

Fortgeschrittenen Praktikum

Patrick Friebl, WS 24

Versuch F44

Lennart Bederke, Yago Obispo Gerster

1 Motivation

In diesem Experiment befassen wir uns sowohl mit dem Zeeman-Effekt als auch mit der Aufzeichnung von Spektren. Der Zeeman-Effekt bezeichnet die experimentelle Beobachtung, dass ein äußeres Magnetfeld zu einer Aufspaltung der Spektrallinien führt. Der niederländische theoretische Physiker Hendrik Antoon Lorentz sagte Ende des 19. Jahrhunderts erstmals voraus, dass ein äußeres Magnetfeld eine dreifache Aufspaltung der Spektrallinien bewirken würde. Diese Vorhersage wurde 1896 experimentell durch seinen Landsmann Pieter Zeeman bestätigt und nach ihm benannt: „(normaler) Zeeman-Effekt“. Für diese Leistungen wurden die beiden Wissenschaftler 1902 mit dem Nobelpreis für Physik ausgezeichnet, was entscheidend zur Etablierung der Vorstellung beitrug, dass das Elektron ein Bestandteil des Atoms ist.

Zur Analyse von Lichtspektren können Gitter- und Prismenspektrometer eingesetzt werden, die Licht je nach Wellenlänge in unterschiedlichen Winkeln beugen und brechen. Betrachtet man das von denselben Atomen emittierte Licht und analysiert es mit einem Spektrometer, so findet man spezifische Linien, die charakteristisch für diese Atome sind. In Kombination mit den Experimenten von Rutherford zu Beginn des 20. Jahrhunderts und der von Planck formulierten Erklärung des Schwarzkörperspektrums, die auf Lichtquanten basiert, ergab sich ein besseres Verständnis der Atomstruktur, das 1913 von Bohr formalisiert wurde. Durch Fortschritte in der Spektroskopie wurde das Atommodell in den folgenden Jahren verfeinert, um verschiedene quantenmechanische Effekte zu berücksichtigen.

Im Rahmen des Experiments werden wir den Zeeman-Effekt beobachten und ein Czerny-Turner-Spektrometer, das ein optisches Gitter verwendet, nutzen, um das Spektrum von Neon und Cadmium zu analysieren.

2 Physikalische Grundlagen

2.1 Normaler Zeemann-Effekt

Wenn man eine anschauliche Erklärung für den Einfluss eines äußeren Magnetfelds auf das Emissionsspektrum eines Atoms geben möchte, ist es hilfreich, ein Atom als einen positiv geladenen Kern zu beschreiben, der von einem negativ geladenen Elektron umkreist wird. Durch diese Vereinfachung wird der Kern von einem Strom I umgeben. Da ein elektrischer Strom vorliegt, entsteht ein magnetisches Moment $\vec{\mu}$. Die Situation lässt sich durch die folgende Gleichung beschreiben:

$$\vec{\mu} = I \cdot \mathbf{A} = I \cdot \pi r^2 \mathbf{n} = \frac{evr}{2} \cdot \mathbf{n} \quad \text{mit :} \quad I = -e \cdot \frac{v}{2\pi r} \quad (1)$$

Da sich das Elektron in einer Ebene bewegt, kann der Flächenvektor \mathbf{A} als $\mathbf{A} = \pi r^2 \mathbf{n}$ beschrieben werden. Wird ein äußeres Magnetfeld \mathbf{B} angelegt, so interagiert dieses mit dem

magnetischen Drehmoment des Elektrons. Aufgrund dieser Wechselwirkung verändert sich die potenzielle Energie des Elektrons, was sich in der folgenden Gleichung ausdrückt:

$$\Delta E_{pot} = -\vec{\mu} \cdot \mathbf{B} = \frac{e}{2m_e} \cdot \mathbf{l} \cdot \mathbf{B} \quad \text{mit : } \mathbf{l} = \mathbf{r} \times \mathbf{p} = m_e \cdot r \cdot v \cdot \mathbf{n} \quad (2)$$

Da der Drehimpuls des Elektrons quantisiert ist, lässt sich dieser wie folgt beschreiben, wobei die Richtung des B -Feldes als Quantisierungsachse z verwendet wird:

$$\begin{aligned} |\mathbf{l}| &= \sqrt{l(l+1)}\hbar \quad \text{mit : } l = 0, 1, \dots, n-1 \\ l_z &= m_l \cdot \hbar \quad \text{mit : } -l \leq m_l \leq l \end{aligned} \quad (3)$$

Somit kann Gleichung (2) umgeschrieben werden, wobei das Bohrsche Magneton μ_B verwendet wird:

$$\Delta E_{pot} = \frac{e\hbar}{2m_e} \cdot m_l \cdot B = \mu_B \cdot m_l \cdot B \quad (4)$$

Abschließend lässt sich festhalten, dass das äußere Magnetfeld die ursprünglichen Energieniveaus in $2l+1$ Niveaus aufspaltet, die denselben l -Wert besitzen, aber unterschiedliche m_l -Werte. Diese Niveaus waren zuvor entartet.

2.2 Annomaler Zeemann-Effekt

Die im vorherigen Abschnitt dargestellte Erklärung des Zeeman-Effekts gilt nur, wenn der Gesamtdrehimpuls $S = 0$ ist. In diesem Fall spricht man vom normalen Zeeman-Effekt, da eine quantenmechanische Beschreibung für die Erklärung noch nicht erforderlich ist. Ist jedoch der Gesamtdrehimpuls ungleich null, bezeichnet man dies als anomalen Zeeman-Effekt, der im Folgenden beschrieben wird.

Zunächst wird der ungestörte Hamiltonoperator \hat{H}_0 , der das Atom beschreibt, um einen Störterm \hat{H}_B erweitert, welcher durch das Magnetfeld verursacht wird. Wenn das angelegte Magnetfeld schwach ist (bei einem starken Magnetfeld spricht man vom Paschen-Back-Effekt) und die Spin-Bahn-Kopplung der Elektronen dominiert, werden zunächst die Spins \mathbf{s}_i und die Bahndrehimpulse \mathbf{l}_i der einzelnen Elektronen zum Gesamtdrehimpuls \mathbf{S} bzw. zum Gesamtbahndrehimpuls \mathbf{L} zusammengefasst. Anschließend werden \mathbf{S} und \mathbf{L} zum Gesamtdrehimpuls \mathbf{J} addiert. Diese Beschreibung ist gültig, solange die Kopplung zwischen den einzelnen Bahndrehimpulsen \mathbf{l}_i wesentlich stärker ist als die Kopplung zwischen dem Bahndrehimpuls und dem Spin eines Elektrons.

Mathematisch lässt sich dies wie folgt ausdrücken:

$$\begin{aligned} \mathbf{L} &= \sum_i \mathbf{l}_i \quad \text{mit: } |\mathbf{L}| = \sqrt{L(L+1)}\hbar \\ \mathbf{S} &= \sum_i \mathbf{s}_i \quad \text{mit: } |\mathbf{S}| = \sqrt{S(S+1)}\hbar \\ \mathbf{J} &= \mathbf{L} + \mathbf{S} \quad \text{mit: } |\mathbf{J}| = \sqrt{J(J+1)}\hbar \\ J_z &= M_J \cdot \hbar \quad \text{mit: } -J \leq M_J \leq J \end{aligned} \quad (5)$$

Der Hamiltonoperator für diese Situation sowie die resultierenden Energieniveausverschiebungen ΔE_{pot} unter Berücksichtigung des Landé-Faktors g_J lauten:

$$\hat{H}_0 + \hat{H}_B = \hat{H}_0 - \frac{\mu_B}{\hbar} (\mathbf{L} + \mathbf{S}) \cdot \mathbf{B}$$

$$\Delta E_{pot} = \mu_B \cdot B \cdot M_J \cdot g_J$$

$$g_J = 1 + \frac{J(J+1) + S(S+1) - L(L+1)}{2J(J+1)}$$
(6)

Beim normalen Zeeman-Effekt hängt die Aufspaltung der Linien ausschließlich vom magnetischen Quantenzahl m_l ab, sodass die Aufspaltung für alle Energieniveaus identisch ist. Beim anomalen Zeeman-Effekt hingegen variiert die Größe der Aufspaltung zwischen den Niveaus, da sie von den drei Quantenzahlen J, L und S abhängt.

Abbildung 1 zeigt eine Darstellung des normalen und des anomalen Zeeman-Effekts.

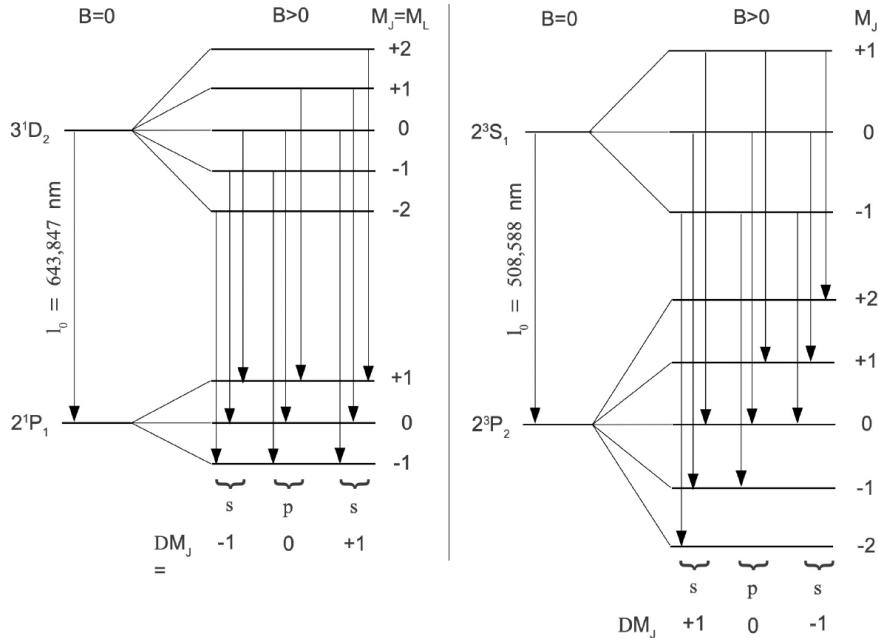


Abbildung 1: Links wird der normale Zeeman-Effekt dargestellt, bei dem die Aufspaltung der Linien von der magnetischen Quantenzahl des Bahndrehimpulses abhängt. Es ist klar erkennbar, dass sich die Linien für 3^1D_2 und 2^1P_1 äquidistant aufspalten, entsprechend der Quantenzahl l . Rechts ist der anomale Zeeman-Effekt abgebildet, bei dem die Aufspaltung von der magnetischen Quantenzahl des Gesamtdrehimpulses J abhängt. Hierbei ist die Abstandsgröße der Aufspaltung der Linien für 2^3S_1 und 2^3P_2 unterschiedlich

2.3 Auswahlregeln

Wie in Abbildung 1 zu erkennen, treten nur spezifische Linien auf. Für einen Übergang ist neben der Erhaltung von Energie, Impuls und Drehimpuls sowie Symmetrieeigenschaften erforderlich, dass eine Komponente des Übergangsmatrixelements ungleich null ist. Dies führt zu der Regel, dass Übergänge nur stattfinden, wenn die Differenz der magnetischen Quantenzahlen $\Delta M_J = M_{J,i} - M_{J,k} = 0, \pm 1$ beträgt. Für elektrische Dipolübergänge gilt zudem, dass die Änderung des Bahndrehimpulses $\Delta L = L_i - L_k = \pm 1$ ist, während der Spin

unverändert bleibt. Das Übergangsmatrixelement für die einzelnen Komponenten kann wie folgt berechnet werden :

$$(M_{ik})_q = e \int \Psi_i^* q \Psi_k \, dq \quad \text{mit} \quad q = x, y, z \quad (7)$$

2.4 Polarisierung des ausgesendeten Lichtes

Bei Übergängen zwischen Energieniveaus unterscheidet die Polarisierung des Lichts zwischen π -Übergängen (für linear polarisiertes Licht) und σ -Übergängen (für zirkular polarisiertes Licht). Wenn beispielsweise das Magnetfeld in z-Richtung orientiert ist, ist $(M_{ik})_z$ das einzige nicht verschwindende Übergangsmatrixelement für $\Delta M_J = 0$. In diesem Fall schwingt das elektrische Dipolmoment parallel zum Magnetfeld und emittiert Licht in einer Ebene, die senkrecht dazu liegt. Das Licht ist linear polarisiert, da der elektrische Feldvektor nur in eine Richtung schwingt.

Bleibt man beim Beispiel eines Magnetfelds in z-Richtung, so gilt für Übergänge mit $\Delta M_J = \pm 1$, dass das Übergangsmatrixelement $(M_{ik})_z$ null ist, während die Übergangsmatrixelemente $(M_{ik})_x$ und $(M_{ik})_y$ nicht null sind. Folglich wird Licht in longitudinaler Richtung entlang der z-Achse beobachtet, das eine zirkulare Polarisierung aufweist. Dies liegt daran, dass die überlagerten Dipole gleich stark sind, jedoch eine Phasenverschiebung von 90° aufweisen.

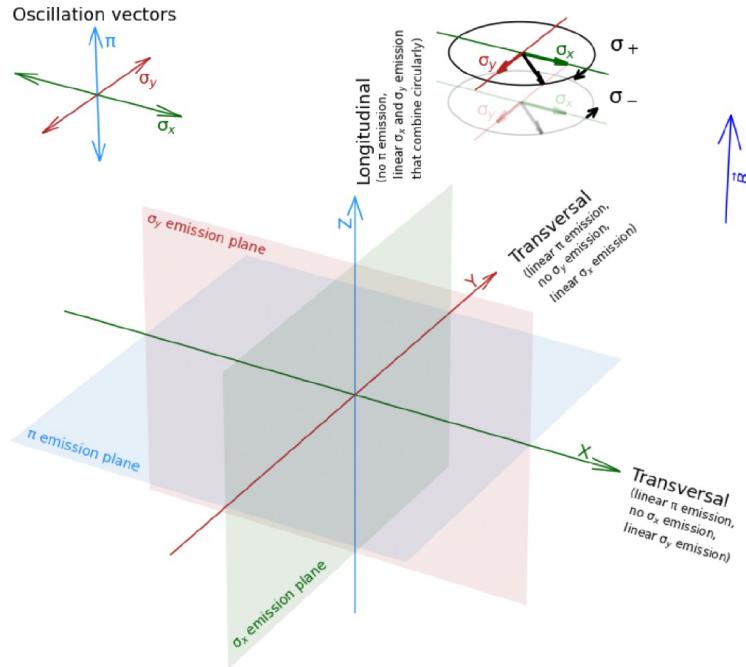


Abbildung 2: Die oben beschriebenen Überlegungen werden für ein spezifisches Beispiel visualisiert. Oben links sind die Richtungen dargestellt, in denen das elektrische Dipolmoment schwingt. Oben rechts wird die zirkulare Polarisierung des Lichts für den Fall der σ -Polarisation gezeigt. Unterhalb der Hauptabbildung sind die Emissionsebenen für die unterschiedlichen Polarisierungen dargestellt.

2.4.1 Lummer-Gehrcke-Platte

Um die Linienaufspaltung durch den Zeeman-Effekt sichtbar zu machen, ist eine sehr hohe Auflösung erforderlich. Diese wird durch die Lummer-Gehrcke-Platte erreicht, die aus Quarzglas mit parallelen Flächen besteht und unten dargestellt ist. Zwischen den Quarzplatten wird das Licht nahezu vollständig reflektiert, ein kleiner Anteil des Lichts tritt jedoch bei jeder Reflexion aus dem Quarzglas aus. Diese Lichtstrahlen interferieren miteinander, was zu einer hohen Auflösung führt. Das Licht wird anschließend mit einem Teleskop hinter der Quarzplatte beobachtet.

Der optische Gangunterschied zwischen zwei Lichtstrahlen, der im Folgenden mit Δ bezeichnet wird, muss ein ganzzahliges Vielfaches der Wellenlänge λ sein, damit konstruktive Interferenz auftritt. Für den Fall, dass die Platte einen Brechungsindex n besitzt und von Luft mit dem Brechungsindex $n = 1$ umgeben ist, ergibt sich der optische Gangunterschied aus:

$$\Delta = \Delta_1 - \Delta_2 = 2 \cdot d \cdot \sqrt{n^2 - 1} \quad (8)$$

Ob konstruktive Interferenz auftritt, hängt somit von der Wellenlänge λ des Lichts, dem Einfallswinkel β und dem Austrittswinkel α ab. Für den freien Spektralbereich der Lummer-Gehrcke-Platte ergibt sich die folgende Gleichung:

$$\Delta\lambda = \frac{\lambda^2}{2d \cdot \sqrt{n^2 - 1}} \quad (9)$$

Für die sehr kleinen Wellenlängenänderungen, die im Zeeman-Effekt auftreten, ergibt sich, wobei δa der Abstand zwischen λ und $\delta\lambda$ ist und Δa der Abstand zwischen zwei Ordnungen k und $k + 1$ der Wellenlänge λ darstellt:

$$\Delta\lambda = \frac{\delta a}{\Delta a} \cdot \frac{\lambda^2}{2d \cdot \sqrt{n^2 - 1}} \quad (10)$$

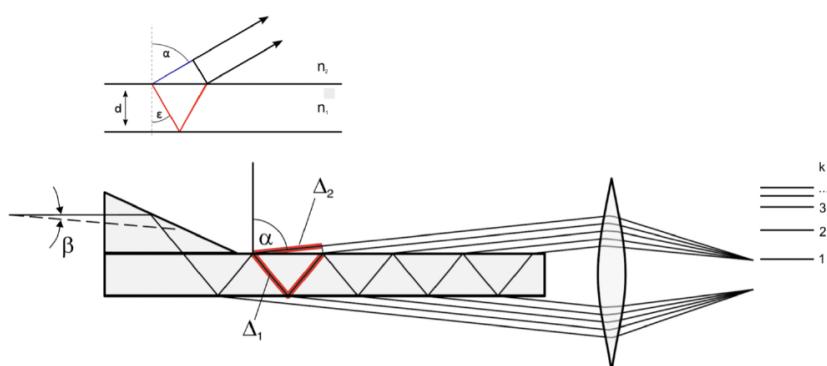


Abbildung 3: Darstellung der Lummer-Gehrcke Platte

3 Versuchsaufbau

In diesem Abschnitt wird der experimentelle Aufbau dargestellt, und die einzelnen Komponenten werden erläutert. In Abbildung 4 sind die Polstücke des Magneten mit a) markiert. Die Cd-Lampe, bezeichnet mit b), ist zwischen den Polstücken positioniert. Das Licht der Lampe fällt durch einen roten Filter bei c) auf die Lummer-Gehrcke-Platte bei d). Nach dem Durchgang durch das Teleskop bei e) kann das Licht mit dem Auge bei f) beobachtet werden. In der Abbildung sind außerdem Justierschrauben mit h) und i) gekennzeichnet. Nicht in der Abbildung gezeigt ist die Kamera, die bei f) angebracht werden kann, um die Spektroskopie-Bilder auf dem PC zu betrachten. Zusätzlich kann vor dem Teleskop ein Polarisationsfilter sowie eine $\lambda/4$ -Platte angebracht werden.

Im zweiten Teil des Experiments wird ein Czerny-Turner-Monochromator verwendet. Hier tritt Licht durch den Eingangs-Spalt ein, wird durch das optische Gitter spektral zerlegt, und anschließend wird die gewünschte Wellenlänge durch Anpassung der Breite des Ausgangs-Spalts ausgewählt. Dies ermöglicht eine präzise Kontrolle und Messung spezifischer Lichtwellenlängen.

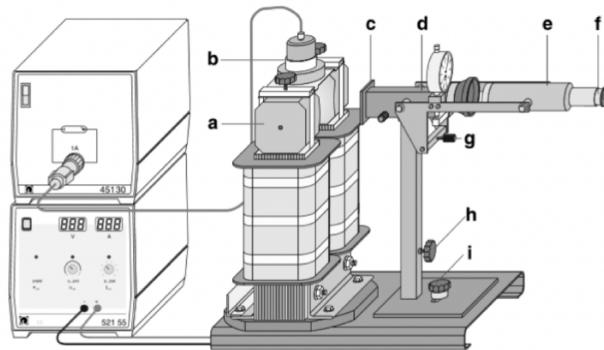


Abbildung 4: Versuchsaufbau

4 Spektroskopie des Zeemann Effektes

4.1 Magnetfeldstärke und Hysterese

Zunächst wurde die entsprechende magnetische Feldstärke für verschiedene Stromstärken gemessen. Um die Hysterese zu bestimmen, wurde der Strom zunächst bis auf 13 A erhöht und anschließend wieder auf 0 A reduziert. Ein erster Blick auf Abbildung 5 zeigt, dass die Hysterese vernachlässigbar ist, da sich die magnetischen Feldstärken bei steigendem und fallendem Strom kaum unterscheiden.

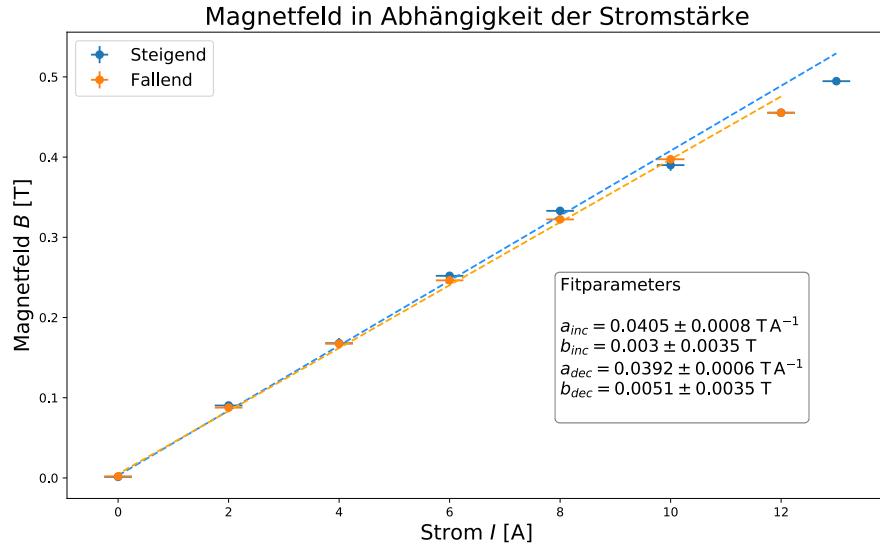


Abbildung 5: Magnetfeld gegen Stromstärke

Um die Hysterese weiter zu untersuchen, wurde eine Funktion der Form

$$f(x) = a \cdot x + b \quad (11)$$

an die Messwerte angepasst. Die Parameter der beiden Anpassungen sind in Abbildung 5 angegeben und unterscheiden sich ebenfalls kaum. Für alle weiteren Berechnungen der magnetischen Feldstärke als Funktion des Stroms wurde der Fit für den steigenden Strom verwendet, da dieser geringfügig höhere Werte liefert als der Fit für den abnehmenden Strom. Somit wird die theoretisch existierende, aber praktisch nicht messbare Hysterese besser berücksichtigt.

