FZD-496

Die numerische Auswertung von Kleinwinkelstreukurven

Roland Küchler

Mai 2008

Wissenschaftlich-Technische Berichte
FZD-496 2008 · ISSN 1437-322X

WISSENSCHAFTLICH-TECHNISCHE BERICHTE

Forschungszentrum
Dresden Rossendorf

Wissenschaftlich-Technische Berichte
FZD-496
Mai 2008

Roland Küchler

Die numerische Auswertung von Kleinwinkelstreukurven

Forschungszentrum
Dresden Rossendorf

Die numerische Auswertung von Kleinwinkelstreukurven

Roland Küchler

Kurzfassung

Aus dem Streubild der Kleinwinkelstreuung kann im Allgemeinen, die die Streuverteilung erzeugende Struktur nicht eindeutig rekonstruiert werden. Die Ursache dafür wird erörtert und die damit verbundenen Einschränkungen bei der rechnerischen Auswertung der Streukurven an Beispielen veranschaulicht. Dies geschieht an Streukurven, die mit bekannten Größenverteilungen berechnet wurden. Weiterhin wird untersucht, welche Fit-Ansätze sich zur Auswertung der Kleinwinkelstreuexperimente am besten eignen. Als Fit- Ansätze wurden Reihenentwicklungen nach Trigonometrischen- und Polynomfunktionen und eine theoretisch motivierte Funktion verwendet. Neben dem entscheidenden Vergleich mit der Streukurve der Ausgangsfunktion werden die Ergebnisse auch den Rechnungen gegenübergestellt, die mit der weit verbreiteten Glatter- Methode erzielt werden.

Abstract

In general, the structure generating the scattering distribution can not be reconstructed from the scattering pattern of small-angle scattering. The report discusses the causes for this and illustrates the associated restrictions during computational evolution of the scattering curves with the aid of examples. Such constraints occur when scattering curves were computed with well-known size distributions. Furthermore, it will be examined which fit setup is best for the evaluation of size distribution by the small angle scattering. Power series expansions of trigonometric functions and orthogonal polynomials were used as fit functions, as well as a theoretically motivated function. The results will be compared with the diffraction pattern of the base function. Finally, the results will be faced with calculations on the basis of the popular Glatter method.

Inhalt

1. Einleitung

Im kernnahen Bereich verändert der Reaktordruckbehälter unter dem Einfluss der Neutronenbestrahlung seine Eigenschaften. Das Phänomen ist als Neutronenversprödung bekannt und insbesondere für Reaktoren vom Typ WWER von hoher sicherheitstechnischer Relevanz. Zur Vertiefung des Verständnisses und zur quantitativen Beschreibung der Zusammenhänge zwischen Gefüge, mechanischen Eigenschaften und der Strahlenbelastung wurden an WWER- Reaktordruckbehälterwerkstoffen Mikrostrukturuntersuchungen durchgeführt. Als besonders geeignet erwies sich dabei die Neutronenkleinwinkelstreuung. Dafür existieren Routineverfahren, mit denen die Anzahldichte, die Größenverteilung und der mittlere Radius der durch Bestrahlung erzeugten Strukturerscheinungen aus den Neutronenkleinwinkelstreuexperimenten bestimmt werden können. Die Strahlendefekte sind nanodisperse Inhomogenitäten mit Radien bis ca. 3 nm. Aus den experimentell gemessenen Streukurven werden dann, mit Hilfe existierender Auswertemethoden die Größenverteilungen bestimmt. Die existierenden Methoden dafür können in drei Gruppen eingeteilt werden. Zur ersten Gruppe gehören alle Verfahren, bei denen die Verteilungsfunktion durch die Annahme einer speziellen Funktion, z. B. logarithmische Normalverteilung, ermittelt wird. In der zweiten Gruppe werden allgemeine numerische Methoden zur Bestimmung der Größenverteilung genutzt (z. B. Glatter 1980). Die dritte Gruppe nutzt analytische Näherungen zur Bestimmung der Größenverteilung (Snyder et al. 1999).
In der folgenden Arbeit werden Auswerteansätze, die in die ersten zwei Gruppen gehören, vorgestellt und analysiert. Die Listings, der dafür geschriebenen lauffähigen FORTRAN Programme, sind im Anhang B angegeben.

2. Neutronenkleinwinkelstreuung

Die Kleinwinkelneutronenstreuung (SANS) ist die diffuse elastische Streuung bei kleinen Streuwinkeln ($\theta < 5°$) an Strukturen, deren Größenskala im Bereich 0.5 nm bis 100 nm liegt. Die Beschränkung auf kleine Winkel ist notwendig, um das zunehmend störende Streubild der Neutronen am Gitter auszublenden. Die Streuung von Neutronen erfolgt durch die Wechselwirkung mit dem Kern oder durch die Wechselwirkung mit dem magnetischen Moment eines Atoms. Das heißt, durch die einfallende Welle werden die Atome angeregt, infolge dessen strahlen sie Sekundärwellen (Kugelwellen) ab, die sich überlagern und das strukturabhängige Streubild erzeugen. Aus dem Betragsquadrat der resultierenden Streuamplitude aller Sekundärwellen erhält man für homogene Teilchen, die in ein umgebendes Medium eingebettet sind, für den makroskopisch differenziellen Streuquerschnitt die Form (z. B. Lindner and Zemb 2002):

$$\frac{d\sigma}{d\Omega}(q) = \eta \frac{N_S}{V_g} \int_0^{r_N} v(r)^2 \mid f(q,r) \mid^2 w(r) dr\,, \quad f(q,r) = \frac{1}{v(r)} \int_{v(r)} e^{-i\vec{q}\cdot\vec{r}}\, d^3\vec{r} \tag{1}$$

dabei ist q der Betrag des Streuvektors, der aus der Differenz der Wellenzahlvektoren $\vec{k}_0$ und $\vec{k}_s$ des einfallenden und des gestreuten Stahls, gebildet wird.

$$q = \mid \vec{k}_0 - \vec{k}_s \mid = \frac{4\pi}{\lambda} \sin\left(\frac{\vartheta}{2}\right) \tag{2}$$

ϑ ist dabei der Winkel zwischen beiden Vektoren und λ die Wellenlänge. Der differenzielle Streuquerschnitt liefert die Zahl der pro Zeit in den Raumwinkel $\delta\Omega$ gestreuten Neutronen in Bezug auf die ankommende Teilchenstromdichte:

$$\frac{d\sigma}{d\Omega}\delta\Omega = \frac{\textit{Zahl der pro Zeit in } \delta\Omega \textit{ gestreuten Teilchen}}{\textit{ankommende Teilchenstromdichte}},$$

die am Detektor gemessen werden. Die weiteren in (1) vorhandenen Parameter und Funktionen werden wie folgt bezeichnet:
V_g: Informationsvolumen (durchstrahltes Volumen)
N_S: Zahl der Streuobjekte (Cluster) in V_g
$v(r)$: Volumen des Streuteilchens mit dem Radius r (in nm)
$w(r)$: Verteilungsfunktion kugelförmiger Streuteilchen in Abhängigkeit vom Radius r (Größenverteilung)
$f(q,r)$: Einteilchenformfaktor
η: Streukontrast, $\eta = 6.4 \cdot 10^{-7}\,\mathrm{nm}^{-4}$ und $\eta_{mag} = 2.57 \cdot 10^{-7}\,\mathrm{nm}^{-4}$ sind z. B. die Werte für Eisen. $\frac{N_S}{V_g} w(r) dr = C(r, t_{fest}) dr$ ist dabei die Zahl der Teilchen bzw. Cluster mit dem Radius r im Volumen V_g. (Clusterdichte).

Wird angenommen, dass die vorliegenden Streuobjekte kugelsymmetrisch sind, dann ist das Integral des Einteilchenformfaktors analytisch gegeben, sein Betragsquadrat hat die Gestalt:

$$|f(q,r)|^2 = \frac{9(\sin(qr) - qr\cdot\cos(qr))^2}{(qr)^6} = \frac{9\pi}{2}\frac{(J_{3/2}(qr))^2}{(qr)^3}. \tag{3}$$

Ziel der Streuexperimente ist es nun, aus den gemessenen Streukurven die dazugehörigen Größenverteilungen *w(r)* der Streuobjekte zu ermitteln. Zur Lösung dieser Aufgabe wurden Fortran- Programme geschrieben, die das Minimum der Gauß'schen Fehlerquadratfunktion (Fehlerquadratmethode, $K = \eta N_S / V_g$)

$$\chi^2(a_0,\ldots,a_N) = \sum_{i=1}^{M}\left(\frac{d\sigma}{d\Omega}(q_i)_{\exp} - K\int_0^{r_{\max}} v(r)^2\,|f(q_i,r)|^2\; w(r;a_0,\ldots,a_N)dr\right)^2 = \text{Min} \quad \rightarrow \tag{4}$$

$$\sum_{i=1}^{M}\left(\frac{d\sigma}{d\Omega}(q_i)_{\exp} - K\int_0^{r_{\max}} v^2\,|f(q_i,r)|^2\; w(r;\mathbf{a})dr\right)\int_0^{r_{\max}} v^2\,|f(q_i,r)|^2\;\frac{\partial w(r;\mathbf{a})}{a_k}dr = 0,\ k = 0,\ldots,N.$$

für unterschiedliche Fit- Ansätze $w(r;\mathbf{a} = a_0,\ldots,a_N)$ berechnen. Der minimale Wert von χ^2 ist ein Maß für die Qualität des Fits. Damit die Anpassung (4) ein echtes Ausgleichsproblem ist, muss die Anzahl der Messpunkte *M* viel größer als die Zahl der Fit- Parameter *N* sein. Die Begründung der Fit- Ansätze und die damit erzielten Ergebnisse sind Gegenstand der folgenden Ausführungen.

2.1. Differentieller Streuquerschnitt und Größenverteilung

Mit dem kugelsymmetrischen Formfaktor und dem entsprechenden Volumen der Streukörper $v(r) = 4/3\pi\cdot r^3$ ergibt sich für den differenziellen Streuquerschnitt die Form:

$$\frac{d\sigma}{d\Omega}(q) = (4\pi)^2 K\int_0^{r_N}\frac{(\sin(qr) - qr\cdot\cos(qr))^2}{q^6}w(r)dr\,. \tag{5}$$

Jede physikalisch sinnvolle Größenverteilung *w(r)* ist bekanntlich formal als Fourierintegral darstellbar:

$$\begin{aligned} w(r) &= \int_0^{\infty} A(\omega)\cos(\omega\cdot r)d\omega + \int_0^{\infty} B(\omega)\sin(\omega\cdot r)d\omega \\ A(\omega) &= \frac{1}{\pi}\int_{-\infty}^{\infty} w(r)\cos(\omega\cdot r)dr \quad B(\omega) = \frac{1}{\pi}\int_{-\infty}^{\infty} w(r)\sin(\omega\cdot r)dr. \end{aligned} \tag{6}$$

Mit diesem allgemeinen Ansatz für die Größenverteilung erhält der Streuquerschnitt, nach einigen Umformungen, die Gestalt:

$$\frac{d\sigma}{d\Omega}(q) = \frac{4\pi^2 K}{q^7}\int_0^{\infty}(A(\omega)h_{\cos}(\omega,q) + B(\omega)h_{\sin}(\omega,q))d\omega,$$

mit (7)

$$\begin{pmatrix} h_{\cos}(\omega,q \\ h_{\sin}(\omega,q) \end{pmatrix} = \int_0^{2qr_m}\left(1+\frac{x^2}{4}-(1-\frac{x^2}{4})\cos(x)-x\cdot\sin(x)\right)\begin{pmatrix}\cos\left(\frac{\omega}{2q}x\right)\\ \sin\left(\frac{\omega}{2q}x\right)\end{pmatrix}dx.$$

Die Integrale $h_{\cos}(\omega,q)$ bzw. $h_{\sin}(\omega,q)$ sind analytisch leicht zu berechnen, ihre Funktionsausdrücke sind jedoch sehr lang. Zur Veranschaulichung des Funktionsverlaufs werden deshalb für drei *q*-Werte in Abb. 1 die Integrale der Kosinuskomponente graphisch dargestellt. Das Entscheidende ist, dass beide Integrale bei $\omega = 2q$ Maxima in den Schwingungsamplituden zeigen, und der größte Term dieses Integrals klingt für $\omega > 2q$ mit wachsendem ω wie $1/(\omega - 2q)$ ab. Alle anderen Beiträge gehen mindestens wie $1/(\omega - 2q)^2$ gegen Null.

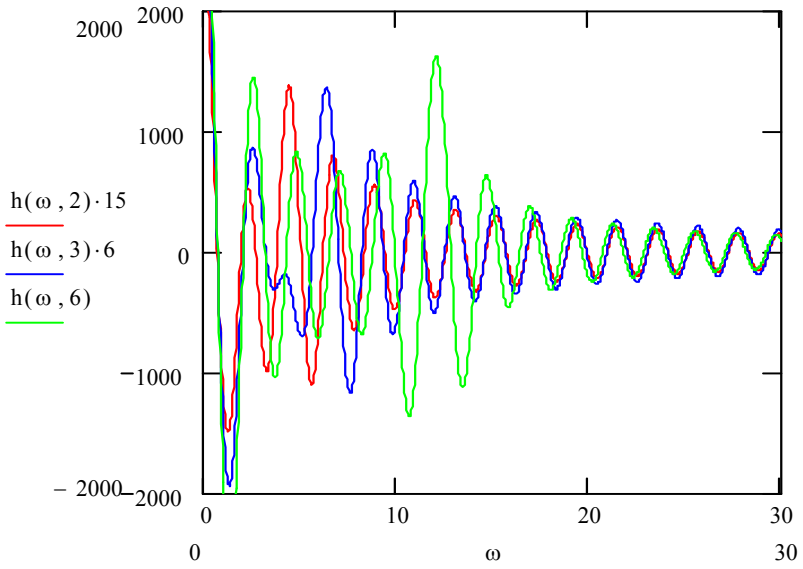


Abb. 1. Graphische Darstellung des Integrals $h_{\cos}(\omega,q)$ für *q*=2, 3 und 6.

Das heißt, bei der Auswertung der Kleinwinkelstreuung werden die „Schwingungsamplituden“ $A(\omega)$, $B(\omega)$ nur bis zu $\omega = 2q_{\max}$ richtig berechnet, wodurch die Rücktransformation nicht mehr vollständig möglich ist. Im folgenden Beispiel wird dieser Sachverhalt veranschaulicht. Die Dreiecksfunktion

$$w(r) = r\Phi(r_0 - r) \tag{8}$$

als ω- abhängiges Fourierintegral geschrieben hat die Gestalt

$$w(r,\omega) = r\left[\frac{Si(\omega\cdot r) + Si(\omega(r_0 - r))}{\pi} + \frac{\cos(\omega\cdot r) - \cos(\omega(r_0 - r))}{\pi\omega}\right],\ w(r) = w(r,\infty)\,, \tag{8a}$$

wobei *Φ(x)* die Heaviside- Funktion und *Si(x)* der Integralsinus (Abramovica i Stigan) ist. In der Abb. 2 wird diese Funktion für drei kleine ω_{max}- Werte graphisch dargestellt.

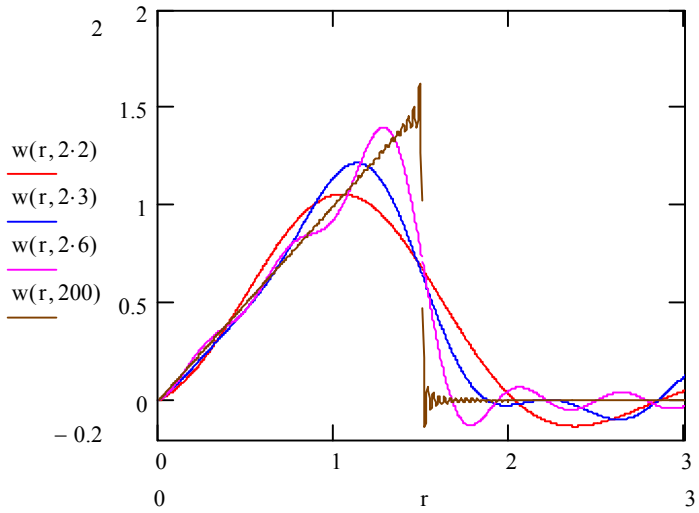


Abb. 2. Das Fourierintegral für kleine ω- Grenzwerte (ω_{max}=2*2, 2*3, 2*6*)*.

