinsamen Unsicherheiten verschobenen Messwerten durchführt und die sich ergebenden Abweichungen der erhaltenen Ergebnisse als jeweilige Unsicherheit verwendet. Am Ende ergibt sich die Gesamtunsicherheit als die Wurzel der Summe aller so ermittelten, quadrierten Unsicherheiten.

A2.3 Einfache Durchführung von Modellanpassungen

Die Grundlagen der Anpassung von Modellen an Messdaten sind in der Literatur und in den oben bereits erwähnten Skripten und Jupyter-Tutorials ausführlich beschrieben.

Vor allem die vielfältigen Methoden zur analytischen Behandlungen von linearen, quadratischen oder nichtlinearen Regression lassen das Thema unnötig komplex erscheinen. Im wesentlichen geht es lediglich darum, ein geeignetes Abstandsmaß zwischen einer Modellfunktion $f()$ und einer Anzahl von N mit Unsicherheiten behafteten Messdaten y_i bezüglich der Parameter a_k zu minimieren:

$\min_{a_j} D(f(x_i;\, a_1, .., a_{n_p}),\, y_i)$.

Eine ganz einfache Lösungsmethode besteht schon darin, den Abstand D als Funktion der Parameter grafisch darzustellen und dann das Minimum zu suchen. Dazu ist kein ausgefeilter Algorithmus notwendig, sondern es reichen einige Zeilen *Python* Code.

Als Beispiel zeigt die rechte Seite der Grafik unten 10 Messwerte $m_1, ..., m_{10}$ und einige konstante Funktionen. Welche davon passt am besten zu den Daten?

Um eine quantitative Antwort zu finden, wählen wir die Summer der Residuenquadrate als Abstandsmaß $D(C) = S(C)$; aufgefasst als Funktion des Parameters C ergibt sich die Gleichung einer Parabel:

$$D(C) = \sum_{i=1}^{10} \frac{(y_i - C)^2}{\sigma^2}\,.$$

Das Abstandsmaß als Funktion von C ist im linken Teil der Grafik dargestellt. Die Parabelform und auch die Position des Minimums $\hat{C}$ sind klar sichtbar. Der Wert von $\chi^2 = D(\hat{C})$ und die sich daraus ergebende χ^2-Wahrscheinlichkeit sind ebenfalls angegeben.

Übrigens entspricht der so bestimmte Wert von $\hat{C}$ dem Mittelwert der Messwerte. Dies kann man leicht durch analytische Bestimmung des Minimums von $D(C)$ verifizieren.

Auch bei komplizierterem Verlauf der Funktion $f(x_i; a_k)$ bleibt die numerische Herangehensweise die gleiche - im Ausdruck für D wird lediglich C durch $f(x_i; a_k)$ ersetzt. Wir können das Anpassungsproblem also durch Verwendung eines einfachen Funktionen-Plotters lösen, indem man $D(a_k)$ grafisch darstellt.