Der Fit wurde insbesondere verwendet, um die folgenden Werte für die magnetische Feldstärke zu bestimmen, die für nachfolgende Berechnungen genutzt werden.

| I [A] | 10.00 ± 0.25 | 11.00 ± 0.25 | 12.00 ± 0.25 | 13.00 ± 0.25 |
|---------|-------------------|-------------------|-------------------|-------------------|
| B [A] | 0.408 ± 0.013 | 0.448 ± 0.014 | 0.489 ± 0.014 | 0.529 ± 0.015 |

Tabelle 1: Caption

Diese Werte wurden wie folgt berechnet:

$$B(I) = a_{inc} \cdot I + b_{inc} \quad (12)$$

$$\Delta B(I) = \sqrt{(\Delta a_{inc} \cdot I)^2 + (\Delta b_{inc})^2 + (a_{inc} \cdot \Delta I)^2}$$

Dabei stellt $B(I)$ die berechnete magnetische Feldstärke bei einem Strom I dar, und $\Delta B(I)$ ist die entsprechende Unsicherheit, die sich aus den Unsicherheiten der Fit-Parameter (Δa_{inc} und Δb_{inc}) sowie der Strommessung (ΔI) ergibt.

4.2 Qualitative Beobachtung der π - und σ -Linien

4.2.1 Transversal

Diese und alle folgenden Beobachtungen des Zeeman-Effekts wurden mit der roten Cd-Linie ($\approx 643,8$ nm) durchgeführt, die mittels eines roten Filters vor der Lummer-Gehrcke-Platte isoliert wurde.

Zunächst (ohne Polarisationsfilter oder $\lambda/4$ -Platte) konnten wir in transversaler Richtung die Zeeman-Aufspaltung einer Spektrallinie in drei unterscheidbare Linien beobachten (siehe Abbildung 6), wie es aufgrund der Auswahlregel $\Delta M_J = 0, \pm 1$ zu erwarten ist. Zusätzlich ist zu erkennen, dass der Abstand zwischen den drei Linien mit zunehmender magnetischer Feldstärke wächst. Dies ist ebenfalls erwartungsgemäß, da die Energieverschiebung durch den Zeeman-Effekt, ΔE , direkt proportional zum Magnetfeld B ist.



Abbildung 6: Cd-Linie mit Magnetfeld

Durch das Hinzufügen und Drehen eines Polarisationsfilters vor dem Teleskop konnten entweder die π -Linie (zentrale Linie) oder die σ -Linien (beide äußeren Linien) beobachtet werden. Dies liegt daran, dass das magnetische Dipolmoment der π -Linie entlang der Achse des Magnetfeldes oszilliert und die π -Linie daher linear entlang dieser Achse polarisiert ist. Die σ -Linien hingegen sind zirkular polarisiert. Ihre Projektionen, bei Beobachtung in transversaler Richtung zum Magnetfeld, sind linear polarisiert, wobei ihre Polarisationsachsen senkrecht zum Magnetfeld bzw. zur π -Linie stehen. Als Folge dessen muss der Polarisationsfilter um 90° gedreht werden, um die σ -Linien anstelle der π -Linie zu sehen.

Wird der Polarisationsfilter in eine Position gedreht, in der nur die π -Linie sichtbar ist, und anschließend eine $\lambda/4$ -Wellenplatte vor dem Filter angebracht (das Licht passiert zuerst die Wellenplatte), können durch Drehen der Wellenplatte alle drei Linien wieder sichtbar gemacht werden, allerdings mit verringriger Intensität. Wenn der Winkel zwischen der optischen Achse der Wellenplatte und der Polarisationsachse der Projektionen der σ -Linien 45° beträgt, wandelt die Wellenplatte das linear polarisierte Licht der σ -Linienprojektionen in zirkular polarisiertes Licht um. Zirkular polarisiertes Licht kann jedoch einen Polarisationsfilter passieren, wodurch alle drei Linien in dieser Konfiguration sichtbar werden.

Wird lediglich die $\lambda/4$ -Wellenplatte ohne Polarisationsfilter vor das Teleskop gesetzt, bleiben alle drei Linien sichtbar, und es gibt keine Veränderung in den Beobachtungen.

4.2.2 Longitudinal

In longitudinaler Richtung konnte ohne ein angelegtes Magnetfeld erneut nur eine Spektrallinie beobachtet werden. Nach Einschalten des Magnetfeldes waren jedoch lediglich zwei Linien erkennbar. Der Grund hierfür ist, dass das magnetische Dipolmoment der π -Linie in Richtung des Magnetfeldes oszilliert und daher in dieser Richtung keine Emission erfolgt. Wird ein Polarisationsfilter vor das Teleskop gesetzt, bleiben weiterhin zwei Linien sichtbar, da die σ -Linien zirkular polarisiert sind.

Wird jedoch eine Kombination aus $\lambda/4$ -Wellenplatte und Polarisationsfilter verwendet (das Licht durchläuft zunächst die Wellenplatte), kann untersucht werden, dass eine der σ -Linien rechtszirkular und die andere linkszirkular polarisiert ist. Dies liegt daran, dass die Wellenplatte das rechtszirkular und linkszirkular polarisierte Licht in linear polarisiertes Licht mit senkrecht zueinander stehenden Polarisationsachsen umwandelt. Dreht man den Polarisationsfilter nun so, dass dessen optische Achse mit der Polarisationsachse einer der beiden Linien übereinstimmt, ist nur diese Linie sichtbar.

4.3 Unterscheidung zwischen Polarisationsfilter und Wellenplatte

Der Polarisationsfilter und die Wellenplatte lassen sich unterscheiden, indem man durch beide hindurch auf einen Computerbildschirm schaut. Der Computerbildschirm emittiert nämlich linear polarisiertes Licht. Der Polarisationsfilter blockiert daher das Licht vom Bildschirm, wenn die optische Achse des Filters korrekt ausgerichtet ist. Die Wellenplatte hingegen lässt das Licht vom Bildschirm in allen Orientierungen durch, da sie das polarisierte Licht in eine andere Polarisation umwandelt, ohne es vollständig zu blockieren.

4.4 Lage der π - und σ -Linien in den Zeeman-Spektren

Im nächsten Schritt wurden die Zeeman-Spektren der roten Cd-Linie in transversaler Richtung ohne Polarisationselemente oder Wellenplatte für vier verschiedene Ströme

$I = \{10, 11, 12, 13\} \text{ A}$ aufgenommen. Bevor die Spektren sinnvoll ausgewertet werden konnten, mussten diese zunächst normalisiert werden. Da die verschiedenen Linien bei höheren Ordnungen zunehmend dichter werden, ist die aufgezeichnete Intensität bei höheren Ordnungen stets größer. Zusätzlich wiesen die aufgezeichneten Intensitätswerte negative Werte auf. Zur Korrektur dieser beiden Punkte wurde zunächst eine lineare Funktion an die Minima angepasst und anschließend von den aufgezeichneten Intensitäten subtrahiert. Daraufhin wurde das gesamte Spektrum durch Addition eines konstanten Wertes zu allen Intensitäten in den positiven Bereich verschoben. Diese beiden Schritte führten zu einem Spektrum mit positiven Intensitätswerten, bei dem alle Interferenzordnungen ebenfalls annähernd die gleiche Intensität aufwiesen. Abschließend wurde die Intensität durch Division aller Intensitäten durch die maximale Intensität normalisiert. Eine dreifache Gauß-Funktion der Form

$$f(x) = a_1 \cdot \exp\left(\frac{(x - b_1)^2}{2 \cdot c_1^2}\right) + a_2 \cdot \exp\left(\frac{(x - b_2)^2}{2 \cdot c_2^2}\right) + a_3 \cdot \exp\left(\frac{(x - b_3)^2}{2 \cdot c_3^2}\right) + d \quad (13)$$

wurde anschließend auf die verschiedenen Ordnungen der aufgezeichneten Zeeman-Spektren angepasst, um die Position der π - und σ -Linien zu bestimmen. Eine Gauß-Funktion wurde als allgemeine Anpassungsfunktion für die Spektrallinien gewählt, da die Linienverbreite-

rung hauptsächlich durch die Doppler-Verbreiterung aufgrund der hohen Temperatur der Cd-Lampe verursacht wird, und die Doppler-Verbreiterung zu einer Gauß'schen Form der Spektrallinien führt. Darüber hinaus passt die Gauß'sche Form auch gut zu den aufgezeichneten Daten. Diese Anpassungen ergaben insgesamt 10 Anpassungsparameter für jede Interferenzordnung, von denen nur die Parameter b_1, b_2 und b_3 als Positionen der σ_{--} , π - und σ_+ -Linien verwendet wurden, während die Parameter c_1, c_2 und c_3 als Fehler dieser Positionen dienten, da die Wurzeln der Parameter b_1, b_2 und b_3 deutlich kleiner als ein Pixel sind. Es sei auch darauf hingewiesen, dass nur die ersten sechs Interferenzordnungen zur Auswertung herangezogen wurden, anstatt der in der Anleitung geforderten neun Ordnungen. Der Grund dafür ist, dass ab der siebten Ordnung die σ_+ -Linie (die rechte σ -Linie des Triplets) kein lokales Maximum mehr im Spektrum darstellte, sondern vielmehr einen Wendepunkt oder oft nicht mehr als einen kleinen Knick im Anstieg des Randes der zentralen Gauß'schen Glockenkurve. Dies machte die sinnvolle Anpassung einer Gauß-Kurve an diese σ_+ -Linien nahezu unmöglich.

Aus Gründen der besseren Lesbarkeit befinden sich die restlichen Abbildungen im Anhang, und nur die Abbildung für $I = 12\text{A}$ wird hier gezeigt.

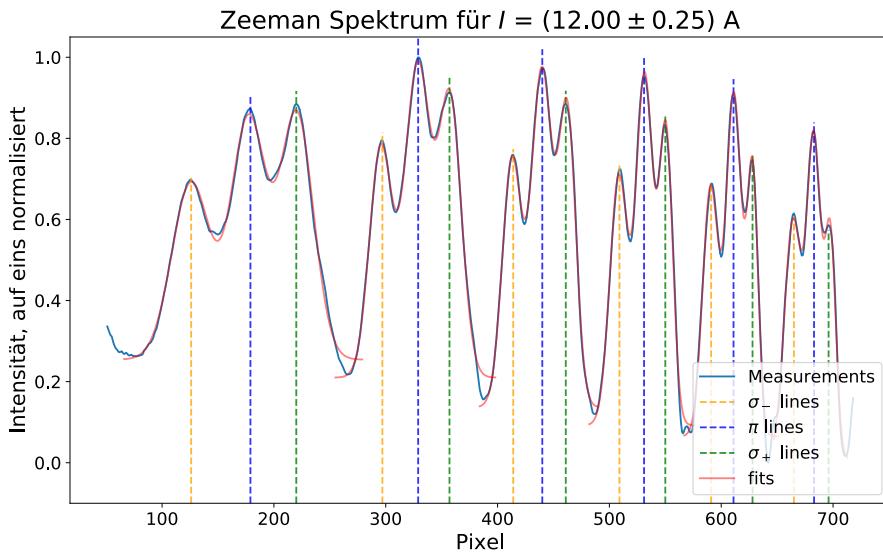


Abbildung 7: Cd-Spektrum

Die Positionen der π - und σ -Linien, die in den folgenden Tabellen dargestellt sind, wurden aus den Anpassungen ermittelt.

| Order | σ_- [px] | π [px] | σ_+ [px] |
|-------|-----------------|--------------|-----------------|
| 1 | 134 ± 18 | 176 ± 13 | 216 ± 18 |
| 2 | 301 ± 11 | 328 ± 10 | 354 ± 13 |
| 3 | 417 ± 9 | 440 ± 9 | 461 ± 9 |
| 4 | 513 ± 9 | 534 ± 7 | 551 ± 7 |
| 5 | 595 ± 8 | 614 ± 7 | 629 ± 5 |
| 6 | 669 ± 10 | 686 ± 5 | 699 ± 5 |

Tabelle 2: $I = (10.00 \pm 0.24)A$

| Order | σ_- [px] | π [px] | σ_+ [px] |
|-------|-----------------|--------------|-----------------|
| 1 | 126 ± 17 | 177 ± 16 | 221 ± 15 |
| 2 | 295 ± 10 | 329 ± 12 | 358 ± 10 |
| 3 | 413 ± 9 | 439 ± 9 | 462 ± 8 |
| 4 | 508 ± 8 | 532 ± 8 | 551 ± 6 |
| 5 | 590 ± 7 | 611 ± 7 | 628 ± 5 |
| 6 | 664 ± 8 | 683 ± 6 | 698 ± 5 |

Tabelle 3: $I = (11.00 \pm 0.24)A$

| Order | σ_- [px] | π [px] | σ_+ [px] |
|-------|-----------------|--------------|-----------------|
| 1 | 130 ± 17 | 177 ± 14 | 219 ± 17 |
| 2 | 298 ± 11 | 328 ± 11 | 356 ± 12 |
| 3 | 415 ± 9 | 440 ± 9 | 462 ± 8 |
| 4 | 510 ± 8 | 532 ± 7 | 551 ± 6 |
| 5 | 592 ± 7 | 612 ± 7 | 628 ± 5 |
| 6 | 666 ± 9 | 684 ± 6 | 698 ± 5 |

Tabelle 4: $I = (12.00 \pm 0.24)A$

| Order | σ_- [px] | π [px] | σ_+ [px] |
|-------|-----------------|--------------|-----------------|
| 1 | 124 ± 15 | 179 ± 17 | 223 ± 13 |
| 2 | 294 ± 10 | 329 ± 12 | 359 ± 10 |
| 3 | 412 ± 8 | 440 ± 10 | 464 ± 8 |
| 4 | 508 ± 8 | 532 ± 8 | 552 ± 6 |
| 5 | 590 ± 7 | 612 ± 7 | 629 ± 5 |
| 6 | 663 ± 7 | 683 ± 7 | 699 ± 5 |

Tabelle 5: $I = (13.00 \pm 0.24)A$

4.5 Bestimmung des Bohr-Magnetons

Um das Bohrsche Magneton zu berechnen, wurde im nächsten Schritt die Position der π -Linien in Pixeln gegen die Interferenzordnung aufgetragen und eine Polynomfunktion zweiten Grades angepasst. Ein Polynom zweiten Grades

$$k(x) = a \cdot x^2 + b \cdot x + c \quad (14)$$

ist der niedrigste Ordnungsgrad, der die Daten hinreichend beschreibt. Die Anpassung erfolgte unter der Bedingung, dass $\Delta k = k(\pi_{k+1}) - k(\pi_k) = 1$ gilt. Solche Plots und Anpassungen wurden für alle verwendeten Ströme durchgeführt, wobei auch die Positionen der σ -Linien eingetragen wurden, um ihnen ebenfalls eine Interferenzordnung zuzuweisen. Wiederum wird

hier nur der Graph für $I = 12$ A dargestellt. Die restlichen Diagramme befinden sich im Anhang. Die Parameter des angepassten Polynoms zweiten Grades sind in den jeweiligen Diagrammen angegeben und werden im nächsten Schritt verwendet, um die Wellenlängenverschiebung der σ -Linien zu bestimmen.

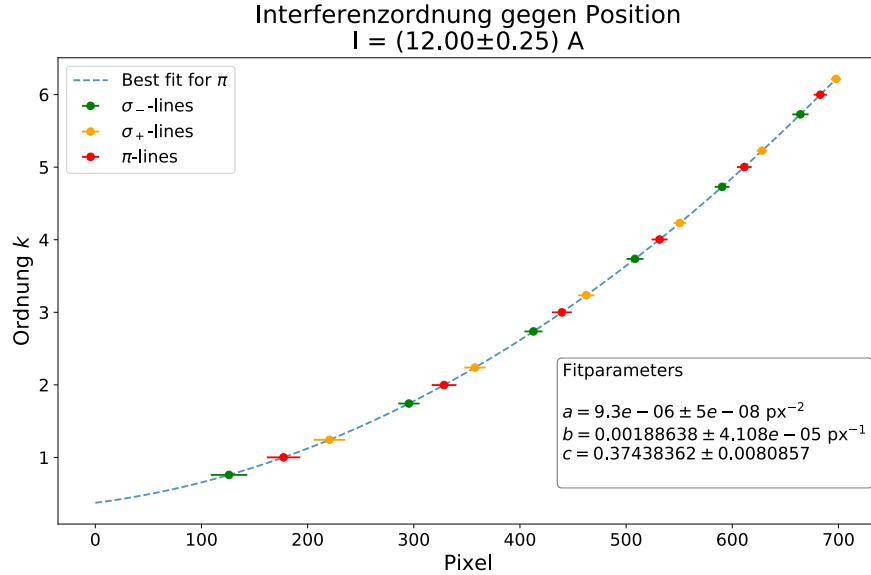


Abbildung 8: Quadratischer Fit: Ordnung gegen Position

Zur Berechnung der Wellenlängenverschiebung $\delta\lambda$ werden die beiden Näherungen

$$\frac{\delta\lambda}{\Delta\lambda} = \frac{\delta a}{\Delta a} = \frac{\delta k}{\Delta k} \quad (15)$$

und

$$\delta\lambda = \frac{\delta a}{\Delta a} \cdot \frac{\lambda^2}{2d \cdot \sqrt{n^2 - 1}} \quad (16)$$

verwendet. Durch Kombination dieser beiden Gleichungen ergibt sich:

$$\delta\lambda = \frac{\delta k}{\Delta k} \cdot \frac{\lambda^2}{2d \cdot \sqrt{n^2 - 1}} \quad (17)$$

Da die Anpassungen der Interferenzordnungen sicherstellen, dass $\Delta k = 1$ gilt, vereinfacht sich dies weiter zu:

$$\delta\lambda = \delta k \cdot \frac{\lambda^2}{2d \cdot \sqrt{n^2 - 1}} \quad (18)$$

Der Wert δk wird nun anhand der Anpassungen für die Interferenzordnungen und der zugehörigen Parameter wie folgt berechnet:

$$\delta k = \frac{|k(\sigma_+) - k(\sigma_-)|}{2} \quad (19)$$

Da für jeden Strom insgesamt 6 Interferenzordnungen analysiert wurden, ergeben sich 6 Werte für δk pro Strom. Diese werden gemittelt, und der mittlere Fehler des Mittelwerts wird als Fehler verwendet, um letztlich nur einen Wert für δk pro Strom zu erhalten.

Unter der Annahme $n = 1.4567$, $d = 4.04$ mm, sowie der Verwendung von Gleichung (24) und der im zweiten Teil bestimmten Wellenlänge der roten Cd-Linie, ergibt sich die folgende Tabelle 6 mit den Wellenlängenverschiebungen.

| I [A] | 10.00 ± 0.25 | 11.00 ± 0.25 | 12.00 ± 0.25 | 13.00 ± 0.25 |
|----------------------------------|---------------------|---------------------|---------------------|---------------------|
| δk | 0.2147 ± 0.0019 | 0.2326 ± 0.0017 | 0.2466 ± 0.0011 | 0.2586 ± 0.0009 |
| $\delta\lambda$ [10^{-11} m] | 1.040 ± 0.010 | 1.127 ± 0.008 | 1.194 ± 0.006 | 1.253 ± 0.005 |

Tabelle 6: Wellenlängenverschiebung für versch. Ströme

Der Fehler der Wellenlängenverschiebung $\Delta(\delta\lambda)$ wurde gemäß folgender Formel berechnet:

$$\Delta(\delta\lambda) = \sqrt{\left(\frac{\Delta(\delta k) \cdot \lambda^2}{2d \cdot \sqrt{n^2 - 1}}\right)^2 + \left(\frac{2 \cdot \delta k \cdot \lambda \cdot \Delta\lambda}{2d \cdot \sqrt{n^2 - 1}}\right)^2} \quad (20)$$

Die Wellenlängenverschiebungen $\delta\lambda$ können nun verwendet werden, um die Energieverschiebung ΔE und schließlich das Bohrsche Magneton μ_B zu berechnen. Für die Energieverschiebung gilt:

$$\begin{aligned} \Delta E &= hc \cdot \left(\frac{1}{\lambda} - \frac{1}{\lambda + \delta\lambda} \right), \\ \Delta(\Delta E) &= hc \cdot \sqrt{\left(\frac{\Delta\lambda}{(\lambda + \delta\lambda)^2} - \frac{\Delta\lambda}{\lambda^2} \right)^2 + \left(\frac{\Delta(\delta\lambda)}{(\lambda + \delta\lambda)^2} \right)^2}. \end{aligned} \quad (21)$$

Da $S = 0$ und somit $\Delta M_J = \Delta M_L = 1$ gilt, kann das Bohrsche Magneton wie folgt berechnet werden:

$$\begin{aligned} \mu_B &= \frac{\Delta E}{\Delta M_L \cdot B} = \frac{\Delta E}{B}, \\ \Delta\mu_B &= \mu_B \cdot \sqrt{\left(\frac{\Delta(\Delta E)}{\Delta E} \right)^2 + \left(\frac{\Delta B}{B} \right)^2}. \end{aligned} \quad (22)$$

Hierbei beschreibt $\Delta(\Delta E)$ den Fehler der Energieverschiebung und ΔB den Fehler des Magnetfeldes. Die oben dargestellten Gleichungen liefern somit die Basis für die präzise Berechnung des Bohrschen Magnetons und seines Fehlers unter Berücksichtigung aller relevanten Messunsicherheiten.

| I [A] | 10.00 ± 0.25 | 11.00 ± 0.25 | 12.00 ± 0.25 | 13.00 ± 0.25 |
|--|-------------------|-------------------|-------------------|-------------------|
| B [A] | 0.408 ± 0.013 | 0.448 ± 0.014 | 0.489 ± 0.014 | 0.529 ± 0.015 |
| ΔE [10^{-24} J] | 4.986 ± 0.011 | 5.403 ± 0.012 | 5.726 ± 0.012 | 6.006 ± 0.013 |
| μ_B [10^{-24} $\frac{\text{J}}{\text{T}}$] | 12.2 ± 0.4 | 12.0 ± 0.4 | 11.7 ± 0.3 | 11.3 ± 0.3 |

Tabelle 7: Energieverschiebung und Bohrmagneton für verschiedene Ströme

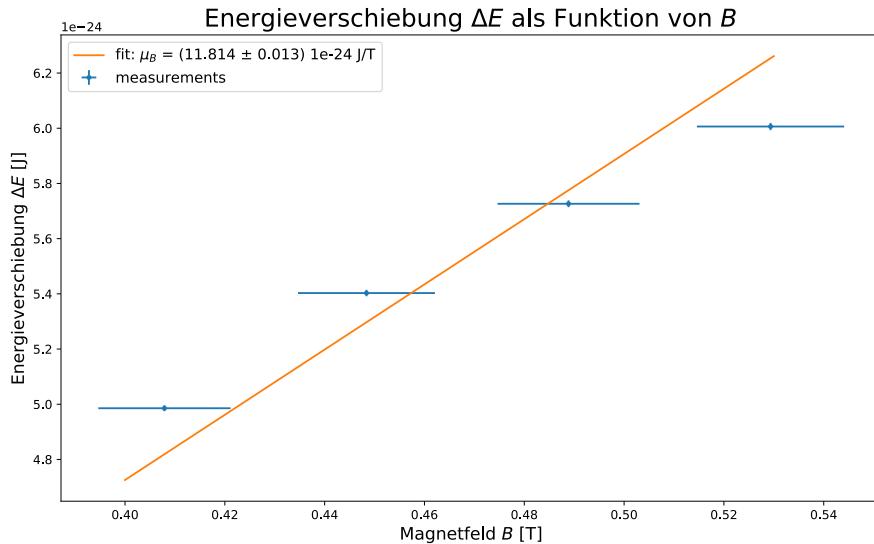
Abschließend kann aus diesen vier Werten für das Bohrsche Magneton der Mittelwert berechnet werden:

$$\bar{\mu}_B = (11.83 \pm 0.17) \cdot 10^{-24} \frac{\text{J}}{\text{T}} \quad (23)$$

Der mittlere Fehler des Mittelwertes wurde erneut als Fehler angegeben.