Wie zu erwarten, zeigt die Abb. 2, dass durch die fehlenden größeren Werte von ω starke Kurvenkrümmungen nicht wiedergegeben werden. Erst mit q_{max}=100 wäre die Ausgangkurve eindeutig zu identifizieren. Das heißt, mit der Kleinwinkelstreuung werden mit einem allgemeinen Ansatz nur Größenverteilungen richtig ermittelt, die als Fourierintegral mit der oberen Integralgrenze $\omega = 2q_{\max}$, statt $\omega \to \infty$, näherungsweise darstellbar sind. Das Gesagte gilt auch für die Fourierreihe, die für die praktische Rechnung geeigneter ist. Die Bestimmung ihrer Koeffizienten führt zur Lösung eines linearen Gleichungssystems und nicht wie beim Fourierintegral auf die Lösung von Integralgleichungen (siehe Gleichung (7)).

Die Größenverteilung im Intervall $0 \le r \le r_{\max}$ als Fourierreihe geschrieben lautet:

$$w(r,N) = a_0 + \sum_{i=1}^{N} a_i \cos\left(\frac{2\pi\cdot i}{r_{\max}} r\right) + b_i \sin\left(\frac{2\pi\cdot i}{r_{\max}} r\right). \tag{9}$$

Statt (9) können auch die Reihen:

$$w(r,N) = a_0 + \sum_{i=1}^{2N} a_i \sin\left(\frac{\pi\cdot i}{r_{\max}} r\right) \text{ bzw. } w(r,N) = b_0 + \sum_{i=1}^{2N} b_i \cos\left(\frac{\pi\cdot i}{r_{\max}} r\right) \tag{9a}$$

zur Approximation herangezogen werden. Die Bedingung $\omega = \frac{2\pi \cdot i}{r_{\max}} = 2q_{\max}$ begrenzt hier die sinnvollen Entwicklungskoeffizienten auf:

$$N = \text{größte ganze Zahl} \geq \frac{q_{\max} \cdot r_{\max}}{\pi}. \qquad (10)$$

Die Fourierreihe (9) für $w(r) = r\Phi(r_0 - r)$ hat die Koeffizienten:

$$a_0 = \frac{r_0^2}{2 \cdot r_{\max}},\ a_i = \frac{r_0}{\pi}\left(\frac{(\cos(\varpi \cdot i) - 1)}{\varpi \cdot i^2} + \frac{\sin(\varpi \cdot i)}{i}\right),\ b_i = \frac{r_0}{\pi}\left(\frac{\sin(\varpi \cdot i)}{\varpi \cdot i^2} - \frac{\cos(\varpi \cdot i)}{i}\right),\ \varpi = 2\pi \frac{r_0}{r_{\max}}.$$

Die Abb. 3 zeigt nun als Vergleich die Approximationsgenauigkeit für $w(r) = r\Phi(r_0 - r)$ mit dem Fourierintegral *w(r,2q_{max})* und der Fourierreihe *wr(r,N_q)*.

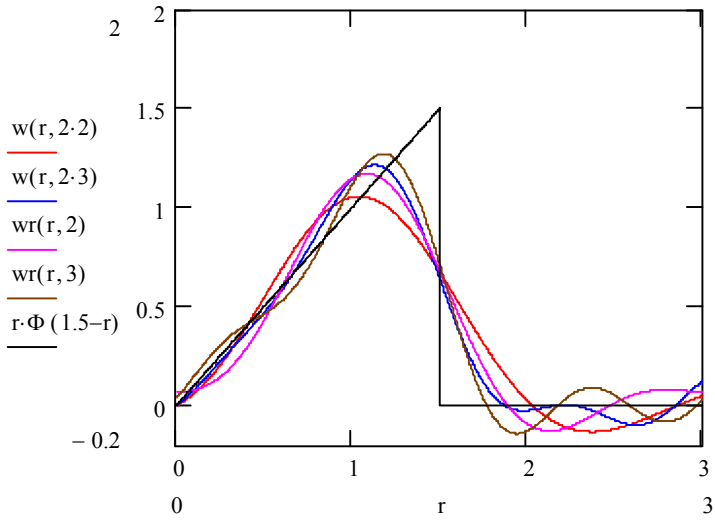


Abb.3 Fourierintegral *w(r,2q_{max})* und Fourierreihe *wr(r,N_q)* für q_{max}=2 und q_{max}=3 im Vergleich.

Der Vergleich zeigt, dass beide Ansätze annähernd gleichwertige Funktionsverläufe ergeben. Beim Fourierreihenansatz ist die gesuchte Funktion allerdings auf das vorgegebene Intervall $0 \leq r \leq r_{\max}$ beschränkt. Die Entwicklungskoeffizienten sind jedoch leicht zu bestimmen.

Die Zahl der Entwicklungskoeffizienten für die Fourierreihe kann auch zur Abschätzung des Polynoms mit dem höchsten sinnvollen Grad bei der Approximation durch Polynome dienen. Aus der Theorie der orthogonalen Funktionen ist nämlich bekannt (Sauer und Szabo 1968), dass die optimale Lösung $w(r;\mathbf{a}) = \sum_{i=0}^{N} a_i P_i(r)$ um die zu approximierende Funktion *w(r)* oszilliert, und zwar treffen sich *w(r;**a**)* und *w(r)* in mindestens *N+1*

Stellen. Diese Aussage trifft natürlich auch für die orthogonalen trigonometrischen Funktionen der Fourierreihe zu, in der die Kosinusfunktion $\cos(2\pi \cdot N)$ mit $2N$ die meisten Nullstellen aufweist. Polynome mit dem Grad $> 2 \cdot N$ sollten deshalb keine Verbesserung mehr in der Anpassung hervorrufen.

Glatter (Glatter 1980) verwendet als Basisfunktionen für die Größenverteilung N kubische B- Splines und erhält für die Zahl der durch die Streuformel bestimmbaren Koeffizienten die gleiche Beschränkung (Glatter and Kratky 1982):

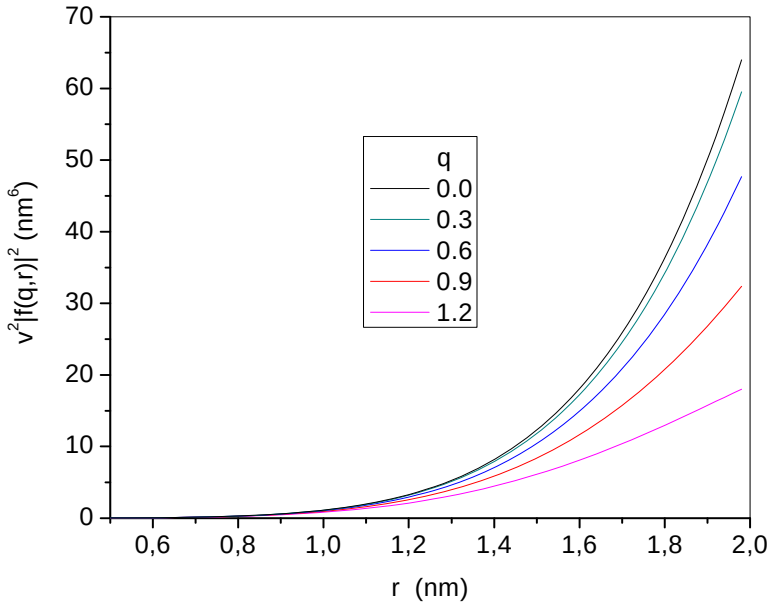
$$N_{\max} \leq \frac{2q_{\max}r_{\max}}{\pi} \tag{11}$$

Diese Bedingung gilt natürlich auch für die Anzahl der Stützstellen, wenn die Größenverteilung durch lineare Interpolation in N Intervallen der Breite Δr approximiert wird (Polygonzug).

Eine weitere, die Bestimmung der Größenverteilung einschränkende Eigenschaft des Integralkerns, Formfaktor mal Streuvolumenquadrat, wird am Kurvenverlauf dieser Funktion für kleine r-Werte deutlich (siehe Abb. 4):

$$\mathrm{v(r)}^2 \mid f(q,r) \mid^2 \approx r^6\left(1-\frac{(q\cdot r)^2}{5}+\frac{3(q\cdot r)^4}{5^2\cdot 7}-\frac{4(q\cdot r)^6}{3^3\cdot 5^2\cdot 7}+\ldots\right). \tag{12}$$

Der größte Term dieser Taylorentwicklung ist von 6-ter Potenz in r, d. h. nur für r>1 liefert der Integrand, Größenverteilung mal Integralkern (12), deutlich anwachsende Beiträge zum differenziellen Streuquerschnitt.



:

Abb. 4. Kurvenverlauf des Integralkerns der Streuformel für kleine Radien.

Der Kurvenverlauf des Integralkerns in Abb. 4 verdeutlicht dieses Verhalten optisch. Die Größenverteilung *w(r)* ist deshalb für Radien

$$r < 1 \tag{13}$$

als unsicher einzuschätzen. Bei der numerischen Auswertung der Streukurven mit den Programmen zeigt sich dieses Verhalten beim Ändern der Rechengenauigkeit. So verbesserten sich die Fits zu kleineren *r*-Werten hin, wenn mit doppelter Genauigkeit gerechnet wird.

Für den oberen Grenzradius gibt Bracewell (1965) die Bedingung:

$$q_{\min} r_{\max} < \pi \tag{14}$$

an. Diese Ungleichung folgt aus dem Sampling- Theorem, das besagt, dass eine Funktion *F(q)* durch eine begrenzte Anzahl von Datenpunkten beschrieben werden kann, unter der Voraussetzung, dass die Fouriertransformierte dieser Funktion in einem endlichen Intervall der Variablen *r* nicht Null ist (Glatter 1980). Zusammenfassend ist festzustellen, dass auf Grund der Bedingungen (10, 11, 13, 14) eine eindeutige Bestimmung der Größenverteilung mit der Kleinwinkelstreuung nicht möglich ist. Wie sich diese Bedingungen auf die Form der Kurven auswirken, wird im nächsten Abschnitt an Testbeispielen veranschaulicht.

3. Fit- Ansätze

3.1. Die Modellfunktion

Es hat sich gezeigt, dass sich die Ergebnisse der Modellrechnungen zur Bildung von Leerstellen- und Kupferclustern (siehe Anhang A) sehr gut mit der Funktion

$$w(r;\mathbf{p}) = a \frac{r^{\alpha}}{\left[1 + \exp(\beta(r - r_0))\right]}, \tag{15}$$

durch die Bestimmung der vier konstanten Parameter $\mathbf{p} = (a, \alpha, \beta, r_0)$, beschreiben lassen. Hierbei verschiebt der Parameter r_0 die Funktion entlang der r- Achse**.** Die Flankensteilheit der Kurve wird durch die Parameter α, β bestimmt. Liegt eine relativ steile rechte Flanke vor, dann ist der mittlere Streuradius in guter Näherung durch $\bar{r} = (\alpha + 1) r_0 / (\alpha + 2)$ gegeben. Ohne nennenswerten Fehler wird durch die Funktion (15) auch die Form der logarithmischen Normalverteilung und die Verteilung der Lifshitz-Slyozow-Wagner (LSW)- Theorie (Landau, L. D.; Lifshitz, E. M. 1987) wiedergegeben, die im Rahmen der neutroneninduzierten Clusterbildung ebenfalls von Interesse sind. In der Abb. 5 wird dieses Fit- Fähigkeit veranschaulicht

Bemerkung: Obwohl die logarithmische Normalverteilung als Fit- Funktion zur Auswertung der Streukurven reichlich genutzt wird, ist sie kein geeigneter Ansatz zur Beschreibung der Clusterbildung in Metallen, da sie für die Größen- und Teilchenverteilung

die gleiche Form liefert, die in Wirklichkeit völlig unterschiedlich sind (vergleiche A2, A3 im Anhang A). Das heißt, aus der Anzahlverteilung

$$w(n) = \frac{1}{\sigma\sqrt{2\pi}\cdot n}\exp\left(-\frac{(\ln(n)-\mu)^2}{2\sigma^2}\right)$$

erhält man durch die Transformation auf den Teilchenradius $r = r_0\sqrt[3]{n} \rightarrow dn = 3r^2/r_0^3 dr$ die gleiche *r*-Abhängigkeit für die Größenverteilung

$$w(r) = \frac{1}{\sigma_r\sqrt{2\pi}\cdot r}\exp\left(-\frac{(\ln(r)-\mu_r)^2}{2\sigma_r^2}\right), \quad \text{mit}: \ \sigma_r = \sigma/3, \ \mu_r = \ln(r_0) + \mu/3\,.$$

Mit der logarithmischen Normalverteilung als Fit- Funktion ist auch die Anpassung an so steile Flanken, wie sie die Modellverteilungen zeigen, nicht möglich.

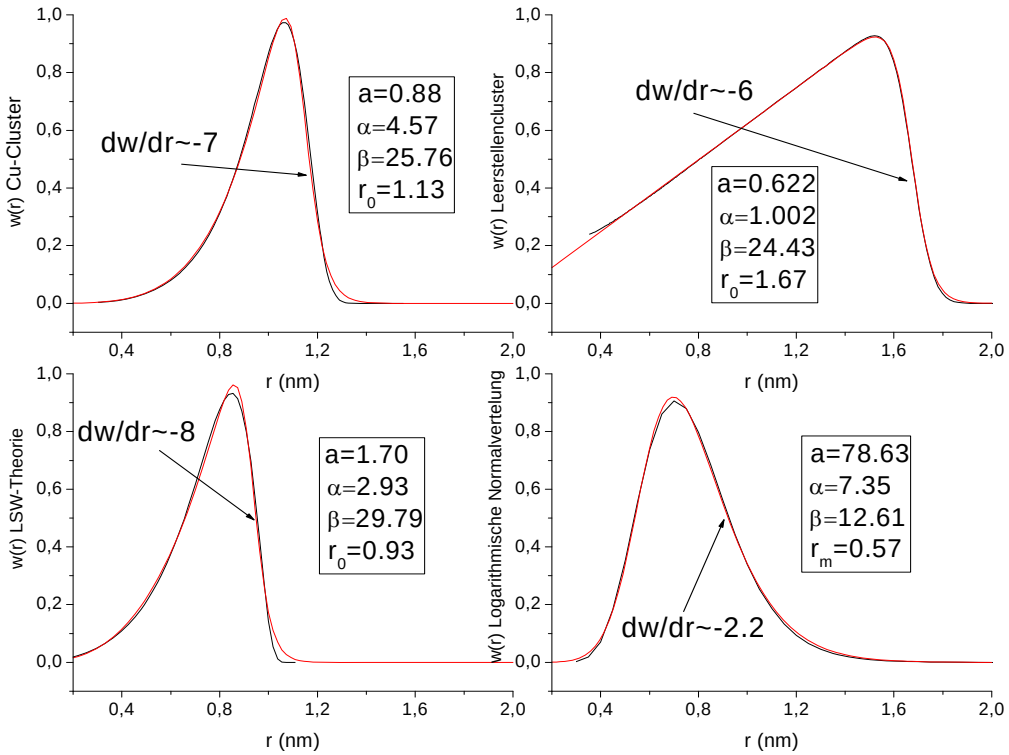


Abb. 5. Typische Größenverteilungen der Clustertheorie mit den roten Fit- Kurven der Funktion $w(r;\mathbf{p}) = a\cdot r^{\alpha}/\left[1+\exp(\beta(r-r_0))\right]$.

Auf Grund der guten Fit- Eigenschaft von $w(r;\mathbf{p})$ an die Modellrechnungen liegt es nahe, diese Funktion zur Auswertung der Streuexperimente zur Neutronenversprödung

heranzuziehen. Vorteilhaft ist dabei noch, dass die Abhängigkeit zwischen den Fit- Parametern und den Modell- Parametern näherungsweise bekannt ist. Ein großer Nachteil ist jedoch die nichtlineare Abhängigkeit dieser Funktion von den zu bestimmenden Parametern, d.h. die Fit- Prozedur konvergiert nur mit geeigneten Startwerten, deren Ermittlung sich gelegentlich als recht schwierig herausstellte. Zur Berechnung des Fehlerquadrates (4) muss das zweite Gleichungssystem, die Extremwertbedingung, in diesem Fall linearisiert werden. Die Berechnung der dabei auftretenden Integrale,

$$\int_0^{r_{\max}} r^6 \mid f(q,r) \mid^2 w(r;\mathbf{p}) dr, \quad \int_0^{r_{\max}} r^6 \mid f(q,r) \mid^2 \frac{\partial w(r;\mathbf{p})}{\partial p_i} dr, \qquad i = 1,\ldots,4 ,$$

ist nur numerisch möglich und erfolgt nach der Simpsonschen Regel.