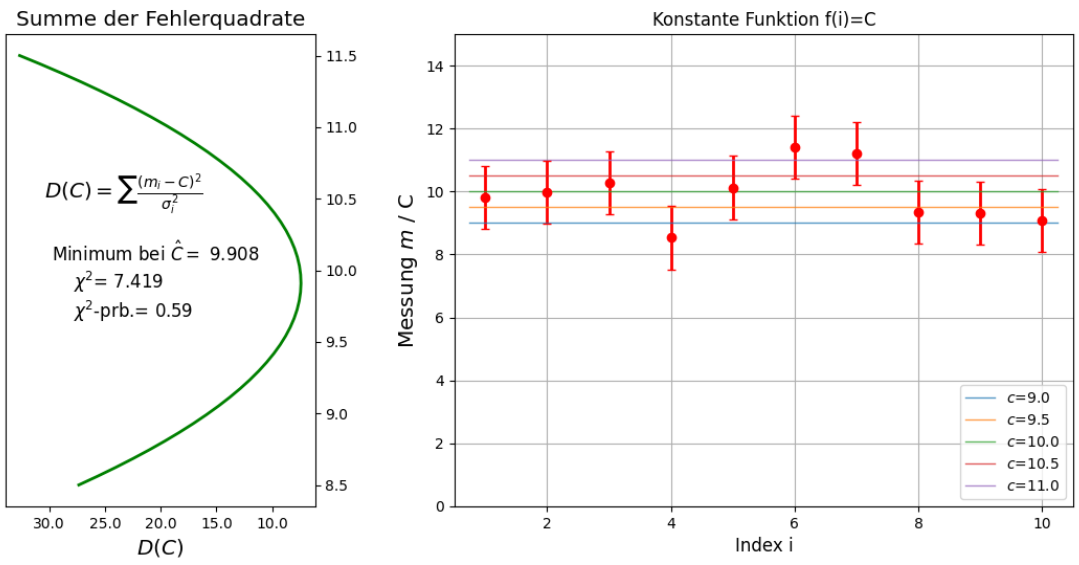


Abbildung 2: In der rechten Grafik sind 10 Messwerte und 5 mögliche konstante Funktionen gezeigt. Der Wert des Abstandsmaßes für verschiedene Werte der Konstanten ist links dargestellt.

Das skizzierte Verfahren funktioniert noch gut mit zwei Parametern, bei drei und mehr wird es numerisch sehr ineffizient; dann setzt man spezielle Algorithmen der numerischen Optimierung ein, um das Optimum bezüglich aller n_p Parameter mit möglichst wenigen Funktionsauswertungen zu finden. Diese Algorithmen sind in Programmpaketen implementiert, die im Folgenden besprochen werden. Sie übernehmen auch die zur Bestimmung der Unsicherheiten notwendige Analyse des Verlaufs des Abstandmaßes in der Nähe des Minimums und optional die grafische Darstellung der Ergebnisse.

A2.3.1 Praktische Werkzeuge zur Funktionsanpassung

Ähnlich wie zur Darstellung von Messdaten gibt es fertige *Python*-Programme, die Anpassungen an Daten ausführen und die auf die Anforderungen in den physikalischen Praktika ausgerichtet sind. Eines davon ist *kafe2go*, ein anderes *run_phyFit.py* aus dem Paket PhyPraKit. Beide Programme nutzen (kompatible) Eingabedaten im *yaml*-Format. *kafe2go* bietet mehr Möglichkeiten an als das einfacher gehaltene *run_phyFit.py*.

Wir können die Eingabe aus der Beispieldatei von oben verwenden, geben diesmal aber ein:

```
python3 run_phyFit.py data.ydat
```

oder

kafe2go data.ydat‘ .

Anstatt die Modellfunktion einfach nur grafisch darzustellen, wird nun eine Anpassung durchgeführt, und in der Grafik und auf der Textkonsole werden die besten Parameterwerte, deren Unsicherheiten und die χ^2-Wahrscheinlichkeit ausgegeben. Das Ergebnis ist in der Grafik unten dargestellt. Zusätzlich wird noch ein schattiertes Band um die Funktion herum angezeigt, das den Unsicherheitsbereich der Funktionsvorhersage wiedergibt und mittels Fehlerfortpflanzung aus den Parameterunsicherheiten berechnet wird.

Die Ergebnis mit *kafe2go* ist identisch, wie in der folgenden Grafik gezeigt.

A2.3.2 Komplexere Unsicherheiten

Die große Stärke der Pakte *kafe2* und *phyFit* ist deren Fähigkeit, mit komplexen Arten von Unsicherheiten umgehen zu können. Solche Unsicherheiten betreffen die $x-$ und/oder y-Werte, sie können unabhängig voneinander oder korreliert, relativ oder absolut sein. Korrelierte Unsicherheiten betreffen gegebenenfalls nur Untergruppen der Daten, z.B. jeweils in gleichen Messbereichen eines Multimeters gemessene Ströme