(Zusatz): Neben der Mittelung gibt es eine weitere Möglichkeit, das Bohrsche Magneton aus der Tabelle zu bestimmen. Hierfür wird die Energieverschiebung gegen das Magnetfeld aufgetragen, und die Datenpunkte werden mit einer Ursprungsgesetzen angepasst:

$$\Delta E(B) = \mu_B \cdot B \quad (24)$$

**Abbildung 9:** Darstellung der Energieverschiebung als Funktion von B

Die Steigung des Graphen korrespondiert nun mit dem Bohrmagneton. Bei dieser Methode erhalten wir einen Wert von:

$$\mu_B = (11.814 \pm 0.013) \cdot 10^{-24} \frac{\text{J}}{\text{T}} \quad (25)$$

Wir sehen direkt mit einem Blick auf die Fehler, dass die Werte in einem ein Sigma Bereich liegen und somit miteinander vereinbar sind.

5 Präzisionsspektroskopie

5.1 Feststellung der Wellenlänge der roten Cd-Linie

Im zweiten Teil des Experimentes wollen wir nun die Wellenlänge eruieren. Basierend auf der NIST-Datenbank wird erwartet, dass die rote Cd-Linie eine Wellenlänge von etwa 643,8 nm aufweist. Um diese Wellenlänge zu bestimmen, wird zusätzlich eine Ne-Lampe verwendet, die als Referenzspektrum für die Kalibrierung dient. Hierzu wird das Licht der Cd- und der Ne-Lampe entweder gemeinsam oder getrennt auf das Czerny-Turner-Spektrometer gerichtet. Beim Vergleich der Spektren erkennt man, dass die beiden hellen Linien auf der rechten Seite von Abbildung 10 zum Ne-Spektrum gehören, ebenso wie die blassen Linien auf der linken Seite. Die blassen Linie in der Mitte entspricht daher der roten Cd-Linie. Im Vergleich zur NIST-Datenbank wäre im Wellenlängenbereich der roten Cd-Linie eine vierte Ne-Linie zu erwarten gewesen. Aufgrund der begrenzten Bandbreite des Spektrometers konnte diese jedoch nicht mehr im Bild erfasst werden.

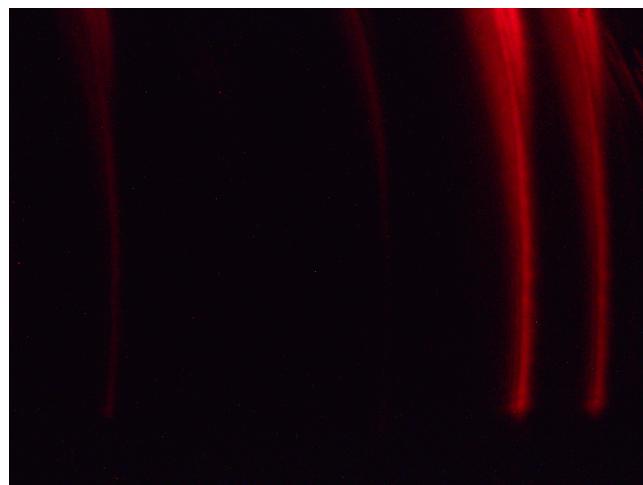


Abbildung 10: Darstellung des Cd und Ne Spektrums

Basierend auf den relativen Abständen zwischen den Ne-Spektrallinien und deren relativer Intensität zueinander treffen wir eine fundierte Annahme und ordnen die äußerste linke Linie der Wellenlänge

$$\lambda_1 = 650.653 \text{ nm}, \quad (26)$$

die mittlere und hellste Linie der Wellenlänge

$$\lambda_2 = 640.225 \text{ nm}, \quad (27)$$

und die äußerste rechte Linie der Wellenlänge

$$\lambda_3 = 638.299 \text{ nm} \quad (28)$$

zu. Mit diesem Wissen wird nun das kombinierte Ne – Cd-Spektrum aufgetragen, und eine

Gauß-Funktion

$$G(x) = a \cdot \exp \left(\frac{(x - b)^2}{2 \cdot c^2} \right) + d \quad (29)$$

wird an jede der vier erkennbaren Linien angepasst.

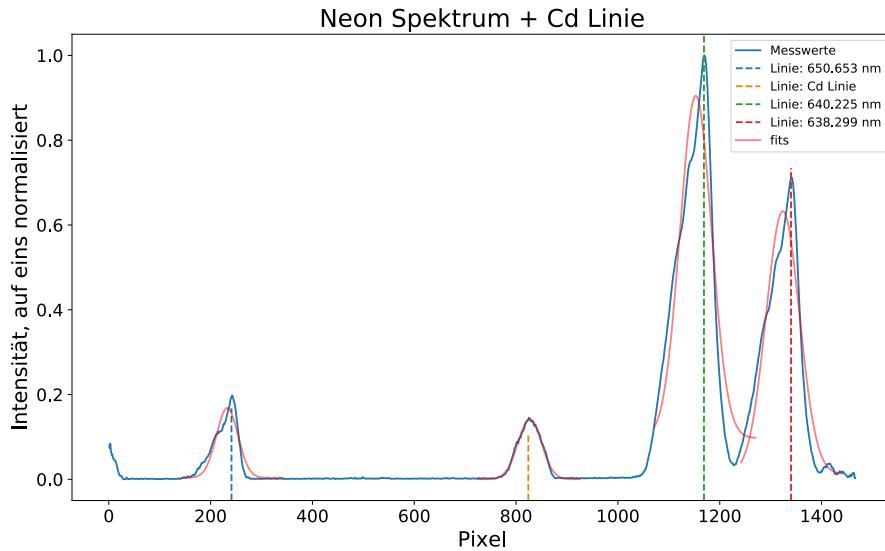


Abbildung 11: Darstellung der Spektrallinien in Abhängigkeit der Pixel

Um nun eine Aussage über die Wellenlänge machen zu können müssen wir die Einheit Pixel noch auf SI umrechnen. Dafür verwenden wir die Informationen über Pixelposition und Wellenlänge die wir aus dem Neonspektrum erhalten und konstruieren eine Kalibrierungslinie:

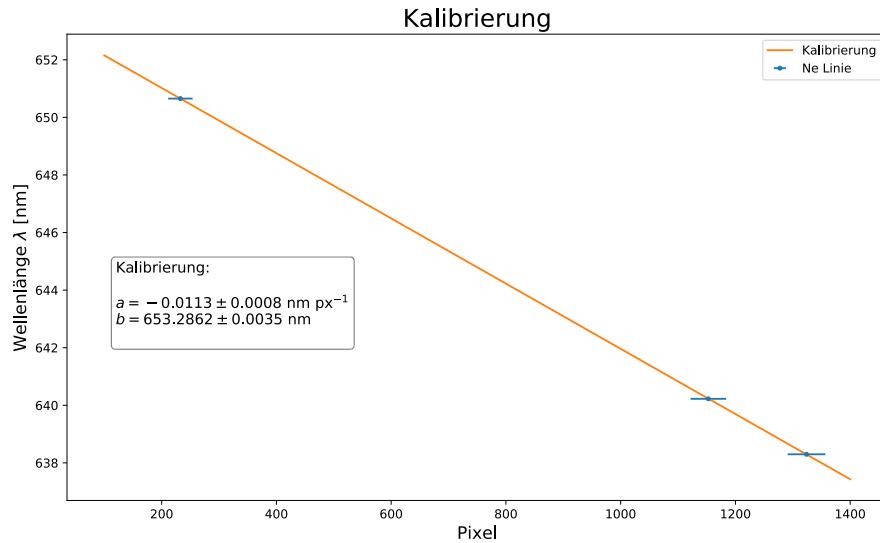


Abbildung 12: Kalibrierung

Betrachten wir nun die Position der Peaks in Abbildung 11 so erhalten wir:

| line | Ne: $\lambda_1 = 650.653 \text{ nm}$ | Cd | Ne: $\lambda_2 = 640.225 \text{ nm}$ | Ne: $\lambda_3 = 638.299 \text{ nm}$ |
|-------------------------|--------------------------------------|--------------|--------------------------------------|--------------------------------------|
| position $x[\text{px}]$ | 233 ± 21 | 827 ± 23 | 1153 ± 31 | 1324 ± 33 |

Tabelle 8: Position der Peaks

Rechnen wir nun mittels der in Abbildung 12 errechneten Steigung die Pixel in Wellenlänge um erhalten wir für Cd:

$$\lambda_{\text{Cd},1} = (643.9 \pm 0.7) \text{ nm} \quad (30)$$

Der entsprechende Plot ergibt sich zu:

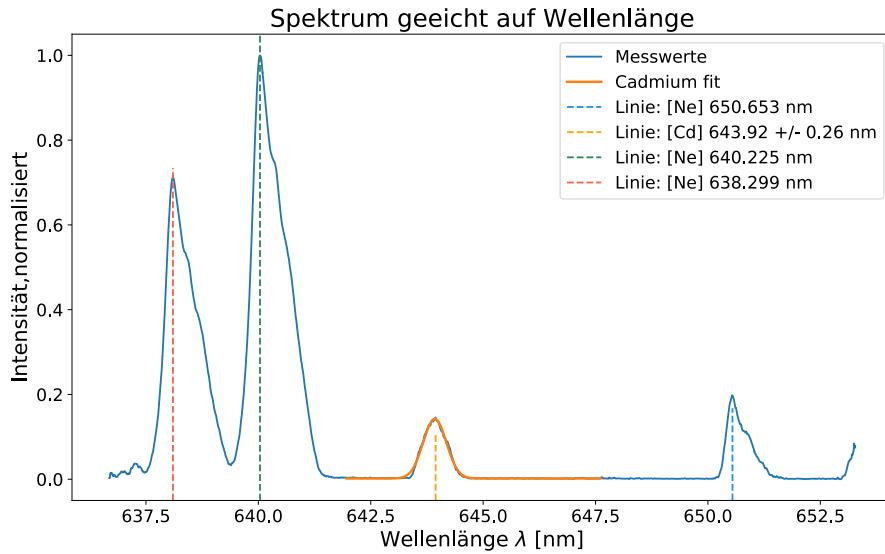


Abbildung 13: Cd+Ne Spektrum

6 Diskussion

Für den Vergleich von zwei Messwerten wird im Folgenden diese Formel genutzt:

$$\Delta = \frac{|\lambda_i - \lambda_k|}{\sqrt{(\Delta\lambda_i)^2 + (\Delta\lambda_k)^2}} \quad (31)$$

6.1 Hysterese

Wie bereits in Bezug auf Abbildung 5 festgestellt wurde, war auf den ersten Blick erkennbar, dass die Hysterese für dieses Experiment vernachlässigbar ist. Dies lässt sich insbesondere daran erkennen, dass die gemessenen Magnetfeldstärken für steigenden und fallenden Strom im Rahmen ihrer Fehler miteinander kompatibel sind. Diese Beobachtung kann jedoch noch einmal mit Hilfe der Fit-Parameter in Abbildung 5 unterstrichen werden. Zu diesem Zweck werden die σ -Abweichungen der Parameter gemäß Gleichung (31) berechnet. Die Ergebnisse sind in der folgenden Tabelle dargestellt.

| Parameter | Inc. | Dec. | σ deviation |
|--------------------------------|---------------------|---------------------|--------------------|
| $a \left[\frac{T}{A} \right]$ | 0.0405 ± 0.0008 | 0.0392 ± 0.0006 | 1.3σ |
| $b [A]$ | 0.003 ± 0.004 | 0.005 ± 0.004 | 0.4σ |

Tabelle 9: Fehlernalyse Hysterese

6.2 Bohr Magneton

Im nächsten Schritt werden die für das Bohrsche Magneton bestimmten Werte mit dem Literaturwert verglichen. Der Literaturwert kann aus der Anleitung entnommen werden: $\mu_B = 9.27400949 \cdot 10^{-24} \frac{J}{T}$.

Vergleichen wir zunächst den aus Tabelle 7 bestimmten Mittelwert $\bar{\mu}_B = (11.83 \pm 0.17) \cdot 10^{-24} \frac{J}{T}$ mit dem Literaturwert gemäß Gleichung (36), so ergibt sich eine signifikante Abweichung von 15.3σ . Da im nächsten Schritt festgestellt wird, dass die Wellenlänge der roten Cd-Linie korrekt bestimmt wurde, müssen die Parameter $\frac{\delta a}{\Delta a}$ und δk sowie die Magnetfeldstärke B als mögliche Fehlerquellen untersucht werden. Wenn wir uns nochmals Abbildung 7 ansehen, sehen wir deutlich, dass die dreifachen Gauß-Fits die verschiedenen Interferenzordnungen der Zeeman-Spektren angemessen beschreiben. Abbildung 8 zeigt ebenfalls klar, dass die Beziehung zwischen den Positionen der π - und σ -Linien durch den verwendeten Fit in Form eines zweiten Grades Polynom korrekt beschrieben wird. Da die Annäherung $\frac{\delta \lambda}{\Delta \lambda} = \frac{\delta a}{\Delta a} = \frac{\delta k}{\Delta k}$ ebenfalls als ausreichend genau angenommen werden kann, um nicht als Erklärung für die signifikante Abweichung zu dienen, bleibt nur noch die Normalisierung der Spektren als möglicher Einfluss auf den Parameter $\frac{\delta a}{\Delta a}$ oder δk . Allerdings sollte die Normalisierung nur die Intensitäten beeinflusst haben, nicht jedoch die Position der π - und σ -Linien, sodass der Parameter $\frac{\delta a}{\Delta a}$ oder δk nahezu als Fehlerquelle ausgeschlossen werden kann. Infolgedessen kann angenommen werden, dass die Bestimmung der Magnetfeldstärke die entscheidende Fehlerquelle darstellt. Diese wurde nicht nur mit einer veralteten Hall-Sonde gemessen, sondern die Hall-Sonde konnte auch nicht im Magneten fixiert werden, sodass nicht sichergestellt werden konnte, dass die Magnetfeldstärke tatsächlich an dem Punkt gemessen wurde, an dem sich später die Cd-Lampe befand. In jedem Fall ist jedoch sicher, dass ein systematischer Fehler in der Messung vorliegen muss, da die Werte für das Bohrsche Magneton aus Tabelle 7 innerhalb der Fehlergrenzen gemäß Gleichung (31) miteinander kompatibel sind.

In der Versuchsanleitung wurde eine weitere Methode dargelegt, welche auch zur Bestimmung des Magnetons verwendet werden kann. Trotz der Tatsache, dass dies keine lange Ausarbeitung ist habe ich es für sinnvoll erachtet die weitere Bestimmung durchzuführen, um einen weiteren Referenzwert zu erhalten den wir mit dem jetzt bestimmten Wert abgleichen können. Wenn wir nun auch den Wert $\mu_B = (11.814 \pm 0.013) \cdot 10^{-24} \frac{J}{T}$, der aus Abbildung 9 unter Verwendung eines Fits bestimmt wurde, mit dem Literaturwert vergleichen, ergibt sich eine deutlich größere Abweichung von 199σ . Neben den bereits diskutierten Fehlerquellen ist der Hauptgrund für diese extrem signifikante Abweichung der sehr kleine relative Fehler von ca. 0.1%. Dieser unterschätzt den tatsächlichen Fehler, da nur die Fehler der Energieverschiebung ΔE und nicht die Fehler der Magnetfeldstärke B in den kleinsten Quadranten des Fits in Abbildung 9 berücksichtigt werden konnten. Hinsichtlich Abbildung 9 ist jedoch sofort klar, dass der Fehler der Magnetfeldstärke B hier tatsächlich der entscheidende Fehler ist. Es ist an der Stelle jedoch anzumerken, dass die beiden berechneten Werte in ihren Fehlergrenzen miteinander vereinbar sind.

6.3 Wellenlänge der Cd-Linie

Die NIST-Datenbank zeigt, dass der Literaturwert der Wellenlänge der roten Cd-Linie $\lambda_{Cd} = 643.8469$ nm beträgt. Der durch uns bestimmte Wert $\lambda_{Cd,1} = (643.9 \pm 0.7)$ nm ist eindeutig mit dem Literaturwert innerhalb ihrer Fehlergrenzen kompatibel, mit Abweichungen von 0.1σ . Dies zeigt erneut deutlich, wie präzise hier die Spektroskopie durchgeführt werden konnte, da trotz der Tatsache, dass der Fehler der Umrechnung von Pixeln zu Wellenlängen nicht in die Berechnung von $\lambda_{Cd,2}$ einbezogen werden konnte, der Wert immer noch sehr gut mit dem Literaturwert übereinstimmt.

6.4 Urteil

Abschließend lässt sich sagen, dass der erste Teil – die Bestimmung des Bohrschen Magneton – nicht als Erfolg gewertet werden kann, da die berechneten Werte für das Bohrsche Magneton deutlich vom Literaturwert abweichen. Um zukünftig genauere Ergebnisse zu erzielen, wäre es sicherlich sinnvoll, die Magnetfeldstärke auf eine andere, genauere Weise zu bestimmen oder zumindest die Möglichkeit zu bieten, die Hall-Sonde in der richtigen Position zu fixieren. Der zweite Teil – die präzise Spektroskopie – war hingegen weitgehend ein Erfolg, da die rote Cd-Linie korrekt bestimmt werden konnte.

Insgesamt hat dieses Experiment trotz einiger kleiner technischer Probleme deutlich gezeigt, wie der Zeeman-Effekt funktioniert.

7 Quellen

1. Instructions F44 Zeeman effect [Online].
2. Wikipedia: Zeeman-Effekt.
3. PEP-3 Skript - Pernice
4. NIST, National Institute for Standards and Technology, spectral line database [Online]