3.2. Reihen von Orthogonalfunktionen

Das obige, nichtlineare Problem entfällt, wenn man die Verteilungsfunktion als Summe linear unabhängiger Basisfunktionen oder besser als Summe von Orthogonalfunktionen darstellt:

$$w(r;\mathbf{a}) = \sum_{i=0}^{N} a_i \cdot \varphi_i(r / r_{\max}) . \tag{16}$$

Der Vorteil der Orthogonalfunktionen besteht darin, dass die Entwicklungskoeffizienten direkt durch Integration, ohne die Lösung eines Gleichungssystems, zu bestimmen sind. Diese Eigenschaft geht allerdings bei der Lösung von (4) verloren, da hier die Summe der *q*- abhängigen Integrale

$$J_j(q) = \int_0^{r_{\max}} r^6 \mid f(q,r) \mid^2 \varphi_j(r / r_{\max}) dr \tag{17}$$

$$\rightarrow \chi^2(a_0,\ldots,a_N) = \sum_{i=1}^{M} \left(\frac{d\sigma}{d\Omega}(q_i)_{\exp} - \sum_{j=0}^{N} a_j J_j(q_i) \right)^2 = \text{Min}$$

anzupassen ist. Trotzdem ist es sinnvoll, derartige Basisfunktionen zu nutzen, da ihre funktionalen Gesetzmäßigkeiten in Funktionstafeln vorliegen und damit jede weitere Rechnung vereinfachen. Weiterhin ist die Genauigkeit bei der praktischen Rechnung mit den Orthogonalpolynomen größer im Vergleich zu der entsprechenden Potenzreihe. Mit den Basisfunktionen $J_i(q)$ könnte man auch, mit Hilfe des Schmidtschen Orthogonalisierungsverfahrens, im *q*- Raum Orthogonalfunktionen erzeugen. Die Entwicklung solch einer Fit- Prozedur ist aber erst dann sinnvoll, wenn sich eine geeignete Reihenentwicklung ergibt, die die typischen Effekte des Streumechanismus besonders gut erfasst und damit zur Auswertung von Experimenten geeignet ist. Als Bestimmungsgleichung für die Koeffizienten a_i erhält man ein $N+1$-dimensionales lineares Gleichungssystem.

Im Rahmen der Approximation der Größenverteilung durch eine Linearkombination bekannter Basisfunktionen wurden die Fourierreihen (9), (9a) und die Entwicklung nach

Legendreschen Polynomen, die in das Intervall $0 \le r \le r_{\max}$ transformiert wurden, in Programmen realisiert. Die transformierten Legendreschen Polynome haben die Gestalt:

$$
\begin{aligned}
&P_0(x) = 1,\ P_1(x) = \sqrt{3}(1-2x),\ P_2(x) = \sqrt{5}(1-6x+6x^2) \\
&P_3(x) = \sqrt{7}(1-12x+30x^2-20x^3),\ P_4(x) = \sqrt{9}(1-20x+90x^2-140x^3+70x^4) \\
&P_5(x) = \sqrt{11}(1-30x+210x^2-560x^3+630x^4-252x^5) \\
&P_6(x) = \sqrt{13}(1-42x+420x^2-1680x^3+3150x^4-2772x^5+924x^6) \\
&P_7(x) = \sqrt{15}(1-56x+756x^2-4200x^3+11550x^4-16632x^5+12012x^6-3432x^7).
\end{aligned}
\tag{18}
$$

Diese Polynome erhält man auch, wenn im Intervall [0,1] das Schmidtsche Orthogonalisierungsverfahren auf die linear unabhängigen Funktionen *1, x, x^2,..., x^7* angewendet wird. Das Integral der Streuformel (5) ist mit den trigonometrischen Funktionen und auch mit den Polynomen (18) analytisch lösbar. Die Fit- Programme mit diesen Basisfunktionen kommen deshalb ohne jede numerische Näherung aus.

3.3 Die Glatter-Methode

Zum Vergleich mit einem bewährten, weit verbreiteten Auswerteverfahren wird im fünften Programm die sogenannte Glatter- Methode (Glatter 1977, 1980, Glatter and Kratky 1982, Lindner and Zemb 2002) realisiert. Glatter verwendet für die Basisfunktionen $\varphi_i(r/r_{\max})$ in der Reihenentwicklung (16) kubische B (Basis)- Splines. Da wegen (11) mit wachsendem N die Entwicklung (16) immer kräftiger um die tatsächliche Funktion oszilliert, (die höheren Entwicklungskoeffizienten sind nicht mehr brauchbar oder die Ausgleichsbedingung $M>N$ ist verletzt), fügt Glatter „Stabilisierungsgleichungen“ in die Fit- Prozedur ein. Diese Stabilisierung versteht man am besten, wenn man sich an das Verhalten von Oszillatoren erinnert, die durch Federn (Federkonstante λ) miteinander verbunden sind.

$$m_i \ddot{y}_i + k(y_i - y_i^0) = \lambda(y_{i-1} - 2y_i + y_{i+1}).$$

Eine Extremalform dieser Gleichung ist bekanntlich

$$\frac{1}{2}\sum_i \int_{t_a}^{t_b} \left[m_i \dot{y}_i^2 - k(y_i - y_i^0)^2 + \lambda(y_i - y_{i-1})^2 \right] dt = Min .$$

Durch die Kopplung können die einzelnen Teilchen im stationären Zustand nicht mehr ihre Gleichgewichtslagen y_i^0 einnehmen. Die Verschiebung zu den neuen Gleichgewichtslagen erfolgt in Abhängigkeit vom k/λ- Verhältnis. Für $k/\lambda \to 0$ nähern sich alle Massenpunkte einer Geraden. Diesen Glättungseffekt der Kopplung führt Glatter als Nebenbedingung in (4) ein:

$$\chi^2_{neu}(y_0,\ldots,y_M) = \chi^2(y_0,\ldots,y_M) + \lambda\sum_{i=1}^{M}(y_i - y_{i-1})^2 = Min , \tag{19}$$

dabei sind die y_i jetzt die Stützstellen der Splinfuktionen. Diese „Zwangskraft“ ermöglicht allerdings nur die kontinuierliche Darstellung der Fit- Kurve an fast beliebig vielen Stützstellen ($N>M$ ist möglich). Der geeignetste λ- Wert wird an Testkurven ermittelt, indem man das kleinste λ bestimmt, das das Oszillieren der Kurve unterdrückt. Erstaunlich dabei ist, dass dann deutlich höhere Beträge von λ die Fit- Kurve wenig ändern, obwohl das Fehlerquadrat (4) zunimmt. Die Annäherung der Anpassung an die wirkliche Verteilung verbessert diese Nebenbedingung nicht. Den optimalen Fit mit Splines erhält man, wenn die Zahl der Stützstellen nach (11) begrenzt wird. Im eigenen Programm werden statt der Splinefunktionen, die Werte zwischen den Stützstellen durch lineare Funktionen interpoliert. Dies sollte zu keinem nennenswerten Unterschied im Fit- Ergebnis führen, da durch die Integration über das Produkt, „Polygonzug“ mal Formfaktor, der Wirkungsquerschnitt $d\sigma / d\Omega$ ohnehin geglättet wird. Ob der Glättungseffekt der Integration noch durch die Glättungswirkung der Splinefunktionen verbessert wird, ist fraglich.

Glatter selbst, bezeichnet λ als Lagrangefaktor oder Stabilitätsparameter. Würde man als Nebenbedingung $\sum_{i=1}^{M}(y_i - y_{i-1})^2 = L$ vorgeben, dann wäre (19) ein Extremwertproblem mit Nebenbedingung in Lagrangescher Formulierung (Methode des Lagrangeschen Multiplikators).

4. Testrechnungen und Ergebnisanalyse

Zur Einschätzung und zum gegenseitigen Vergleich wurden diese Programme, mit den fünf unterschiedlichen Fit- Ansätzen, an theoretisch berechneten Streukurven getestet. Der Vorfaktor $(4\pi)^2 K$ wurde dabei gleich 1 gesetzt.

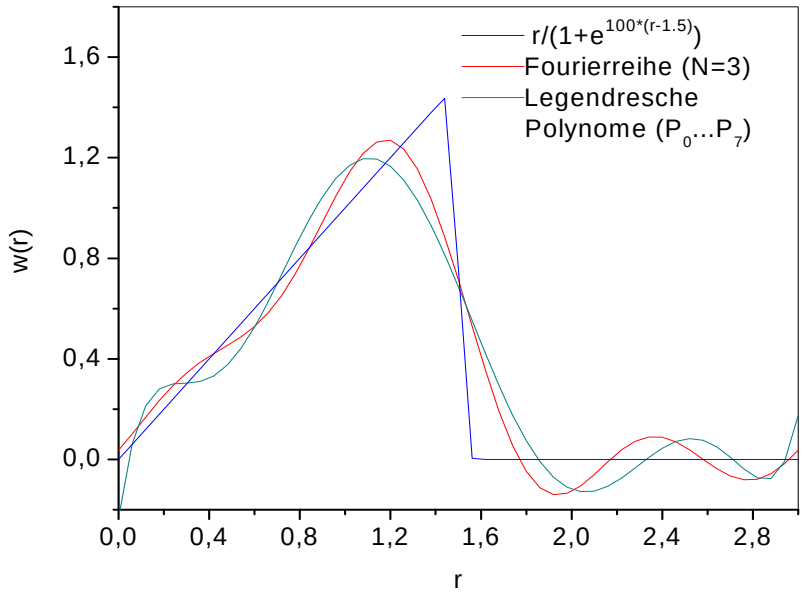


Abb. 6. Approximation der Dreiecksfunktion (8) durch Fourierreihe und Legendresche Polynome mit der begrenzten Anzahl von Entwicklungskoeffizienten.

4.1 Dreiecksfunktion als Größenverteilung

Im ersten Testbeispiel wird mit den Programmen die Streukurve ausgewertet, die die Dreiecksfunktion (8) erzeugt. In der Abb. 6 wird vorher zum Vergleich die Anpassung der Fit- Funktionen an (8) mit den wenigen, abgeschätzten relevanten Koeffizienten dargestellt (q_{max}=*2.8,* r_{max}=*3.3*). Mit (15) kann man (8) natürlich beliebig genau approximieren ($\beta \to \infty$).

Das Ergebnis der Fits auf der Grundlage der Fehlerquadratmethode (4) bzw. (19) zeigt die Abb. 7. Am besten wird der Verlauf der Ausgangsfunktion mit dem Ansatz (15) ermittelt. Das ist nicht verwunderlich, da es mit den vier Parametern möglich ist, diese Funktion hervorragend zu approximieren. Die anderen Fits zeigen deutlich die erwarteten Abweichungen. Die Anzahl der Schnittpunkte zwischen den Fit- Kurven der Fourier- bzw. Polynomentwicklung und der Ausgangsfunktion zeigen deutlich, dass die Fehlerfunktion (4) nur für die abgeschätzte, begrenzte Anzahl von Entwicklungskoeffizienten das Minimum annimmt (siehe Abb.3). Die Wirkung der Nebenbedingung bei der Glatter-Methode ist bei diesem Beispiel besonders schön an der senkrecht auf die Ordinate auftreffende Kurve zu erkennen. Auf das obige physikalische Bild zurückgreifend, heißt das, für die kleinen *r*-Werte (Abb. 4) geht die Verschiebungsstärke *k* gegen Null, was die Anordnung der Fit- Punkte auf einer geraden Linie zur Folge hat. Wird der Wert der ersten Stützstelle in der Nebenbedingung auf Null gesetzt, dann fällt die Fit- Kurve steil zur Abszisse hin ab (gestrichelte Kurve).

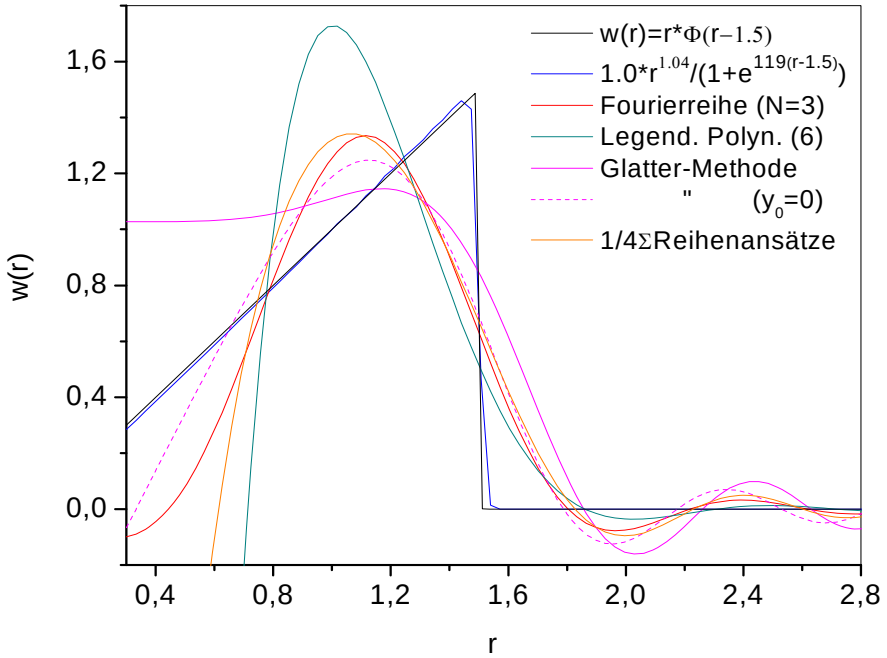


Abb. 7. Die mit den vier Ansätzen ermittelten Größenverteilungen.

Aus diesem Verhalten wird besonders deutlich, dass die ermittelte Verteilung für kleine Radien völlig falsch sein kann. Die Kurvenverläufe im Beispiel sind schon für Radien $r < 1\,(\mathrm{nm})$ unbrauchbar. Die Anstiege der Fit- Funktionen an der Stelle r=1.5 variieren geringfügig um den Wert $w'(r_0) \approx -2.4$. Dies ist ein Hinweis dafür, dass mit einer Reihenentwicklung steilere Flanken in der Größenverteilung mit der Kleinwinkelstreuung nicht mehr nachweisbar sind.

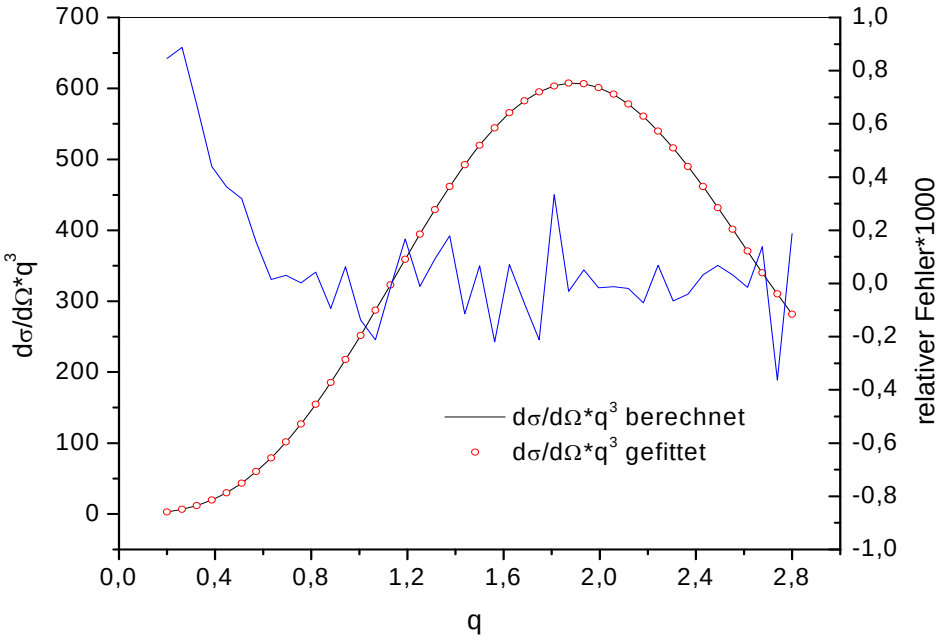


Abb. 8. Die Fit- Qualität für den Fall der Legendreschen Polynome.

Zu erwähnen ist noch, dass bei allen Testrechnungen die Kurve des modellierten Streuquerschnitts und die vorgegebene Streukurve optisch nicht zu unterscheiden sind. Stellvertretend für alle Fits wird diese Aussage zur Genauigkeit der Anpassung in der Abb. 8, am Beispiel der Entwicklung nach den Legendreschen Polynomen, veranschaulicht. Die Polynomreihe ist in diesem Beispiel der schlechteste Fit an die Ausgangskurve.