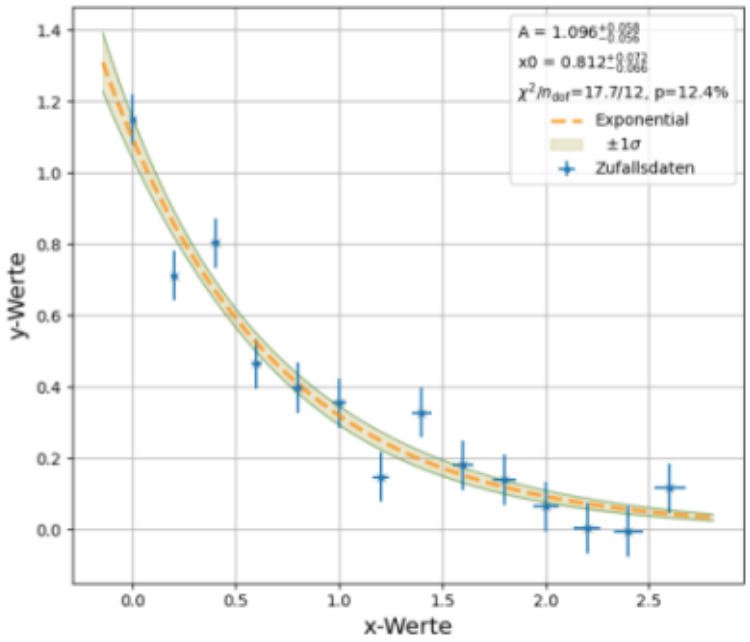


Abbildung 3: Anpassung einer Exponentialfunktion an Messdaten mit *run_phyFit.py*

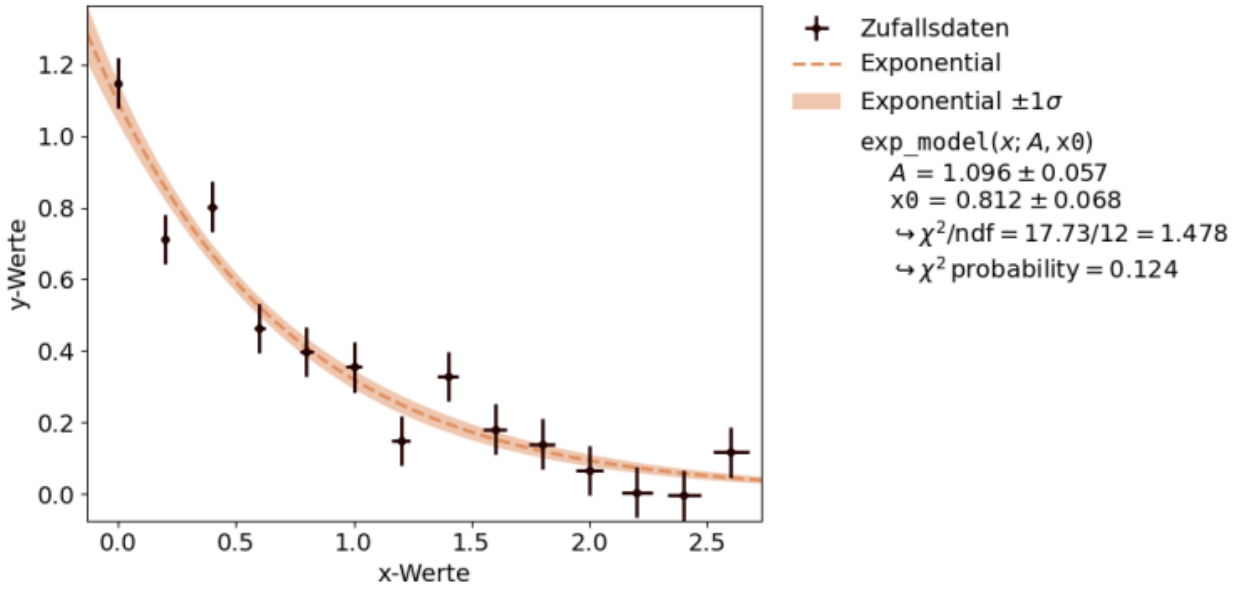


Abbildung 4: Anpassung einer Exponentialfunktion an Messdaten mit *kafe2go*

oder Spannungen. Solche Szenarien lassen sich mit den Programmen korrekt abbilden. Die Struktur der Unsicherheiten kann in *yaml* abgebildet werden und sieht ganz allgemein so aus:

```
# general structure of error block
y_errors:  ( OR x_errors:)
- error_value: <float> OR <[list of floats]>
  correlated:  0. OR 1.
  relative:    false OR true
# possibly, add more kinds of uncertainties ...
- error_value ...
  ...
```