8 Appendix

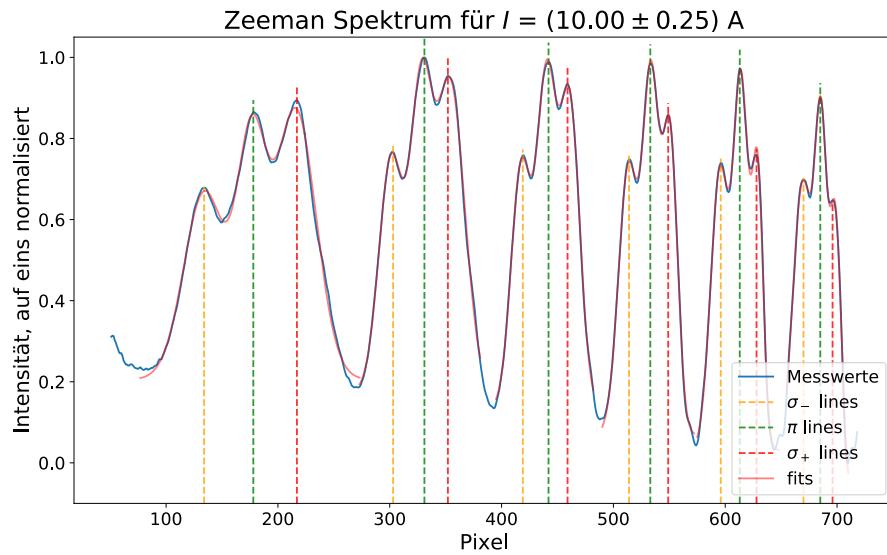


Abbildung 14: Zeemann-Spektrum für: $I = (10.00 \pm 0.25) \text{ A}$

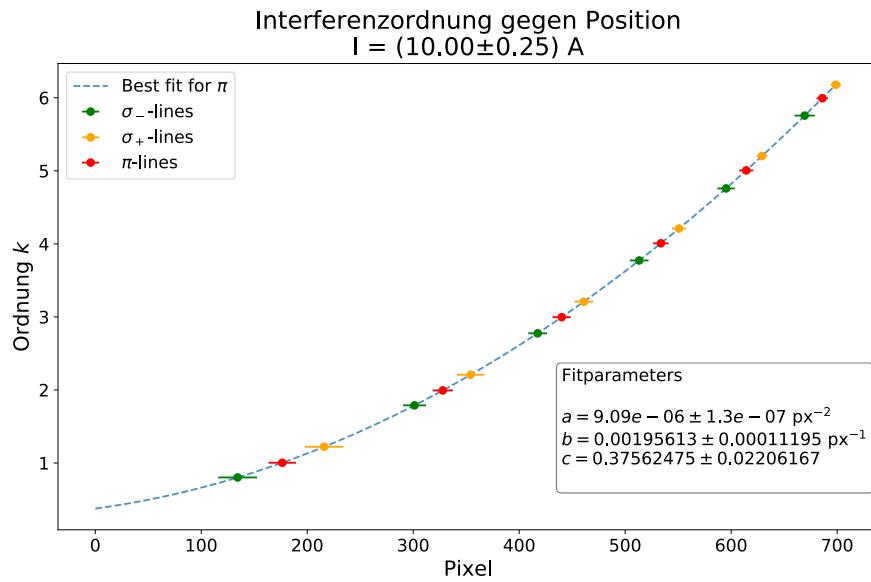


Abbildung 15: Interferenzordnung für: $I = (10.00 \pm 0.25) \text{ A}$

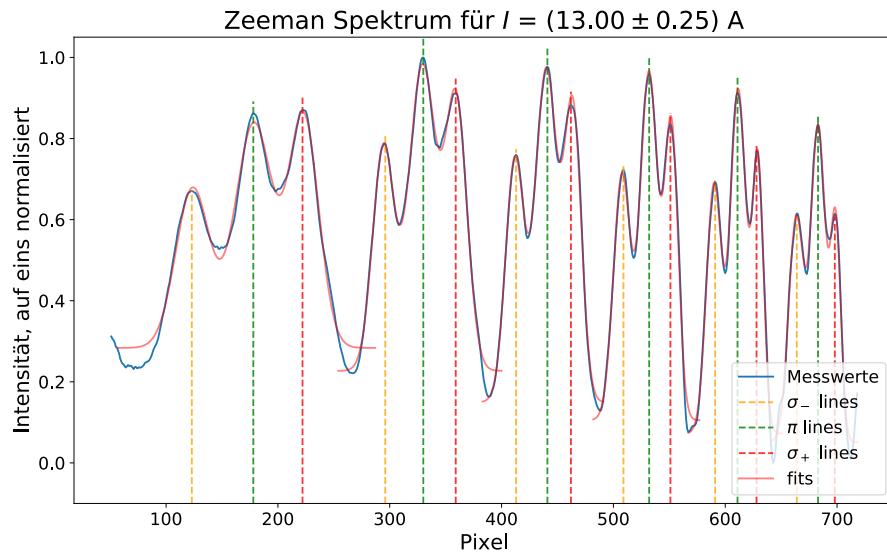


Abbildung 18: Zeemann-Spektrum für: $I = (13.00 \pm 0.25) A$

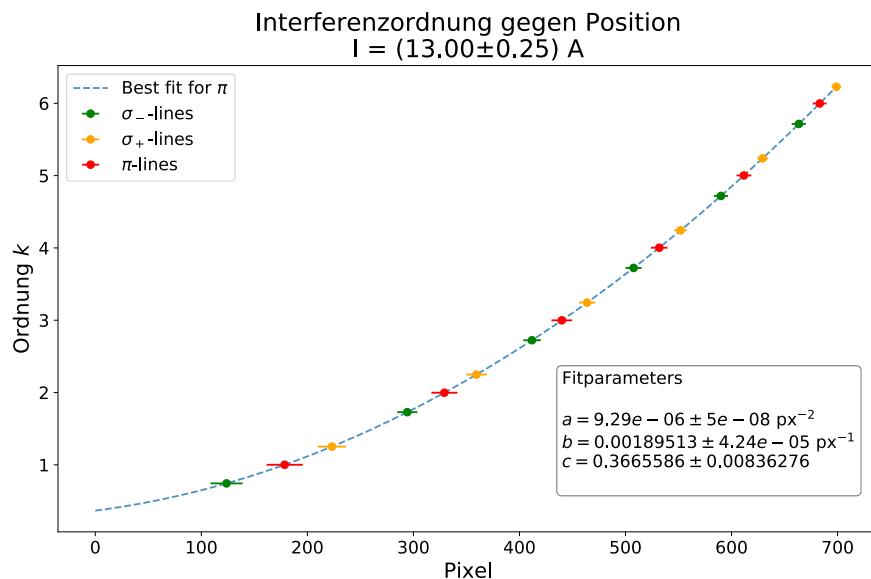


Abbildung 19: Interferenzordnung für: $I = (13.00 \pm 0.25) A$

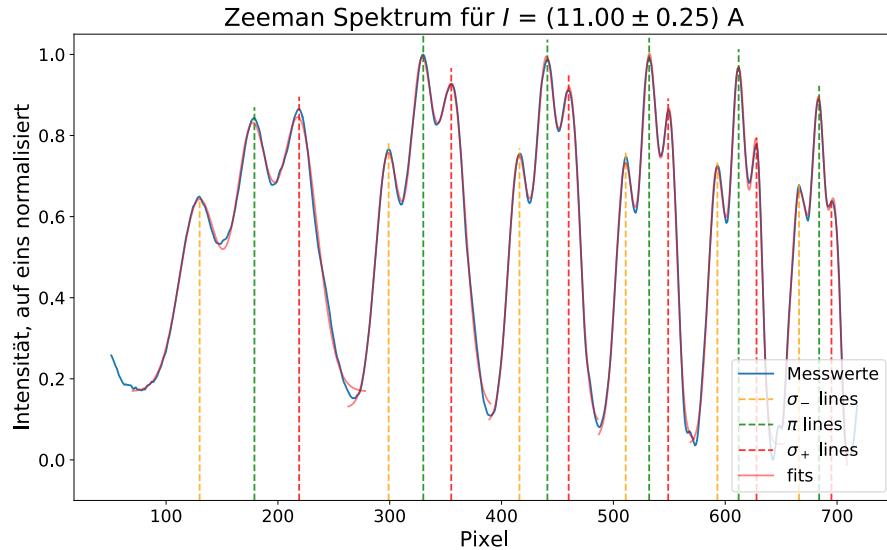


Abbildung 16: Zeemann-Spektrum für: $I = (11.00 \pm 0.25) \text{ A}$

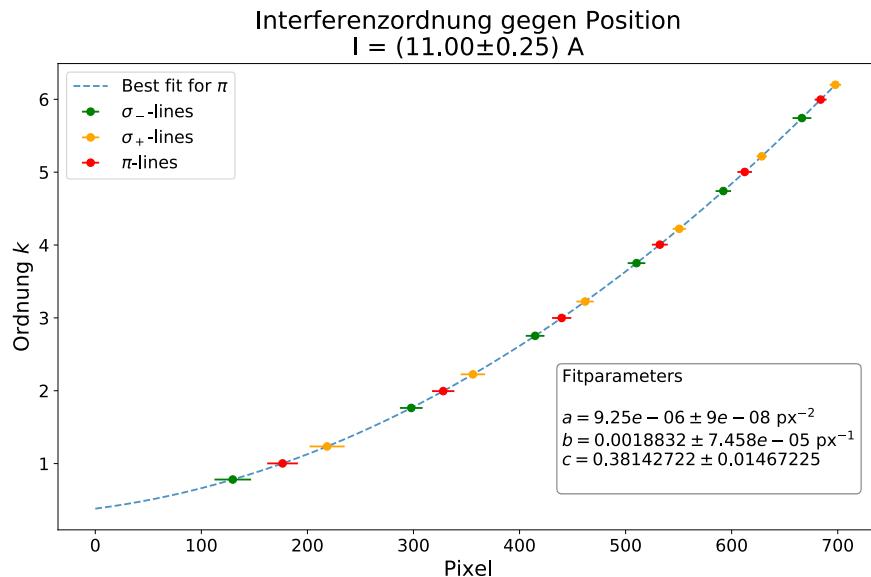


Abbildung 17: Interferenzordnung für: $I = (11.00 \pm 0.25) \text{ A}$

```
In [2]: import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
%matplotlib inline
#%matplotlib qt5
import numpy as np
from numpy import exp, sqrt, log, pi
from scipy.optimize import curve_fit
from scipy.stats import chi2
from glob import glob
from astropy.io import fits
from astropy.visualization import simple_norm
import matplotlib.patches as mpatches
import statistics as st
import pandas as pd
from scipy.signal import find_peaks
```

```
In [3]: # box design
props = dict(boxstyle = 'round', facecolor = 'white', alpha = 0.5)

# make a box with parameters in plot
def textstrer(title, params, err_params, units, labels, a, b, r =
2,
            fonts = 14):
    textstr = np.array([title + '\n'])

    for i in range(len(params)):
        textstr = np.append(textstr,
                            '$\{} = \{} \pm \{} \cdot \{}$' .
                            format(labels[i],
                                   round(params[i], r),
                                   round(err_params[i], r),
                                   units[i]
                            )) # Create a textstring w/ all p
    parameters
    plt.text(a, b, ''.join(map(lambda x: x[:] + '\n', textstr))
             , va='top', fontsize = fonts, bbox = props)
```

Hysterese

```
In [4]: # Get the mean from every 3-point measurement
def meaner(B):
    B_mean = np.array([])
    B_std = np.array([])

    for array in B:
        mean = np.mean(array)
        std = np.std(array)
        B_mean = np.append(B_mean, mean)
        B_std = np.append(B_std, std)

    return B_mean, B_std
```

```
In [5]: I_inc = np.array([0,2,4,6,8,10,12,13])
I_dec = np.array([12,10,8,6,4,2,0])
err_I=0.25
B_inc=np.array([[1,1,2], [89,90,92],[159,171,174],[250,254,252],[33
3,332,334],[380,396,394],[452,456,458],[490,494,500]])*1e-3# T
B_dec = np.array([[452,456,458],[395,398,399],[324,321,322],[245,24
3,251],[166,167,169],[87,90,86],[1,2,3]])*1e-3 # T
#meaner
B_mean_inc, B_std_inc = meaner(B_inc)
B_mean_dec, B_std_dec = meaner(B_dec)
```

```
In [6]: # Fit a line
def line(x, a, b):
    return a * x + b

x_inc = np.linspace(I_inc[0], I_inc[-1], 200)
x_dec = np.linspace(I_dec[0], I_dec[-1], 200)
popt_inc, pcov_inc = curve_fit(line, I_inc, B_mean_inc ,sigma = B_s
td_inc)
popt_dec, pcov_dec = curve_fit(line, I_dec, B_mean_dec, sigma = B_s
td_dec)

params_inc, err_params_inc = popt_inc, sqrt(np.diag(pcov_inc))
params_dec, err_params_dec = popt_dec, sqrt(np.diag(pcov_dec))

params = np.append(params_inc, params_dec)
err_params = np.append(err_params_inc, err_params_dec)
```

In [7]: # Plot

```

plt.figure(figsize = (12, 7))
plt.title('Magnetfeld in Abhangigkeit der Stromstrke',
           size = 20)
plt.errorbar(I_inc, B_mean_inc, xerr = err_I, yerr = B_std_inc,
              fmt = 'o', label = 'Steigend')
plt.errorbar(I_dec, B_mean_dec, xerr = err_I, yerr = B_std_dec,
              fmt = 'o', label = 'Fallend')

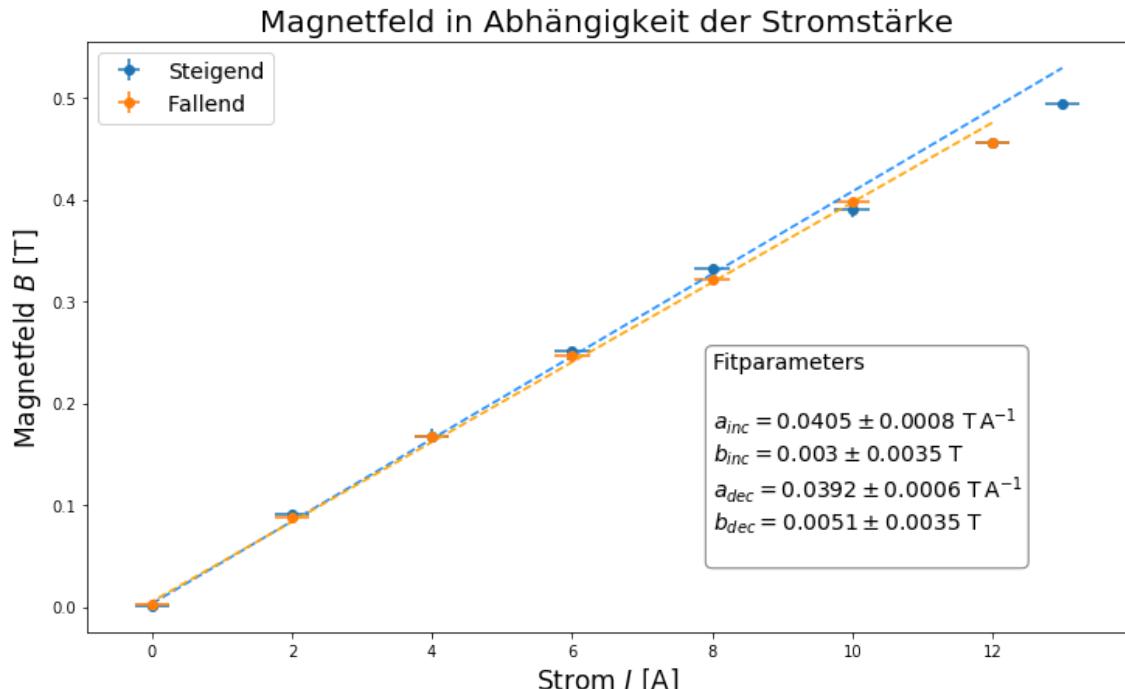
plt.plot(x_inc, line(x_inc, *popt_inc),
          ls = '--', color = 'dodgerblue')
plt.plot(x_dec, line(x_dec, *popt_dec),
          ls = '--', color = 'orange')

plt.xlabel('Strom $I$ [A]', size = 17)
plt.ylabel('Magnetfeld $B$ [T]', size = 17)

plt.legend(loc = 'best', fontsize = 14)

labels = ['a_{inc}', 'b_{inc}', 'a_{dec}', 'b_{dec}']
units = ['\mathrm{T} \backslash, \mathrm{A}^{-1}', '\mathrm{T} ', '\mathrm{T} \backslash, \mathrm{A}^{-1}', '\mathrm{T} ']
textstrer('Fitparameters', params, err_params, units, labels, 8, 0.
25, 4)
plt.savefig('diagrams/B_I_inc_dec.pdf')

```



Zeeman effect

```
In [8]: # some Gaussians for later
def gaussian(x, a, b, c, d):
    return a * exp(- (x - b) ** 2 / (2 * c ** 2)) + d

def double_gaussian(x, a1, b1, c1, a2, b2, c2, d):
    return (gaussian(x, a1, b1, c1, d = 0) +
            gaussian(x, a2, b2, c2, d))

def triple_gaussian(x, a1, a2, a3, b1, b2, b3, c1, c2, c3, d):
    return (gaussian(x, a1, b1, c1, d = 0)
            + gaussian(x, a2, b2, c2, d = 0)
            + gaussian(x, a3, b3, c3, d))
```

```
In [9]: from functools import wraps # This convenience func preserves name
        and docstring
from scipy.signal import argrelextrema # tool to find local extrema

class OmniTool:
    pass

def add_method(cls):
    def decorator(func):
        @wraps(func)
        def wrapper(self, *args, **kwargs):
            return func(*args, **kwargs)

        setattr(cls, func.__name__, wrapper)
        # Note we are not binding func, but wrapper which accepts
        # self but does exactly the same as func
        return func # returning func means func can still be used normally
    return decorator
```

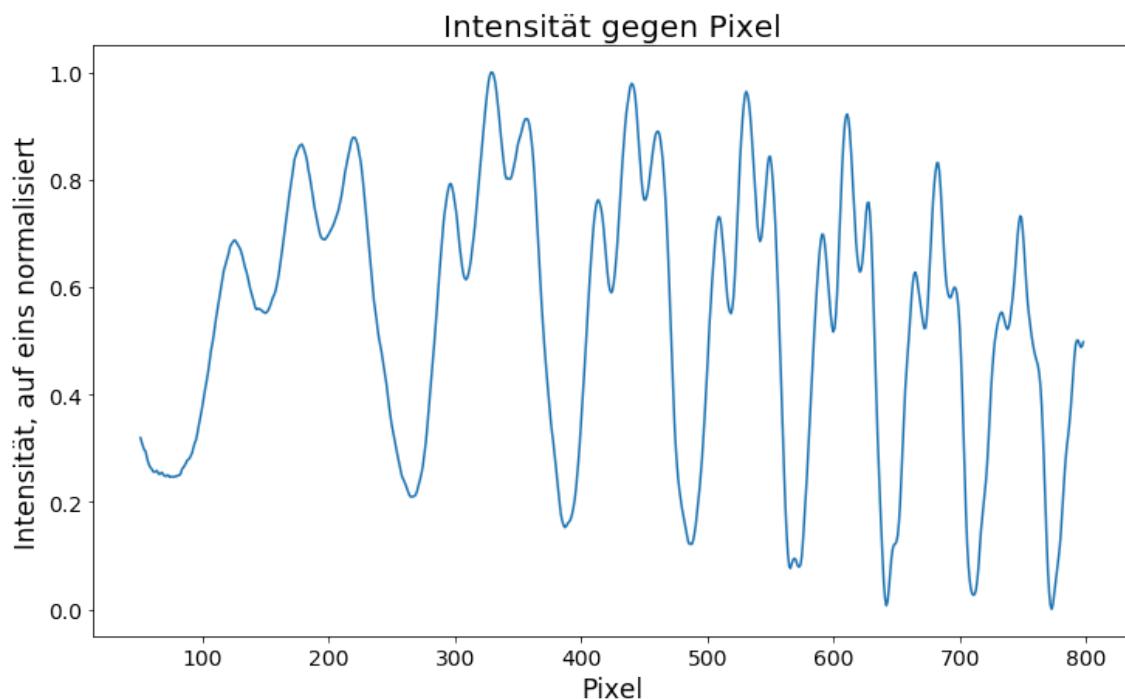
```
In [10]: class OmniTool:  
    pass  
  
    # We add the method tolino to read the data from the txt  
    @add_method(OmniTool)  
    def tolino(file,upper, lower):  
        data_i = pd.read_csv(file, delimiter ='\t')  
        pos_i = data_i['x'].values  
        int_i = data_i['intensity'].values  
        int_i = (int_i - min(int_i))*1e-7# shifting the negative intens  
        ities up  
        #int_i = int_i/(max(int_i))# normalizing to 1  
        mask = ((pos_i < upper) & (pos_i>lower))  
        pos_i = pos_i[mask]  
        int_i = int_i[mask]  
        # for local minima  
        minima = argrelextrema(int_i, np.less)  
        popt, pcov = curve_fit(line, pos_i[minima], int_i[minima])  
        int_i = int_i - line(pos_i,*popt)# solving the problem of incre  
        asing intensity, it may be that a slight increase remains, but sinc  
        e we are only interested in the position that isnt important  
        int_i = int_i+abs(min(int_i))  
        int_i = int_i/(max(int_i))# normalising to the maximum  
  
        return pos_i[:-1], int_i[:-1]  
  
file_12 = 'misc/12A.txt'  
data_12 = OmniTool().tolino(file_12,800,50)  
  
# Need the () after OmniTool to initialize the class itself or sth
```

```
In [11]: @add_method(OmniTool)
def plotter(file,upper,lower):
    data = OmniTool().tolino(file,upper, lower)

    x = np.linspace(data[0][0], data[0][-1], 2000)

    fig = plt.figure(figsize = (12, 7))
    plt.plot(data[0], data[1], label = 'Messwerte')
    plt.title('Title', size = 20)
    plt.xlabel('Pixel', size = 17)
    plt.xticks(size = 14)
    plt.ylabel('Intensität, auf eins normalisiert', size = 17)
    plt.yticks(size = 14)

    return fig
fig = OmniTool().plotter(file_12, 800, 50)
# We can still manually alter this graph if we want to deviate from
# the standard parameters:
plt.title('Intensität gegen Pixel', size = 20)
plt.show()
```