Teilt man in (4) die Differenz im Fehlerquadrat durch den statistischen Fehler der Messung, dann kann man mit der Anpassung der experimentellen Daten durch die theoretischen Werte in etwa zufrieden sein, wenn der minimale Wert von χ^2 der Anzahl der Freiheitsgrade des Fits entspricht. Diese ist als die Differenz der Anzahl der experimentellen Datenpunkte und der Anzahl der Fitparameter definiert. Selbst mit einem viel kleineren Rechenfehler, statt einem Messfehler, ist diese Bedingung im Beispiel sehr gut erfüllt. Das heißt, schon am ersten Beispiel wird ersichtlich, dass diese schöne Regel nicht auf die Größenverteilungsfunktion übertragbar ist, denn Ausgangsfunktion und Fit-Funktion weichen ersichtlich deutlich voneinander ab. Die Berechnung der Fehlerquadratsumme zwischen der Ausgangsfunktion und der Fit- Funktion als Genauigkeitsmaß hat auch wenig Sinn, da dieser Wert sehr stark von der Wahl der Summationsgrenzen abhängt.

4.2 Trapezverteilung

Im zweiten Beispiel werden mit einer Trapezverteilung die Fit- Programme geprüft. Das Ergebnis der Anpassungen an diese Streukurve zeigt die Abb. 9. Es ist ersichtlich, dass alle Fit- Ansätze die relativ steile, rechte Flanke der Funktion ($w' = -2.5$) gut wiedergeben. Die Maximalwerte der Fit- Funktionen sind jedoch um rund 15% zu hoch. Wie im vorangegangen Beispiel ist der Kurvenverlauf aller Funktionen auch hier im Bereich $r \leq 1\,\mathrm{nm}$ unkorreliert. Nicht richtig erfasst wird auch die Breite der Verteilung. Mit (15) ist es natürlich nicht möglich, diese Trapezfunktion darzustellen. Überraschend ist jedoch, dass die allgemeinen Reihen kaum bessere Anpassungen liefern. Die Trapezverteilung wurde durch die Fourierreihe (9) mit N=40 Entwicklungsgliedern dargestellt. Nur wenn man in dieser Reihe N=3 Glieder berücksichtig und damit die Streukurve berechnet, erhält man im Radiusbereich über 0.9 nm die identische Kurve zurück.

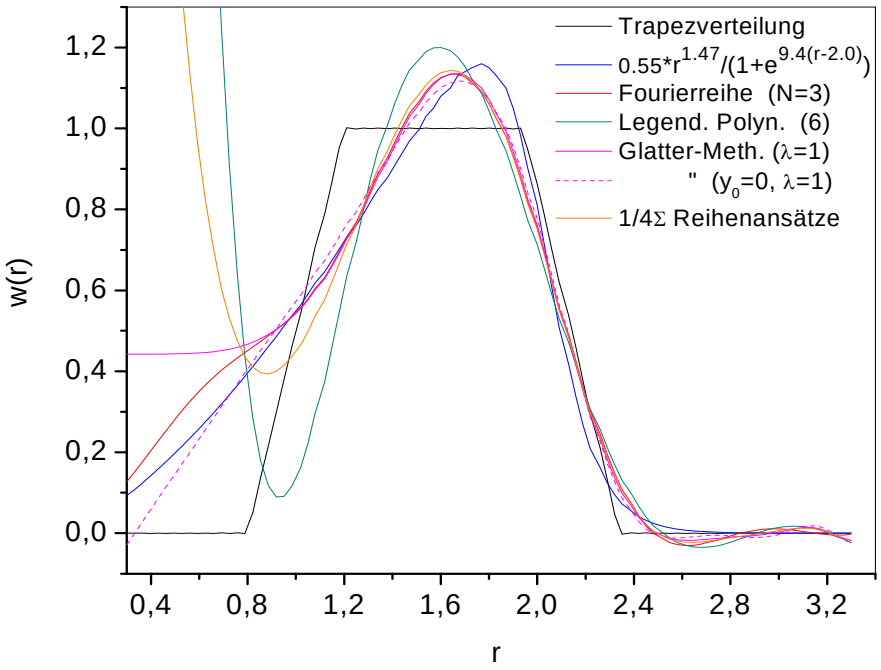


Abb. 9. Anpassungen an den Streuquerschnitt einer Trapezverteilung.

4.3 Bimodale Verteilung

An der folgenden Verteilung wird getestet, welche Informationen die Fits liefern, wenn die Ursache der Streuung durch die Überlagerung zweier unterschiedlicher Größenverteilungen gegeben ist (bimodale Verteilung). In der Praxis kann sich diese Art der Verteilung sehr schnell, durch das Vorhandensein zweier Ausscheidungsprozesse, einstellen.

Aus der Streukurve dieser etwas komplexeren Funktion (Abb. 10) wird nur der Anstieg der rechten Flanke der Verteilung akzeptabel wiedergegeben. Der Anstieg aller Ver-

teilungen im Intervall $0.8 < r < 1.3$ ist ein Hinweis für eine Zunahme der Konzentration im Bereich der kleineren Radien, eine Aussage über die Form dieses Maximums ist jedoch nicht möglich. Die Mittelung über die vier Reihenansätze gibt die Ausgangskurve, natürlich bis auf den Bereich $r < 1$ nm, am besten wieder. Die Aussagen zum Fit mit (15) im vorangegangenen Beispiel sind auch hier zutreffend. Natürlich ist es durch die Überlagerung zweier Verteilungen der Gestalt (15) im Prinzip möglich, für diese bimodale Verteilung einen deutlich besseren Fit anzugeben. Ohne Hinweise zur Lage der Extremwerte wird jedoch hier die Bestimmung des größeren Startvektors mit Sicherheit zum Problem.

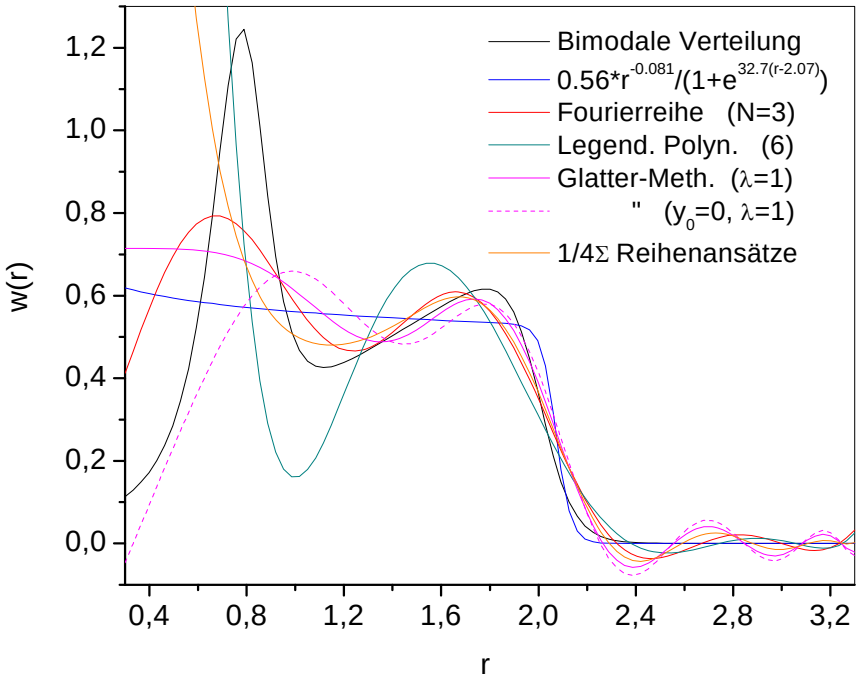


Abb. 10. Die χ^2-Fits an den Streuquerschnitt der bimodalen Verteilung

4.4 Logarithmische Normalverteilung

Die oft als Fit- Funktion verwendete, logarithmische Normalverteilung (z. B. Solt et al. 1999, Eberbeck and Bläsing 1999, Wirth et al. 1999, Gokhman et al. 2000, Böhmert et al. 2003) dient als Ausgangsfunktion für die folgenden zwei Tests:

$$w(r,\sigma,r_0) = \frac{1}{\sigma\sqrt{2\pi}\cdot r}\exp\left(-\frac{(\ln(r/r_0))^2}{2\sigma^2}\right). \qquad (19)$$

Im ersten Fall wird mit den Programmen die Streukurve $d\sigma(q)/d\Omega$ ausgewertet, die die Größenverteilung $w(r,0.25,0.75)$ hervorruft. Die Lage und Form dieser Verteilung ähnelt vielen Größenverteilungsfunktionen, die mit Hilfe der Glatter- Prozedur aus gemessenen Streukurven berechnet wurden (Ulbricht et al. 2006, Ulbricht 2005, Gokhman et al. 2000). Das Resultat der Anpassungen an die Streukurve dieser Funktion ist im folgenden Diagramm abgebildet.

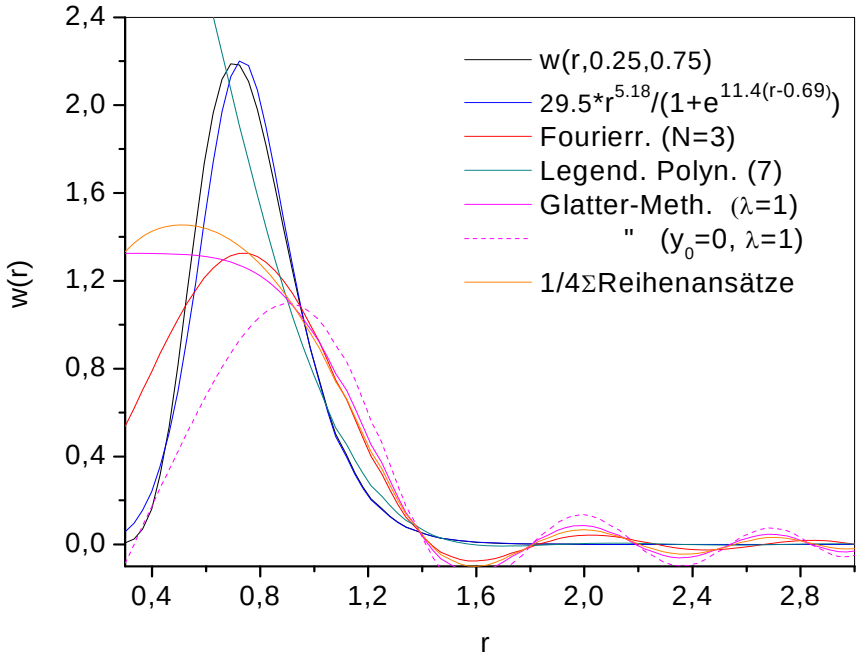


Abb. 11. χ^2- Fits an den Streuquerschnitt der logarithmischen Normalverteilung $w(r,0.25,0.75)$.

Wie eingangs festgestellt, gelingt es mit (15) die logarithmische Normalverteilung sehr gut zu approximieren (siehe Abb. 5). Es ist klar, dass sich diese Anpassungsfähigkeit auch bei der Minimierung des Systems (4) vorteilhaft auswirkt. Wie in den vorangegangen Beispielen, zeigt auch hier die Entwicklung nach den Legendreschen Polynomen im Bereich kleiner *r*-Werte drastische Abweichungen von der Ausgangsfunktion. Es ist wiederholt festzustellen, dass die Werte der Reihenentwicklungen für Radien unter einem Nanometer deutlich auseinandergehen. Die Ursache dafür ist natürlich, der eingangs angeführte, drastische Abnahme der Streudichte im Bereich kleiner *r*- Werte (Abb. 4). Die Realisierung der Bedingung $w(r=0)=0$ bei der Glatter- Methode führt hier zu einer leichten Verschlechterung des Fits im Bereich der stabilen Anpassung. Da es nicht möglich ist, die Verteilung im Bereich der kleinen Radien zu bestimmen, kann auch auf diese Festlegung verzichtet werden.

Die Situation bessert sich, wenn die nachzuweisenden Cluster im Bereich größerer Radien angesiedelt sind. Das zeigen die Fits an die Streukurve der nach rechts versch-

benen logarithmischen Normalverteilung ($\sigma = 0.25$, $r_0 = 1.5$, Abb. 12). Auch dieses Bild der Verteilung findet man als Ergebnis von Streumessungen an Reaktorstählen in den oben zitierten Arbeiten. Die in den zitierten Arbeiteten veröffentlichten Verteilungen unterscheiden sich im Allgemeinen etwas von den Abbildungen dieser Arbeit, da in ihnen gewöhnlich die Untergrundstreuung enthalten ist, und es wird auch oft, statt der Größenverteilung die Form der radiusbezogenen „Streuvolumendichte“ $\propto r^3 w(r)$ [vol%/nm] abgebildet.

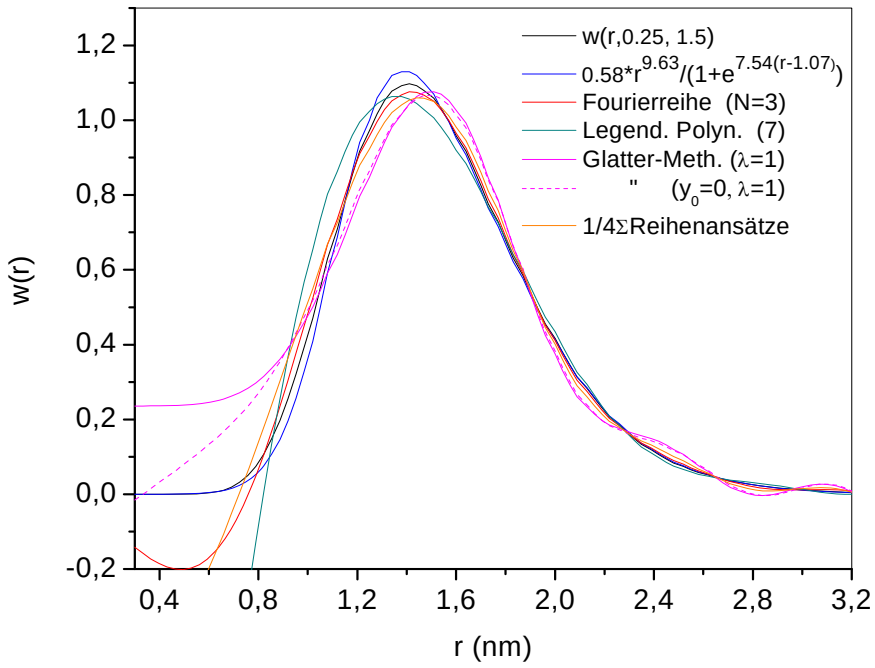


Abb. 12. χ^2- Fits an den Streuquerschnitt der logarithmischen Normalverteilung *w(r,*0.25,1.5*)*.

Alle vier Fit- Ansätze geben die Ausgangsfunktion sehr gut wieder. Auch damit werden die eingangs angegebenen Bedingungen bestätigt, denn diese Form der Verteilung erhält man, in guter Näherung, schon mit wenigen Entwicklungskoeffizienten in den angesetzten Reihen. Dass die logarithmische Normalverteilung selbst, als Fit- Funktion, sehr gut funktioniert, ist auch auf diese Tatsache zurückzuführen. Dieses Beispiel macht jedoch deutlicht klar, dass die alleinige Prüfung der Fit- Programme an einer glücklich gewählten Testfunktion zur Überbewertung der ermittelten Größenverteilung führen kann.