Für *kafe2go* können von solchen Arten von Unsicherheiten beliebig viele angegeben werden, die dann in einer gemeinsamen Kovarianzmatrix aller Unsicherheiten zusammengefasst werden. In *phyFit* wird momentan nur eine Angabe pro Unsicherheits-Typ unterstützt.

Für relative Unsicherheiten, die auf den Modellwert bezogen werden, oder für Unsicherheiten in x-Richtung, die unter Annahme des Modells per Taylor-Entwicklung auf Unsicherheiten in y-Richtung transformiert werden, muss diese Matrix in jedem Schritt der numerischen Optimierung neu berechnet werden. Das ist zwar ein sehr aufwändiges Verfahren, auf modernen Laptops aber überhaupt kein Problem. Einige auch komplexere Beispiele sind im Verzeichnis `examples/` im Paket *PhyPraKit* enthalten und können als Basis zur Lösung von eigenen Anpassungsproblemen dienen.

Auch durch externe Kenntnisse eingeschränkte Parameter, wie in 3.5 beschrieben, können mit diesen Werkzeugen behandelt werden. Der entsprechende Eintrag in der *yaml*-Datei sieht so aus:

```
parameter_constraints:
  <name1>:
     value: <v>
     uncertainty <u>
  <name2>:
    ...
```

A2.3.3 Vereinfachte Behandlung komplexer Unsicherheiten

Viele der gängigen Programme zur Funktionsanpassung unterstützen keine Unsicherheiten in x-Richtung oder korrelierte Unsicherheiten. Bei deren Verwendung muss dann auf andere Verfahren zurückgegriffen werden, die lediglich einfache Anpassungen nur unter Berücksichtigung der statistisch unabhängigen Unsicherheiten in y-Richtung erfordern. Eventuell vorhandene weitere korrelierte Unsicherheiten, relative Unsicherheiten und/oder Unsicherheiten in x-Richtung werden dann in nachfolgenden Schritten hinzugefügt, indem man die Unsicherheiten in y-Richtung neu berechnet und die Anpassung ggf. mehrfach wiederholt.

Solche Verfahren, wie sie unten kurz skizziert sind, waren auch üblich, als Computer noch weniger leistungsfähig waren; sie werden auch heute noch eingesetzt, wenn die Berechnung der Modellfunktion sehr aufwändig ist.

a) Behandlung von Unsicherheiten in x-Richtung

Unsicherheiten der Datenpunkte bzgl. der Abszissenachse, d.h. der x_i, können folgendermaßen berücksichtigt werden:

1. zunächst erfolgt eine Anpassung ohne Berücksichtigung der Unsicherheiten der Abszissenwerte;
2. im einem zweiten Schritt werden diese dann mit Hilfe der ersten Ableitungen $f'(x_i)$ der im ersten Schritt angepassten Funktion f in entsprechende Unsicherheiten in y-Richtung umgerechnet und quadratisch zu den bereits vorhandenen Unsicherheiten in y addiert:
   $\sigma_i{}^2 = \sigma_{y_i}^2 + (f'(x_i) \cdot \sigma_{xi})^2$. Mit diesen modifizierten Unsicherheiten wird die Residuensumme S minimiert.
3. Ein dritter Schritt, der der Vorgehensweise beim zweiten Schritt entspricht, dient zur Verbesserung des Ergebnisses und zur Fehlerkontrolle - der Wert S_0 am Minimum darf sich vom zweiten zum dritten Schritt nicht signifikant ändern, ansonsten muss nochmals iteriert werden.