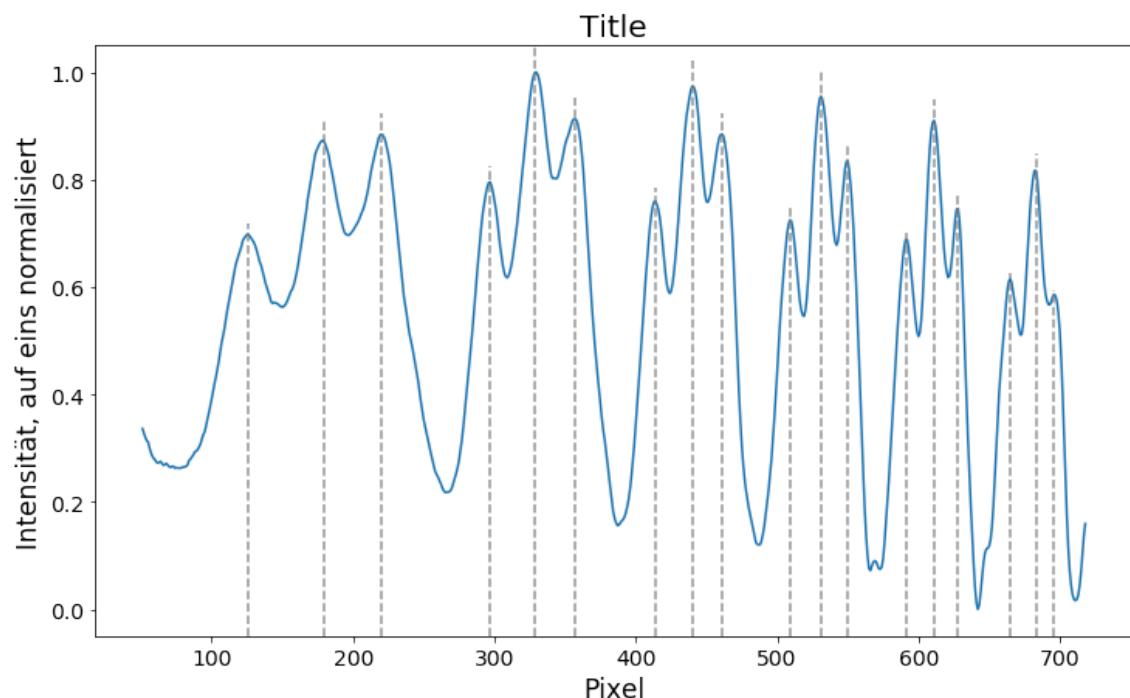
```
In [12]: @add_method(OmniTool)
def peaker(file,upper, lower, ds = 8, prom = 0.008, ht = 0.4,
           dis_b = None, dis_a = None):
    data = OmniTool().tolino(file, upper, lower)

    peaks, _ = find_peaks(data[1], distance = ds,
                          prominence = prom, height = ht)
    #peaks = peaks[dis_a:dis_b] # Cut the last one out, only two peaks visible
    peaks_pos = data[0][peaks]
    heights = data[1][peaks]

    return peaks, peaks_pos, heights
peaks, peaks_pos, heights = OmniTool().peaker(file_12, 720, 50)
```

```
In [13]: @add_method(OmniTool)
def peak_marker(peaks_pos, heights, col = 'gray', lab = None, alph = 0.8):
    for i in range(len(peaks_pos)):
        if i==0:
            plt.axvline(peaks_pos[i], ls = '--', alpha = alph,
                        color = col, label = lab,
                        ymax = heights[i])
        else:
            plt.axvline(peaks_pos[i], ls = '--', alpha = alph,
                        color = col,
                        ymax = heights[i])

fig = OmniTool().plotter(file_12, 720, 50)
OmniTool().peak_marker(peaks_pos, heights)
```



```
In [14]: @add_method(OmniTool)
def grouper(file, upper, lower, peaks, heights):
    data = OmniTool().tolino(file, upper, lower)
    #    peaks, heights = OmniTool().peaker(file)

    a = int(len(peaks) / 3)
    ps = np.array([[0, 0, 0]]) # need this to append to the right dimension
    maxs = np.array([[0, 0, 0]])

    for i in range(a): # Find the position and the height of the peak
        pl_i, maxl_i = peaks[0::3][i], data[1][peaks[0::3][i]]
        pc_i, maxc_i = peaks[1::3][i], data[1][peaks[1::3][i]]
        pr_i, maxr_i = peaks[2::3][i], data[1][peaks[2::3][i]]

        p_i, max_i = [pl_i, pc_i, pr_i], [maxl_i, maxc_i, maxr_i]
        ps, maxs = (np.append(ps, [p_i], axis = 0),
                     np.append(maxs, [max_i], axis = 0))

    ps = ps[1:]
    maxs = maxs[1:]

    return ps, maxs

groups = OmniTool().grouper(file_12, 720, 50, peaks, heights)
```

```
In [15]: @add_method(OmniTool)
def gausser(file, upper, lower, group,
            col = 'C1',
            alph = 0.5,
            dim = 10,
            symcut = 50):
    # col, alph parameters for the plot
    # dim = num of parameters of triple gaussian (10)
    # symcut for plotting interval for every order
    data = OmniTool().tolino(file, upper, lower)

    #c = [5, 10, 5] # guessed parameters for width
    d = 0 # and offset

    # need for right dimensions when appending
    popts = np.empty([1, dim])
    pcovs = np.empty([1, dim])

    for i in range(len(group[0])):
        b = group[0][i] # groups outputs positions, intensities
        a = group[1][i]
        c = np.array([1,1,1])*int((b[2]-b[1])/2) # guessed parameters for width

        # closeup on one order
        data_ab = (data[0][b[0]-3*c[0]:b[2]+3*c[0]], 
                   data[1][b[0]-3*c[0]:b[2]+3*c[0]])

        # estimated parameters based on peak finding algorithm
        p0_i = np.append(a, data[0][b])
        p0_i = np.append(p0_i, c)
        p0_i = np.append(p0_i, d)

        popt_i, pcov_i = curve_fit(triple_gaussian, data_ab[0],
                                    data_ab[1], p0 = p0_i)

        popts = np.append(popts, [popt_i], axis = 0)
        pcovs = np.append(pcovs, [sqrt(np.diag(pcov_i))], axis = 0)

        x_ab = np.linspace(data_ab[0][0], data_ab[0][-1], 200)

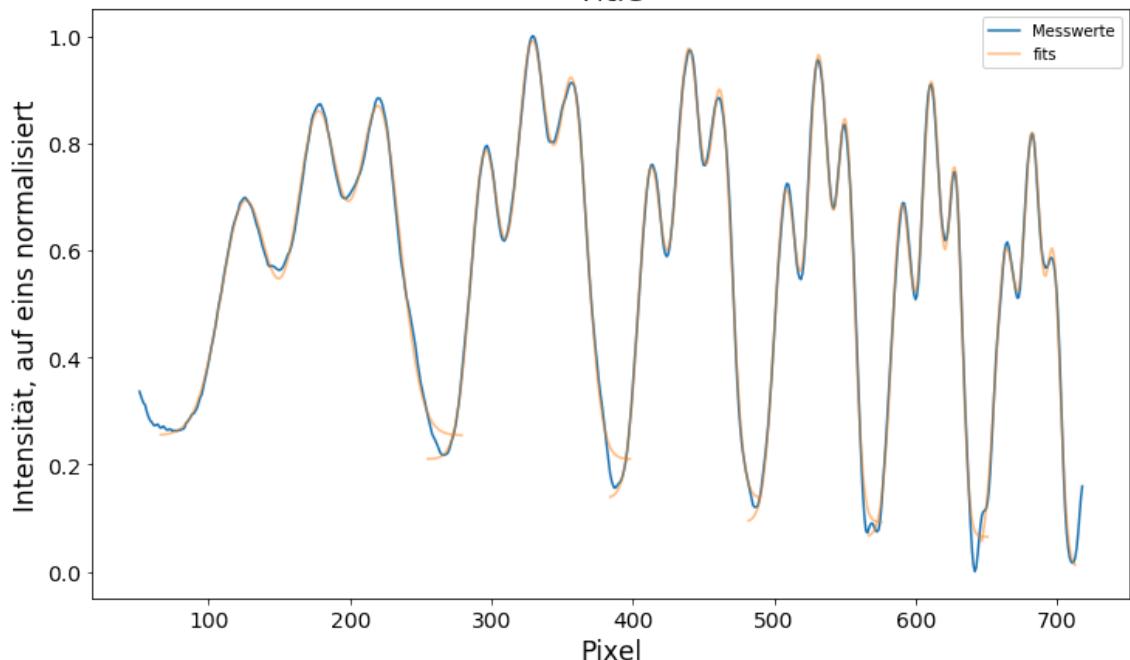
        if i==0:
            plt.plot(x_ab, triple_gaussian(x_ab, *popt_i),
                      color = col, alpha = alph, label = 'fits')
        else:
            plt.plot(x_ab, triple_gaussian(x_ab, *popt_i),
                      color = col, alpha = alph)

    popts = popts[1:] # eliminate that empty
    pcovs = pcovs[1:]
    plt.legend()

    return popts, pcovs

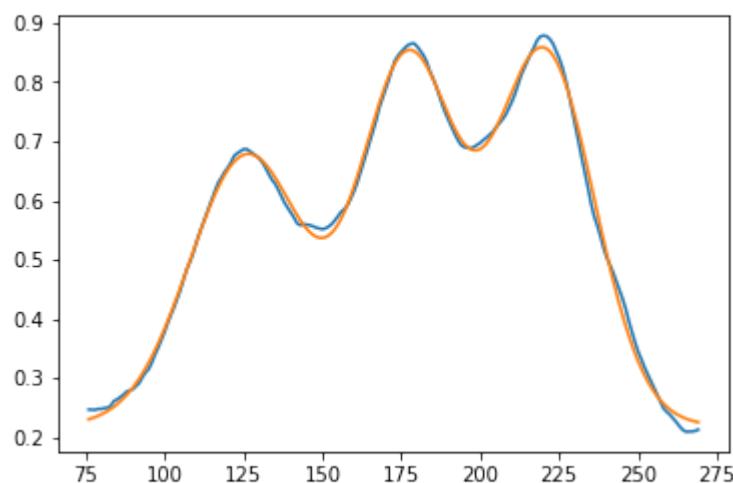
fig = OmniTool().plotter(file_12, 720, 50)
fits = OmniTool().gausser(file_12, 720, 50, groups)
```

Title



```
In [16]: a = groups[1][0]
b = groups[0][0]
data_ab = (data_12[0][b[0]-50:b[2]+50],
            data_12[1][b[0]-50:b[2]+50])
plt.plot(data_ab[0],data_ab[1])
p_0 = np.append(a,(data_12[0][b]))
p_0 = np.append(p_0,[5,5,5,0])
popt, pcov = curve_fit(triple_gaussian, data_ab[0],data_ab[1],p0= p_0)
plt.plot(data_ab[0],triple_gaussian(data_ab[0],*popt) )
```

Out[16]: <matplotlib.lines.Line2D at 0x7f274a5c2eb8>



```
In [17]: # 1. Plot the data
file = file_12
data = OmniTool().tolino(file,720,50)

fig = OmniTool().plotter(file,720,50)
plt.title('Zeeman Spektrum für  $I = (12.00 \pm 0.25) \text{ A}$ ', size = 20)
plt.ylim(-0.1, 1.05)

# 2. Find the peaks
peaks_12, peaks_pos_12, heights_12 = OmniTool().peaker(file, 720, 50)

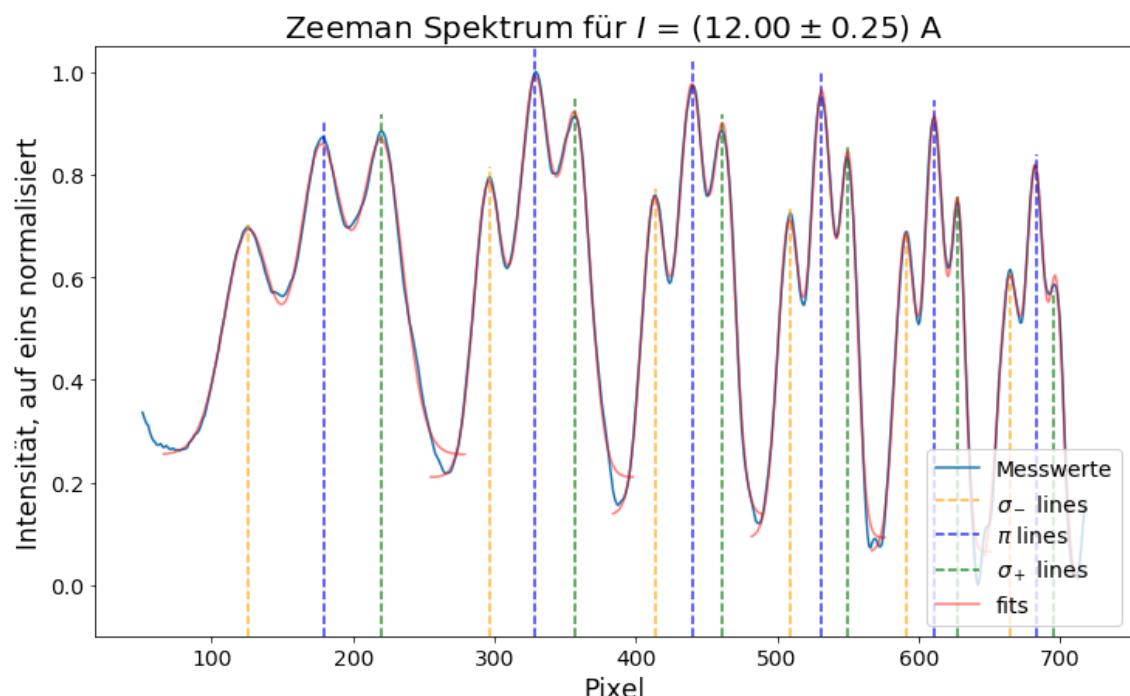
peaks_sig_l, heights_sig_l = peaks_pos_12[0::3], heights_12[0::3]
peaks_pi, heights_pi = peaks_pos_12[1::3], heights_12[1::3]
peaks_sig_r, heights_sig_r = peaks_pos_12[2::3], heights_12[2::3]

OmniTool().peak_marker(peaks_sig_l, heights_sig_l,
                       col = 'orange', lab = ' $\sigma_-$  lines')
OmniTool().peak_marker(peaks_pi, heights_pi,
                       col = 'blue', lab = ' $\pi$  lines')
OmniTool().peak_marker(peaks_sig_r, heights_sig_r,
                       col = 'green', lab = ' $\sigma_+$  lines')

# 3. Group triplets
groups = OmniTool().grouper(file,720,50, peaks_12, heights)

# 4. Fit a triple Gauss on each triplet
fits_12 = gausser(file,720,50, groups, col = 'red')

# 5. Decorate the plot a little more if you like
plt.legend(loc = 'lower right', fontsize = 14)
# Little window
# Fit for order a
# HERE WOULD HAVE BEEN BIN[4]
plt.savefig('diagrams/zeemanspectrum12.pdf')
```



```
In [18]: @add_method(OmniTool)
def announcer_fits(fits, title = 'Parameters', r = [2, 0],
                    labels = ['sigma_l', 'pi', 'sigma_r']):
    print(title + '\n')

    for i in range(len(fits[0])):
        as_i, err_as_i = fits[0][i][0:3], fits[1][i][0:3]
        bs_i, err_bs_i = fits[0][i][3:6], fits[1][i][3:6]
        cs_i, err_cs_i = fits[0][i][6:9], fits[1][i][6:9]

        print('Order: {}'.format(i+1))
        print('  {}: a = {} +/- {}  b = {} +/- {} [cov] +/- {} [st'
              'at] [px]'.
              format(labels[0],
                     round(as_i[0], r[0]),
                     round(err_as_i[0], r[0]),
                     round(bs_i[0], r[1]),
                     round(err_bs_i[0], r[1]),
                     round(cs_i[0], r[1]),
                     ))
        print('  {}: a = {} +/- {}  b = {} +/- {} [cov] +/- {} [st'
              'at] [px]'.
              format(labels[1],
                     round(as_i[1], r[0]),
                     round(err_as_i[1], r[0]),
                     round(bs_i[1], r[1]),
                     round(err_bs_i[1], r[1]),
                     round(cs_i[1], r[1]),
                     ))
        print('  {}: a = {} +/- {}  b = {} +/- {} [cov] +/- {} [st'
              'at] [px]'.
              format(labels[2],
                     round(as_i[2], r[0]),
                     round(err_as_i[2], r[0]),
                     round(bs_i[2], r[1]),
                     round(err_bs_i[2], r[1]),
                     round(cs_i[2], r[1]),
                     ))
```

```
In [19]: announcer_fits(fits_12)
```

Parameters

```
Order: 1
    sigma_l: a = 0.44 +/- 0.0    b = 126.0 +/- 0.0 [cov] +/- 17.0 [stat]
    pi: a = 0.59 +/- 0.0    b = 177.0 +/- 0.0 [cov] +/- 16.0 [stat]
    sigma_r: a = 0.6 +/- 0.01    b = 221.0 +/- 0.0 [cov] +/- 15.0 [stat]
    Order: 2
    sigma_l: a = 0.56 +/- 0.01    b = 295.0 +/- 0.0 [cov] +/- 10.0 [stat]
    pi: a = 0.77 +/- 0.01    b = 329.0 +/- 0.0 [cov] +/- 12.0 [stat]
    sigma_r: a = 0.67 +/- 0.01    b = 358.0 +/- 0.0 [cov] +/- 10.0 [stat]
    Order: 3
    sigma_l: a = 0.6 +/- 0.01    b = 413.0 +/- 0.0 [cov] +/- 9.0 [stat]
    pi: a = 0.83 +/- 0.0    b = 439.0 +/- 0.0 [cov] +/- 9.0 [stat] [px]
    sigma_r: a = 0.72 +/- 0.01    b = 462.0 +/- 0.0 [cov] +/- 8.0 [stat]
    Order: 4
    sigma_l: a = 0.61 +/- 0.01    b = 508.0 +/- 0.0 [cov] +/- 8.0 [stat]
    pi: a = 0.86 +/- 0.01    b = 532.0 +/- 0.0 [cov] +/- 8.0 [stat]
    sigma_r: a = 0.71 +/- 0.01    b = 551.0 +/- 0.0 [cov] +/- 6.0 [stat]
    Order: 5
    sigma_l: a = 0.61 +/- 0.01    b = 590.0 +/- 0.0 [cov] +/- 7.0 [stat]
    pi: a = 0.84 +/- 0.01    b = 611.0 +/- 0.0 [cov] +/- 7.0 [stat]
    sigma_r: a = 0.64 +/- 0.02    b = 628.0 +/- 0.0 [cov] +/- 5.0 [stat]
    Order: 6
    sigma_l: a = 0.59 +/- 0.01    b = 664.0 +/- 0.0 [cov] +/- 8.0 [stat]
    pi: a = 0.78 +/- 0.01    b = 683.0 +/- 0.0 [cov] +/- 6.0 [stat]
    sigma_r: a = 0.54 +/- 0.02    b = 698.0 +/- 0.0 [cov] +/- 5.0 [stat]
```

```
In [20]: @add_method(OmniTool)
def pi_sigl_sigr(fits):
    for i in range(len(fits[0])):
        if i ==0:
            pi = np.array(fits[0][i][4])
            pi_err = np.array(fits[0][i][7])
            sig_l = np.array(fits[0][i][3])
            sig_l_err =np.array(fits[0][i][6])
            sig_r = np.array(fits[0][i][5])
            sig_r_err = np.array(fits[0][i][8])
        else:
            pi = np.append(pi, fits[0][i][4])
            pi_err = np.append(pi_err, fits[0][i][7])
            sig_l = np.append(sig_l, fits[0][i][3])
            sig_l_err = np.append(sig_l_err, fits[0][i][6])
            sig_r = np.append(sig_r, fits[0][i][5])
            sig_r_err = np.append(sig_r_err, fits[0][i][8])
    return pi, pi_err, sig_l, sig_l_err, sig_r, sig_r_err
```

```
In [21]: pi_12, pi_err_12, sig_l_12, sig_l_err_12, sig_r_12, sig_r_err_12 =
pi_sigl_sigr(fits_12)
k_12 = range(1,len(pi_12)+1,1)

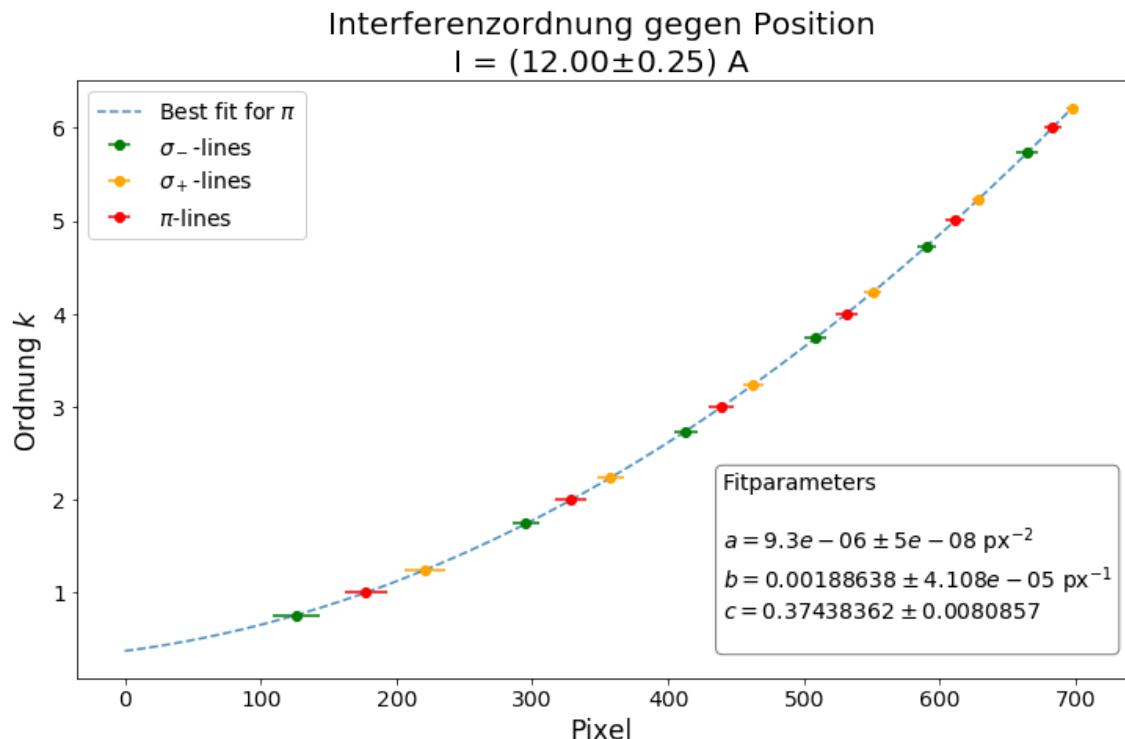
def quad(x,a,b,c):
    return(a*x**2+ b*x + c)
# Error for lateral lines
# params: [a, b, c], lines
def error_polynom(params, err_params, lines, err_lines):
    return sqrt( (err_params[0] * lines ** 2) ** 2
                + (2 * params[0] * err_lines) ** 2
                + (err_params[1] * lines) ** 2
                + (params[1] * err_lines) ** 2
                + (err_params[2]) ** 2)

@add_method(OmniTool)
def order_fit(pi, pi_err, sig_l, sig_l_err, sig_r, sig_r_err, k, num, a = 400, b = 1.2, r = 8 ):
    popt,pcov = curve_fit(quad, pi, k)
    x = np.linspace(0,sig_r[len(sig_r)-1],1000)
    plt.figure(figsize = (12, 7))
    plt.title('Interferenzordnung gegen Position\nI = (' + num + '.
00$\pm 0.25$) A',
               size = 20)
    plt.errorbar(sig_l, quad(sig_l,*popt), xerr = sig_l_err,fmt =
'o', color = 'green', label = '$\sigma_{-$lines'})
    plt.errorbar(sig_r, quad(sig_r,*popt), xerr = sig_r_err,fmt =
'o', color = 'orange', label = '$\sigma_{+$lines}')
    plt.errorbar(pi, quad(pi,*popt), xerr = pi_err, fmt = 'o',color =
'red', label = '$\sigma_{pi$-lines}')
    plt.plot(x, quad(x, *popt),
              label = 'Best fit for $\pi$',
              color = 'C0', alpha = 0.8, ls = '--')
    plt.xlabel('Pixel', size = 17)
    plt.xticks(size = 14)
    plt.ylabel('Ordnung $k$', size = 17)
    plt.yticks(size = 14)
    #plt.ylim(0,None)
    plt.legend(loc = 'best', fontsize = 14)

    params = popt
    err_params = sqrt(np.diag(pcov))
    textstrer('Fitparameters', params, err_params,
              ['$px$^{-2}', '$px$^{+1}', ''],
              ['a', 'b', 'c'], 440, 2.3, r)

    return params, err_params
```

```
In [22]: params_12, err_params_12 = order_fit(pi_12, pi_err_12, sig_l_12, sig_l_err_12, sig_r_12, sig_r_err_12, k_12, '12')
plt.savefig('diagrams/interferenceposition12.pdf')
```