4.5 Fourierreihen im Vergleich

Da es durch kleine Änderungen im Fit- Programm der Fourierreihe (9) möglich ist, die Koeffizienten der Reihenentwicklungen (9a) zu berechnen, lag es nahe, auch diese Ansätze auf ihre Eignung zur Ermittelung der Größenverteilung zu prüfen. Dabei zeigte

sich, dass sich die Fit- Kurven der Fourierreihen (9a), wie bei den Legendreschen Polynomen, im Bereich kleiner *r*-Werte völlig unkorreliert verhalten. Im anpassungsfähigen Bereich werden die Testfunktionen jedoch etwas schlechter als durch die Fourierreihe (9) wiedergegeben. In der Abb. 13 sind diese Unterschiede dargestellt. Die Ursache für die Abweichungen in der Approximationsgenauigkeit bei den Reihenentwicklungen ist, wie schon festgestellt, auf die wenigen gültigen Entwicklungskoeffizienten zurückzuführen, und außerdem gehen die dazugehörigen Integrale $\int_0^{r_{max}} r^6 \,|\, f(q,r) \,|^2 \; \varphi_i(r / r_{max}) dr$, abhängig von der jeweiligen Basisfunktion $\varphi_i(r / r_{max})$ unterschiedlich gewichtet in die Bestimmungsgleichungen der a_i bzw. b_i ein. Bei einem anderen Formfaktor, z. B. dem Formfaktor zylindrischer Teilchen, könnte die Genauigkeit der Anpassung für die einzelnen Reihen völlig anders ausfallen. Die Minimierung von (4) mit unterschiedlichen Reihenansätzen ist schon dadurch sinnvoll. Weiterhin erlauben es nur die Fit- Ergebnisse verschiedener Reihen, den ungültigen Bereich der Größenverteilung für die kleinen *r*-Werte, durch das starke Auseinanderdriften der Kurvenverläufe, ziemlicht sicher einzugrenzen

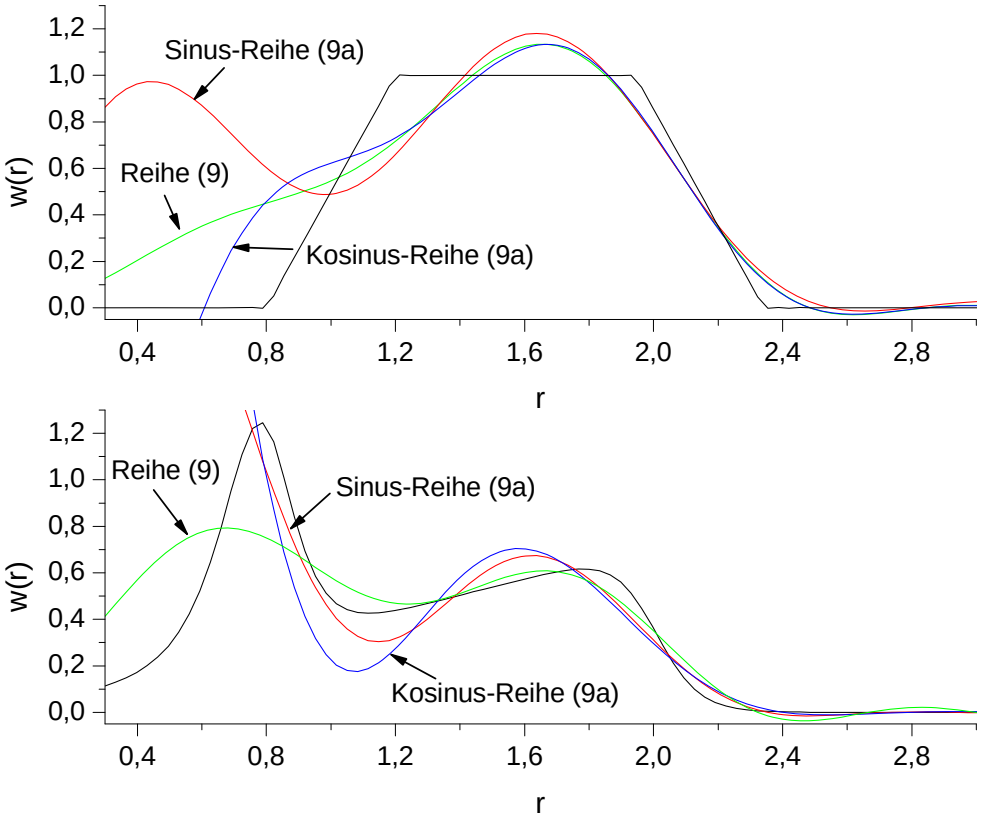


Abb. 13. Die Fourierreihen im Vergleich.

Wird einmal von der nichtlinearen Anpassung abgesehen, kann man sagen, dass die Fourierreihe (9) als allgemeiner Ansatz, die gewählten Testfunktionen in der Streuformel am besten „erkennt“. Im Bereich der stabilen Anpassung ist der mittlere Funktionswert aller

Reihen wahrscheinlich die sicherste Annäherung an die Größenverteilung. Für die Validierung von Modellansätzen durch das Experiment, bietet sich natürlich die Verteilung (15) an. Die damit verbundene, lästige Suche nach geeigneten Startwerten bei einer routinemassigen Anwendung dieser Prozedur wird durch die Levenberg- Marquardt Methode (Press, W. H. et al. 1992) erleichtert. Denn diese Methode ermöglicht die Lösung von (4) mit Startwerten, die weiter vom Minimumvektor entfernt liegen. Dieses Verfahren besteht aus einer Kombination zwischen der Newton-Methode und der Gradienten-Methode, die durch die Wahl eines Parameters unterschiedlich wirksam werden. Die Gradienten-Methode ($\mathbf{x}_{n+1} = \mathbf{x}_n - konst. \cdot \nabla\chi^2(\mathbf{x}_n)$) konvergiert sehr langsam, konvergiert allerdings auch bei einem schlechteren Startvektor. Geeignete Startwerte sind jedoch auch bei dieser Prozedur erforderlich.

Alle Testrechnungen wurden mit doppelter Genauigkeit durchgeführt. Wie schon erwähnt, verschlechtert das Rechnen mit der einfachen Genauigkeit die Fit- Fähigkeit der Programme im Bereich der kleinen *r*-Werte merklich, d. h. auch, das die numerischen Werte der Streukurven in diesem kritischen Bereich das Ergebnis stark beeinflussen. Es ist daher anzunehmen, dass diese „Sensibilität“ bei der Auswertung fehlerbehafteter Messwerte eine zusätzliche Unsicherheit bei der Bestimmung der Form der Kurven mit sich bringt.

4.6 Polynomentwicklungen im Vergleich

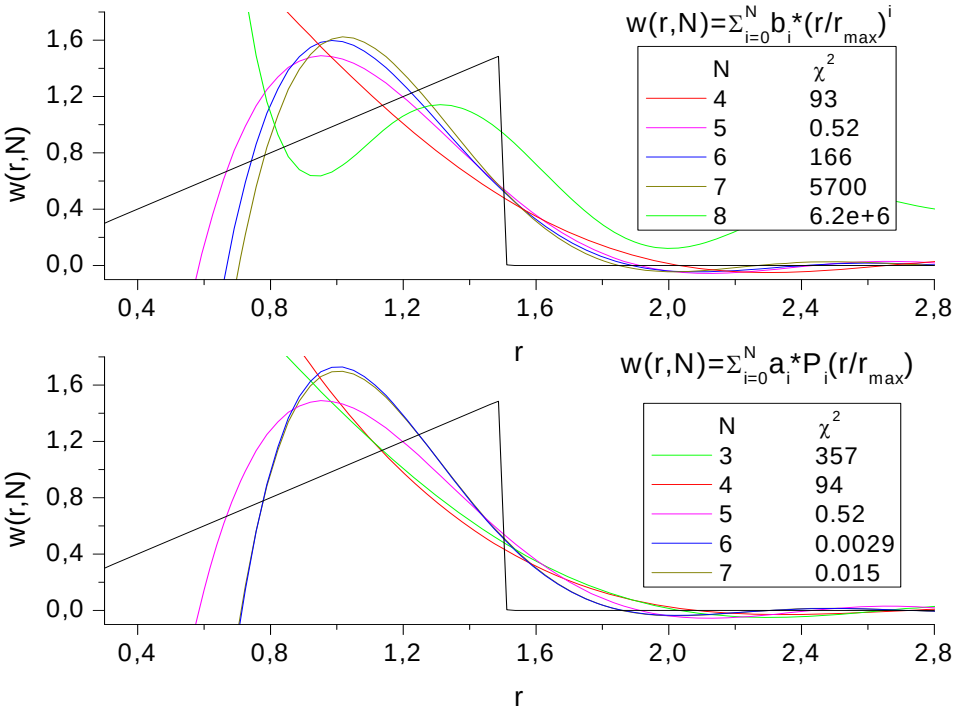


Abb. 14. Rechnung mit unterschiedlichen Basispolynomen.

Mit der letzten Abbildung (Abb. 14) werden noch einmal entscheidende Eigenschaften der allgemeinen Fit- Prozeduren hervorgehoben. Bei diesen Rechnungen wird der Entwicklung nach den gewöhnlichen Polynomen die entsprechende Reihe nach Legendreschen Polynomen gegenübergestellt. Der Vergleich zeigt noch einmal, mit welchen Problemen man bei allgemeinen Fit- Ansätzen rechnen muss. Beide Anpassungen bestätigen die Bedingung (10) N=6, obwohl die Fehlerquadratsumme bei der gewöhnlichen Polynomreihe diesen Schluss nicht zulässt. Die Ursache dafür sind die deutlich zunehmenden Beträge in den Entwicklungskoeffizienten (siehe Tabelle 1), die beim Aufsummieren der Potenzen im Bereich der größeren r-Werte Ungenauigkeiten erzeugen, die zu kräftigeren Schwingungen um die Null führen. Dieser Test zeigt am Verlauf des χ^2- Wertes auch, dass es sinnvoll ist, mit den besser konvergierenden Orthogonalpolynomen zu rechnen. Wie schon erwähnt, ist auch hier zu erkennen, dass die gesuchte Größenverteilung nicht immer durch die minimale Fehlerquadratsumme gegeben ist.

Tabelle 1. Vergleich der Entwicklungskoeffizienten

	1	x	x^2	x^3	x^4	x^5	x^6
b_i	-35.15	389.48	-1588.37	3200.00	-3434.07	1884.06	-415.92
Poly. (8)	$P_0(x)$	$P_1(x)$	$P_2(x)$	$P_3(x)$	$P_4(x)$	$P_5(x)$	$P_6(x)$
a_i	-3.57	-5.80	-6.27	-5.18	-3.14	-1.27	-0.26

5. Zusammenfassung

Die Testrechnungen haben deutlich gezeigt, dass es sinnvoll ist, mit mehreren allgemeinen Funktionsansätzen die Auswertung von Streukurven vorzunehmen. So erkennt man aus dem annähernd gemeinsamen Kurvenverlauf aller Fits den wirklich gültigen Verteilungsbereich. Das Auseinanderdriften der Kurven im Bereich der kleinen Radien markiert den Grenzradius, unter dem keine Aussage mehr zur Größenverteilung gemacht werden sollte. Für den Bereich der kleinen Radien müssen andere Messverfahren herangezogen werden, wie z. B. die Methode der Positronenannihilation. Die Prozedur für die Legendreschen Polynome ist sehr schnell, durch den Austausch der Entwicklungskoeffizienten, auch mit anderen Polynomen nutzbar, zum Beispiel mit den Tschebyscheff- Polynomen, die beim normalen Fit, im Allgemeinen am besten konvergieren. Die Legendreschen Polynome wurden gewählt, weil sie im Vergleich zu den Tschebyscheff- Polynomen im Bereich der Messgenauigkeit der Kleinwinkelstreuung eine größere Nullstellendichte (Abtastfähigkeit) besitzen. Mit der nichtlinearen Fit- Funktion (15) ist es möglich, etwas über den Mechanismus der Clusterbildung zu erfahren, falls die streuenden Teilchen nur aus einer Komponente bestehen. Spezielle Fit- Ansätze, in die theoretische Überlegungen eingehen und dadurch mit weniger freien Parametern auskommen, sind natürlich der sicherste Weg, aussagekräftige Hinweise zu den Modellansätzen zu erhalten. Die Rechnung mit (15) hat leider auch gezeigt, dass bei der Minimierung von (4) mehrere lokale Minima auftreten können, was die Wahl der Startpunkte gelegentlich zusätzlich erschwert. Alle Testrechnungen bestätigen die allgemein bekannte Aussage, dass aus dem Wirkungsquerschnitt der Kleinwinkelstreuung die Größenverteilung der streuenden Teilchen nicht eindeutig bestimmt werden kann.

Literaturverzeichnis

Abramovica, M.; Stigan, I. (1979)
Spravočnik po special'nym funkcijam. Moskwa Nauka

Böhmert, J.; Gokhman, A.; Große, M.; Ulbricht, A. (2003)
Nachweis, Interpretation und Bewertung bestrahlungsbedingter Gefügeänderungen in WWER-Reaktor-Druckbehälterstählen. FZR-381, Juni 2003

Landau, L. D.; Lifshitz, E. M. (1987)
Lehrbuch der Theoretischen Physik V. Akademie Verlag Berlin

Eberbeck, D.; Bläsing, J. (1999)
Investigation of particle size distribution and aggregate structure of various ferrofluids by small-angle scattering experiments.
J. Appl. Cryst. **32**, 273-280.

Glatter, O. (1977)
A New Method for the Evaluation of Small-Angle Scattering Data. *J. Appl. Cryst.* **10**, 415-421.

Glatter, O. (1980)
Determination of Particle-Size Distribution Functions from Small-Angle Scattering Data by Means of the Indirect Transformation Method. . J. Appl. Cryst. **13**, 7-11.

Glatter, O.; Kratky, O. (1982)
Small Angle X-ray Sacttering, Academic Press

Gokhman, A.; Böhmert, J.; Ulbricht, A. (2000)
Contribution to the Determination of Microstructural Parameters from Small Angle Scattering Experiments at Reactor Pressure Vessel Steels. FZR-288, Februar 2000

Lindner, P.; Zemb, Th. (2002)
Neutrons, X-rays and Light: Scattering Methods Applied top Soft Condensed Matter. Elsevier

Press, H.; Teukoolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (1992)
Numerical recipes in FORTRAN, Cambridge University Press

Sauer, R.; Szabo, I. (1968)
Mathematische Hilfsmittel des Ingenieurs, Band III. Springer

Smirnow, W. I. (1968)
Lehrgang der höheren Mathematik, Teil II. VEB Deutscher Verlag der Wissenschaften, Berlin

Snyder, R. L..; Fiala, J.; Bunge, H. J. (1999)
Defect and Microstructure Analysis by Diffraction. Reprint 2005 Oxford

Solt, G.; Frisius, F; Waeber, W.B.; Bührer, W. (1999)
SANS and DENS Study of Irradiation Damage in a Reactor Pressure Vessel Material with a Systematic Variation of Irradiation Dose and Heat Treatments.
14th International Symposium (Vol. II), ASTM STP 1045, N. H. Packan, R. E. Stoller, A. S: Kumar, Eds., ASTM, Philadelphia, 1999, pp. 154

Ulbricht, A. (2005)
Untersuchungen an neutronenbestrahlten Reaktordruckbehälterstählen mit Neutronen-Kleinwinkelstreuung. Dissertation, TU Bergakademie Freiberg

Ulbricht, A.; Bergner, F.; Dewhurst, C.D.; Heinemann, A. (2006)
Small-angle neutron scattering study of post-irradiation annealed neutron irradiated pressure vessel steels. J. Nucl. Mater. 353, 27-34

Gokhman, A., Bergner, F., Ulbricht, A. (2004)
Modeling of vacancy cluster evolution in neutron irradiated iron. FZR-420

Anhang A

1. Leerstellencluster

Das Differenzialgleichungssystem zur Beschreibung der Leerstellenclusterbildung hat die Form (Gokhman et. al. 2004):

$$\frac{dc_n}{dt} = g_{1,\ldots,8} + a(n-1)^{1/3} c_{n-1} - a\left(n^{1/3}(2-\delta) + (n-1)^{1/3} b \cdot d^{(n^{2/3}-(n-1)^{2/3})} \right) \cdot c_n + a\left((n+1)^{1/3}(1-\delta) + n^{1/3} b \cdot d^{((n+1)^{2/3}-n^{2/3})} \right) \cdot c_{n+1} \quad \text{(A1)}$$

Dabei ist c_n die Konzentration eines Clusters mit n Leerstellen pro Volumen, die g_n sind strahlungsinduzierte Quellterme. Die Konstanten in den Koeffizienten des Differenzialgleichungssystems sind durch das Diffusionsvermögen der Leerstellen, der Zwischengitteratome und durch Beziehungen der Ausscheidungskinetik bestimmt:

$$\delta = 1 - D_i C_i / D_V C_V \approx \frac{1}{6}, \quad a = 4\pi \cdot r_0 D_V C_V, \quad a \cdot \delta = 6.187 a_0 (D_V C_V - D_i C_i),$$

$$b = C_{veg} / C_1, \quad d = \exp\left(\frac{4\pi \cdot r_0^2 \gamma}{k_B T} \right); \quad \gamma = \text{Grenzflächenenergie}.$$

Für Clustergrößen $n > 50$ kann die Ausgangsgleichung (A1) mit Hilfe der Taylorreihe in eine partielle Differenzialgleichung überführt werden. An dieser hinreichend bekannten „Transportgleichung“ ist der Prozess der Clusterentwicklung besser abzulesen.