b) Behandlung von Parametern mit externen Einschränkungen

Modellparameter, die durch externes Wissen im Rahmen ihrer (gaußförmigen) Unsicherheiten eingeschränkt sind, lassen auch mit einem Anpassungswerkzeug ohne die Möglichkeit zur Berücksichtigung von Constraints (s. 3.5) behandeln, indem man Anpassungen mit jeweils um $\pm\sigma_{p_k}$ veränderten Werten des betreffenden Parameters p_k durchführt. Dabei verändern sich natürlich die Werte der angepassten Parameter um $\Delta_j^{\pm}$. Um symmetrische Unsicherheiten zu erhalten, mittelt man beide Werte und erhält so den Wert der systematischen Unsicherheit $\Delta_j = (\Delta_j^+ + \Delta_j^-)/2$, den man quadratisch zur ursprünglichen Parameterunsicherheit σ_j^0 addiert:

$\sigma_{a_j}{}^2 = \sigma_{a_j}^0{}^2 + \Delta_j^2 \,.$

c) Behandlung von korrelierten Unsicherheiten

Korrelierte Unsicherheiten σ_i^c der Datenpunkte können berücksichtigt werden, indem man sie in der Anpassung zunächst vernachlässigt und in vor einer weiteren Anpassung zu den Datenpunkten addiert bzw. davon subtrahiert:

$y_i' = y_i \pm \sigma_i^c$ bzw.

$x_i' = x_i \pm \sigma_i^c$ im Fall von Unsicherheiten in x-Richtung.

Die nach wiederholten Anpassungen gemittelten Änderungen Δ_j der interessierenden Parameter a_j addiert man wie im Fall b) quadratisch zu den ursprünglichen Parameterunsicherheiten σ_{a_j}.

Wenn das verwendete Anpassungswerkzeug die Berücksichtigung eingeschränkter Parameter erlaubt, kann man korrelierte Unsicherheiten auch als Verschiebung $o = 0$ bzw. als als Skalierungsfaktor $s = 1$ mit Unsicherheiten σ_o bzw. σ_s auffassen und im Modell $f(x_i)$ durch $o + s \cdot f(x_i)$ bzw. $f(o + s \cdot x_i)$ im Fall von Unsicherheiten in x-Richtung ersetzen. Die Unsicherheiten bzgl. Verschiebung oder Skalierung werden so korrekt auf die Parameterunsicherheiten propagiert.

d) Behandlung von relativen Unsicherheiten

Wenn Unsicherheiten von den Messwerten selbst oder von den Modellparametern abhängen, also z.B. auf den wahren Wert bezogenen relative Unsicherheiten oder auch statistische Unsicherheiten in Zählexperimenten, sind die Grenzen der Methode der kleinsten Fehlerquadrate erreicht. In solchen Fällen müssen zur Vermeidung von Verzerrungen die Unsicherheiten in jedem Schritt der Minimierung neu auf Grund der jeweils aktuellen Parameterwerte berechnet werden.

Man kann relative Unsicherheiten zunächst auf die gemessenen Werte beziehen und eine erste Anpassung durchführen. Dann werden für einen zweiten Schritt die Unsicherheiten auf Grund der nun näherungsweise bekannten Modellwerte erneut berechnet und die endgültige Anpassung durchgeführt, aus der man die finalen Parameterunsicherheiten entnimmt.

A3: Arbeiten mit *Jupyter Notebooks*

Das quelloffene Projekt *Jupyter* bietet eine Web-basierte interaktive Umgebung für wissenschaftliche Berechnungen und Datenauswertung Die zu Grunde liegenden Dateien sind sogenannte „Notebooks“, die sowohl formatierten Text als auch Programmcode u.a. in der Sprache *Python* und die entsprechende Programmausgabe in Form von Text und Grafiken in einem einzigen Dokument vom Typ `.ipynb` speichern.

Grundlage der Texteingabe und -formatierung bildet das oben schon erwähnte, von Menschen und Maschinen gut zu lesende *Markdown*-Format. Bei diesem modernen Konzept dient ein beliebiger Web-Browser als grafisches Interface zu einem Jupyter-Server, der entweder im Netzwerk oder auch lokal auf dem gleichen Computer laufen kann. Die Installation eines lokalen *Jupyter*-Sververs als Möglichkeit einer Umgebung zur Datenauswertung wurde bereits oben in Kap. 4. besprochen. Damit lassen sich die Vorgaben zur Dokumentation von Messwerten und der damit durchgeführten Analysen inklusive des verwendeten Programmcodes vollständig nachvollziehbar und reproduzierbar dokumentieren.