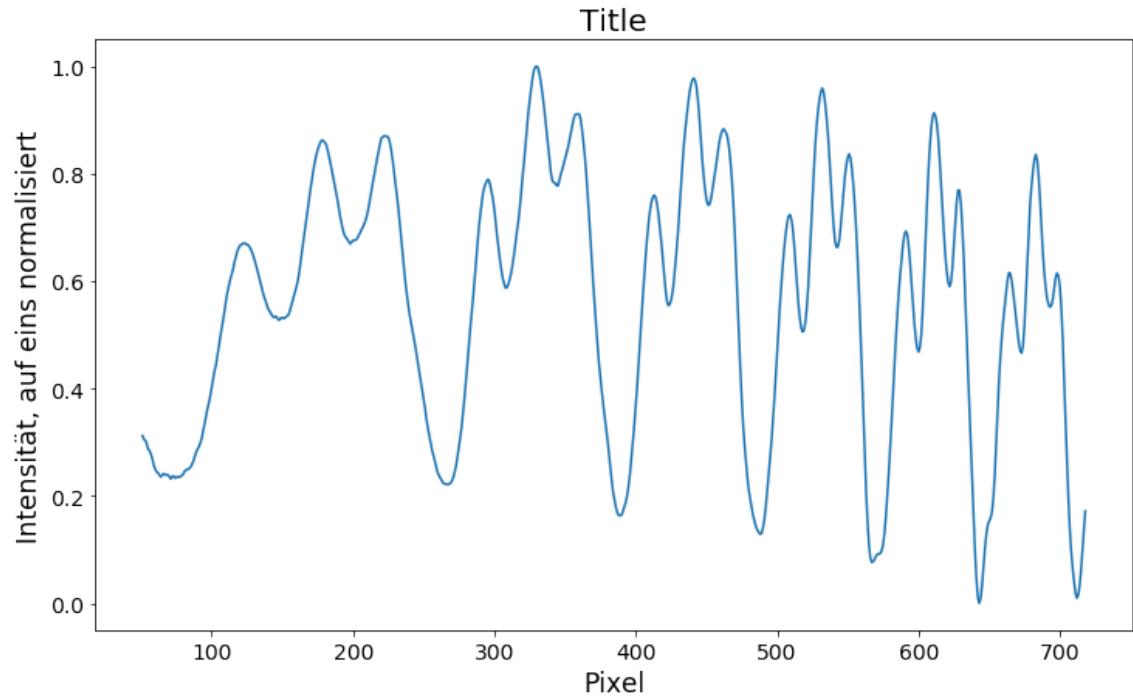
```
In [23]: @add_method(OmniTool)
def delta_k(sig_l, sig_r, params):
    for i in range(len(sig_l)):
        if i == 0:
            del_k = np.array(abs(quad(sig_l[i], *params) - quad(sig_r[i], *params))/2)
        else:
            del_k = np.append(del_k, abs(quad(sig_l[i], *params) - quad(sig_r[i], *params))/2)
    err_del_k = np.std(del_k)/np.sqrt(len(del_k))
    del_k = np.mean(del_k)
    print('delta k = {} +- {}'.format(e = del_k, d = err_del_k))
    return del_k, err_del_k
```

```
In [24]: del_k_12, err_del_k_12 = delta_k(sig_l_12, sig_r_12, params_12)

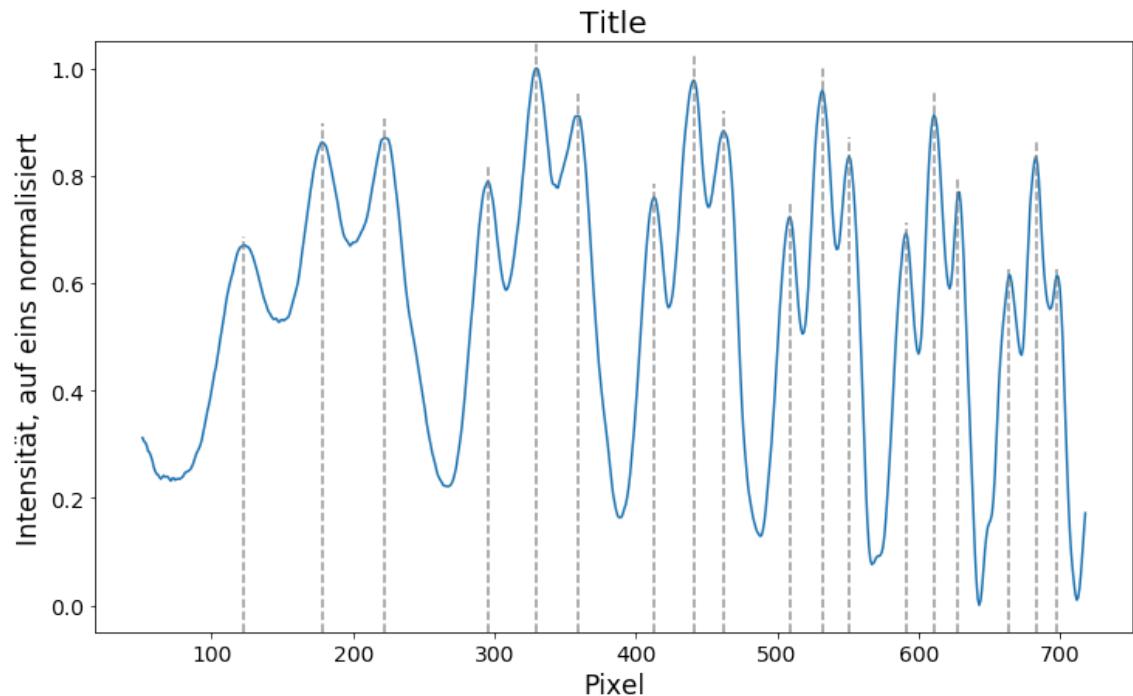
delta k = 0.24655661156862307 +- 0.0011433696482865774
```

13 A

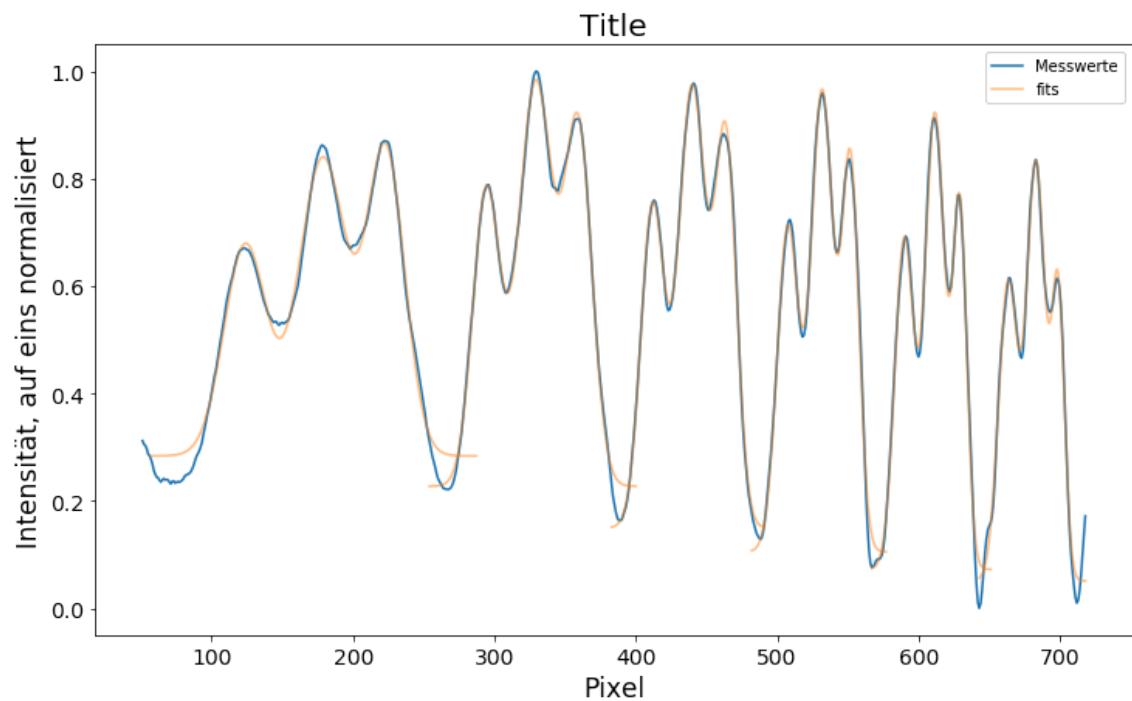
```
In [25]: file_13 = 'misc/13A.txt'  
#data_13 = OmniTool().tolino(file_13,800,50)  
fig_13 = OmniTool().plotter(file_13, 720, 50)
```



```
In [26]: peaks_13, peaks_pos_13, heights_13 = OmniTool().peaker(file_13, 720, 50, prom=0.002)  
fig_13 = OmniTool().plotter(file_13, 720, 50)  
OmniTool().peak_marker(peaks_pos_13, heights_13)
```



```
In [27]: groups_13 = OmniTool().grouper(file_12, 720 ,50, peaks_13, heights_13)
fig_13 = OmniTool().plotter(file_13, 720, 50)
fits_13 = OmniTool().gausser(file_13, 720, 50, groups_13)
```



```
In [28]: # 1. Plot the data
file = file_13
data = OmniTool().tolino(file,720,50)

fig = OmniTool().plotter(file,720,50)
plt.title('Zeeman Spektrum für  $I = (13.00 \pm 0.25) \text{ A}$ ', size = 20)
plt.ylim(-0.1, 1.05)

# 2. Find the peaks
peaks_13, peaks_pos_13, heights_13 = OmniTool().peaker(file, 720, 50)

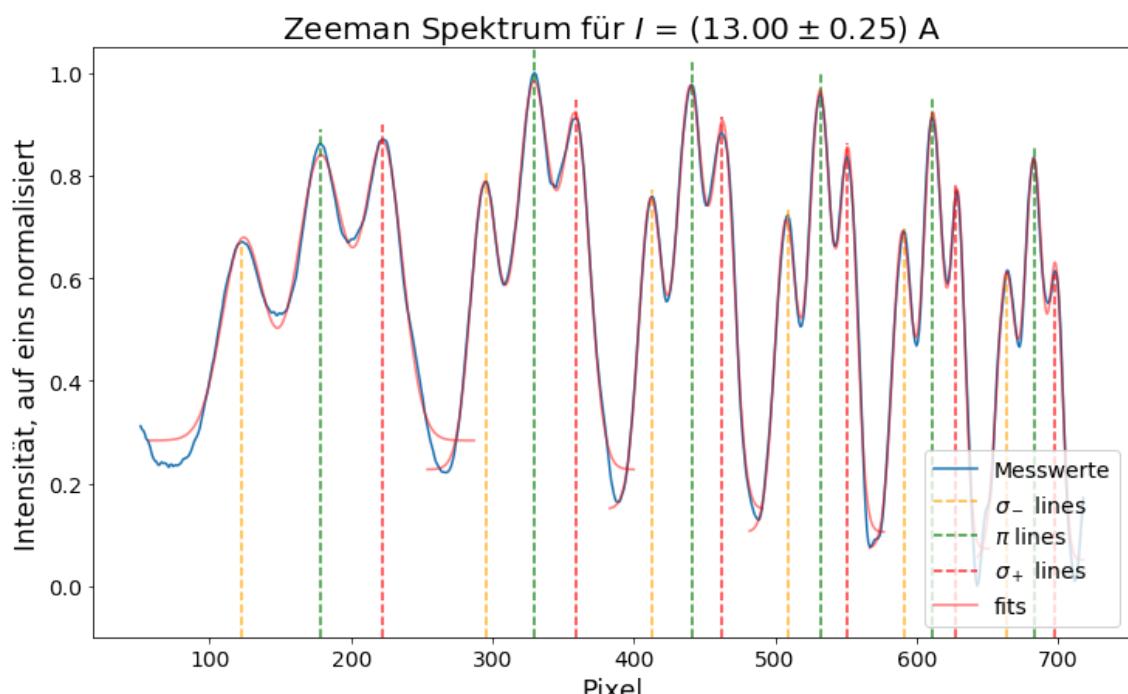
peaks_sig_l, heights_sig_l = peaks_pos_13[0::3], heights_13[0::3]
peaks_pi, heights_pi = peaks_pos_13[1::3], heights_13[1::3]
peaks_sig_r, heights_sig_r = peaks_pos_13[2::3], heights_13[2::3]

OmniTool().peak_marker(peaks_sig_l, heights_sig_l,
                       col = 'orange', lab = ' $\sigma_-$  lines')
OmniTool().peak_marker(peaks_pi, heights_pi,
                       col = 'green', lab = ' $\pi$  lines')
OmniTool().peak_marker(peaks_sig_r, heights_sig_r,
                       col = 'red', lab = ' $\sigma_+$  lines')

# 3. Group triplets
groups = OmniTool().grouper(file,720,50, peaks_13, heights_13)

# 4. Fit a triple Gauss on each triplet
fits_13 = gausser(file,720,50, groups, col = 'red')

# 5. Decorate the plot a little more if you like
plt.legend(loc = 'lower right', fontsize = 14)
# Little window
# Fit for order a
# HERE WOULD HAVE BEEN BIN[4]
plt.savefig('diagrams/zeemanspectrum13.pdf')
```



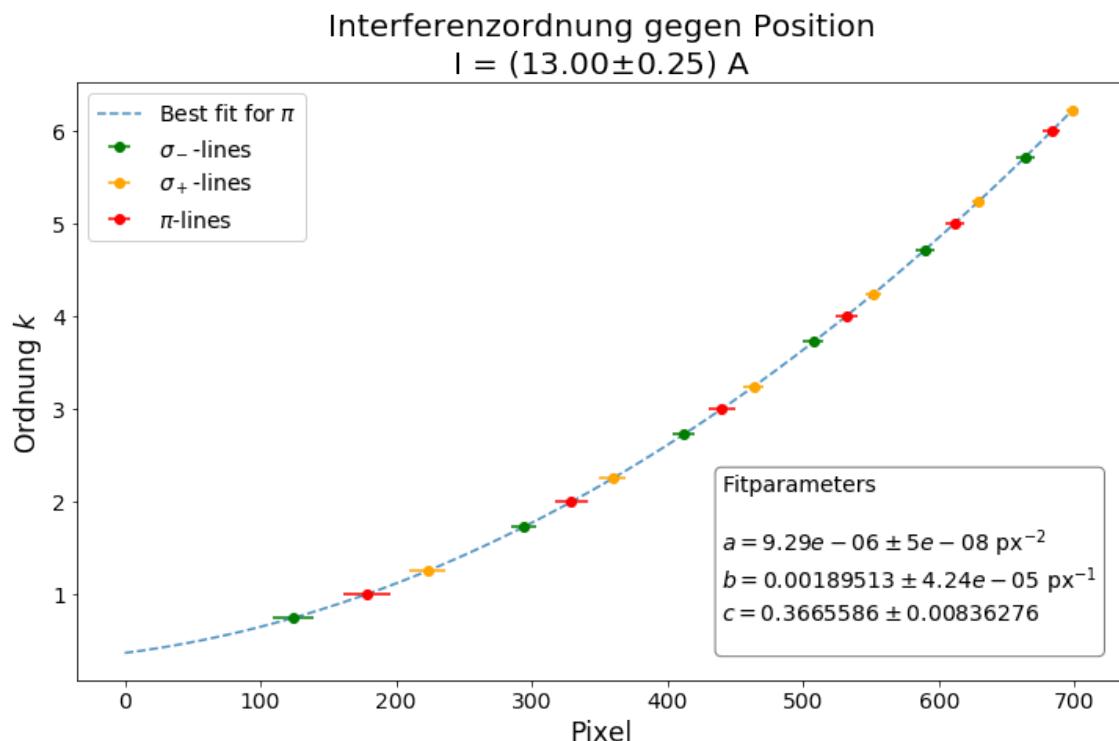
In [29]: `announcer_fits(fits_13)`

Parameters

```
Order: 1
  sigma_l: a = 0.39 +/- 0.01    b = 124.0 +/- 1.0 [cov] +/- 15.0 [stat]
  pi: a = 0.55 +/- 0.02    b = 179.0 +/- 1.0 [cov] +/- 17.0 [stat]
  sigma_r: a = 0.56 +/- 0.02    b = 223.0 +/- 1.0 [cov] +/- 13.0 [stat]
Order: 2
  sigma_l: a = 0.55 +/- 0.01    b = 294.0 +/- 0.0 [cov] +/- 10.0 [stat]
  pi: a = 0.75 +/- 0.01    b = 329.0 +/- 0.0 [cov] +/- 12.0 [stat]
  sigma_r: a = 0.65 +/- 0.01    b = 359.0 +/- 0.0 [cov] +/- 10.0 [stat]
Order: 3
  sigma_l: a = 0.59 +/- 0.01    b = 412.0 +/- 0.0 [cov] +/- 8.0 [stat]
  pi: a = 0.82 +/- 0.01    b = 440.0 +/- 0.0 [cov] +/- 10.0 [stat]
  sigma_r: a = 0.71 +/- 0.01    b = 464.0 +/- 0.0 [cov] +/- 8.0 [stat]
Order: 4
  sigma_l: a = 0.6 +/- 0.01    b = 508.0 +/- 0.0 [cov] +/- 8.0 [stat]
  pi: a = 0.86 +/- 0.01    b = 532.0 +/- 0.0 [cov] +/- 8.0 [stat]
  sigma_r: a = 0.72 +/- 0.01    b = 552.0 +/- 0.0 [cov] +/- 6.0 [stat]
Order: 5
  sigma_l: a = 0.61 +/- 0.01    b = 590.0 +/- 0.0 [cov] +/- 7.0 [stat]
  pi: a = 0.85 +/- 0.01    b = 612.0 +/- 0.0 [cov] +/- 7.0 [stat]
  sigma_r: a = 0.66 +/- 0.02    b = 629.0 +/- 0.0 [cov] +/- 5.0 [stat]
Order: 6
  sigma_l: a = 0.55 +/- 0.01    b = 663.0 +/- 0.0 [cov] +/- 7.0 [stat]
  pi: a = 0.77 +/- 0.01    b = 683.0 +/- 0.0 [cov] +/- 7.0 [stat]
  sigma_r: a = 0.53 +/- 0.02    b = 699.0 +/- 0.0 [cov] +/- 5.0 [stat]
```

In [30]: `pi_13, pi_err_13, sig_l_13, sig_l_err_13, sig_r_13, sig_r_err_13 = pi_sigl_sigr(fits_13)`
`k_13 = range(1, len(pi_13)+1, 1)`

```
In [31]: params_13, err_params_13 = order_fit(pi_13, pi_err_13, sig_l_13, si
g_l_err_13, sig_r_13, sig_r_err_13, k_13, '13')
plt.savefig('diagrams/interferenceposition13.pdf')
```

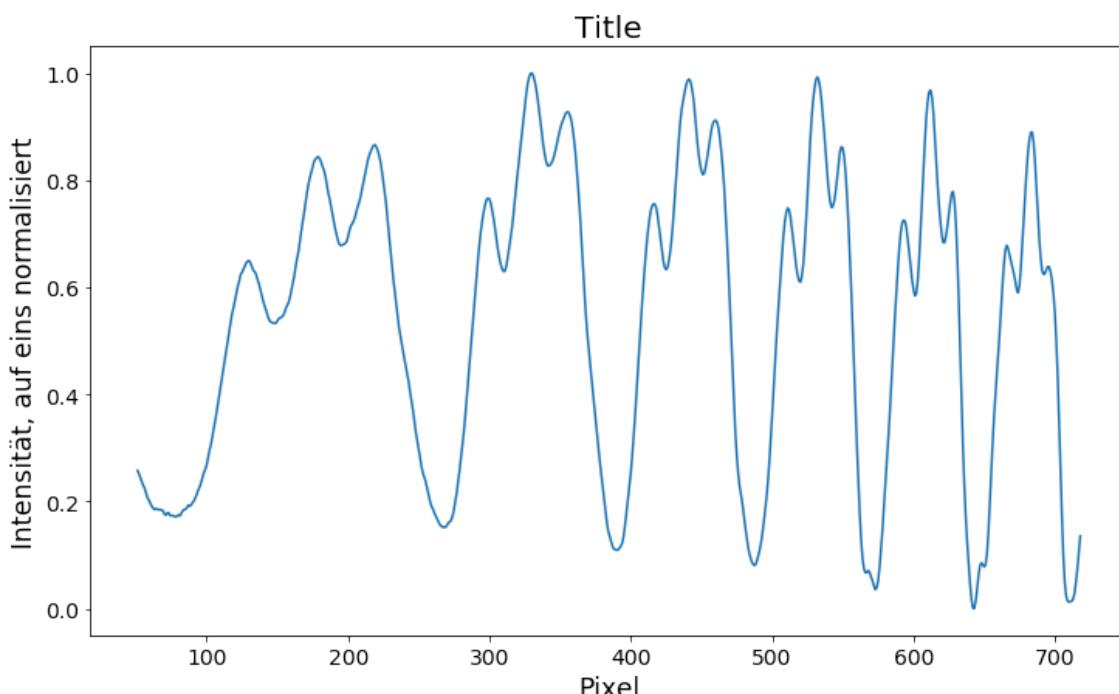


```
In [32]: del_k_13, err_del_k_13 = delta_k(sig_l_13, sig_r_13, params_13)
```

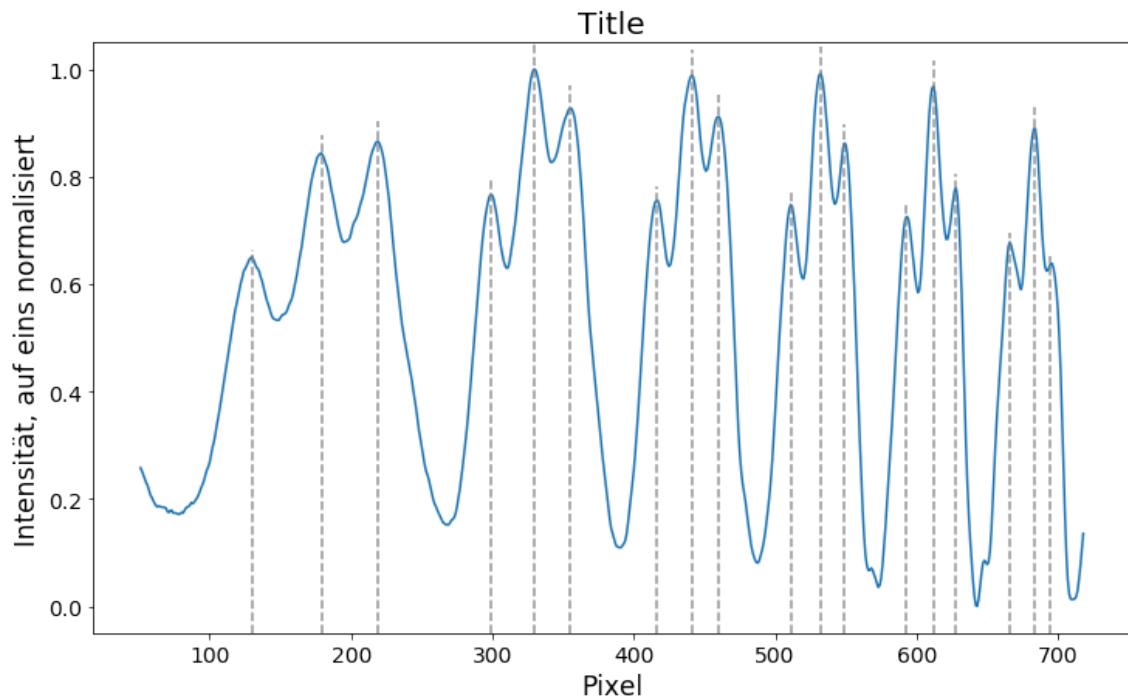
$\text{delta k} = 0.25860872501254134 \pm 0.0008799789101427808$