$$\frac{\partial c}{\partial t} = a \cdot \sqrt[3]{n} \left[\frac{(2-\delta+b)}{2} \frac{\partial^2 c}{\partial n^2} + \left(-\delta + b + \frac{2-\delta}{3n} + \frac{2b\ln(d)}{3\sqrt[3]{n}} \right) \frac{\partial c}{\partial n} - \frac{\delta - b}{3n} \cdot c \right] \quad \text{(A2)}$$

Mit der Transformation, $r = r_0 \sqrt[3]{n} \rightarrow c(n,t) = C(r,t)$, erhält man direkt die Gleichung für die Größenverteilung der Cluster:

$$\frac{\partial C}{\partial t} = a \frac{r_0^2}{3r} \left[\frac{r_0^3}{3r^2} \frac{(2-\delta+b)}{2} \frac{\partial^2 C}{\partial r^2} + \left(-(\delta - b) - \frac{r_0^3 b}{3r^3} + \frac{2 r_0 b \ln(d)}{3r} \right) \frac{\partial C}{\partial r} - \frac{\delta - b}{r} C \right]. \quad \text{(A3)}$$

2. Kupfercluster

Im Gegensatz zur Leerstellenclusterausbreitung, deren Ursache die bestrahlungsinduzierten Quellterme g_n sind, bilden sich die Kupfercluster aus einer vorgegebenen Anfangsverteilung c_1 von Einzelatomen:

$$\frac{dc_n}{dt} = a(n-1)^{1/3}\frac{c_1}{c_{eq}}c_{n-1} - a(n^{1/3}\frac{c_1}{c_{eq}} + (n-1)^{1/3}\cdot d^{(n^{2/3}-(n-1)^{2/3})})\cdot c_n + a(n^{1/3}\cdot d^{((n+1)^{2/3}-n^{2/3})})\cdot c_{n+1} \quad \text{(A4)}$$

c_1 ist dabei mit allen Konzentrationen durch den Erhalt der Kupferatome

$$c_{Cu}^0 = c_1 + \sum_{n=2}^{n_{\max}} n\cdot c_n$$

verknüpft. Die Konstanten sind wie oben, aus Elementarprozessen abgeleitete Beziehungen:

$$a = 2\pi\cdot r_{Cu}D_{Cu}C_v, \quad d = \exp\left(\frac{4\pi\cdot r_{Cu}^2\gamma}{k_BT}\right).$$

Auch hier kann für große n das Gleichungssystem durch eine partielle DGL ersetzt werden:

$$\frac{\partial c}{\partial t} = a\cdot\sqrt[3]{n}\left[\frac{(1+c_1/c_{eq})}{2}\frac{\partial^2 c}{\partial n^2} + \left(1-c_1/c_{eq}(1-\frac{1}{3n}) + \frac{2\ln(d)}{3\sqrt[3]{n}}\right)\frac{\partial c}{\partial n} + \frac{(1-c_1/c_{eq})}{3n}\left(1+\frac{1}{3n}\right)\cdot c\right]$$

beziehungsweise

$$\frac{\partial C}{\partial t} = a\frac{r_0^2}{3r}\left[\frac{r_0^3}{3r^2}\frac{(1+C_1/C_{eq})}{2}\frac{\partial^2 C}{\partial r^2} + \left(1-C_1/C_{eq} - \frac{r_0^3}{3r^3} + \frac{2r_0\ln(d)}{3r}\right)\frac{\partial C}{\partial r} + \frac{(1-C_1/C_{eq})}{r}(1+\frac{r_0^3}{3r^3})C\right]$$

.

Die beiden Systeme lassen sich numerisch relativ bequem mit Hilfe der impliziten Differenzengleichungen (Crank-Nicolson- Methode) lösen:

$$\frac{c_n^t - c_n^{t-\Delta t}}{\Delta t} = \alpha_{n-1}\left[\vartheta c_{n-1}^t + (1-\vartheta)c_{n-1}^{t-\Delta t}\right] - (\alpha_n + \beta_{n-1})\left[\vartheta c_n^t + (1-\vartheta)c_n^{t-\Delta t}\right] + \beta_n\left[\vartheta c_{n+1}^t + (1-\vartheta)c_{n+1}^{t-\Delta t}\right]$$

mit $\vartheta = 1/2...1. \rightarrow$ Lösung: $\mathbf{c}^t = (E - \vartheta\Delta t\mathbf{M})^{-1}(E + (1-\vartheta)\Delta t\mathbf{M})\mathbf{c}^{t-\Delta t}$

Der Erhalt der Kupferatome $c_{Cu}^0 = \sum_{n=1}^{N} n\cdot c_n^j$ in (A4) erfolgt dabei iterativ.

Mit der Fit- Funktion $w(r;\mathbf{p}) = a\cdot r^{\alpha}/\left[1+\exp(\beta(r-r_0))\right]$ (15) lassen sich die Lösungsfunktionen beider Systeme für feste Beobachtungszeiten t_{fest} ($r_0 \propto t_{fest}^{\kappa}$) in Abhängigkeit vom Radius sehr gut darstellen.

Anhang B

Fortran 90 -Programme zur Auswertung von Kleinwinkelstreukurven

Dieser Anhang enthält die Quelltexte der Programme, mit denen die Testanpassungen durchgeführt wurden. Alle Programme sind in Fortran 90 (Compaq Visual Fortran) geschrieben. Die Programme wurden nicht so „defensiv" gestaltet, dass sie die Eingabe willkürlicher Parameterwerte erlauben, es ist somit möglich, dass ungeeignete Parameterwerte zum Zusammenbruch der Rechnung führen.

Im Programm der nichtlinearen Anpassung „modell_fit" besteht die Möglichkeit, in ersten Versuchen einen der vier Parameter festzuhalten. Da es sehr unwahrscheinlich ist, bei einer unbekannten Streukurve gleich vier geeignete Startwerte zu erraten, wird damit das Herantasten an einen geeigneten Startvektor erleichtert. Weiterhin ist mit λ (la) ein Dämpfungsparameter gegeben, bei dem das Verfahren bei Startwerten konvergiert, die etwas weiter vom Minimumvektor entfernt liegen. λ muss kleiner Eins sein und ergibt sich durch Probieren.

Das Programm „modell_mrqmin" löst das gleiche Problem mit Hilfe der Levenberg-Marquardt Methode. Die Lösung basiert auf dem Programm xmrqmin.for mit den dazugehörigen Subroutinen mrqmin.for, covsrt.for, mrqcof.for, und gaussj.for, die den Numerical Recipes in Fortran, William H. Press et al., Cambridge University Press, New York, 1992, entnommen wurden. Dieses Programm ist in der aufgelisteten Form nicht lauffähig, da die oben angegebenen Unterprogramme aus der Programmbibliothek der Numerical Recipes einzubinden sind.

Bei den folgenden Programmen der allgemeinen linearen Regression ist nur zu beachten, dass die Zahl der Basisfunktionen wie angegeben beschränkt wird. Den optimalen Fit erhält man, indem die Zahl der Basisfunktionen in der Entwicklung um den angegebenen Richtwert variiert wird.

Die Inputfiles zu den Beispielen wurden mit den entsprechenden Verteilungen in der Streuformel (5) mit Mathcad 2001i berechnet. Diese Files können aber auch mit der jeweiligen Größenverteilung und dem eingerahmten Teil des Programms „modell_fit" erzeugt werden.

```
! Fit-Funktion: w(r)=A*r**a/(1+exp(b*(r-r0)))
! xa: Startvektor für (A, a, b, r0)
! nr: Zahl der Stützstellen bei der Simpson-Regel
! ra-rb: Radiusbereich, qa-qb: q-Bereich des Streuvektors
! psi(i): Streukurve an den nq-Stützstellen q(i)
! ip=k hält den k-ten Parameter fest (k=1-A, 2-a, 3-b, 4-r0)
! la <1.0: Dämpfungsparameter

module messwerte
    integer,parameter          ::nq=42       ! Zahl der Messwerte
    real,parameter             ::qa=0.2, qb=2.8
    real,dimension(0:nq)       ::q, psi, phi, diff
end module messwerte

module verteilung
    integer,parameter          ::nr=92       ! nr=gerade Zahl
    real,parameter             ::ra=0.3, rb=3.3
    real,dimension(0:nr)       ::r, w, wm
end module verteilung

program modell_fit
use messwerte
use verteilung
implicit none
real,parameter             ::A=1011.0        !16*pi**2*Eta*1.e+5)

integer                    ::i, j, k, it, neu, ip
real                       ::integ, eps, amp, h, dq
real                       ::wr, chi2, chin, fq, la
real,dimension(0:3,0:nq)   ::phidp
real,dimension(0:3,0:nr)   ::wdp, wrdp
real,dimension(0:3)        ::xa, xb, xh, epsv, dwdp

open(unit=2,file='d:\daten\phi_q_r.dat',status='old',action='read')
open(unit=4,file='d:\daten\r_verteilung.dat',action='write')
open(unit=6,file='d:\daten\vergleich.dat',action='write')
eps=1.0; it=0; neu=0; chin=1.e+6

xa=(/1.0,1.0,50.,1.45/) ! Startvektor
ip=0                    ! ip=1,2,3,4 hält den jeweiligen Parameter fest
la=1.0

h=(rb-ra)/nr; dq=(qb-qa)/nq
if(mod(nr,2) /= 0) print*, 'nr muss gerade Zahl sein!'
if(h>0.05) print*, 'nr zu klein=>Integrationsfehler!'; print*
do j=0, nr;  r(j)=ra+h*j;  enddo
do  i=0, nq
    q(i)=qa+dq*i
    read(2,*) psi(i)
    psi(i)=psi(i)*q(i)**3
end do
close(unit=2)
do while(eps > 1.e-6)
    select case(neu)
    case(0)
    xb=xa
    case(1)
    print*, 'Startvektor neu festlegen!'
    exit
    end select

    select case (ip)
```

```
    case(0)
    do j=0, nr
        call mod_funk(r(j),xb,wr,dwdp)
        wm(j)=wr; wrdp(:,j)=dwdp(:)
    enddo
        call para_opt(xh,3)
        xa=xa+xh
    case(1)
    do j=0, nr
        call mod_funk(r(j),xb,wr,dwdp); wm(j)=wr
        wrdp(0,j)=dwdp(1); wrdp(1,j)=dwdp(2); wrdp(2,j)=dwdp(3)
    enddo
        call para_opt(xh,2)
        xa(1)=xa(1)+xh(0); xa(2)=xa(2)+xh(1); xa(3)=xa(3)+xh(2)
    case(2)
    do j=0, nr
        call mod_funk(r(j),xb,wr,dwdp); wm(j)=wr
        wrdp(0,j)=dwdp(0); wrdp(1,j)=dwdp(2); wrdp(2,j)=dwdp(3)
    enddo
        call para_opt(xh,2)
        xa(0)=xa(0)+xh(0); xa(2)=xa(2)+xh(1); xa(3)=xa(3)+xh(2)
    case(3)
    do j=0, nr
        call mod_funk(r(j),xb,wr,dwdp); wm(j)=wr
        wrdp(0,j)=dwdp(0); wrdp(1,j)=dwdp(1); wrdp(2,j)=dwdp(3)
    enddo
        call para_opt(xh,2)
        xa(0)=xa(0)+xh(0); xa(1)=xa(1)+xh(1); xa(3)=xa(3)+xh(2)
    case(4)
    do j=0, nr
        call mod_funk(r(j),xb,wr,dwdp); wm(j)=wr
        wrdp(0,j)=dwdp(0); wrdp(1,j)=dwdp(1); wrdp(2,j)=dwdp(2)
    enddo
        call para_opt(xh,2)
        xa(0)=xa(0)+xh(0); xa(1)=xa(1)+xh(1); xa(2)=xa(2)+xh(2)
    end select

    do j=0, 3; epsv(j)=abs(1-xb(j)/xa(j)); enddo
    eps=maxval(epsv)
    if(chi2 > chin ) neu=1
    it=it+1; chin=chi2
    ! Iterationsfolge
    write(*, '(2x,a5, i3, 2x,a4,e9.4,3x,a5,e9.4)') &
              'Iter.',it,'eps=',eps,'chi2=',chi2/nq
    print*
    write(*,'(4(e10.4,3x))') xa(0), xa(1), xa(2), xa(3)
    print*
end do
    ! Berechnung der Streukurve aus der Größenverteiung
!-------------------------------------------------------
    do k=0, nq
        do j=0, nr
        call mod_funk(r(j),xb,wr,dwdp)
        wm(j)=wr
        w(j) =wr*fqr(q(k),r(j))
        enddo
        amp=A/q(k)**3
        call simpson(ra,rb,w,integ,nr)
        phi(k)=integ*amp
    enddo
!-------------------------------------------------------
    ! Größenverteilung
```

```
    do j=0, nr
        write(4,'(f8.2,2x,(e9.3))') r(j), wm(j)
    enddo
    ! Vergleich der Streuquerschnitte
    do i=0, nq
        write(6,'(f8.2,2x,2(x,e9.3))') q(i), psi(i), phi(i)
    enddo

contains

real function fqr(q,r)
real          ::qr, q, r
    qr=q*r
    fqr=0.5*(1.+qr*qr)-qr*sin(2.0*qr)+0.5*(qr*qr-1)*cos(2.0*qr)
end function

subroutine para_opt(xh,np)
implicit none
integer                          ::np
real,dimension(0:np,0:np)        ::m, mi
real,dimension(0:np)             ::d, xh
    do k=0, nq
        do j=0, nr
        fq=fqr(q(k),r(j))
        w(j)=wm(j)*fq
        do i=0, np; wdp(i,j)=wrdp(i,j)*fq; enddo
        enddo
        amp=A/q(k)**3
        call simpson(ra,rb,w,integ,nr)
        phi(k)=integ*amp
        do i=0, np
        call simpson(ra,rb,wdp(i,:),integ,nr)
        phidp(i,k)=integ*amp
        enddo
    enddo
        diff=psi-phi
        chi2=dot_product(diff,diff)
        do i=0, np
        d(i)=dot_product(diff,phidp(i,:))
        enddo
        do i=0, np
            do j=i, np
            m(i,j)=dot_product(phidp(i,:),phidp(j,:))
            enddo
        enddo
        do i=1, np
            do j=0, i; m(i,j)=m(j,i); enddo
        enddo
        call inverse(m,mi,np)
        xh=la*matmul(mi,d)
end subroutine

end program

subroutine simpson(a,b,f,sg,m)
implicit none
integer               ::m, i
real                  ::a, b, h, sg, su
real,dimension(0:m)   ::f
    h=(b-a)/m/3.0
    sg=0.0; su=0.0
    do i=1, m-1, 2
```

```
        sg=sg+f(i+1); su=su+f(i)
    end do
    sg=(4.*su+2*sg+f(0)-f(m))*h
end subroutine

subroutine inverse(M,IN,n)
implicit none
integer                                 ::i,j,n
real,dimension(0:n,0:n),intent(in)  ::M
real,dimension(0:n,0:n)             ::IN,R
R=M
in=0
do i=0,n
    in(i,i)=1.0
end do
do i=0,n-1
    do j=i+1, n
    if(R(j,i)/=0.0) then
        in(j,:)=R(i,i)/R(j,i)*in(j,:)-in(i,:)
        R(j,:)=R(i,i)/R(j,i)*R(j,:)-R(i,:)
    end if
    end do
end do
do i=1,n
    do j=0,i-1
    if(R(j,i)/=0.0) then
        in(j,:)=R(i,i)/R(j,i)*in(j,:)-in(i,:)
        R(j,:)=R(i,i)/R(j,i)*R(j,:)-R(i,:)
    end if
    end do
end do
do i=0, n
    in(i,:)=in(i,:)/R(i,i)
end do
end subroutine

subroutine mod_funk(r,x,wr,dwdp)
implicit none
real                ::r, wr, exr
real,dimension(0:3) ::x, dwdp
    exr=exp(x(2)*(r-x(3)))
    wr=x(0)*r**x(1)/(1.0+exr)
    dwdp(0)=wr/x(0)
    dwdp(1)=wr*log(r)
    dwdp(2)=wr*(x(3)-r)*exr/(1.0+exr)
    dwdp(3)=wr*x(2)*exr/(1.0+exr)
end subroutine
```

```
! Fit-Funktion: w(r)=A*r**a/(1+exp(b*(r-r0)))
! xa: Startvektor für (A, a, b, r0)
! nr: Zahl der Stützstellen bei der Simpson-Regel
! ra-rb: Radiusbereich, qa-qb: q-Bereich des Streuvektors
! psi(i): Streukurve an den nq-Stützstellen q(i)

module messwerte
    integer,parameter          ::nq=43       ! Zahl der Messwerte
    real,parameter             ::qa=0.2, qb=2.8
    real,dimension(nq)         ::q, psi, sig, phi_b
end module messwerte

module verteilung
    integer,parameter          ::nr=92       ! nr=gerade Zahl
    real,parameter             ::ra=0.3, rb=3.3
    real,dimension(0:nr)       ::r, w, wm
end module verteilung

program modell_mrqmin
use messwerte
use verteilung
implicit none
integer,parameter          ::np=4
integer                    ::i, j, k, iter, itst, mfit, ia(np)
real,parameter             ::A=1011.0        !16*pi**2*Eta*1.e+5)
real                       ::integ, eps, amp, h, dq, modell, fqr
real                       ::wr, chi2, gasdev, fq, alamda
real                       ::ochisq, chisq, phi
real,dimension(np)         ::phidp
real,dimension(np,0:nr)    ::wdp, wrdp
real,dimension(np)         ::xa, dwdp
real,dimension(np,np)      ::covar, alpha
external modell

open(unit=2,file='d:\daten\phi_q_r.dat',status='old',action='read')
open(unit=4,file='d:\daten\r_verteilung.dat',action='write')
open(unit=6,file='d:\daten\vergleich.dat',action='write')
eps=1.0

xa=(/.8,1.0,30.,1.3/) ! Startvektor

h=(rb-ra)/nr; dq=(qb-qa)/(nq-1)
if(mod(nr,2) /= 0) print*, 'nr muss gerade Zahl sein!'
if(h>0.05) print*, 'nr zu klein=>Integrationsfehler!'; print*
    do j=0, nr;  r(j)=ra+h*j;  enddo
    do  i=1, nq
        q(i)=qa+dq*(i-1)
        read(2,*) psi(i)
        psi(i)=psi(i)*q(i)**3; sig(i)=0.001*psi(i)
    end do
    close(unit=2)

    mfit=np
    do i=1, mfit; ia(i)=1; enddo
    do iter=1, 2
    alamda=-1
    call mrqmin(q,psi,sig,nq,xa,ia,np,covar,alpha,np,chisq, &
                modell,alamda)
    k=1; itst=0
10  if (mod(k,10)==0) then
    write(*,'(/1x,a,i2,t18,a,f10.4,t43,a,e9.2)') 'Iteration #',k, &
               'Chi-squared:',chisq,'alamda:',alamda
```

```
    write(*,'(1x,t5,a,t14,a,t23,a,t32,a)') 'xa(1)', &
                'xa(2)','xa(3)','xa(4)'
    write(*,'(1x,4f9.4)') (xa(i),i=1,4)
    endif
    k=k+1
    ochisq=chisq
    call mrqmin(q,psi,sig,nq,xa,ia,np,covar,alpha,np,chisq, &
                modell,alamda)
        if (chisq > ochisq) then
          itst=0
        else if (abs(ochisq-chisq) < 0.1) then
          itst=itst+1
        endif
        if (itst < 60) then
        goto 10
        endif
        alamda=0.0

    call mrqmin(q,psi,sig,nq,xa,ia,np,covar,alpha,np,chisq, &
                modell,alamda)
        write(*,*) 'Uncertainties:'
        write(*,'(1x,4f9.3/)') (sqrt(covar(i,i)),i=1,4)
        if (iter==3) then
          write(*,*) 'press return to continue with constraint'
          read(*,*)
          write(*,*) 'Holding xa(2) and xa(3) constant'
          do j=1,np; xa(j)=xa(j)+.1; enddo
          xa(2)=1.041
          ia(2)=0
          xa(3)=119.0
          ia(3)=0
        endif
     enddo

    print*
    write(*, '(4(e10.4,3x))') xa(1), xa(2), xa(3), xa(4)
    print*

    do k=1, nq
        do j=0, nr
        call mod_funk(r(j),xa,wr,dwdp)
        wm(j)=wr
        w(j) =wr*fqr(q(k),r(j))
        enddo
        amp=A/q(k)**3
        call simpson(ra,rb,w,integ,nr)
        phi_b(k)=integ*amp
    enddo
    ! Größenverteilung
    do j=0, nr
        write(4,'(f8.2,2x,(e9.3))') r(j), wm(j)
    enddo
    ! Vergleich der Streuquerschnitte
    do i=1, nq
        write(6,'(f8.2,2x,2(x,e9.3))') q(i), psi(i), phi_b(i)
    enddo
end program

real function fqr(q,r)
real          ::qr, q, r
    qr=q*r
    fqr=0.5*(1.+qr*qr)-qr*sin(2.0*qr)+0.5*(qr*qr-1)*cos(2.0*qr)
```

```
end function

subroutine modell(q,xa,phi,phidp,np)
use verteilung
implicit none
integer                          ::np, i, j
real                             ::q, phi, fq, wr, amp, integ, fqr
real,parameter                   ::A=1011.0
real,dimension(np)               ::xa, dwdp, phidp
real,dimension(np,0:nr)          ::wdp

        do j=0, nr
        fq=fqr(q,r(j))
        call mod_funk(r(j),xa,wr,dwdp)
        w(j)=wr*fq
        do i=1, np; wdp(i,j)=dwdp(i)*fq; enddo
        enddo
        amp=A/q**3
        call simpson(ra,rb,w,integ,nr)
        phi=integ*amp
        do i=1, np
        call simpson(ra,rb,wdp(i,:),integ,nr)
        phidp(i)=integ*amp
        enddo
end subroutine

subroutine simpson(a,b,f,sg,m)
implicit none
integer                ::m, i
real                   ::a, b, h, sg, su
real,dimension(0:m)    ::f
    h=(b-a)/m/3.0
    sg=0.0; su=0.0
    do i=1, m-1, 2
        sg=sg+f(i+1); su=su+f(i)
    end do
    sg=(4.*su+2*sg+f(0)-f(m))*h
end subroutine

subroutine mod_funk(r,x,wr,dwdp)
implicit none
real                 ::r, wr, exr
real,dimension(4)    ::x, dwdp
    exr=exp(x(3)*(r-x(4)))
    wr=x(1)*r**x(2)/(1.0+exr)
    dwdp(1)=wr/x(1)
    dwdp(2)=wr*log(r)
    dwdp(3)=wr*(x(4)-r)*exr/(1.0+exr)
    dwdp(4)=wr*x(3)*exr/(1.0+exr)
end subroutine
```

```
! Fit-Funktion: Fouierreihe
! w(r)=a0+a1*sin(2pi*r/rb)+b1*cos(2pi*r/rb)+...
!                  +an*sin(2pi*n*r/rb)+bn*cos(2pi*n*r/rb)
! nf: Zahl der Koeffizienten = ganze Zahl > (qb*rb/pi)
! ra-rb: Radiusbereich, qa-qb: Bereich des Streuvektors
! psi(i): Streuquerschnitt an den nq Stützstellen q(i)

module messwerte
    integer,parameter           ::nq=42    ! Zahl der Messwerte
    real,parameter              ::qa=0.2, qb=2.8
    real,dimension(0:nq)        ::q, psi, psi_m
end module messwerte

module verteilung
    integer,parameter           ::nr=92     !nr=Stuetzstellen
    real,parameter              ::ra=0.3, rb=3.3
    real,dimension(0:nr)        ::r, w
end module verteilung

program Fourierreihe_2pi
use messwerte
use verteilung
implicit none
integer,parameter               ::nf=3, mf=2*nf
integer                         ::i,j,k
real,parameter                  ::A=1011.0, pi=3.141592654
real                            ::p, pr, h, dq
real                            ::amp, sp, cp, sn, cn, si, co, omp, omn
real                            ::gs, gi, qk, chi2
real,dimension(0:mf,0:nq)       ::phi
real,dimension(0:mf)            ::xa, d
real,dimension(0:mf,0:mf)       ::m, mi

open(unit=2,file='d:\daten\Itrapez_q.dat',status='old',action='read')
open(unit=4,file='d:\daten\r_verteilung.dat',action='write')
open(unit=6,file='d:\daten\vergleich.dat',action='write')

h=(rb-ra)/nr; p=2.*pi/rb; dq=(qb-qa)/nq
do i=0, nq
    q(i)=qa+i*dq
    read(2,*) psi(i)
    psi(i)=psi(i)*q(i)**3
enddo

do i=0, nr; r(i)=ra+h*i; enddo

do k=0, nq
    qk=q(k); amp=A/(qk**3)
    phi(0,k) =amp*fqr(qk,rb)
    do i=1, nf
    si=sin(i*p*rb); co=cos(i*p*rb)
    omp=i*p+2*qk;   omn=i*p-2*qk
    sp=sin(omp*rb); sn=sin(omn*rb)
    cp=cos(omp*rb); cn=cos(omn*rb)
    phi(i,k)=amp*(hqc1(qk,i,rb,si,co,p)+hqc2(qk,rb,omp,omn,sp,cp,sn,cn)
 &
            +hqc3(qk,rb,omp,omn,sp,sn)+hqc4(qk,rb,omp,omn,sp,cp,sn,cn))

    j=i+nf
    phi(j,k)=amp*(hqs1(qk,i,rb,si,co,p)+hqs2(qk,rb,omp,omn,sp,cp,sn,cn)
 &
            +hqs3(qk,rb,omp,omn,cp,cn)+hqs4(qk,rb,omp,omn,sp,cp,sn,cn))
```

```
    enddo
enddo

do i=0, mf
        gs=dot_product(phi(i,:),psi)
        d(i)=gs
    do j=i, mf
        gs=dot_product(phi(i,:),phi(j,:))
        m(i,j)=gs; m(j,i)=gs
    enddo
enddo

call inverse(m,mi,mf)
xa=matmul(mi,d)
    gi=0.0
    do k=0, nq
        gs=dot_product(xa,phi(:,k))
        gi=gi+(gs-psi(k))**2; psi_m(k)=gs
    enddo
    chi2=gi/nq

    do i=0, nr
        gs=0.0; pr=r(i)*p
        do j=0, nf
            gs=gs+xa(j)*cos(j*pr)+xa(j+nf)*sin(j*pr)
        enddo
        w(i)=gs
    enddo
    ! Größenverteilung
    do i=0, nr
    write(4,'(f8.3,x,e12.4)') r(i), w(i)
    enddo
    ! Vergleich der Streuquerschnitte
    do i=0, nq
    write(6,'(f8.3,2(x,e9.3))') q(i), psi(i), psi_m(i)
    enddo
    print*
    write(*,'(I3,x,e12.4)') nf, chi2
contains

real function fqr(q,r)
real        ::qr, q, r
    qr=q*r
    fqr=0.5*r*((1.+qr*qr/3.)+0.5*(qr-2.5/qr)*sin(2.*qr)+1.5*cos(2.*qr))
end function

real function hqs1(q,n,r,sx,cx,p)
integer     ::n
real        ::qr, q, r, sx, cx, p
    hqs1=(-0.5*(1.+(r*r-2./(n*p)**2.)*q*q)*cx+q*q*r*sx/(n*p))/(n*p) &
         +(0.5-q*q/(n*p)**2.)/(n*p)
end function

real function hqc1(q,n,r,sx,cx,p)
integer     ::n
real        ::qr, q, r, sx, cx, p
    qr=q*r
    hqc1=(0.5*(1.+(r*r-2./(n*p)**2.)*q*q)*sx+q*q*r*cx/(n*p))/(n*p)
end function

real function hqs2(q,r,omp,omn,sp,cp,sn,cn)
```

```
real         ::q, r, sp, sn, cp, cn, omp, omn
    hqs2=-0.5*q*((cn-1.+omn*r*sn)/(omn*omn)-(cp-1.+omp*r*sp)/(omp*omp))
end function

real function hqc2(q,r,omp,omn,sp,cp,sn,cn)
real         ::q, r, sp, sn, cp, cn, omp, omn
    hqc2=-0.5*q*((sp-omp*r*cp)/(omp*omp)-(sn-omn*r*cn)/(omn*omn))
end function

real function hqs3(q,r,omp,omn,cp,cn)
real         ::q, r, cp, cn, omp, omn
    hqs3=0.25*((cp-1.)/omp+(cn-1.)/omn)
end function

real function hqc3(q,r,omp,omn,sp,sn)
real         ::q, r, sp, sn, omp, omn
    hqc3=-0.25*(sp/omp+sn/omn)
end function

real function hqs4(q,r,omp,omn,sp,cp,sn,cn)
real         ::q, r, sp, sn, cp, cn, omp, omn
    hqs4=0.25*q*q*(((2.-omp*omp*r*r)*cp+2.*omp*r*sp-2.)/omp**3 &
                  +((2.-omn*omn*r*r)*cn+2.*omn*r*sn-2.)/omn**3)
end function

real function hqc4(q,r,omp,omn,sp,cp,sn,cn)
real         ::q, r, sp, sn, cp, cn, omp, omn
    hqc4=-0.25*q*q*(((2.-omp*omp*r*r)*sp-2.*omp*r*cp)/omp**3 &
                   +((2.-omn*omn*r*r)*sn-2.*omn*r*cn)/omn**3)
end function

end program
```

```
! Fit-Funktion: sin und cos- Fourierreihe
! ws(r)=a0+a1*sin(pi*r/rb)+...+an*sin(pi*n*r/rb)
! wc(r)=b0+b1*cos(pi*r/rb)+...+bn*cos(pi*n*r/rb)
! nf: Zahl der Koeffizienten = ganze Zahl > (2*qb*rb/pi)
! ra-rb: Radiusbereich, qa-qb: Bereich des Streuvektors
! psi(i): Streukurve an den nq Stützstellen q(i)

program sin_cos_reihe
implicit none
integer,parameter               ::nq=42, nr=92, nf=6
integer                         ::i,j,k
real,parameter                  ::A=1011.0, pi=3.141592654
real                            ::ra, rb, h, qa, qb, dq, p, pr

real                            ::amp, sp, cp, sn, cn, si, co, omp, omn
real                            ::qk, gs, gc, gi, gj, chi2s, chi2c
real,dimension(0:nf,0:nq)       ::phis, phic
real,dimension(0:nq)            ::q, psi, ys, yc
real,dimension(0:nf)            ::xs, xc, ds, dc
real,dimension(0:nf,0:nf)       ::ms, mc, mi
real,dimension(0:nr)            ::r, ws, wc

open(unit=2,file='d:\daten\Itrapez_q.dat',status='old',action='read')
open(unit=4,file='d:\daten\r_verteilung.dat', action='write')
open(unit=6,file='d:\daten\vergleich.dat', action='write')

ra=0.3; rb=3.3; h=(rb-ra)/nr; p=pi/rb
qa=0.2; qb=2.8; dq=(qb-qa)/nq

do i=0, nq
    q(i)=qa+i*dq
    read(2,*) psi(i)
    psi(i)=psi(i)*q(i)**3
enddo

do i=0, nr; r(i)=ra+h*i; enddo

do k=0, nq
   qk=q(k); amp=A/(qk**3)
   phis(0,k) =amp*fqr(qk,rb)
   phic(0,k) =phis(0,k)
   do i=1, nf
   si=sin(i*p*rb); co=cos(i*p*rb)
   omp=i*p+2*qk; omn=i*p-2*qk
   sp=sin(omp*rb); sn=sin(omn*rb)
   cp=cos(omp*rb); cn=cos(omn*rb)

   phis(i,k)=amp*(hqs1(qk,i,rb,si,co,p)+hqs2(qk,rb,omp,omn,sp,cp,sn,cn)
 &
            +hqs3(qk,rb,omp,omn,cp,cn)+hqs4(qk,rb,omp,omn,sp,cp,sn,cn))
   phic(i,k)=amp*(hqc1(qk,i,rb,si,co,p)+hqc2(qk,rb,omp,omn,sp,cp,sn,cn)
 &
            +hqc3(qk,rb,omp,omn,sp,sn)+hqc4(qk,rb,omp,omn,sp,cp,sn,cn))

   enddo
enddo

do i=0, nf
    gs=dot_product(phis(i,:),psi)
    gc=dot_product(phic(i,:),psi)
    ds(i)=gs; dc(i)=gc
    do j=0, nf
```

```
        gs=dot_product(phis(i,:),phis(j,:))
        gc=dot_product(phic(i,:),phic(j,:))
        ms(i,j)=gs; ms(j,i)=gs
        mc(i,j)=gc; mc(j,i)=gc
    enddo
enddo

call inverse(ms,mi,nf)
xs=matmul(mi,ds)
call inverse(mc,mi,nf)
xc=matmul(mi,dc)

gi=0.0; gj=0.0
do k=0, nq
    gs=dot_product(xs,phis(:,k))
    gc=dot_product(xc,phic(:,k))
    gi=gi+(gs-psi(k))**2; ys(k)=gs
    gj=gj+(gc-psi(k))**2; yc(k)=gc
enddo
    chi2s=gi/nq; chi2c=gj/nq

do i=0, nr
    gs=0.0; gc=0.0; pr=r(i)*p
    do j=0, nf
        gs=gs+xs(j)*sin(j*pr)
        gc=gc+xc(j)*cos(j*pr)
    enddo
    ws(i)=gs; wc(i)=gc
enddo
    ! Größenverteilungen
    do i=0, nr
    write(4,'(f8.3,2(x,e12.4))') r(i), ws(i), wc(i)
    enddo
    ! Vergleich für die Sinusreihe
    do i=0, nq
    write(6,'(f8.3,2(x,e9.3))') q(i), psi(i), ys(i)
    enddo
    print*
    write(*,'(x,I3,2(x,e12.4))') nf, chi2s, chi2c
contains

real function fqr(q,r)
real     :: qr, q, r
    qr=q*r
    fqr=0.5*r*((1.+qr*qr/3.)+0.5*(qr-2.5/qr)*sin(2*qr)+1.5*cos(2*qr))
end function

real function hqs1(q,n,r,sx,cx,p)
integer  ::n
real     ::qr, q, r, sx, cx, p
    hqs1=(-0.5*(1.+(r*r-2./(n*p)**2.)*q*q)*cx+q*q*r*sx/(n*p))/(n*p) &
         +(0.5-q*q/(n*p)**2.)/(n*p)
end function

real function hqc1(q,n,r,sx,cx,p)
integer  ::n
real     ::qr, q, r, sx, cx, p
    qr=q*r
    hqc1=(0.5*(1.+(r*r-2./(n*p)**2.)*q*q)*sx+q*q*r*cx/(n*p))/(n*p)
end function

real function hqs2(q,r,omp,omn,sp,cp,sn,cn)
```

```
real     ::q, r, sp, sn, cp, cn, omp, omn
    hqs2=-0.5*q*((cn-1.+omn*r*sn)/(omn*omn)-(cp-1.+omp*r*sp)&
                 /(omp*omp))
end function

real function hqc2(q,r,omp,omn,sp,cp,sn,cn)
real     ::q, r, sp, sn, cp, cn, omp, omn
    hqc2=-0.5*q*((sp-omp*r*cp)/(omp*omp)-(sn-omn*r*cn)&
                 /(omn*omn))
end function

real function hqs3(q,r,omp,omn,cp,cn)
real     ::q, r, cp, cn, omp, omn
    hqs3=0.25*((cp-1.)/omp+(cn-1.)/omn)
end function

real function hqc3(q,r,omp,omn,sp,sn)
real     ::q, r, sp, sn, omp, omn
    hqc3=-0.25*(sp/omp+sn/omn)
end function

real function hqs4(q,r,omp,omn,sp,cp,sn,cn)
real     ::q, r, sp, sn, cp, cn, omp, omn
  hqs4=0.25*q*q*(((2.-omp*omp*r*r)*cp+2.*omp*r*sp-2.)/omp**3 &
                 +((2.-omn*omn*r*r)*cn+2.*omn*r*sn-2.)/omn**3)
end function

real function hqc4(q,r,omp,omn,sp,cp,sn,cn)
real     ::q, r, sp, sn, cp, cn, omp, omn
    hqc4=-0.25*q*q*(((2.-omp*omp*r*r)*sp-2.*omp*r*cp)/omp**3 &
                   +((2.-omn*omn*r*r)*sn-2.*omn*r*cn)/omn**3)
end function

end program
```

```
! Fit-Funktion: w(r)=a0*P0(r/rb)+a1*P1(r/rb)+...+an*Pn(r/rb)
! Pn(x)=Legendresches Polynom n-ten Grades, Intervall [0,1]
! np: Zahl der gewünschten Polynome, np= ganze Zahl >(2*qb*rb/pi)
! np_max=7
! ra-rb: Radiusbereich, qa-qb: Bereich des Streuvektors
! psi(i): Streukurve an den nq Stützstellen q(i)

module messwerte
    integer,parameter           ::nq=42    ! Zahl der Messwerte
    real,parameter              ::qa=0.2, qb=2.8
    real,dimension(0:nq)        ::q, psi, psi_m
end module messwerte

module verteilung
    integer,parameter           ::nr=92     !nr=Stuetzstellen
    real,parameter              ::ra=0.3, rb=3.3
    real,dimension(0:nr)        ::r, w, x
end module verteilung

program Polynom_fit
use messwerte
use verteilung
implicit none
integer,parameter           ::np=6

integer                     ::i,j,k
real,parameter              ::A=1011.0
real                        ::xr, zo, amp, h, dq
real                        ::sp, cp, si, co, gs, gi, qk, chi2
real,dimension(0:7,0:nq)    ::phi
real,dimension(0:7)         ::S, P
real,dimension(0:np)        ::xa, d
real,dimension(0:np,0:np)   ::m, mi

open(unit=2,file='d:\daten\phi_q_r.dat',status='old',action='read')
open(unit=4,file='d:\daten\dichte_r.dat',action='write')
open(unit=6,file='d:\daten\vergleich.dat',action='write')

h=(rb-ra)/nr; dq=(qb-qa)/nq
do i=0, nq
    q(i)=qa+i*dq
    read(2,*) psi(i)
    psi(i)=psi(i)*q(i)**3
enddo

do i=0, nr; r(i)=ra+h*i; x(i)=r(i)/rb; enddo

do k=0, nq
    qk=q(k); amp=A/(4.0*qk**4)
    zo=2*qk*rb
    call potin(S,zo)
    phi(0,k)=amp*S(0)
    phi(1,k)=amp*(S(0)-2.0*S(1))
    phi(2,k)=amp*(S(0)+6.0*S(2)-6.0*S(1))
    phi(3,k)=amp*(S(0)-20.0*S(3)+30.0*S(2)-12.0*S(1))
    phi(4,k)=amp*(S(0)+70.0*S(4)-140.0*S(3)+90.0*S(2)-20.0*S(1))
    phi(5,k)=amp*(S(0)-252.0*S(5)+630.0*S(4)-560.0*S(3)+210.0*S(2)- &
                      30.0*S(1))
    phi(6,k)=amp*(S(0)+924.0*S(6)-2772.0*S(5)+3150.0*S(4)-1680.0*S(3)+&
                      420.0*S(2)-42.0*S(1))
    phi(7,k)=amp*(S(0)-3432.0*S(7)+12012.0*S(6)-16632.0*S(5)+11550.0* &
                      S(4)-4200.0*S(3)+756.0*S(2)-56.0*S(1))
```

```
    ! Potenzreihe
    ! do i=, np; phi(i,k)=amp*S(i); enddo
enddo

do i=0, np
    gs=dot_product(phi(i,:),psi)
    d(i)=gs
    do j=i, np
        gs=dot_product(phi(i,:),phi(j,:))
        m(i,j)=gs; m(j,i)=gs
    enddo
enddo
print*
    call inverse(m,mi,np)
    xa=matmul(mi,d)
    gi=0.0
    do k=0, nq
        gs=0.0
        do i=0, np
            gs=gs+xa(i)*phi(i,k)
        enddo
        gi=gi+(gs-psi(k))**2; psi_m(k)=gs
    enddo
    chi2=gi/nq
    write(*,'(x,8(f8.3,x))') (xa(i), i=0,np)
    do i=0, nr
        xr=x(i)
        call poly(P,xr,np)
        gs=0.0
        do j=0, np; gs=gs+xa(j)*P(j); enddo
        w(i)=gs
    enddo
   !Ergebnisse des Fits
    do i=0, nq
    write(6,'(f8.3,2x,2(f9.4,x))') q(i), psi(i), psi_m(i)
    enddo
    do i=0, nr
    write(4,'(x,f8.3,x,e12.4)') r(i), w(i)
    enddo
    print*
    write(*,'(x,e12.4)')  chi2

contains

subroutine potin(S,x)
!Integration von f(q,r)**2*x**n, n=0...7
implicit none
real                    ::x, xx, co, si, q, a
real                    ::p1, p2, p3, p4, p5, p6, p7, p8
real, dimension(0:7)    ::S

xx=x*x; co=cos(x); si=sin(x)
p1=xx-2.0; p2=(xx-6.0)*x; p3=(xx-12.0)*xx+24.0
p4=((xx-20.0)*xx+120.0)*x; p5=((xx-30.0)*xx+360.0)*xx-720.0
p6=(((xx-42.0)*xx+840.0)*xx-5040.0)*x
p7=(((xx-56.0)*xx+1680.0)*xx-20160.0)*xx+40320.0
p8=((((xx-72.0)*xx+3024.0)*xx-60480.0)*xx+362880.0)*x
S(0)=(xx/12.0+1.0)*x-(2.0-0.25*p1)*si+1.5*x*co
S(1)=(xx/16.0+0.5)*x-((3.0*x-0.25*p2)*si-(1.75*p1-1.0)*co-4.5)/x
S(2)=(xx/20.0+1./3.)*x-((4.0*p1-0.25*p3)*si-(2.0*p2-2.0*x)*co)/xx
S(3)=(xx/24.0+0.25)*x-((5.0*p2-0.25*p4)*si-(2.25*p3-3.0*p1)*co+60.0)&
         /(xx*x)
```

```
S(4)=(xx/28.0+0.2)*x-((6.0*p3-0.25*p5)*si-(2.5*p4-4.0*p2)*co)/(xx*xx)
S(5)=(xx/32.0+1./6.)*x-((7.0*p4-0.25*p6)*si-(2.75*p5-5.0*p3)*co- &
         2100.0)/x**5
S(6)=(xx/36.0+1./7.)*x-((8.0*p5-0.25*p7)*si-(3.0*p6-6.0*p4)*co)/xx**3
S(7)=(xx/40.0+1./8.)*x-((9.0*p6-0.25*p8)*si-(3.25*p7-7.0*p5)*co+ &
        136080.0)/x**7
end subroutine

subroutine poly(P,x,n)
! Legendresche Polynome fuer das Intervall [0,1}
implicit none
integer                 ::i,n
real                    ::x, x2, xi
real,dimension(0:7)     ::P

x2=1.0-2*x
P(0)=1.0
P(1)=x2
do i=2, n
    P(i)=((2.0*i-1.0)*x2*P(i-1)-(i-1)*P(i-2))/i
enddo
   !Potenzreihe
   !do i=1, n; P(i)=x*P(i-1); enddo
end subroutine

end program
```

```
! Glatter-Methode
! Fit-Funktion: Lineare Interpolation zwischen
! nr äquidistanten Stützstellen
! nr: Zahl der Stützstellen, N= ganze Zahl >(2*qb*rb/pi) ist die
! Anzahl der reell wirkenden Stützstellen
! ra-rb: Radiusbereich, qa-qb: Bereich des Streuvektors
! psi(i): Streukurve an den nq Stützstellen q(i)
! lambda: Feld der "Glättungsparameter"

module messwerte
    integer,parameter           ::nq=42    ! Zahl der Messwerte
    real,parameter              ::qa=0.2, qb=2.8
    real,dimension(0:nq)        ::q, psi
    real,dimension(0:nq,0:8)    ::psi_m
end module messwerte

module verteilung
    integer,parameter           ::nr=92     !nr=Stuetzstellen
    real,parameter              ::ra=0.3, rb=3.3
    real,dimension(0:nr)        ::r, w, xa, d
end module verteilung

program glatter_fit
use messwerte
use verteilung
implicit none
integer                         ::i, j, k, jop
real,parameter                  ::A=1011.0
real                            ::amp, diff, la, gi, gk, h, dq
integer,dimension(1)            ::j_op
real,dimension(0:nr,0:8)        ::xl
real,dimension(0:nr,0:nq)       ::phi
real,dimension(0:nr,0:nr)       ::m, ml, mi, ela
real,dimension(0:8)             ::lambda, rho, chi2

open(unit=2,file='d:\daten\Ilogn2_q.dat', status='old', action='read')
open(unit=4,file='d:\daten\r_verteilung.dat', action='write')
open(unit=5,file='d:\daten\roh_lambda.dat', action='write')
open(unit=6,file='d:\daten\vergleich.dat', action='write')

h=(rb-ra)/nr; dq=(qb-qa)/nq
do i=0, nq
    q(i)=qa+i*dq; read(2,*) psi(i)
    psi(i)=psi(i)*q(i)**3
end do

do i=0, nr; r(i)=ra+h*i; enddo; chi2=0.0
!   lambda=(/0.1,1.0,100.0,1000.0,1.e+4,1.e+6,1.e+8,1.e+10,1.e+12/)
    lambda=(/-1000.,-100.,-50.,-10.,0.5,10.,50.,100.,1000./)
!   lambda=(/-40.,-20.,-10.,-5.,1.,5.,10.,20.,40./)

do k=0, nq
   amp=A/(h*q(k)**3)
   phi(0,k) =amp*(r(1)*(fqr(q(k),r(1))-fqr(q(k),r(0)))-hqr(q(k),r(1)) &
                 +hqr(q(k),r(0)))
   do i=1, nr-1
        gk=r(i+1)*fqr(q(k),r(i+1))-2*r(i)*fqr(q(k),r(i))+ &
                                              r(i-1)*fqr(q(k),r(i-1))
        phi(i,k)=amp*(gk-hqr(q(k),r(i+1))+2*hqr(q(k),r(i))- &
                                                 hqr(q(k),r(i-1)))

   enddo
```

```
        phi(nr,k) =amp*(-r(nr-1)*(fqr(q(k),r(nr))-fqr(q(k),r(nr-1))) &
                    +hqr(q(k),r(nr))-hqr(q(k),r(nr-1)))
enddo

do i=0, nr
    gk=dot_product(phi(i,:),psi)
    d(i)=gk
    do j=i, nr
        gk=dot_product(phi(i,:),phi(j,:))
        m(i,j)=gk; m(j,i)=gk
    enddo
enddo
!Stabilisierung
do j=0, 8
    la=lambda(j)
    ela=0.0
    ela(0,0)=+la*0.0; ela(1,0)=-la !la*0.0 ---> psi_m(0)=xa(0)=0.0
    do i=1, nr-1
        ela(i-1,i)=-la; ela(i,i)=2*la; ela(i+1,i)=-la
    end do
    ela(nr-1,nr)=-la;   ela(nr,nr)=la
    ml=m+ela
    call inverse(ml,mi,nr)
    xa=matmul(mi,d)
    xl(:,j)=xa
    gk=0.0
    do i=0, nr-1
       gk=gk+(xa(i+1)-xa(i))**2
    enddo
    rho(j)=gk    ! Nebenbedingung
enddo
    do j=0, 8
        gi=0.0
        do k=0, nq
            gk=dot_product(xl(:,j),phi(:,k))
            gi=gi+(gk-psi(k))**2; psi_m(k,j)=gk
        enddo
        chi2(j)=gi/nq
    enddo
    j_op=minloc(chi2); jop=j_op(1)-1

! Ergebnis des Fits
print*
write(*,'(a35,I2,2x,a8,e8.3)')'kleinstes Fehlerquadrat fuer lambda'&
        ,jop,',chi**2=',chi2(jop)
do i=0, nr
!   write(4,'(f8.3,(x,e14.6))') r(i), xl(i,jop)         ! optimaler Fit
    write(4,'(f8.3,9(x,e14.6))') r(i), (xl(i,j), j=0,8) ! alle Fits
enddo
do i=0, nq; write(6,'(f8.3,2(x,e9.3))') q(i), psi(i), psi_m(i,jop); end
do
do j=0, 8;  write(5,'(3(x,e9.3))') lambda(j), rho(j), chi2(j); enddo
contains

real function fqr(q,r)
real        ::qr, q, r
    qr=q*r
    fqr=0.5*r*((1.+qr*qr/3.)+0.5*(qr-2.5/qr)*sin(2*qr)+1.5*cos(2*qr))
end function

real function hqr(q,r)
real        ::qr, q, r
```

```
    qr=q*r
    hqr=0.25*r*r*((1.+qr*qr/2.)+(qr-4.5/qr)*sin(2*qr) &
        +0.5*(7.0-4.5/(qr*qr))*cos(2*qr))
end function

end program
```