Die Fakultät für Physik betreibt einen Jupyter-Server, der Studierenden zur Verfügung steht. Informationen dazu findet man in den einführenden [Tutorials zur Nutzung von Jupyter Notebooks]. Dieser Server ermöglicht Datenauswertungen ohne die Notwendigkeit der Installation eigener Software. Eine *Jupyter* Testumgebung auch ohne Zugang zum Server der Fakultät wird auf der [Homepage des Autors] bereit gestellt. Dort finden

sich auch eine Reihe an grundlegenden und weiter führenden Tutorials zur Anwendung der *Jupyter*-Umgebung, zur Einführung in die Grundlagen der Statistik und Fehlerrechnung sowie zu fortgeschritteneren Themen wie die Nutzung des Programmierinterfaces von *kafe2*, zur Maximum-Likelihood-Methode und zum modernen Datenverwaltungs- und -analyse-Werkzeug *pandas*.

Kenntnisse der Sprache *Python* und die sehr komfortable *Jupyter*-Umgebung bilden heute die Grundlage für die wissenschaftliche Datenauswertung in vielen Disziplinen. In den Veranstaltungen zu den Grundlagen der Rechnernutzung und der computergestützten Datenauswertung wurden *Jupyter Notebooks* als Entwicklungsumgebung für *Python*-Programme bereits eingeführt.
Die gut strukturierte Oberfläche ist intuitiv erlernbar. In den Praktika werden überdies Vorlagen bereit gestellt, die die Versuchsanleitungen sowie notwendige Hilfsmittel wie Formeln, fertige Programme oder auch Code-Fragmente schon enthalten.

***Jupyter*-Notebook mit stand-alone Programmen zur Datenauswertung** An dieser Stelle möchten wir explizit auf das Jupyter-Notebook *tutorial_StandAloneTools* aus dem Paket *PhyPraKit* verweisen, das Sie im Verzeichnis *PhypraKit/ipynb_examples* finden.

Es enthält einige Beispiele, die Darstellung und Auswertung von Daten auch ohne eigenen Programmcode mit Hilfe der in *A1 c)* beschriebenen Programme *plotData.py*, *kafe2go.py* oder *run_phyFit.py* bewerkstelligen. Die Eingabedaten werden dazu in Textdateien im ebenfalls oben beschriebenen *yaml*-Format bereit gestellt. Als Namenskonvention wurden die Endungen *.ydat* für Daten und *.yfit* für Eingabedateien gewählt.

Die in diesem Tutorial beschriebenen Methoden und Vorgehensweisen bilden die Grundlage für die Datenauswertung und Dokumentation im Physikalischen Anfängerpraktikum.

A4: Tipps für MS Windows

A4.1. Wichtige Befehle für die Kommandozeile

Anmerkung: Bitte die *Windows Konsole* und nicht die neuere *PowerShell* benutzten, da letztere ggf. einige zusätzliche Konfigurationen erfordert, um die hier vorausgesetzte Verhaltensweise zu erzeugen!

Tipp: Starten Sie am Besten die *Windows Konsole* und probieren Sie die Bespiele gleich aus !

Befehle für Windows Eingabeaufforderung:

- `cd <Ordnerpfad>` (= change directory): in anderes Verzeichnis wechseln
- `cd ..` ins Verzeichnis darüber wechseln
- `dir` zeigt eine Liste der Dateien und Ordner
- `move <Quelldatei> <Zielordner>` verschieben von Dateien
- `copy <Quelldatei(-Pfad)> <Zielordner>` Kopieren von Dateien
- `md <Ordnername>` Erzeugen eines neuen Ordners
- `rd <Ordnername/-pfad>` Löschen eines (leeren) Ordners
- `type <Dateiname>` Ausgabe einer Text-Datei auf Bildschirm

Windows-Programme lassen sich ebenfalls bequem von der Kommandozeile aus starten:

- `notepad <Dateiname>` startet den Texteditor
- `idle <Dateiname>` startet die einfache *Python*-Umgebung *idle*
- `pip install <Paketname>` installiert ein *Python*-Paket
- `pip install --upgrade <Paketname>` dient zu Aktualisierung von *Python*-Paketen
- `pip install --upgrade --pre <Paketname>` berücksichtigt auch Vorabversionen bei der Aktualisierung
- `<name>.py [parameter]` startet ein *Python*-Programm mit dem angegebenen Parameter, wenn die Verknüpfung von Dateien mit der Endung `.py` mit *Python.exe* hergestellt und sich das *Python*-Skript im Windows-Suchpfad für ausführbare Programme befindet (s. unten)
- `pdflatex <name>.tex` erzeugt eine Ausgabedatei im *pdf*-Format aus einer *LaTeX*-Datei
- `pandoc <name.md> -o <output-name>.pdf` erzeugt eine Ausgabedatei im *pdf*-Format aus einer Datei im Markdown-Format

A4.2. Installation von *Python* unter Windows

Python für Windows installiert man am besten direkt von der Homepage der Python Foundation. Die jeweils aktuellste sowie auch ältere Versionen können dort als Installationspaket für Windows herunter geladen werden. Für diese Dokumentation wurde `Python 3.10.5` verwendet und auf den Versionen Windows 10 und 11 getestet.

Zur Installation wählen Sie die gewünschte *Python*-Version (3.10.x) aus und laden Sie den passenden Windows Installer herunter - bei den meisten modernen Systemen ist das *Windows Installer (64-bit)*. Starten Sie nach dem Download den Installer und beachten Sie die angebotenen Optionen - idealerweise sollten alle Komponenten im Fenster ausgewählt werden, insbesondere auch der `py launcher`, der als ausführbare Datei im Verzeichnis `C:\Windows\`installiert wird. Im zweiten Fenster des Installationsprogramms gibt es weitere Optionen. Man kann *Python* systemweit für alle Nutzer installieren, wenn man Administrator-Rechte auf dem System hat. Ansonsten werden die Daten für die Installation im Nutzerbereich als Unterordner im Verzeichnis `C:\Users\<name>\AppData` abgelegt. Die Option *Asscociate files with Python* sollte ebenfalls ausgewählt werden - dann werden Dateien mit der Endung `.py` automatisch mit der richtigen Anwendung (*Python*) gestartet. Auch die Option `Add Python to environment variables` ist empfehlenswert, damit alle im Unterverzeichnis `\Scripts` des *Python*-Installationsverzeichnisses abgelegten Programme ohne Pfadangabe ausgeführt werden können.
Als letztes kann noch das Installationsverzeichnis ausgewählt werden;
hier empfiehlt es sich, einen einfachen Pfadnamen auszuwählen, damit man später die Python-Installation wiederfindet und ggf. Desktop-Icons für häufig genutzte Programme anlegen kann.

Die Verknüpfung von *Python*-Programmen mit der passenden Anwendung kann auch noch nach der Installation erfolgen. Dazu mit der rechten Maus auf eine *Python-Datei* klicken und "Öffnen mit" anklicken, dann die Anwendung `C:\Windows\py.exe` auswählen. Nun können *Python*-Programme durch Anklicken oder auch auf der Kommandozeile durch Angabe ihres Namens ausgeführt werden. Ebenfalls angegebene Parameter, wie Ein- oder Ausgabedateien oder Optionen, werden dabei übernommen und an das *Python*-Programm übergeben.

Wenn *Python* systemweit installiert wurde, sollten alle mit *pip* nachinstallierten Pakete ebenfalls mit Administrator-Rechten installiert werden, damit die sie in den korrekten Unterverzeichnissen der *Python*-Installation gespeichert werden.

A4.3. Eigenen Bereich in den Pfad der ausführbaren Programme aufnehmen.