11 A

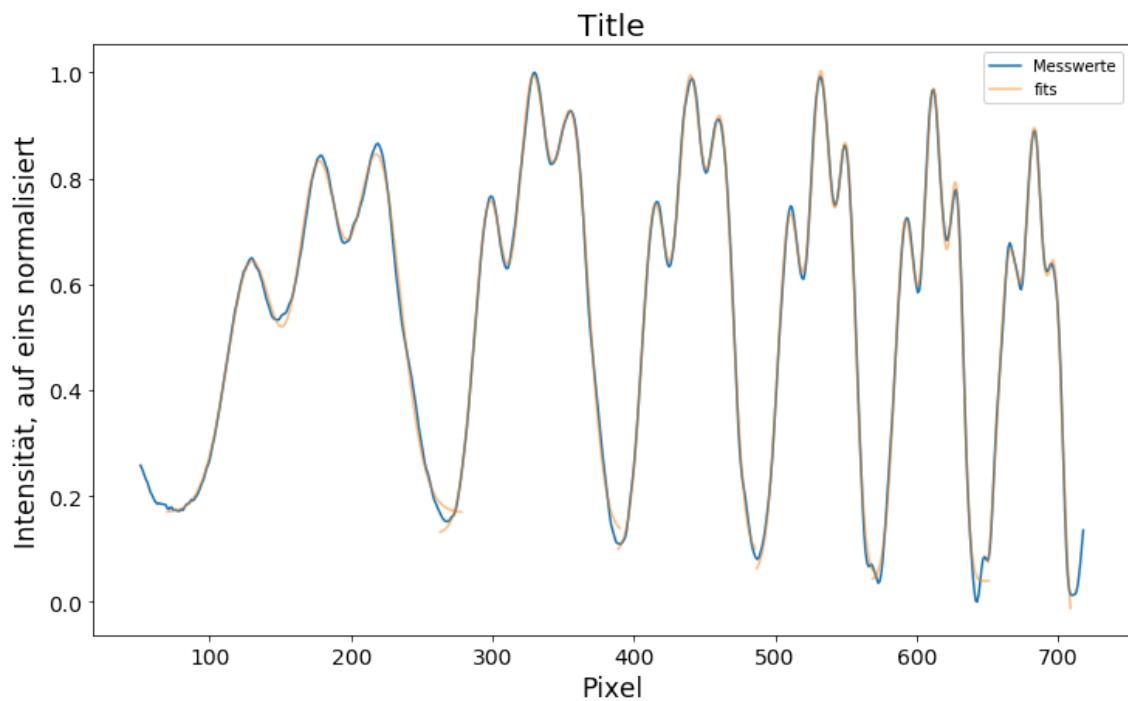
```
In [33]: file_11 = 'misc/11A.txt'
#data_13 = OmniTool().tolino(file_13, 800, 50)
fig_11 = OmniTool().plotter(file_11, 720, 50)
```



```
In [34]: peaks_11, peaks_pos_11, heights_11 = OmniTool().peaker(file_11, 720, 50, prom=0.008)
fig_11 = OmniTool().plotter(file_11, 720, 50)
OmniTool().peak_marker(peaks_pos_11, heights_11)
```



```
In [35]: groups_11 = OmniTool().grouper(file_11, 720, 50, peaks_11, heights_11)
fig_11 = OmniTool().plotter(file_11, 720, 50)
fits_11 = OmniTool().gausser(file_11, 720, 50, groups_11)
```



```
In [36]: # 1. Plot the data
file = file_11
data = OmniTool().tolino(file,720,50)

fig = OmniTool().plotter(file,720,50)
plt.title('Zeeman Spektrum für  $I = (11.00 \pm 0.25) \text{ A}$ ', size = 20)
plt.ylim(-0.1, 1.05)

# 2. Find the peaks
peaks_11, peaks_pos_11, heights_11 = OmniTool().peaker(file, 720, 50)

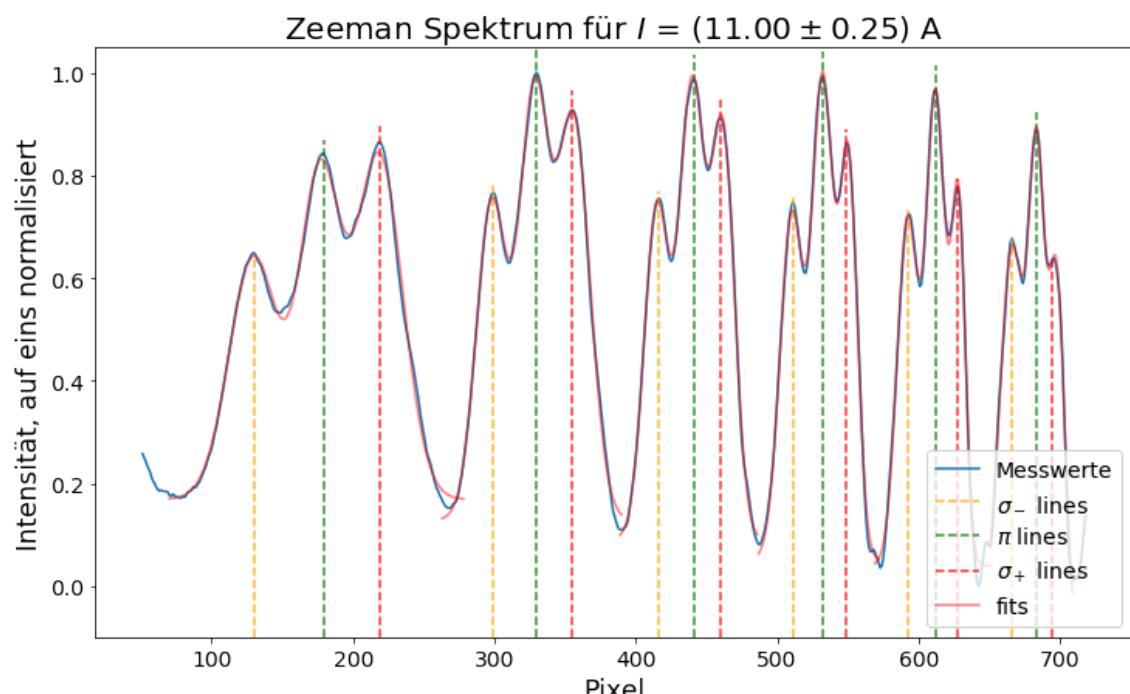
peaks_sig_l, heights_sig_l = peaks_pos_11[0::3], heights_11[0::3]
peaks_pi, heights_pi = peaks_pos_11[1::3], heights_11[1::3]
peaks_sig_r, heights_sig_r = peaks_pos_11[2::3], heights_11[2::3]

OmniTool().peak_marker(peaks_sig_l, heights_sig_l,
                       col = 'orange', lab = ' $\sigma_-$  lines')
OmniTool().peak_marker(peaks_pi, heights_pi,
                       col = 'green', lab = ' $\pi$  lines')
OmniTool().peak_marker(peaks_sig_r, heights_sig_r,
                       col = 'red', lab = ' $\sigma_+$  lines')

# 3. Group triplets
groups = OmniTool().grouper(file,720,50, peaks_11, heights_11)

# 4. Fit a triple Gauss on each triplet
fits_13 = gausser(file,720,50, groups, col = 'red')

# 5. Decorate the plot a little more if you like
plt.legend(loc = 'lower right', fontsize = 14)
# Little window
# Fit for order a
# HERE WOULD HAVE BEEN BIN[4]
plt.savefig('diagrams/zeemanspectrum11.pdf')
```



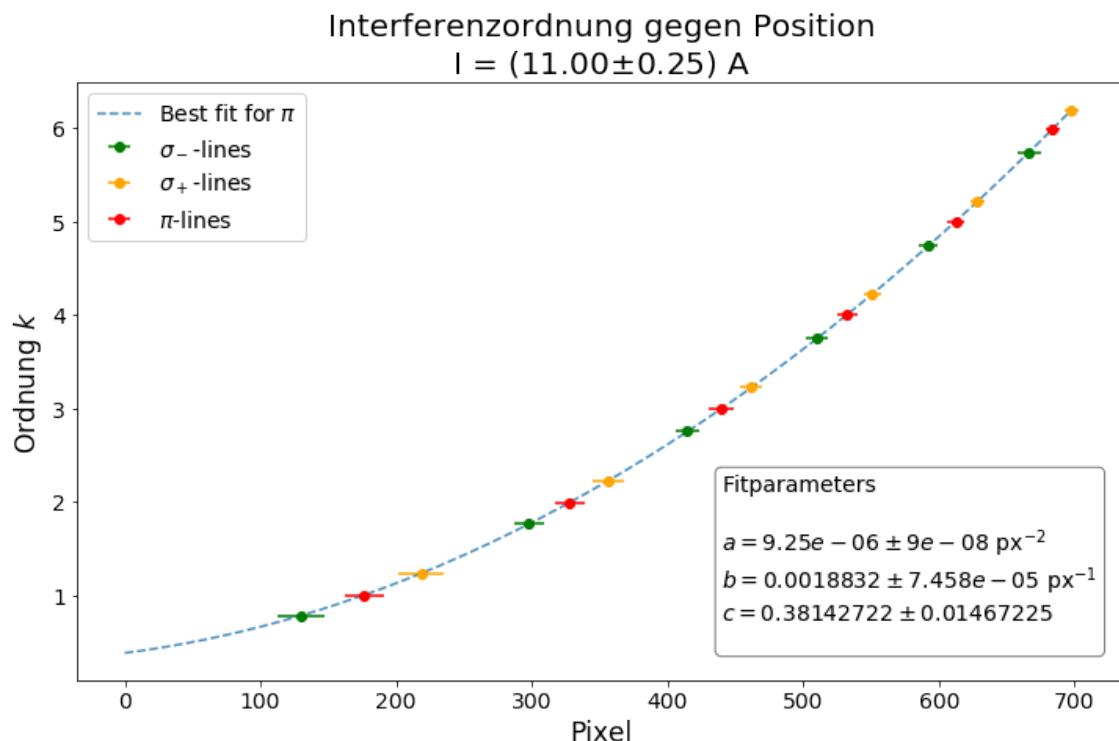
In [37]: `announcer_fits(fits_11)`

Parameters

```
Order: 1
  sigma_l: a = 0.47 +/- 0.0    b = 130.0 +/- 0.0 [cov] +/- 17.0 [stat]
  pi: a = 0.62 +/- 0.0    b = 177.0 +/- 0.0 [cov] +/- 14.0 [stat]
  sigma_r: a = 0.67 +/- 0.0    b = 219.0 +/- 0.0 [cov] +/- 17.0 [stat]
  Order: 2
  sigma_l: a = 0.61 +/- 0.01   b = 298.0 +/- 0.0 [cov] +/- 11.0 [stat]
  pi: a = 0.81 +/- 0.01   b = 328.0 +/- 0.0 [cov] +/- 11.0 [stat]
  sigma_r: a = 0.77 +/- 0.01   b = 356.0 +/- 0.0 [cov] +/- 12.0 [stat]
  Order: 3
  sigma_l: a = 0.64 +/- 0.01   b = 415.0 +/- 0.0 [cov] +/- 9.0 [stat]
  pi: a = 0.87 +/- 0.0    b = 440.0 +/- 0.0 [cov] +/- 9.0 [stat] [px]
  sigma_r: a = 0.77 +/- 0.01   b = 462.0 +/- 0.0 [cov] +/- 8.0 [stat]
  Order: 4
  sigma_l: a = 0.67 +/- 0.01   b = 510.0 +/- 0.0 [cov] +/- 8.0 [stat]
  pi: a = 0.92 +/- 0.01   b = 532.0 +/- 0.0 [cov] +/- 7.0 [stat]
  sigma_r: a = 0.76 +/- 0.01   b = 551.0 +/- 0.0 [cov] +/- 6.0 [stat]
  Order: 5
  sigma_l: a = 0.67 +/- 0.01   b = 592.0 +/- 0.0 [cov] +/- 7.0 [stat]
  pi: a = 0.91 +/- 0.01   b = 612.0 +/- 0.0 [cov] +/- 7.0 [stat]
  sigma_r: a = 0.68 +/- 0.02   b = 628.0 +/- 0.0 [cov] +/- 5.0 [stat]
  Order: 6
  sigma_l: a = 0.75 +/- 0.02   b = 666.0 +/- 0.0 [cov] +/- 9.0 [stat]
  pi: a = 0.86 +/- 0.02   b = 684.0 +/- 0.0 [cov] +/- 6.0 [stat]
  sigma_r: a = 0.69 +/- 0.03   b = 698.0 +/- 0.0 [cov] +/- 5.0 [stat]
```

In [38]: `pi_11, pi_err_11, sig_l_11, sig_l_err_11, sig_r_11, sig_r_err_11 = pi_sigl_sigr(fits_11)`
`k_11 = range(1, len(pi_11)+1, 1)`

```
In [39]: params_11, err_params_11 = order_fit(pi_11, pi_err_11, sig_l_11, sig_l_err_11, sig_r_11, sig_r_err_11, k_11, '11')
plt.savefig('diagrams/interferenceposition11.pdf')
```

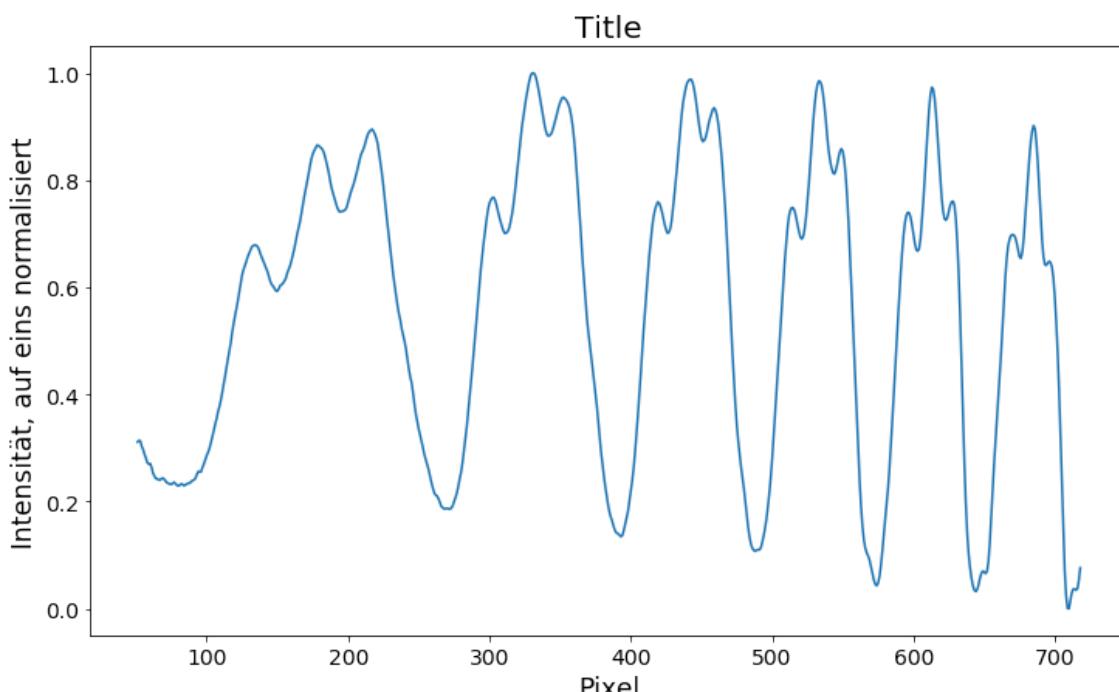


```
In [40]: del_k_11, err_del_k_11 = delta_k(sig_l_11, sig_r_11, params_11)
```

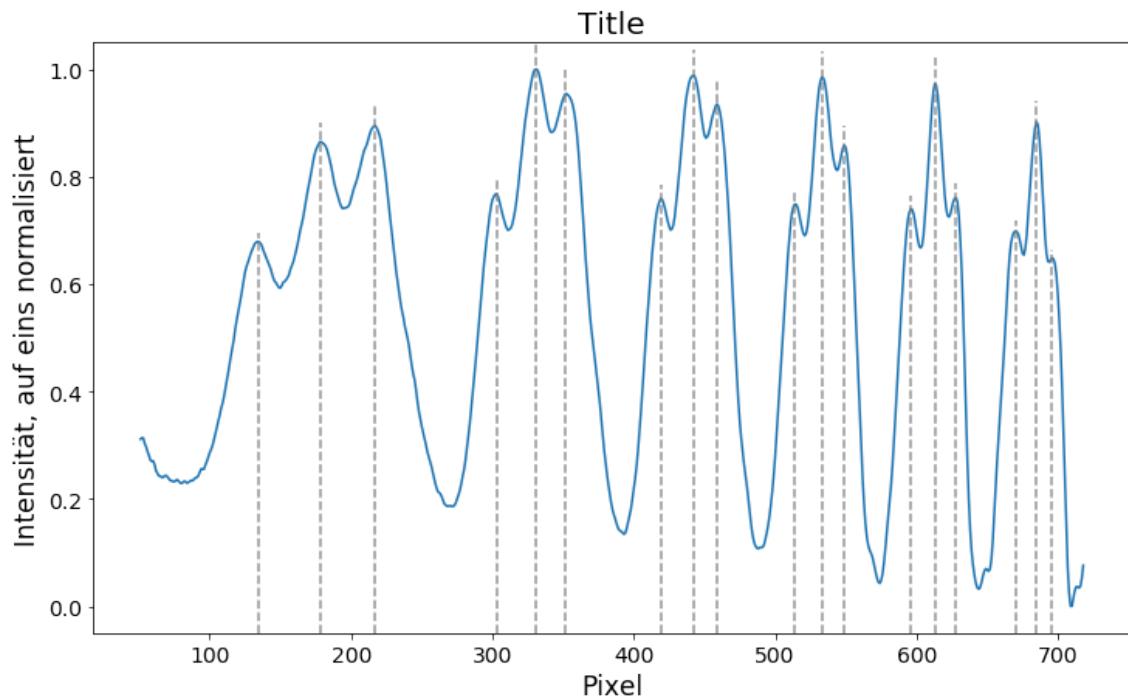
$\delta k = 0.23262876527336562 \pm 0.001660709133027063$

10 A

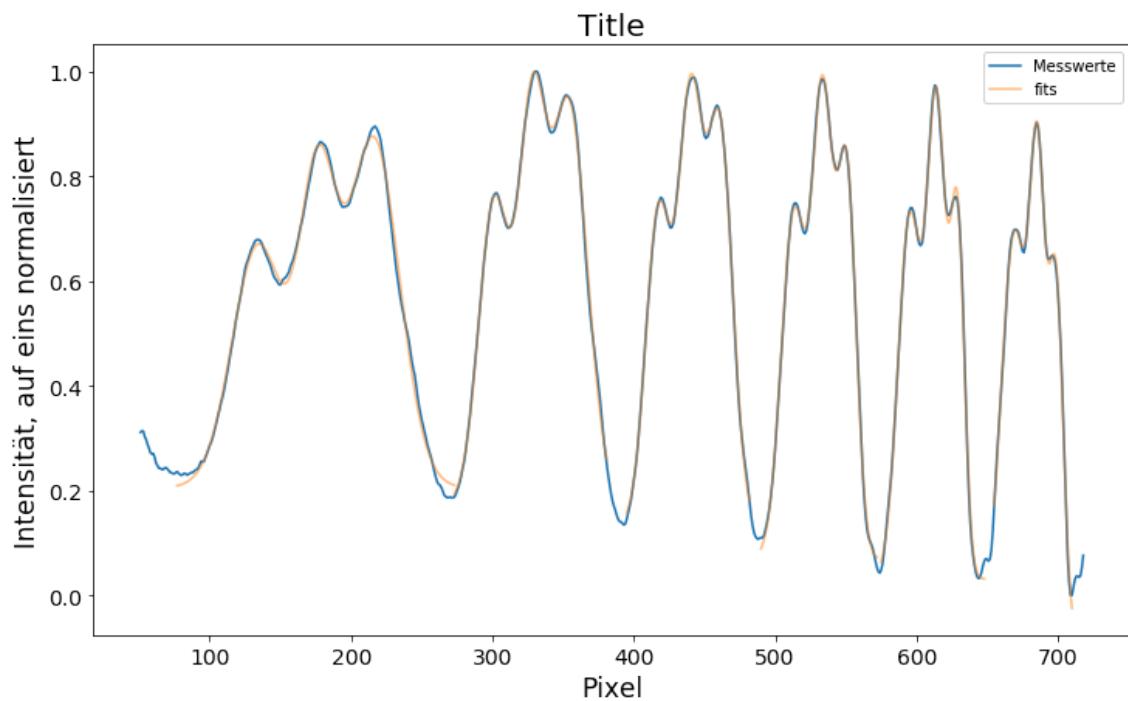
```
In [41]: file_10 = 'misc/10A.txt'
#data_13 = OmniTool().tolino(file_13, 800, 50)
fig_10 = OmniTool().plotter(file_10, 720, 50)
```



```
In [42]: peaks_10, peaks_pos_10, heights_10 = OmniTool().peaker(file_10, 720, 50, prom=0.006)
fig_10 = OmniTool().plotter(file_10, 720, 50)
OmniTool().peak_marker(peaks_pos_10, heights_10)
```



```
In [43]: groups_10 = OmniTool().grouper(file_10, 720, 50, peaks_10, heights_10)
fig_10 = OmniTool().plotter(file_10, 720, 50)
fits_10 = OmniTool().gausser(file_10, 720, 50, groups_10)
```



```
In [44]: # 1. Plot the data
file = file_10
data = OmniTool().tolino(file,720,50)

fig = OmniTool().plotter(file,720,50)
plt.title('Zeeman Spektrum für  $I = (10.00 \pm 0.25) \text{ A}$ ', size = 20)
plt.ylim(-0.1, 1.05)

# 2. Find the peaks
peaks_10, peaks_pos_10, heights_10 = OmniTool().peaker(file, 720, 50, prom=0.006)

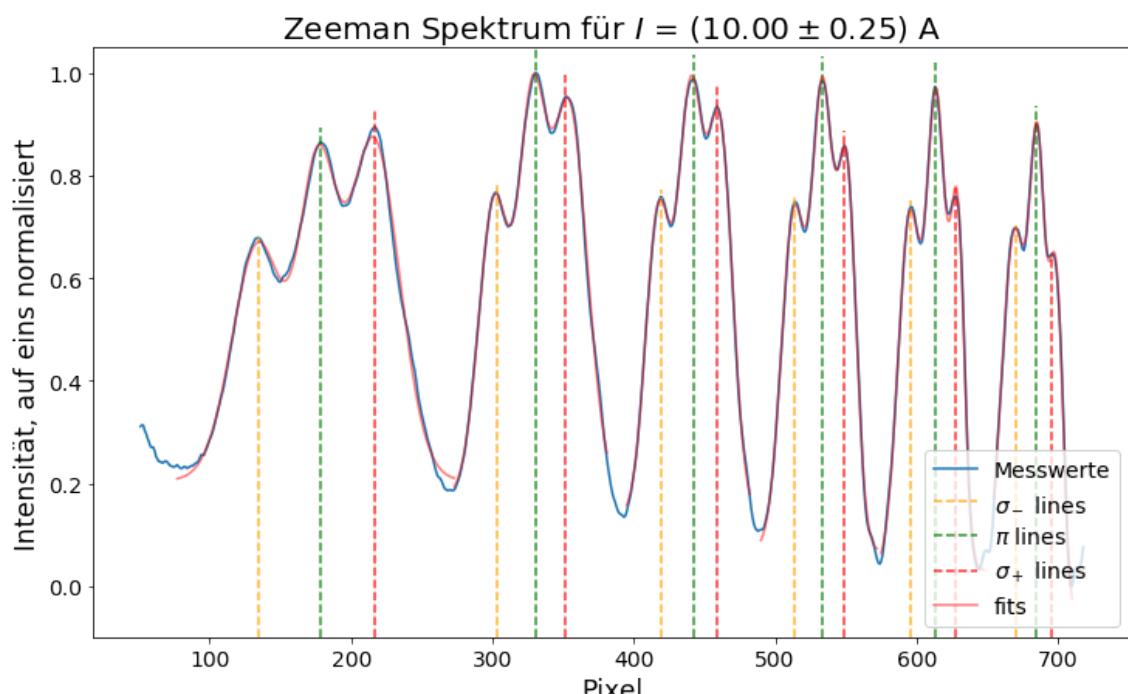
peaks_sig_l, heights_sig_l = peaks_pos_10[0::3], heights_10[0::3]
peaks_pi, heights_pi = peaks_pos_10[1::3], heights_10[1::3]
peaks_sig_r, heights_sig_r = peaks_pos_10[2::3], heights_10[2::3]

OmniTool().peak_marker(peaks_sig_l, heights_sig_l,
                       col = 'orange', lab = ' $\sigma_-$  lines')
OmniTool().peak_marker(peaks_pi, heights_pi,
                       col = 'green', lab = ' $\pi$  lines')
OmniTool().peak_marker(peaks_sig_r, heights_sig_r,
                       col = 'red', lab = ' $\sigma_+$  lines')

# 3. Group triplets
groups = OmniTool().grouper(file,720,50, peaks_10, heights_10)

# 4. Fit a triple Gauss on each triplet
fits_10 = gausser(file,720,50, groups, col = 'red')

# 5. Decorate the plot a little more if you like
plt.legend(loc = 'lower right', fontsize = 14)
# Little window
# Fit for order a
# HERE WOULD HAVE BEEN BIN[4]
plt.savefig('diagrams/zeemanspectrum10.pdf')
```