Für Aufgaben in der Datenauswertung werden meist eine ganze Reihe von Programmen benötigt, die Daten aus einem Arbeitsverzeichnis lesen oder neue darin erzeugen. In Windows wird eine Liste von Pfaden verwaltet, in denen nach den Namen von Programmen gesucht wird. Bei der Installation werden diese Pfade dort üblicherweise eingetragen; allerdings gilt dies nicht für *Python*-Programme, die auf anderem Weg kopiert werden, z.B. die oben angeführten Werkzeuge für grafische Darstellungen oder zur Funktionsanpassung. Damit diese Programme nur mit ihrem Namen ohne Angabe des zum Teil langen Verzeichnispfades ihres Speicherorts aufgerufen werden können, müssen sie in einem Verzeichnis liegen, das sich im Suchpfad befindet. Dazu sollte ein entsprechendes Verzeichnis angelegt, in den Suchpfad eingetragen und dann die benötigten Programme (bzw. Links darauf, siehe unten) dort hin kopiert werden.

Dazu im Verzeichnis des Nutzers ein neues Verzeichnis für die Hilfsprogramme erzeugen, z.B. mit dem Namen `Programme`:

```
> mkdir C:\Users\<name>\Programme
```

Dann auf der Kommandozeile das neue Verzeichnis in die Liste der für ausführbare Programme durchsuchten Pfade eingeben:

```
> setx PATH "%PATH%;C:\Users\<name>\Programme"
```

Nun das Fenster schließen und die Eingabeaufforderung neu starten. Mit

```
> path
```

den aktuellen Suchpfad ausgeben und überprüfen, ob alles korrekt übernommen wurde.

Python-Programme oder andere *.exe*-Dateien, die in das neu angelegte Verzeichnis kopiert wurden, können nun direkt mit ihrem Namen ohne Pfad-Angabe aufgerufen werden.

A4.4. Link für Dateien erzeugen

Programmdateien sind oft sehr groß, und daher legt man häufig nur einen Verweis, einen "Link" anstelle einer Kopie an. In Windows gibt es dazu den sog. Hardlink, den man nur über die Kommandozeile erzeugen kann:

```
> mklink /h <Verknüpfung> <Ziel>
```

Wenn man also im gemeinsamen Bereich aller Hilfsprogramme solche Hardlinks auf häufig benötigte Programme anlegt, kann man sie von jedem Verzeichnis aus durch Eingabe ihres Namens starten.

Mit Eingabe von

```
> mklink /j <Verknüpfung> <Zielverzeichnis>
```

kann man auch auf ein ganzes Verzeichnis verweisen. Dies ist nützlich, um ggf. lange Pfade abzukürzen.

Mit dem Befehl

```
> del <Ziel>
```

kann man solche Links auch wieder löschen, ohne dabei die Originaldaten zu beeinflussen.

A4.5 Arbeiten mit *Origin*

[*Origin*] ist eine weit verbreitete kommerzielle Software ausschließlich für das Betriebssystem Microsoft Windows zur Visualisierung und Analyse von Daten, das nur gegen Lizenzgebühr genutzt werden kann. *Origin* ist im Softwarekatalog für Akademische Forschung und Lehre enthalten und für KIT-Angehörige im [Softwareshop des KIT SCC] kostenlos verfügbar.

Origin zeichnet sich durch ein an Microsoft Excel angelehntes Bedienkonzept aus, das Datenerfassung ähnlich wie in gängigen Programmen zur Tabellenkalkulation ermöglicht. Es gibt eine sehr umfangreiche Auswahl an grafischen Darstellungsmöglichkeiten und Analysewerkzeugen, zu denen auch die Anpassung beliebiger Funktionen an Messdaten mit Unsicherheiten gehört. *Origin* deckt eine große Palette an Anforderungen aus verschiedenen Wissenschaftsbereichen ab, und daher ist die Auswahl der für physikalische Auswertungen sinnvollen Einstellungen und üblichen Darstellungsformen nicht ganz einfach. In der [Kurzanleitung zu Verwendung von Origin für das Praktikum für Lehramt] sind die notwendigen Informationen zusammengestellt.