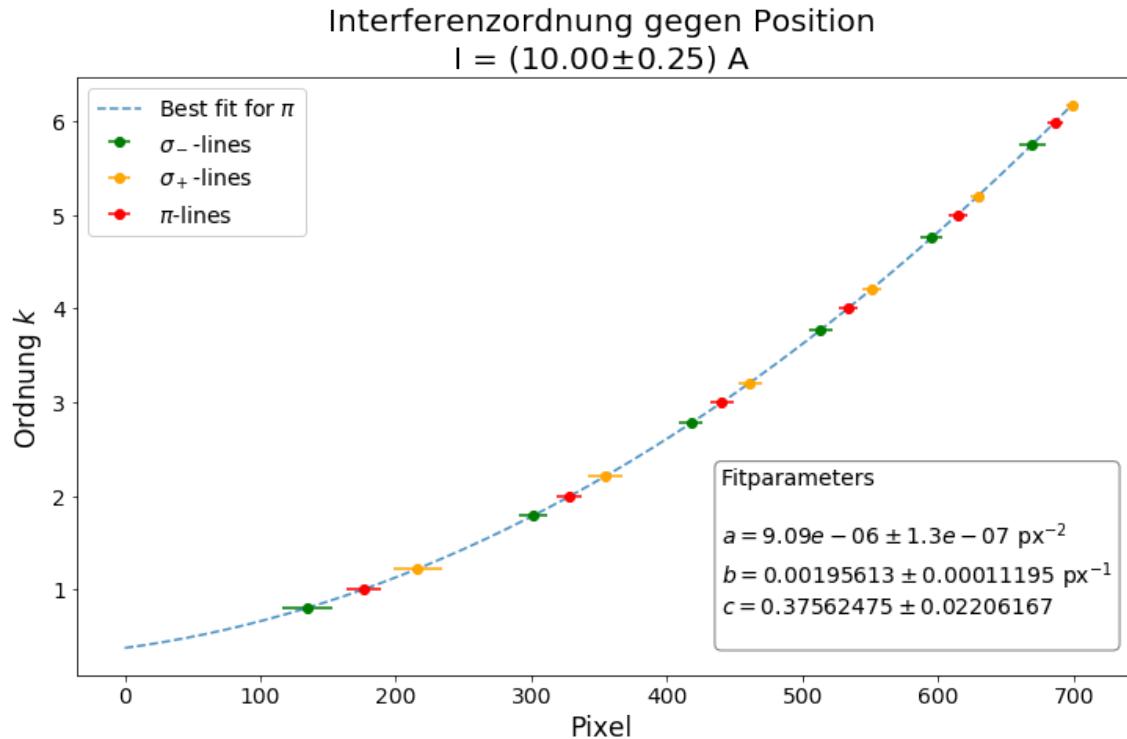
In [45]: `announcer_fits(fits_10)`

Parameters

```
Order: 1
  sigma_l: a = 0.46 +/- 0.0    b = 134.0 +/- 0.0 [cov] +/- 18.0 [stat]
  pi: a = 0.56 +/- 0.01    b = 176.0 +/- 0.0 [cov] +/- 13.0 [stat]
  sigma_r: a = 0.66 +/- 0.0    b = 216.0 +/- 0.0 [cov] +/- 18.0 [stat]
Order: 2
  sigma_l: a = 0.57 +/- 0.01    b = 301.0 +/- 0.0 [cov] +/- 11.0 [stat]
  pi: a = 0.7 +/- 0.01    b = 328.0 +/- 0.0 [cov] +/- 10.0 [stat]
  sigma_r: a = 0.76 +/- 0.01    b = 354.0 +/- 0.0 [cov] +/- 13.0 [stat]
Order: 3
  sigma_l: a = 0.59 +/- 0.01    b = 417.0 +/- 0.0 [cov] +/- 9.0 [stat]
  pi: a = 0.79 +/- 0.01    b = 440.0 +/- 0.0 [cov] +/- 9.0 [stat]
  sigma_r: a = 0.75 +/- 0.01    b = 461.0 +/- 0.0 [cov] +/- 9.0 [stat]
Order: 4
  sigma_l: a = 0.65 +/- 0.01    b = 513.0 +/- 0.0 [cov] +/- 9.0 [stat]
  pi: a = 0.86 +/- 0.01    b = 534.0 +/- 0.0 [cov] +/- 7.0 [stat]
  sigma_r: a = 0.72 +/- 0.01    b = 551.0 +/- 0.0 [cov] +/- 7.0 [stat]
Order: 5
  sigma_l: a = 0.69 +/- 0.01    b = 595.0 +/- 0.0 [cov] +/- 8.0 [stat]
  pi: a = 0.89 +/- 0.01    b = 614.0 +/- 0.0 [cov] +/- 7.0 [stat]
  sigma_r: a = 0.66 +/- 0.01    b = 629.0 +/- 0.0 [cov] +/- 5.0 [stat]
Order: 6
  sigma_l: a = 0.77 +/- 0.02    b = 669.0 +/- 0.0 [cov] +/- 10.0 [stat]
  pi: a = 0.78 +/- 0.02    b = 686.0 +/- 0.0 [cov] +/- 5.0 [stat]
  sigma_r: a = 0.66 +/- 0.02    b = 699.0 +/- 0.0 [cov] +/- 5.0 [stat]
```

In [46]: `pi_10, pi_err_10, sig_l_10, sig_l_err_10, sig_r_10, sig_r_err_10 =
pi_sigl_sigr(fits_10)
k_10 = range(1, len(pi_10)+1, 1)`

```
In [47]: params_10, err_params_10 = order_fit(pi_10, pi_err_10, sig_l_10, sig_l_err_10, sig_r_10, sig_r_err_10, k_10, '10')
plt.savefig('diagrams/interferenceposition10.pdf')
```



```
In [48]: del_k_10, err_del_k_10 = delta_k(sig_l_10, sig_r_10, params_10)
```

```
delta k = 0.21465861684638335 +- 0.0019080176932963274
```

```
In [49]: del_k = np.array([del_k_10, del_k_11, del_k_12, del_k_13])
err_del_k = np.array([err_del_k_10, err_del_k_11, err_del_k_12, err
_del_k_13])
```

part 2

determination of the red Cd line

```
In [50]: @add_method(OmniTool)
def single_gausser(file, upper, lower, peaks, heights,
                    col = 'C1',
                    alph = 0.5,
                    dim = 4,
                    symcut = 100):
    data = OmniTool().tolino(file, upper, lower)
    popts = np.empty([1, dim])
    pcovs = np.empty([1, dim])
    for i in range(len(peaks)):
        # closeup on one order
        data_ab = (data[0][int(peaks[i])] - symcut:int(peaks[i]) + symcut],
                   data[1][int(peaks[i])] - symcut:int(peaks[i]) + symcut])
        # estimated parameters based on peak finding algorithm
        p0_i = np.append(heights[i], data[0][peaks[i]])
        p0_i = np.append(p0_i, symcut/2)
        p0_i = np.append(p0_i, 0)

        popt_i, pcov_i = curve_fit(gaussian, data_ab[0],
                                     data_ab[1], p0 = p0_i)

        popts = np.append(popts, [popt_i], axis = 0)
        pcovs = np.append(pcovs, [sqrt(np.diag(pcov_i))], axis = 0)

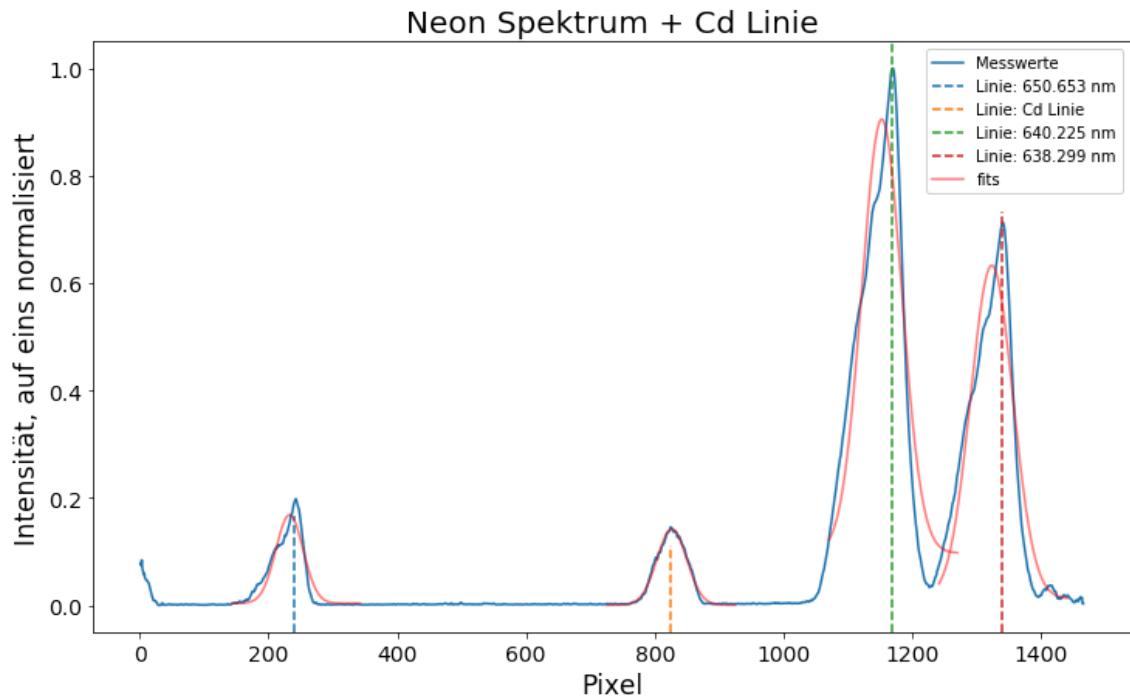
        x_ab = np.linspace(data_ab[0][0], data_ab[0][-1], 200)

        if i==0:
            plt.plot(x_ab, gaussian(x_ab, *popt_i),
                      color = col, alpha = alph, label = 'fits')
        else:
            plt.plot(x_ab, gaussian(x_ab, *popt_i),
                      color = col, alpha = alph)

    popts = popts[1:] # eliminate that empty
    pcovs = pcovs[1:]
    plt.legend()

    return popts, pcovs
```

```
In [51]: file_ne_cd = 'misc/Ne+Cd.txt'
data_ne_cd = OmniTool().tolino(file_ne_cd, 1600, 0)
fig_ne_cd = OmniTool().plotter(file_ne_cd, 1600, 0)
plt.title('Neon Spektrum + Cd Linie', size = 20)
# Peaks
# 2. Find peaks
peaks_ne_cd, peaks_pos_ne_cd, heights_ne_cd = OmniTool().peaker(file_ne_cd, 1600, 0, ds = 1, prom = 0.1, ht = 0.1)
# 2.5 Plot peaks
wls = [650.653, 'Cd Linie', 640.225, 638.299]
labels_wls = [str(wls[0]) + ' nm', str(wls[1]), str(wls[2]) + ' nm',
str(wls[3]) + ' nm']
cols = ['C0', 'C1', 'C2', 'C3']
for i in range(len(peaks_ne_cd)):
    OmniTool().peak_marker([peaks_ne_cd[i]], [heights_ne_cd[i]],
                           col = cols[i],
                           lab = 'Linie: ' + labels_wls[i],
                           alph = 1)
plt.legend()
fits_ne_cd = OmniTool().single_gausser(file_ne_cd, 1600, 0, peaks_pos_ne_cd, heights_ne_cd, col='red')
plt.savefig('diagrams/neon_cd_together.pdf')
```



```
In [52]: @add_method(OmniTool)
def pos_err(fits):
    for i in range(len(fits[0])):
        if i == 0:
            pos = np.array(fits[0][i][1])
            err_pos = np.array(fits[1][i][1])
            err = np.array(fits[0][i][2])
            err_err = np.array(fits[1][i][2])
        else:
            pos = np.append(pos, fits[0][i][1])
            err_pos = np.append(err_pos, fits[1][i][1])
            err = np.append(err, fits[0][i][2])
            err_err = np.append(err_err, fits[1][i][2])
    return pos, err, err_pos, err_err
```

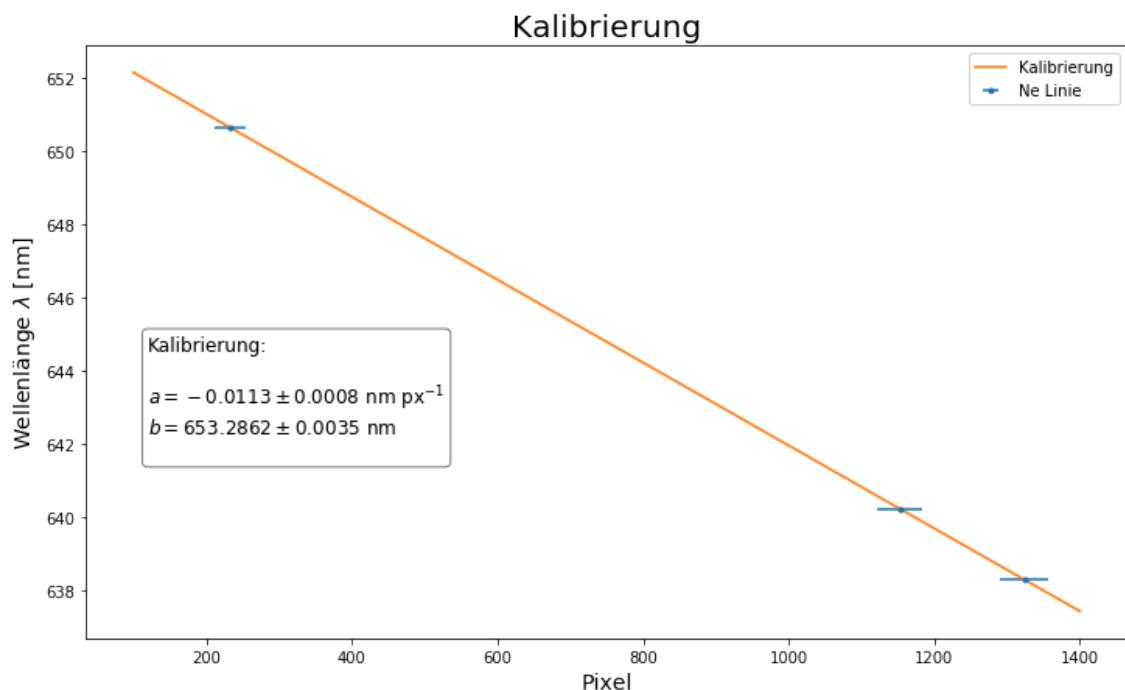
```
In [53]: pos_ne_cd, err_ne_cd, err_pos_ne_cd, err_err_ne_cd = pos_err(fits_ne_cd)
for i in range(len(pos_ne_cd)):
    print('pos = {} +- {} px'.format(e = pos_ne_cd[i], d = err_ne_cd[i]))
pos = 232.60330126040583 +- 21.2724382334289 px
pos = 826.6843589679711 +- 23.041184309198787 px
pos = 1152.8842259421463 +- 31.005685414363615 px
pos = 1323.8579990508072 +- 32.860852417665185 px
```

```
In [54]: popt_cal, pcov_cal = curve_fit(line, [pos_ne_cd[0], pos_ne_cd[2], pos_ne_cd[3]], [650.653, 640.225, 638.299])
```

```
In [55]: params = popt_cal
err_params = sqrt(np.diag(pcov_cal))
units = ['\mathrm{nm\ px}^{-1}', '\mathrm{nm}']
labels = ['a', 'b']

plt.figure(figsize = (12, 7))
plt.title('Kalibrierung', size = 20)
plt.xlabel('Pixel', size = 14)
plt.ylabel('Wellenlänge  $\lambda$  [nm]', size = 14)
textstrer('Kalibrierung:', params, err_params, units,
          labels, 120, 645, r = 4, fonts = 12)
x_lin = np.linspace(100, 1400, 200)
plt.plot(x_lin, line(x_lin, *popt_cal), color = 'C1', label = 'Kalibrierung')
plt.errorbar([pos_ne_cd[0], pos_ne_cd[2], pos_ne_cd[3]], [650.653, 640.225, 638.299], xerr = [err_ne_cd[0], err_ne_cd[2], err_ne_cd[3]], fmt = 'o', label = 'Ne Linie')
plt.legend()

plt.savefig('diagrams/neon_calibration.pdf')
```



```
In [56]: def err_cd_line(pos_cd,err_cd, a, b, err_a, err_b):
    return sqrt((a*err_cd)**2 + (err_a*pos_cd)**2 + (err_b)**2)
wl_cd, err_wl_cd = (line(pos_ne_cd[1],*popt_cal), err_cd_line(pos_ne_cd[1], err_ne_cd[1], params[0], params[1], err_params[0], err_params[1]))
print('Cd Linie = ({d} +- {e}) nm'.format(d = wl_cd, e = err_wl_cd))
wl_cd_lit = 643.84695 #nm
```

Cd Linie = (643.9245320731894 +- 0.6954052495491295) nm

```
In [57]: print('sigma deviation = {e}'.format(e = abs(wl_cd_lit-wl_cd)/err_wl_cd))
```

sigma deviation = 0.11156383021228788

```
In [58]: data_cd_ne = OmniTool().tolino(file_ne_cd,1600,0)
# Adjust the data to give the units in nm
data_cd_ne_wl = params[1] + params[0] * data_cd_ne[0], data_cd_ne[1]

plt.figure(figsize = (12, 7))
plt.plot(data_cd_ne_wl[0], data_cd_ne_wl[1], label = 'Messwerte')

plt.title('Spektrum geeicht auf Wellenl nge', size = 20)
plt.xlabel('Wellenl nge  $\lambda$  [nm]', size = 17)
plt.xticks(size = 14)
plt.ylabel('Intensit t, normalisiert', size = 17)
plt.yticks(size = 14)

# Find peaks
peaks, peaks_pos, heights = OmniTool().peaker(file_ne_cd, 1600, 0, prob = 0.01, ht = 0.1)
peaks = params[1] + params[0] * peaks_pos

# Fit one Gauss
p0_cd = [0.2, 644, 0.01, 0]
x = data_cd_ne_wl[0][500:1000]
y = data_cd_ne_wl[1][500:1000]
popt_cd, pcov_cd = curve_fit(gaussian, x, y, p0 = p0_cd)

line_cd, err_line_cd = np.round(popt_cd[1], 2), np.round(popt_cd[2], 2)

plt.plot(x, gaussian(x, *popt_cd), label = 'Cadmium fit', linewidth = 2)

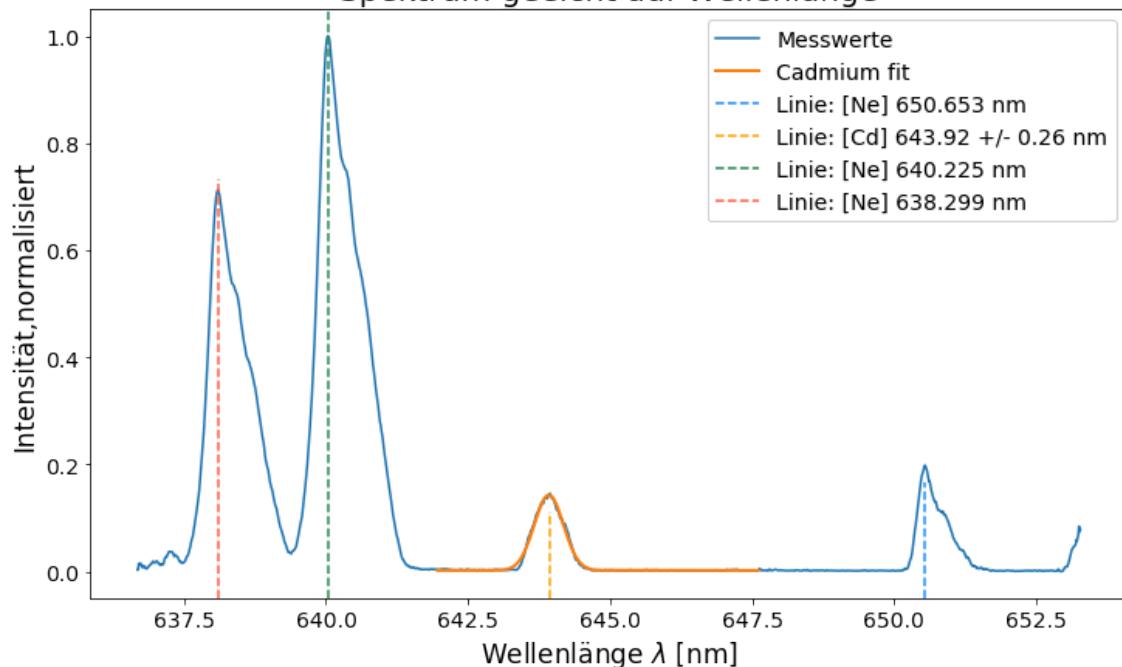
wls = [650.653, 640.225, 638.299]
labels_wls = ['[Ne] ' + str(wls[0]) + ' nm',
              '[Cd] ' + str(line_cd) + ' +/- ' + str(err_line_cd) +
              ' nm',
              '[Ne] ' + str(wls[1]) + ' nm',
              '[Ne] ' + str(wls[2]) + ' nm']
cols = ['dodgerblue', 'orange', 'seagreen', 'tomato']

for i in range(len(peaks)):
    OmniTool().peak_marker([peaks[i]], [heights[i]],
                           col = cols[i],
                           lab = 'Linie: ' + labels_wls[i],
                           alph = 1)

plt.legend(loc = 'best', fontsize = 14)

plt.savefig('diagrams/ne+cd_wl.pdf')
```

Spektrum geeicht auf Wellenlänge



In []: