EIN ALGOL-PROGRAMM FÜR DIE QUANTITATIVE ANALYSE VON STERNSPEKTREN

von

B. Baschek, H. Holweger — Institut für Theoretische Physik und Sternwarte der Universität Kiel
G. Traving — Hamburger Sternwarte

A. Einleitung.

Das im folgenden dargestellte Programm für die Feinanalyse von Sternspektren ist das Ergebnis einer längeren Entwicklung am Institut für Theoretische Physik und Sternwarte der Universität Kiel (1,2) und neuerdings auch an der Hamburger Sternwarte (Bergedorf). Das ursprünglich im Maschinencode geschriebene Programm liegt seit einiger Zeit in der Formelsprache ALGOL 60 (10) vor. Damit ist es unabhängig von der speziellen Maschine formuliert und universell mitteilbar.

Eine Publikation des vollständigen Programmtextes mit einer knappen Beschreibung erscheint als Hilfe und Anregung für entsprechende Untersuchungen an anderen Instituten sowie für einen Vergleich mit deren Ergebnissen sinnvoll. Gleichzeitig werden hiermit die numerischen Methoden der Sternanalysen aus den Instituten in Kiel und Bergedorf zusammengestellt.

Die Darstellung, in der nur die Formeln und eine kurze Funktionsbeschreibung gegeben werden, setzt voraus, daß der Leser mit den Grundlagen der quantitativen Sternspektroskopie vertraut ist und ALGOL 60 beherrscht.

Das Programm wurde auf der Electroligica X1 des Kieler Rechenzentrums entwickelt und vielfach verwendet. Die hier dargestellte Fassung ist für die Rechenanlage TR 4 des Rechenzentrums Hamburg mit ihrem Monitorbetrieb geschrieben und auf ihr ausgeprüft.

B. Physikalische Annahmen und Programmstruktur

Die wesentliche Funktion des Feinanalyseprogramms ist, aus den physikalischen Daten einer Spektrallinie bei gegebener Häufigkeit des Elements die Äquivalentbreite zu berechnen. Hierbei wird vorausgesetzt, daß das Modell der Sternatmosphäre bekannt ist.

Den Rechnungen liegt folgende Schematisierung des Problems zugrunde:

1. homogene planparallel geschichtete Atmosphäre
2. lokales thermodynamisches Gleichgewicht (LTE), wahre Absorption
3. keine Rotation oder Makroturbulenz
4. Voigtprofile für den Linienabsorptionskoeffizienten. Modifizierte asymptotische Näherung nach der Theorie von H. R. Griem für Wasserstofflinien.
5. Keine Variation des kontinuierlichen Absorptionskoeffizienten und der Kirchhoff-Planck-Funktion mit der Wellenlänge im Bereich einer Linie.
6. Keine Variationen der Strahlungsdämpfung mit der optischen Tiefe.

Weitere physikalische Einschränkungen werden nicht gemacht. So wird z. B. die Variation mit der optischen Tiefe von Zustandssummen, Ionisationsgleichgewichten, Kirchhoff-Planck-Funktion, kontinuierlicher Absorption und Linienverbreiterung durch Stoßdämpfung und Dopplereffekt berücksichtigt. Der Linienabsorptionskoeffizient kann aus Komponenten beliebiger Elemente zusammengesetzt sein (Blend).

Zwei Zusatzfunktionen des Programms können wahlweise einzeln oder kombiniert benutzt werden: die Bestimmung der Elementhäufigkeiten 1) durch eine Grobanalyse, 2) in einem iterativen Verfahren unter Verwendung der berechneten Linienintensitäten. Um eine flexible Benutzung zu ermöglichen, ist es vorgesehen, daß durch die Eingabe spezieller Zahlen Sprünge im Programmablauf ausgelöst werden. Gegen die Eingabe fehlerhafter Daten sind Sicherungen einprogrammiert, die soweit wie möglich zur sinnvollen Fortsetzung des Rechenablaufs führen.

In Abb. 1 wird die Blockstruktur des Programms mit Ein- und Ausgabe und den Sprungmöglichkeiten dargestellt.

C. Formeln und Rechenverfahren

In diesem Abschnitt werden die Funktion des Programms beschrieben und die zugrundeliegenden Formeln angegeben, wobei zunächst die Blöcke (I) und dann die von ihnen aufgerufenen Prozeduren (II) besprochen werden. Bezüglich der Details wird auf das Studium des ALGOL-Programms selbst verwiesen.

Der Ablauf der Rechnung wird durch einen Index a gesteuert:

$a < 0$ bewirkt folgende Sprünge im Programm

- -1 nach LIDA
- -2 nach EPS
- -3 nach ION
- -4 nach MODA
- -5 nach OUT (Programmende)

$a = 0$ normale Funktion

$a > 0$ steuert die Berechnung der jeweiligen Linie.

- +1 Ausgabe des Profils
- +2 Iterative Häufigkeitsbestimmung
- +4 Grobanalyse.

Für $a > 0$ bewirkt die Summe der positiven Zahlen die Kombination der entsprechenden Funktionen.

Jede Ionisationsstufe eines Elements wird durch eine Zahl in folgender Weise verschlüsselt:

100 x Ordnungszahl + Ionisationsstufe,

so z. B. neutrales H : 100

einfach ionisiertes Fe: 2601 usw.

Für die Elementhäufigkeiten (nach Atomzahlen) verwenden wir die gebräuchliche Normierung

$$\log \varepsilon_H = 12.00. \qquad (1)$$

Die Summe über alle Elementhäufigkeiten wird durch

$$\sum \varepsilon_i = \varepsilon_H + \varepsilon_{He} \qquad (2)$$

approximiert.

I. Programmblöcke

a) GEDA (Allgemeine Daten)

Eingegeben werden eine mehrfach verwendete Genauigkeitsschranke δ[+)], eine mittlere Tiefe τ_m für die Linienentstehung und die Atomgewichte μ der Elemente, ferner Gewichte w und diskrete Tiefen τ zur Berechnung des Strahlungsstroms

$$F(0) = 2\int_0^\infty E_2(\tau) B(\tau)\, d\tau = \sum_{f=1}^{6} w_f B(\tau_f) \qquad (3a)$$

bzw. der Strahlungsintensität

$$I(0,\vartheta) = \int_0^\infty e^{-\tau/\cos\vartheta} B(\tau)\, d\tau/\cos\vartheta = \sum_{i=1}^{6} w_i B(\tau_i \cos\vartheta). \qquad (3b)$$

Wir verwenden für w_f und τ_f die Zahlenwerte von R. Cayrel (3), für w_i und τ_i die von R. Cayrel und G. Traving (4).

b) MODA (Modelldaten)

Die Daten der Modellatmosphäre werden in Form von Tabellen der folgenden Struktur eingegeben:

1. Anzahl der optischen Tiefen t_{ma} ($3 \leq t_{ma} \leq 31$)
2. Effektive Temperatur T_{eff}, Schwerebeschleunigung log g und Häufigkeitsverhältnis von Helium zu Wasserstoff $\varepsilon_{He}/\varepsilon_H$ nach Atomzahlen
3. Die Skala der repräsentativen optischen Tiefen τ_t. Um die spätere Berechnung der monochromatischen optischen Tiefen τ_λ durch Integration zu erleichtern, machen wir hier die Einschränkung, daß die Schrittweiten in τ_t nur mit Doppelschritten anwachsen (Kontrolle durch das Programm).

+) Die minimale Einsenkung r_{min}, bis zu der das Linienprofil berechnet wird, ist mit δ durch $r_{min} = \delta^2 + 0.002$ gekoppelt.

4. In Abhängigkeit von τ_t:
   a. $\Theta = 5040/T$
   b. $\log P_e$ (Elektronendruck $[\text{dyn cm}^{-2}]$)
   c. $\log P_g$ (Gasdruck $[\text{dyn cm}^{-2}]$)
   d. $-\log \varkappa$ (kontinuierlicher Absorptionskoeffizient pro Kern, zur Skala τ_t zugehörig)
   e. ξ (Geschwindigkeit der Mikroturbulenz $[\text{km/sec}]$)
5. Die Anzahl l_{ma} ($2 \leq l_{ma} \leq 12$) der
6. diskreten Wellenlängen λ_c, für welche
7. $-\log \varkappa_\lambda$, die monochromatischen Absorptionskoeffizienten, als Funktion der optischen Tiefe eingegeben werden. Die kurzwellige Seite einer Absorptionskante wird formal durch negatives λ_c gekennzeichnet.

c) PRE (Vorrechnungen)

Für jede Modellatmosphäre werden in diesem Programmteil durchgeführt

1. die Berechnung des Ionisationsgleichgewichts und der Dopplergeschwindigkeit für Wasserstoff (Index j = 0),
2. die nur vom Modell abhängigen Teile der Berechnung
   a) der Ionisationsgleichgewichte der übrigen Elemente,
   b) der Dämpfung durch Elektronenstoß und
   c) der van der Waals-Dämpfung durch neutrale Atome im Grundzustand.

Zur Kontrolle werden die bisher eingegebenen und berechneten Daten ausgegeben.

d) ION (Ionisationsgleichgewichte)

Vor der Berechnung der Ionisationsgleichgewichte müssen eingelesen werden:

1. die Anzahl j_{ma} ($1 \leq j_{ma} \leq 12$) der Ionisationsstufen verschiedener Elemente, für welche Linien berechnet werden sollen.
2. die Liste dieser Stufen.

Für Wasserstoff wurde das Ionisationsgewicht bereits in PRE berechnet. Die Häufigkeiten der gewünschten Elemente werden zunächst $\log \varepsilon_j = 6$ gesetzt, damit eine Ausgangshäufigkeit für die Grobanalyse verfügbar ist.

Andere ε_j können im Block EPS eingegeben werden.

Die Berechnung der Ionisationsgleichgewichte ist mit der der Zustandssummen

$$Q = g_o + \sum_1^s Q' \tag{4}$$

mit

$$Q' = \sum_{j=1}^{k} g_j \cdot 10^{-\chi_j \Theta} + 2 g_{pr} Q_{as} \cdot 10^{-\chi' \Theta} \tag{5}$$

gekoppelt.

Die Zustandssummen werden zunächst getrennt nach s verschiedenen Konfigurationen des Elternions (Gewicht g_{pr}) berechnet. Mit g_j und χ_j sind die statistischen Gewichte und Anregungsspannungen (in eV) derjenigen Terme bezeichnet, über welche die Summe erstreckt wird. g_o ist das Gewicht des Grundterms, χ' die Ionisationsspannung.

Der Koeffizient Q_{as} im wasserstoffähnlich gerechneten asymptotischen Teil der Zustandssumme wird durch die später zu besprechende Prozedur ASQ (l , n) berechnet. Der asymptotische Teil berücksichtigt die Terme von der effektiven Quantenzahl l an aufwärts, n ist die effektive Kernladungszahl (n = 1 neutrales Atom).

Die Ionisationsrechnung liefert die Besetzungszahl pro Quantenzelle im Grundzustand des n-ten Ionisationszustandes

$$\zeta_n = N_n / Q_n \sum_{i=1}^{n_{ma}} N_i = P_n / S_{n_{ma}} \qquad (6)$$

Die P_n und S_n werden als rekursive Lösung

$$P_1 = 1 \; ; \quad P_{n+1} = P_n \cdot \frac{\text{const.}}{\theta^{5/2} P_e} \cdot 10^{-(\chi_n - n \cdot \Delta\chi)\theta} \qquad (7)$$

und

$$S_1 = 1 \; ; \quad S_{n+1} = S_n + Q_{n+1} \cdot P_{n+1} \qquad (8)$$

des Systems der Saha-Gleichungen

$$\frac{N_{n+1}}{N_n} = \frac{Q_{n+1}}{Q_n} \cdot \frac{\text{const.}}{\theta^{5/2} P_e} \cdot 10^{-(\chi_n - n \cdot \Delta\chi)\theta} \qquad (9)$$

erhalten. Die Erniedrigung $n \cdot \Delta\chi$ der Ionisationsenergie χ_n wird nach der Debye'schen Theorie berechnet:

$$n \cdot \Delta\chi = 4.98 \cdot 10^{-4} n \theta \sqrt{P_e} \qquad [eV] \; . \qquad (10)$$

Das Programm benötigt die Daten in der folgenden Form:

Elementbezeichnung Anzahl der Ionisationsstufen n_{ma}

s	g_o				
k	$2 \cdot g_{pr}$	χ'	l	s - mal	n_{ma} - mal
g_j	χ_j	k - mal			

Diese Größen sind universelle Daten der Atome.

Um die Anzahl k der benötigten Terme möglichst klein zu halten, haben wir die wirklichen Energieniveaus durch fiktive ersetzt, deren g_j und χ_j und durch Tschebyscheff-Approximation gefunden wurden (11). Eine Zusammenstellung der g_j und χ_j für die wichtigsten Elemente ist publiziert (13).

Außer den Ionisationsgleichgewichten werden in ION die Dopplergeschwindigkeiten

$$v_{D,j} = \frac{1}{c}\sqrt{\xi^2 + \frac{83.81}{\theta \cdot \mu_j}} \qquad (11)$$

für die Elemente j der Liste berechnet.

Am Ende des Blocks erfolgt zur Kontrolle die Ausgabe der log ζ_n.

e) EPS (Häufigkeiten)

Dieser durch einen Sprung erreichbare Block ermöglicht das Einlesen geänderter Häufigkeiten log ε_j.

f) LIDA (Liniendaten)

Die Liniendaten werden in folgender Reihenfolge eingelesen:

1. Index a (a $<$ 0: Sprung; a $\geq$ 0: Steuerung des Rechenablaufs)
2. Anzahl b_{ma} der Komponenten der Linie ($1 \leq b_{ma} \leq 10$).
3. mu (mu $>$ 0: Strahlungsintensität für cos ϑ = mu bzw. mu = -1: Strahlungsstrom).
4. gemessene Wellenlänge λ_m in Å und Äquivalentbreite W_m in mÅ .

Die weiteren Daten +) werden in der angegebenen Reihenfolge b_{ma} -mal benötigt:

5. Elementindikation el und Multiplett-Nummer
6. Laboratoriumswellenlänge λ in Å
7. Anregungsspannung χ_b des unteren Termes in eV
8. Oszillatorenstärke in der Form log gf
9. Wechselwirkungskonstanten für van der Waals-Dämpfung und quadratischen Starkeffekt, -log C_6 und -log C_4

Definition C_p $\left[cm^p \text{ sec}^{-1}\right] = r^p \cdot \Delta\omega$ (12)

+) Bei Verwendung von Lochkarten ist es zweckmäßig, die folgenden Angaben, welche physikalische Parameter der Linie sind, auf getrennte Karten zu lochen.

Sonderfälle:

a) $m = -\log C_6 < 10$ bewirkt Berechnung der van der Waals-Dämpfung nach A. Unsöld (14):

$$\log C_6 = \log C_6\,(\text{Unsöld}) + m \qquad (13)$$

mit

$$C_6\,(\text{Unsöld}) = 1.01 \cdot 10^{32}\left[\left(\frac{13.6}{\chi-\chi_o}\right)^2 - \left(\frac{13.6}{\chi-\chi_b}\right)^2\right] . \qquad (14)$$

χ Ionisationsspannung; χ_o Anregungsspannung des oberen Terms.

b) $-\log C_4 < 0$: Dämpfung durch Elektronenstoß wird nicht berücksichtigt.

10. Strahlungsdämpfung γ_{rad} in 10^8 sec^{-1}.

$\gamma_{rad} < 0$ bewirkt Berechnung der Strahlungsdämpfung nach der klassischen Elektronentheorie

$$\gamma_{rad} = 2.21 \cdot 10^7/\lambda^2 \qquad [\lambda \text{ in } \text{Å}] . \qquad (15)$$

Für Wasserstofflinien ist unter 8, 9 und 10 einzulesen:

8 H. log K + 17, definiert wie bei G. Traving (12)

9 H. untere und obere Hauptquantenzahl (m und n) des Übergangs

10 H. Zahl ohne Bedeutung.

Entsprechendes gilt auch für He II-Linien usw., deren Berechnung im vorliegenden Programm nicht vorgesehen ist.

Unter anderen wird geprüft, ob die im Sternspektrum gemessene Wellenlänge λ_m bis auf 5 Å mit dem Mittelwert der Wellenlängen des Blends übereinstimmt, ob sie im verfügbaren Wellenlängenbereich der kontinuierlichen Absorptionskoeffizienten liegt und ob die Daten ζ_n (Gl. 6) und v_D (Gl. 11) für das betreffende Element zur Verfügung stehen. Ist dies der Fall, so werden die Liniendaten durch das Array b_j den in ION berechneten Tabellen zugeordnet. Abschließend wird u. a. die der Linienentstehungstiefe τ_m entsprechende Dopplerbereite $\Delta\lambda_D$ berechnet.

g) CONT (kontinuierliche Absorption und Kirchhoff-Planck-Funktion)

Aus der Tabelle der monochromatischen Absorptionskoeffizienten bei den diskreten Wellenlängen λ_c wird für die Wellenlänge λ_m der Linie $\eta_c = \varkappa_\lambda/\varkappa$ durch Interpolation in allen Tiefen bestimmt. Gleichzeitig wird

$$B_\lambda = \left(e^{28541.7 \cdot \Theta/\lambda_m} - 1\right)^{-1} \qquad (16)$$

als Funktion von τ sowie der Strahlungsstrom $F_c(0)$ bzw. die Intensität $I_c(0)$ im Kontinuum berechnet.

h) KAPGAM (Linienabsorptionskoeffizient und Dämpfung)

Wir schreiben das Verhältnis $\eta(\Delta\lambda)$ von Linien- zu kontinuierlichem Absorptionskoeffizienten in der Form

$$\eta(\Delta\lambda) = \eta_o \cdot \Phi(\Delta\lambda) \quad . \tag{17}$$

Die Profilfunktion Φ ist für Metallinien gleich der Voigt - Funktion

$$\Phi = H(a,v) \tag{18}$$

mit den Parametern

$$a = \gamma / 2\Delta\omega_D \quad \text{und} \quad v = |\Delta\lambda/\Delta\lambda_D| \, .$$

Die Dopplerbreite ist

$$\Delta\lambda_D = \lambda^2/2\pi c \cdot \Delta\omega_D = \lambda \cdot v_D \tag{19}$$

Die Dämpfung γ setzt sich aus Strahlungsdämpfung γ_{rad}, Elektronendämpfung γ_e und van der Waals-Dämpfung γ_v durch Stoß mit neutralen H- und He-Atomen im Grundzustand zusammen:

$$\gamma = \gamma_{rad} + \gamma_e + \gamma_v \quad . \tag{20}$$

In dem Temperatur- und Druckbereich, in dem γ_v von Bedeutung ist, können die Unterschiede in Anregung und Ionisation von H und He vernachlässigt werden, also

$$\gamma_v = \gamma_H + \gamma_{He} = \gamma_H \left(1 + G \, \frac{\varepsilon_{He}}{\varepsilon_H}\right) \quad . \tag{21}$$

gesetzt werden. Die Konstante G ist durch die Polarisierbarkeiten p und Atomgewichte μ genähert festgelegt:

$$G = \left(\frac{P_{He}}{P_H}\right)^{2/5} \cdot \left(\frac{\mu_H}{\mu_{He}}\right)^{3/10} = \frac{1}{2.4192} \quad . \tag{22}$$

Aus den Wechselwirkungskonstanten C_4 und C_6 ergibt sich für die entsprechenden Dämpfungen

$$\log \gamma_e \, [10^8 \, \text{sec}^{-1}] = \frac{2}{3} \log C_4 + 7.7621 + \frac{5}{6} \log \Theta + \log P_e \tag{23}$$

und

$$\log \gamma_H \left[10^8 \mathrm{sec}^{-1}\right] = \frac{2}{5} \log C_6 + 8.6735 + \frac{7}{10} \log \Theta + \log P_H \quad . \qquad (24)$$

Der Partialdruck P_H der H-Atome im Grundzustand ist aus dem Gasdruck P_g zu berechnen:

$$P_H = \frac{\varepsilon_H}{\Sigma \varepsilon_i} \cdot P_g \cdot 2\zeta_H \quad . \qquad (25)$$

Als Profilfunktion für Wasserstofflinien verwenden wir die für den Linienflügel gültige asymptotische Form (siehe Prozedur ASH)

$$\Phi = S(R, \Delta\lambda) \quad .$$

Der Parameter R wird in der von H. R. Griem (5) angegebenen Approximation

$$R = A(n,m) \cdot \sqrt{\Theta} \cdot \left[4.1007 - \log \sqrt{P_e}\, n^2 \Theta^{3/2}\right] \qquad (26)$$

mit

$$A(n,m) = 0.06479 \left(n^5 + m^5\right) / \left(n^2 m^2 \sqrt{n^2 - m^2}\right) \qquad (27)$$

berechnet (n obere, m untere Quantenzahl).

Der Absorptionskoeffizient in der Linienmitte η_o für eine Metallinie ist

$$\eta_o = \frac{2\pi^{3/2} e^2}{mc} \cdot \frac{g f}{\Delta\omega_D} \cdot \frac{\varepsilon}{\Sigma \varepsilon_i} \cdot \zeta \cdot 10^{-\chi_b \Theta} \cdot \frac{1 - e^{-c_2/\lambda T}}{\varkappa} \quad , \qquad (28)$$

wobei $1 - e^{-c_2/\lambda T} = e^{-c_2/\lambda T} / B_\lambda$ der Faktor für erzwungene Emission und $\varkappa$ der kontinuierliche Absorptionskoeffizient pro Kern ist.

Die Berechnung von η_o erfolgt in der Form

$$\eta_o = A \cdot \frac{\varepsilon}{\varepsilon_H} \cdot \frac{10^{-\chi' \Theta}}{v_D B_\lambda \varkappa} \qquad (29)$$

mit

$$\chi' = \chi_b + \frac{12394.7}{\lambda} \qquad (30)$$

und

$$\log A = -32.3014 + \log \frac{\varepsilon_H}{\varepsilon_H + \varepsilon_{He}} + \log gf\lambda \quad . \qquad (31)$$

Der entsprechende Ausdruck für η_o bei Wasserstofflinien ist

$$\eta_{o,H} = (2.603 \cdot e)^{3/2} \cdot g \cdot k \cdot \frac{\varepsilon_H}{\Sigma\varepsilon_i} \cdot \zeta_H \cdot \frac{1 - e^{-c_2/\lambda T}}{\varkappa} \cdot \frac{P_e}{kT} \quad , \qquad (28\,H)$$

wobei $g = 2$ m^2 das statistische Gewicht des unteren Terms ist.

Im Block KAPGAM werden η_o und α (bzw. R) für jede Komponente des Blends als Funktion der optischen Tiefe τ berechnet.

Wahlweise (s. Index a) kann bei Einzellinien die Elementhäufigkeit ε aus der gemessenen Äquivalentbreite W_m und einem für beliebiges ε'' berechneten η_o [+)] durch eine Grobanalyse bestimmt werden:

$$\log \varepsilon' = \log \varepsilon'' + \log \left(\frac{R_c}{r(0)} - 1 \right) \qquad (32a)$$

$$\log \varepsilon = \log \varepsilon' + UCG(W_m) \quad . \qquad (32b)$$

Die Prozedur UCG liefert $\log C$, die Abszisse der universellen Wachstumskurve (s. Gleichung (48). Durch iterative Verwendung von Gleichung (32a) wird die Häufigkeit ε' so bestimmt, daß $\log C$ gleich 0 ist. Gleichung (32a) beruht auf der von K. Hunger (8) angegebenen Wachstumskurve für die Zentraltiefen.

i) EQUIV (Äquivalentbreite)

Die Äquivalentbreite W_{cal} wird nach dem Integrationsverfahren von Simpson mit automatischer Festlegung der Schrittweite $\Delta\Delta\lambda$ berechnet. Die minimale Schrittweite ist 10 mÅ, so daß $\Delta\Delta\lambda$ immer ein ganzzahliges Vielfaches von 10 mÅ ist.

Die Einsenkung r_p im Abstand $\Delta\lambda_p$ von der Linienmitte (bzw. von der kurzwelligsten Komponente bei Blends) liefert die Prozedur DEP, welche gleichzeitig das Resultat speichert, um wiederholtes Berechnen eines Profilpunkts zu vermeiden.

Die Integration von W_{cal} wird abgebrochen, wenn bei einem Simpson-Schritt das Mittel der Einsenkungen kleiner als r_{min} (vgl. GEDA) wird.

+) Wenn keine Häufigkeit eingegeben wurde, so wird η_o zunächst formal mit $\log \varepsilon'' = 6$ berechnet.

k) FIN (Schlußhandlungen)

Zusammen mit den Linienkonstanten werden die berechnete Äquivalentbreite W_{cal}, die Anzahl p_{ma} der Profilpunkte und der Strahlungsstrom (bzw. die Intensität) im Kontinuum

$$\log F_\lambda = 27.0757 + \log F_c(0) - 5 \log \lambda_m \qquad (33)$$
$$F_\lambda \text{ in } [\text{erg cm}^{-2} \text{ ster}^{-1} \text{ sec}^{-1} / \Delta\lambda = 1\text{Å}]$$

ausgegeben (PROT).

Je nach Wahl kann das gespeicherte Profil r ($\Delta\lambda$) ausgegeben werden (PROF).

Ebenfalls wahlweise kann durch Iteration die Elementhäufigkeit zu einer gegebenen Äquivalentbreite W_m bestimmt werden:

Von einer Anfangshäufigkeit mit zugehörigem W_{cal} ausgehend wird die Häufigkeit durch Vergleich von W_{cal} mit W_m verbessert. Der erste Iterationsschritt verwendet hierfür die universelle Wachstumskurve; bei den folgenden Schritten wird in einer Tabelle ($\log W_{cal}$, $\log \varepsilon$) interpoliert, welche bei jedem Schritt um ein Wertepaar wächst.

Die Iteration wird abgebrochen, wenn die Abweichungen von gemessener und berechneter Äquivalentbreite unter einer vorgegebenen Schranke δ liegen

$$\left| \ln (W_m / W_{cal}) \right| < 2\delta \quad .$$

Jedoch werden höchstens 5 Schritte (gezählt durch Index it) durchgeführt.

II. Prozeduren

a) MIN, MAX und REST

An der TR 4 verfügbare Code-Prozeduren, welche die minimale bzw. maximale Zahl von zweien sowie den Rest bei ganzzahliger Division berechnen.

b) LOG (X) und XP (X)

Logarithmus $\log_{10} x$ und Exponentiation 10^x.

c) SUM (I, J, R)

Summation $\sum_{i=1}^{j} R_i$

d) SEARCH (K, X, I, R, N)

$R_i = R(i)$ ist eine monoton wachsende Funktion, welche für die ganzen Zahlen $1 \leq i \leq n$ erklärt ist (z.B. eine Tabelle). Einem beliebigen Argument x wird durch SEARCH die Zahl k zugeordnet, für welche

$$R_k < X < R_{k+1}$$

ist, wenn x im Wertebereich der Tabelle liegt. Falls $x < R_1$ wird $k = 1$, falls $x > R_n$ wird $k = n - 1$ gesetzt.

e) ORDER (X, Y, A, B, K, N)

Der Funktionswert y(x) wird seinem Argument x nach in die Tabelle $b_i(a_i)$ $(1 \leq i \leq n)$ mit wachsenden Argumenten a_i auf den Platz k eingeordnet. Die Länge der Tabelle nach dem Einordnen ist n + 1.

f) IPOL (X, A, B, N)

Zu dem Argument x wird aus der nach wachsenden Argumenten a_i geordneten Tabelle $b_i(a_i)$ der Länge n die Funktion b(x) durch Inter- bzw. Extrapolation bestimmt.

Ist $n > 2$ und liegt x innerhalb des Argumentbereichs der Tabelle, so wird immer quadratisch aus den nächstgelegenen Funktionswerten $b(a_0)$, $b(a_1)$ und $b(a_2)$ interpoliert.

Die Interpolation ist hyperbolisch, wenn b zwischen a_0 und a_2 monoton verläuft:

$$b(x) = b(a_0) + (x-a_0)\cdot[b_0 b_1]\left\{1-(x-a_1)\,\frac{[b_0 b_1 b_2]}{[b_0 b_2]}\right\}^{-1} , \qquad (34)$$

anderenfalls wird parabolisch interpoliert:

$$b(x) = b(a_0) + (x-a_0)\left\{[b_0 b_1] + (x-a_1)\,[b_0 b_1 b_2]\right\} \qquad (35)$$

Zur Abkürzung wurde gesetzt:

$$[b_i b_j] = \frac{b(a_i) - b(a_j)}{a_i - a_j} ,$$

$$[b_i b_j b_k] = \frac{[b_i b_j] - [b_j b_k]}{a_i - a_k} . \qquad (36)$$

Wenn $n = 2$ oder wenn n außerhalb des Tabellenbereichs liegt, wird linear inter- oder extrapoliert.

g) H(A, V)

Die Voigt-Funktion $H(\alpha, v)$ wird für den gesamten Parameterbereich α, v durch verschiedene Reihenentwicklungen mit einem Fehler $< 1\,\%$ dargestellt. Die Koeffizienten, die zum Teil durch Tschebyscheff-Polynome approximiert werden, sind bei G. Traving (12) angegeben.

h) ASH (R, D)

Asymptotische Formel für den Flügel der Wasserstofflinien in Anlehnung an H. R. Griem (6)

$$S(R,\Delta\lambda) = \frac{1+R^*\sqrt{\Delta\lambda}}{\Delta\lambda^{5/2}} \qquad (37)$$

R^* ist durch die Interpolationsformel

$$\left(\frac{1}{R^*\sqrt{\Delta\lambda}}\right)^4 = 1+\left(\frac{1}{R\sqrt{\Delta\lambda}}\right)^4 \qquad (38)$$

mit dem Koeffizienten R der älteren Theorie nach Kolb-Griem-Shen verknüpft. Hierdurch wird der Übergang von der Stoßtheorie zur statistischen Verbreiterung durch Elektronen genähert berücksichtigt.

i) SUMETA (T)

Der Integrand K zur Berechnung der monochromatischen Tiefen wird in der Tiefe τ_t für ein festes $\Delta\lambda$ in der Linie aus kontinuierlichem und Linienabsorptionskoeffizient zusammengesetzt (siehe KAPGAM):

$$K = \eta_c + \sum_{b=1}^{b_{ma}} \eta \qquad (39)$$

k) FLUX (DL)

Die monochromatische Tiefe τ_λ (im Kontinuum oder für ein festes $\Delta\lambda$ in der Linie) als Funktion von τ,

$$\tau_\lambda(\tau) = \int_0^\tau K(\tau')\,d\tau' \qquad (40)$$

wird nach der Simpsonschen Regel integriert, wobei K durch Gleichung (39) gegeben ist.

Die τ -Skala muß hierfür so gewählt sein, daß die Schrittweite sich jeweils nur nach einem Doppelschritt ändert (vgl. MODA).

Der Zuwachs des Integrals für den ganzen Doppelschritt ist

$$\Delta\tau_\lambda(\tau_2) = \frac{\Delta\tau}{3}\left[K_0 + 4K_1 + K_2\right] , \qquad (41)$$

für den halben Schritt

$$\Delta\tau_\lambda(\tau_1) = \frac{\Delta\tau}{12}\left[5K_0 + 8K_1 - K_2\right] . \qquad (42)$$

Der Strahlungsstrom bzw. die Strahlungsintensität wird durch die Summe (3a) bzw. (3b) berechnet, wobei die Kirchhoff-Planck-Funktion für die diskreten τ_f (bzw. $\tau_i \cos\vartheta$) aus der Tabelle $B_\lambda(\tau)$ interpoliert wird.

Wächst das Integral über einem Doppelschritt zu ungleichmäßig an, d.h. ist

$$\Delta\tau_\lambda(\tau_1) < \tfrac{1}{4}\,\Delta\tau_\lambda(\tau_2) \quad \text{oder} \quad \Delta\tau_\lambda(\tau_1) > \tfrac{3}{4}\,\Delta\tau_\lambda(\tau_2) \quad ,$$

so wird die quadratische Interpolation der $B_\lambda\,(\tau_\lambda(\tau))$ ungenau. Wir beschränken daher den ohnehin nur mit geringerer Genauigkeit erhaltenen Wert des Integrals für den halben Doppelschritt künstlich zwischen

$$\frac{1}{4} \le \frac{\Delta\tau_\lambda(\tau_1)}{\Delta\tau_\lambda(\tau_2)} \le \frac{3}{4} \quad .$$

Am Endpunkt eines Doppelschritts ist $\Delta\tau_\lambda(\tau_2)$ jeweils exakt im Rahmen der Simpsonschen Regel.

l) DEP (DL)

Der Profilspeicher $(\Delta\lambda_p, r_p)$ wird durchsucht, ob $r(\Delta\lambda)$ bereits von einem vorangehenden Schritt bei der Integration von W_{cal} her verfügbar ist. Wenn nicht, so wird

$$r(\Delta\lambda) = 1 - \frac{F(\Delta\lambda)}{F_c} \qquad (43)$$

berechnet und das Wertepaar $(\Delta\lambda_p, r_p)$ gespeichert.

m) ASQ (L, N)

Asymptotischer Teil der Zustandssumme, der von der effektiven Quantenzahl l an wasserstoffähnlich gerechnet wird.

$$Q_{as} = \sum_{m=l}^{h} m^2 e^{a/m^2} \qquad (44)$$

mit $a = 31.321\, n^2\Theta$, n effektive Kernladungszahl.

Das Abschneiden der Zustandssumme erfolgt nach der Debye'schen Theorie der Wechselwirkung der Elektronen mit dem gesamten Plasma (vgl. z.B. (7)) bei einer effektiven Quantenzahl

$$h = \sqrt{\frac{13.595 \cdot n}{\Delta\chi}} \quad . \qquad (45)$$

$\Delta\chi$ ist die in ION (Gleichung 8) definierte Erniedrigung der Ionisationsspannung.

Nach Entwicklung des Exponenten in der Summe (44) ergibt sich

$$Q_{as} = \sum_{l}^{h} m^2 + \sum_{l}^{h} a + \frac{1}{2}\sum_{l}^{h} \frac{a^2}{m^2} + \cdots\cdots \qquad (46)$$

$$= \frac{1}{3}\left[h(h+1)(h+\tfrac{1}{2}) - l(l+1)(l+\tfrac{1}{2})\right] + a(h-l) + \frac{a^2}{2}\left(\frac{1}{l} - \frac{1}{h}\right) .$$

n) UCG (W)

Die Abszisse log C der universellen Wachstumskurven für M.E.-Modell mit den numerischen Werten nach K. Hunger (8) hängt von den Parametern

$$\Omega = \log \frac{W}{2\Delta\lambda_D R_c} \qquad (47)$$

und

$$a = \log 2\alpha$$

ab, die für die mittlere Linienentstehungstiefe τ_m berechnet werden. Es bedeuten: W Äquivalentbreite, $\Delta\lambda_D$ Dopplerbreite, $R_c = 1 - B_\lambda(0)/F_c$ maximale Linieneinsenkung und α Dämpfungsparameter (vgl. KAPGAM).

Nach K. Hunger ist

$$\log C = \begin{cases} \Omega + 0.08 & \text{für} \quad \Omega < -1 \\ \sum_{i=0}^{3} w_i \Omega^i - \max(0,\Omega)\cdot\frac{a+1}{0.9} & \text{für} \quad -1 \le \Omega < 0.9 \\ 2\Omega - a + 0.157 & \text{für} \quad 0.9 \le \Omega \end{cases} \qquad (48)$$

Die Wachstumskurve ist im flachen Teil für $\log 2\alpha_0 = -1$ durch ein Tschebyscheff-Polynom in Ω mit den Koeffizienten

$w_0 = 0.563$ $\quad$ $w_2 = 0.643087$

$w_1 = 2.337445$ $\quad$ $w_3 = -0.245805$

dargestellt. Die Abweichungen des Dämpfungsparameters von α_o werden nur näherungsweise berücksichtigt.

o) NEW (DEPS)

Änderung des Linienabsorptionskoeffizienten η_o für eine neue Elementhäufigkeit.

p) TEXT (L)

Ausgabe von Klartext mit anschließendem Sprung.

q) CHEK (K, L, M, N)

Eine Zahl x wird gelesen und wie folgt geprüft:

1) Ist x ganzzahlig im Intervall (-1, -5) ?
   Wenn ja, so wird ein Programmsprung ausgeführt

nach		falls x =
	LIDA	-1
	EPS	-2
	ION	-3
	MODA	-4
	OUT	-5

2) Ist x ganzzahlig im Intervall (1, m) ?

Wenn ja, so nimmt k den Wert x an, anderenfalls liegt ein Fehler vor. In diesem Fall erfolgt ein Sprung nach jump [n] .

r) IN (MI, MA, A)

Einlesen einer Tabelle a_i von i = mi bis i = ma.

s) PROT und PROF

Ausgabeprozeduren für Liniendaten und Profil (siehe FIN).

III. Ein- und Ausgabeprozeduren der TR4

Die Eingabe von Daten geschieht durch die Prozedur READ (A, B, ...)

Die Ausgabeprozeduren OUTPUT entsprechen dem "Proposal for Input-Output-Conventions in ALGOL 60" (9).

Literatur

(1) Baschek, B. u. G. Traving: Z. Astrophysik 54, 7 (1962)

(2) Baschek, B., G. Bode, W.-H. Kegel, K. Kodaira, K. Kohl u. G. Traving: Proc. First Harvard-Smithsonian Conf. on Stellar Atmospheres. SAO Special Report 167, 253 (1964)

(3) Cayrel, R.: Suppl. Ann. d'Astrophys. No 6 (1958)

(4) Cayrel, R. u. G. Traving: Z. Astrophysik 50, 239 (1960)

(5) Griem, H. R.: Astrophys. J. 132, 883 (1960)

(6) Griem, H. R.: Astrophys. J. 136, 422 (1962)

(7) Griem, H. R.: Phys. Rev. 128, 997 (1962)

(8) Hunger, K.: Z. Astrophysik 39, 36 (1956)

(9) Knuth, D. E., L. L. Bumgarner, P. Z. Ingerman, J. N. Merner, D. E. Hamilton, M. P. Lietzke u. D. T. Ross: Communications of the ACM 7, 273 (1964)

(10) Naur, P. ed.: Report on the Algorithmic Language Algol 60, Regnecentralen Copenhagen 1960

(11) Schlender, B. u. G. Traving: Z. Astrophysik 61, 92 (1965)

(12) Traving, G.: Landolt-Börnstein, Zahlenwerte und Funktionen, Neue Serie Gruppe VI Band 1, Astronomie und Astrophysik 5.3.4 pg 445. Springer-Verlag (1965)

(13) Traving, G., B. Baschek u. H. Holweger: Abhandlungen aus der Hamburger Sternwarte VIII, 3 (1966)

(14) Unsöld, A.: Physik der Sternatmosphären, 2. Aufl. Berlin - Göttingen - Heidelberg. Springer 1955.

Symbole

Im Programm:	In den Formeln:	Bedeutung:
a		Sprung- bzw. Steuerindex
alpha	α bzw R	Dämpfungsparameter
b, bma		Index und Anzahl der Blendkomponenten
bj	b_j	Zuordnungsarray Blendkomponente-Elementliste
blam	B_λ	Kirchhoff-Planck-Funktion
boa		Boole'sches Array, berechnet aus $a > 0$
chi	χ_j	Anregungsspannung eines Terms
chib	χ_b	Anregungsspannung der Linie
chij		Ionisationsspannung in der Elementliste
chin	χ_n	Ionisationsspannung
conti		Boole'sche Variable: Kontinuum oder Linie?
dchi	$\Delta\chi$	Erniedrigung der Ionisationsspannung
ddl	$\Delta\Delta\lambda$	Schrittweite bei Integration des Profils
del	δ	Genauigkeitsschranke
dl, dlp	$\Delta\lambda, \Delta\lambda_D$	Abstand im Profil vom Zentrum der kurzwelligsten Komponente
dla	$\Delta\lambda$	Abstände der Blend-Komponenten voneinander
dlam	$\Delta\lambda$	Abstand eines Profilpunkts von den einzelnen Komponenten
dldop	$\Delta\lambda_D$	Dopplerbreite
dtau		Gewichtete Differenzen der optischen Tiefen τ_t
efft	T_{eff}	Effektivtemperatur
el		Elementindikation
elb		Elementindikation der Linie
elj		Elementindikation in der Elementliste
eldp	$\log \gamma_e / C_4^{2/3}$	Elektronenstoßdämpfung
epit		Elementhäufigkeit bei der Iteration
epsb	$\log \varepsilon_b$	Elementhäufigkeit für die Blendkomponente
epsj	$\log \varepsilon_j$	Elementhäufigkeit in der Elementliste
eta	$\eta(\Delta\lambda)$	Verhältnis von Linien zu kontinuierlichem Absorptionskoeffizient
etac	$\eta_c = \varkappa_\lambda / \varkappa_c$	
fc	$F_c(0)$ bzw $I_c(0)$	Strahlungsstrom bzw. Intensität im Kontinuum
g	g_j	statistisches Gewicht eines Terms
go	g_0	statistisches Gewicht des Grundterms
gpr	$2 \cdot g_{pr}$	statistisches Gewicht des Elternions x 2
he	$\varepsilon_{He} / \varepsilon_H$	Heliumhäufigkeit

hy		Boole'sche Variable: Wasserstoff?
i		Summationsindex
it		Zählindex bei der Iteration
j, jma		Index und Anzahl der Ionisationsstufen
k		Anzahl der (g_j , χ_j)-Paare
l, lma		Index und Anzahl der Wellenlängen (für $\varkappa_\lambda$)
lam	λ	Laboratoriumswellenlänge der Linie
lamax		Abstand der äußersten Blendkomponenten
lamc	λ_c	diskrete Wellenlängen (für $\varkappa_\lambda$)
lc 4	$-\log C_4$ (bzw n)	Wechselwirkungskonstante für Elektronenstoß-dämpfung
lc 6	$-\log C_6$ (bzw m)	Wechselwirkungskonstante für van der Waals-dämpfung
lgf	log gf (bzw log K+A)	Oszillatorenstärke
lgg	log g	Schwerebeschleunigung
list		Liste der Elemente
lk	$-\log \varkappa$	mittlerer kontinuierlicher Absorptionskoeffizient
lkl	$-\log \varkappa_\lambda$	monochromatischer kontinuierlicher Absorptions-koeffizient
lpg	$\log P_g$	Gasdruck
lpe	$\log P_e$	Elektronendruck
lwit	$\log W_{cal}$	Äquivalentbreite bei der Iteration
mlam	λ_m	gemessene Wellenlänge der Linie
mtau	τ_m	mittlere Linienentstehungstiefe
mt		Tiefenindex zu τ_m
mu	$\cos\vartheta$	Austrittswinkel der Strahlung
mult		Multiplett-Nummer
my	μ	Atomgewicht
n, nma		Effektive Kernladungszahlen und Anzahl der Ionisationsstufen
p, pma		Index und Anzahl für Profilpunkte
pn	P_n	für Ionisationsgleichgewichte, Gleichung (6)
q	Q	Zustandssumme
rad	γ_{rad}	Strahlungsdämpfung
rmin	r_{min}	minimale Linieneinsenkung
rp	r_p	Linieneinsenkung im Abstand $\Delta\lambda_p$
s		Anzahl der verschiedenen Elektronenkonfigurationen (siehe Gleichung (4))
saha	$const/\Theta^{5/2} P_e$	Saha-Gleichung (7)
single		Boole'sche Variable: Einzellinie?
sn	S_n	für Ionisationsgleichgewichte, Gleichung (6)

t, tma		Index und Anzahl der optischen Tiefen
tau	τ_t	diskrete optische Tiefen
tflx tint ta	τ_f τ_i	diskrete optische Tiefen für Berechnung des Strahlungsstromes bzw. der Intensität
theta	$\theta = 5040/T$	
th	θ bzw $-\theta$	Hilfsgröße
tlam	τ_λ	monochromatische optische Tiefen
vdop	$v_{D,j}$	Dopplergeschwindigkeit / c
vdw	$\log \gamma_H / C_6^{2/5}$	van der Waals-Dämpfung
wcal	W_{cal}	berechnete Äquivalentbreite
wflx wint wa	w_f w_i	Gewichte für die Berechnung des Strahlungsstromes bzw. der Intensität
wm	W_m	gemessene Äquivalentbreite
xi	ξ	Mikroturbulenzgeschwindigkeit
zeta	ζ_n	Besetzungszahl des n-ten Ionisationszustands
u, v, w, x, y, z		Hilfsgrößen

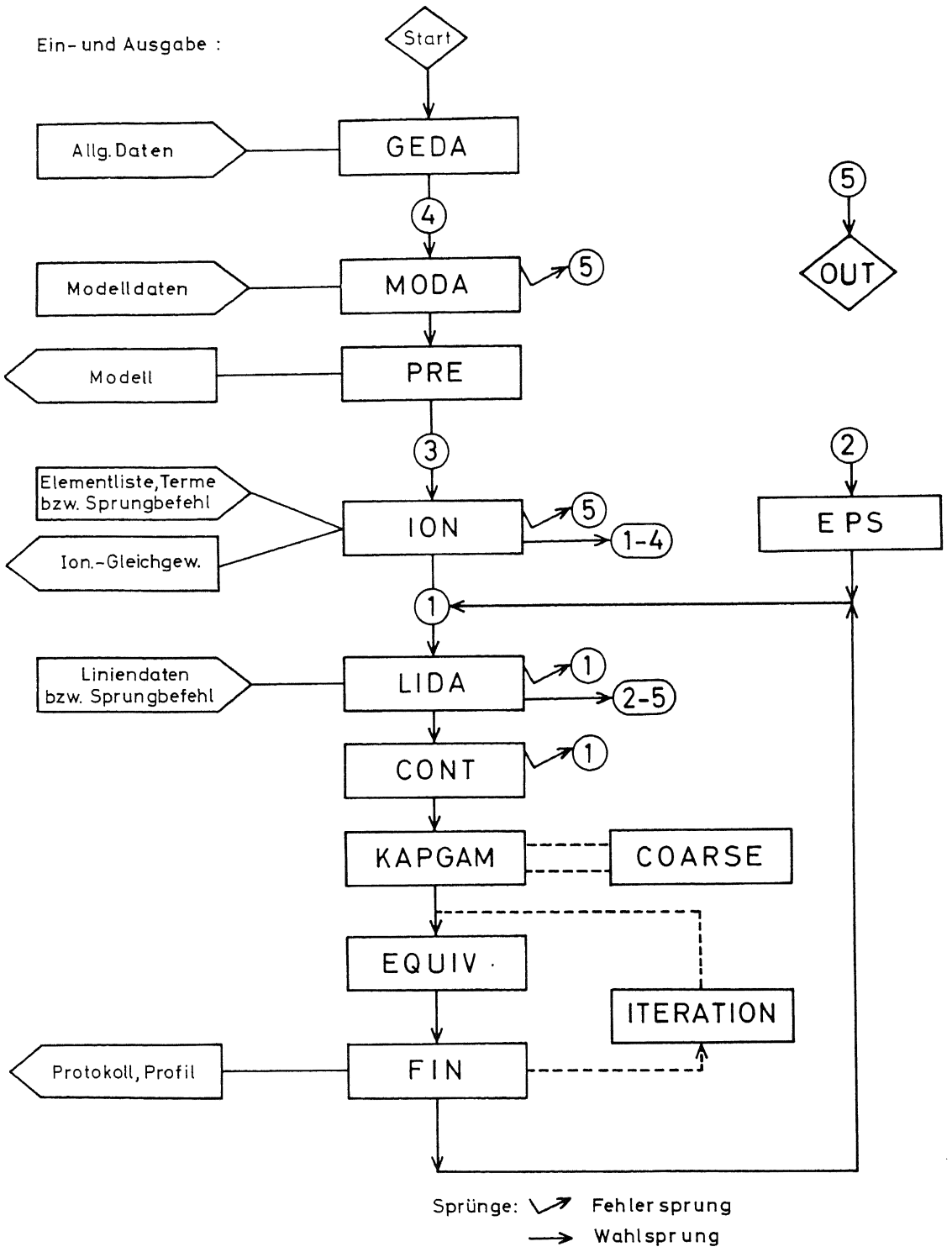

Abb. 1 Blockdiagramm

```
begin comment Analyse 65;
    integer a, b, bma, j, jma, l, lma, n, nma, p, pma, t, tma, el, it, mt, i, k, s;
    real del, mtau, rmin, efft, lgg, he, th, go, gpr, mlam, lamax, dl, ddl, dldop, wm, wcal, fc, mu, u, v, w, x, y, z;
    boolean conti, single;
    integer array list[1:12], elj[0:12], bj,elb,mult[1:10];
    real array tflx,wflx,tint,wint,ta,wa[1:6], my[2:92], tau,dtau,theta,lpe,lpg,lk,xi,etac,blam,tlam,vdw,eldp,saha,
               dchi,sn,q[1:31], lamc[1:12], epsj,chij[0:12], chin[1:7], g,chi[1:8], lam,lgf,chib,lc6,lc4,rad,epsb,
               dlam,dla[1:10], dlp,rp[1:400], lwit,epit[1:5], vdop,zeta[1:31,0:12], lkl[1:31,1:12], pn[1:31,1:7],
               alpha,eta[1:31,1:10];
    boolean array boa[1:3], hy[1:10];

    switch jump := lida, eps, ion, moda, out;

    real procedure min; code;

    real procedure max; code;

    integer procedure rest; code;

    real procedure log(x);
    real x;
    log := .4342945 × ln(x);

    real procedure xp(x);
    real x;
    xp := exp(2.30258 × x);

    real procedure sum(i, j, r);
    value j;
    integer i, j;
    real r;
    begin
        real s;
        s := 0;
        for i := 1 step 1 until j do s := s + r;
        sum := s
    end sum;

    procedure search(k, x, i, r, n);
    value n, x;
    integer i, k, n;
    real x, r;
    begin
        k := 1;
```

```
tst:        if n-k = 1 then go to exi;
            i := (k + n) / 2;
            if r < x
                then k := i
                else n := i;
            go to tst;
exi:    end search;

        procedure order(x, y, a, b, k, n);
        integer k, n;
        real x, y;
        real array a, b;
        begin
            k := n := n + 1;
ost:        if k = 1 then go to oxi;
            if x < a[k-1]
                then
                begin
                    a[k] := a[k-1];
                    b[k] := b[k-1];
                    k := k - 1;
                    go to ost
                end;
oxi:        a[k] := x;
            b[k] := y
        end order;

        real procedure ipol(x, a, b, n);
        value n, x;
        integer n;
        real x;
        real array a, b;
        begin
            integer i, k;
            real u, v, w, y, z;
            search(k, x, i, a[i], n);
            if n = 2 ∨ x < a[1] ∨ x > a[n]
                then ipol := b[k] + (b[k+1] - b[k]) × (x - a[k]) / (a[k+1] - a[k])
                else
                begin
                    if k = n-1 then k := k - 1;
                    u := x - a[k];
                    v := x - a[k+1];
                    w := x - a[k+2];
                    y := (b[k] - b[k+1]) / (v - u);
```

```
            z := (b[k+1] - b[k+2]) / (w - v);
            if sign(y) = sign(z)
                then ipol := b[k] + u × y / (1 - v × (y - z) / (b[k] - b[k+2]))
                else ipol := b[k] + u × (y + v × (y - z) / (w - u))
        end
end ipol;

real procedure h(a, v);
value a, v;
real a, v;
begin
    real vv, aa, h0, h1, h2, hh1, hh2, hh3, hh4, u, uu;
    boolean q;
    vv := v × v;
    q := a < .2;
    if q ∧ v > 5
        then h := ((2.12 / vv + .8463) / vv + .5642) × a / vv
        else
            if -, q ∧ (a > 1.4 ∨ a+v > 3.2)
                then
                begin
                    aa := a × a;
                    u := (aa + vv) × 1.4142;
                    uu := u × u;
                    h := ((((aa - 10 × vv) × aa × 3 + 15 × vv × vv) / uu + 3 × vv - aa) / uu + 1) × a × .79788 / u
                end
                else
                begin
                    h0 := exp(-vv);
                    h2 := (1 - 2 × vv) × h0;
                    if v > 2.4
                        then h1 := ((-.0032783 × vv + .0429913 × v - .188326) × vv + .278712 × v + .55415)
                                   / (vv - 1.5)
                        else
                            if v > 1.3
                                then h1 := (-.220416 × vv + 1.989196 × v - 6.61487) × vv + 9.39456 × v - 4.4848
                                else h1 := (.42139 × vv - 2.34358 × v + 3.28868) × vv - .15517 × v - 1.1247;
                    if q
                        then h := (h2 × a + h1) × a + h0
                        else
                        begin
                            hh1 := h1 + h0 × 1.12838;
                            hh2 := h2 + hh1 × 1.12838 - h0;
                            hh3 := (1 - h2) × .37613 - hh1 × .66667 × vv + hh2 × 1.12838;
                            hh4 := (3 × hh3 - hh1) × .37613 + h0 × .66667 × vv × vv;
```

```
                        h := ((((hh4 × a + hh3) × a + hh2) × a + hh1) × a + h0)
                            × (((-.122727278 × a + .532770573) × a - .96284325) × a + .979895032)
                    end
                end
end h;

real procedure ash(r, d);
value r, d;
real r, d;
begin
    real rd, rrd;
    d := max(.0125, d);
    rd := sqrt(d);
    rrd := r × rd;
    ash := (1 + rrd / sqrt(sqrt(1 + rrd ↑ 4))) / (d × d × rd)
end ash;

real procedure sumeta(t);
value t;
integer t;
sumeta := etac[t] + (if conti then 0 else sum(b, bma, eta[t,b]
          × (if hy[b] then ash(alpha[t,b], dlam[b]) else h(alpha[t,b], dlam[b] / lam[b] / vdop[t,bj[b]]))));

real procedure flux(dl);
value dl;
real dl;
begin
    integer te;
    real k0, k2, d, e;
    if -, conti then for b := 1 step 1 until bma do dlam[b] := abs(dl - dla[b]) / 1000;
    tlam[1] := 0;
    k2 := sumeta(1);
    for t := 1, t+2 while t < tma ∧ tlam[t] < ta[6] do
    begin
        k0 := k2;
        k2 := sumeta(t + 2);
        d := (k0 + 4 × sumeta(t + 1) + k2) × dtau[t];
        e := min(max(d / 4, d / 2 + (k0 - k2) × dtau[t+1]), 3 × d / 4);
        te := t + 2;
        tlam[te] := tlam[t] + d;
        tlam[t+1] := tlam[t] + e
    end t;
    flux := sum(j, 6, wa[j] × ipol(ta[j], tlam, blam, te))
end flux;
```

```
        real procedure dep(dl);
        value dl;
        real dl;
        begin
            pma := rest(pma, 400);
            for p := pma step -1 until 1 do
            begin
                if abs(dlp[p] - dl) < 1 then go to yes;
                if dlp[p] < dl then go to no
            end p;
no:         order(dl, 1 - flux(dl) / fc, dlp, rp, p, pma);
yes:        dep := rp[p]
        end dep;

        real procedure asq(l, n);
        integer n;
        real l;
        begin
            real h;
            h := sqrt(n × 13.595 / dchi[t]);
            asq := (h × (h + 1) × (h + .5) - l × (l + 1) × (l + .5)) / 3
                   - th × n × n × 31.321 × (h - l) × (1 - th × 15.66 × n × n / h / l)
        end asq;

        real procedure ucg(w);
        real w;
        begin
            real om, al;
            al := log(2 × alpha[mt,1]);
            om := log(w / 2 / dldop / (1 - blam[1] / fc));
            if om < -1
                then ucg := om + .08
                else
                    if om < .9
                        then ucg := ((-.245805 × om + .643081) × om + 2.337445) × om + .563 - max(0, om) × (1 + al) / .9
                        else ucg := 2 × om - al + .157
        end ucg;

        procedure new(deps);
        value deps;
        real deps;
        begin
            real x;
            x := xp(deps);
            epsb[1] := epsb[1] + deps;
```

```
        for t := 1 step 1 until tma do eta[t,1] := eta[t,1] × x
    end new;

    procedure text(l);
    integer l;
    begin
        switch tx := l1, l2, l3, l4, l5;
        go to tx[l];
l1:     output1(1, ‹‹wrong tau scale›bbzzd/›, t);
        go to out;
l2:     output1(1, ‹‹lambda 1 negativ›bb+5z.dd/›, lamc[1]);
        go to out;
l3:     output1(1, ‹‹wrong line›bb+5z.dd/›, mlam);
        go to lida;
l4:     output1(1, ‹‹no extrapolation›bb+5z.dd/›, mlam);
        go to lida;
l5:     output1(1, ‹‹no element›bb4d/›, elb[b]);
        go to lida
    end text;

    procedure check(k, l, m, n);
    integer k, l, m, n;
    begin
        real x;
        read(x);
        for k := 1 step 1 until 5 do
        begin
            if abs(x + k) < ₁₀-8 then go to jump[k]
        end k;
        for k := l step 1 until m do
        begin
            if abs(x - k) < ₁₀-8 then go to ok
        end k;
        go to jump[n];
ok:  end check;

    procedure in(mi, ma, a);
    integer mi, ma;
    real array a;
    for i := mi step 1 until ma do read(a[i]);

    procedure prot;
    begin
        if 50 < s
            then
```

```
            begin
                s := 0;
                output0(1, ¢x¢ion   mult  lambda     chi    lg gf  -lg c6  -lg c4  lg rad  ¢
                           ¢lg eps   rc    w obs   w cal  pma  lg f   mu¢//¢)
            end;
        if it = 0
            then
            begin
                s := s + bma + 1;
                output0(1, ¢/¢)
            end
            else s := s + 1;
        for b := 1 step 1 until bma do
        begin
            output10(1, ¢4d,2b3zd,b4zd.2d,6(2b-zd.2d),2b.4d¢, elb[b], mult[b], lam[b], chib[b], lgf[b],
                        lc6[b], lc4[b], log(rad[b]), epsb[b], ipol(dla[b], dlp, rp, pma));
            if b = bma
                then output5(1, ¢2(b5z.d),bzzd,bbd.3d,bb-d.dd/¢,
                                wm, wcal, pma, 27.07573 + log(fc) - 5 × log(mlam), mu)
                else output0(1, ¢/¢)
        end b
    end prot;

    procedure prof;
    begin
        if 50 < s+4+pma/7 then prot;
        s := s + 4 + pma ⌈ 7;
        for p := 1 step 1 until pma do
        begin
            output2(1, ¢3b-5zd,b.4d¢, dlp[p], rp[p]);
            if rest(p, 7) = 0 then output0(1, ¢/¢)
        end p;
        output0(1, ¢//¢)
    end prof;

geda:   begin
        read(del, mtau);
        rmin := del × del + .002;
        s := 0;
        in(1, 6, tflx);
        in(1, 6, wflx);
        in(1, 6, tint);
        in(1, 6, wint);
        in(2, 92, my);
        go to moda
```

```
        end geda;

moda:   begin
            check(tma, 3, 31, 5);
            read(efft, lgg, he);
            in(1, tma, tau);
            for t := 1 step 2 until tma-2 do
            begin
                x := tau[t+2] - tau[t];
                y := tau[t+1] - tau[t];
                if abs(2 - x / y) > .0004 then text(1);
                dtau[t] := x / 6;
                dtau[t+1] := y / 4
            end t;
            in(1, tma, theta);
            in(1, tma, lpe);
            in(1, tma, lpg);
            in(1, tma, lk);
            in(1, tma, xi);
            check(lma, 2, 12, 5);
            for l := 1 step 1 until lma do
            begin
                read(lamc[l]);
                if lamc[1] < 0 then text(2);
                for t := 1 step 1 until tma do read(lkl[t,l])
            end l;
            go to pre
        end moda;

pre:    begin
            if s > 0 then output0(1, ⟨⟩);
            output3(1, ⟨⟨ efft   lg g   he/h⟩//,4zd,3bd.dd,3bd.3d///,
                       ⟨tau quer    theta    temp   lg pe   lg pg    xi    -lg kq    eldamp    vdw⟩//⟩, efft, lgg, he);
            search(mt, mtau, t, tau[t], tma);
            elj[0] := 100;
            epsj[0] := 12;
            w := 8.9745 + log((1 + he / 2.4192) / (1 + he));
            for t := 1 step 1 until tma do
            begin
                th := -theta[t];
                x := ln(-th);
                v := lpe[t];
                z := saha[t] := xp(9.0799 - v - 1.0857 × x);
                u := dchi[t] := -4.98₁₀-4 × th × exp(1.15129 × v);
                y := zeta[t,0] := 1 / (2 + 20.9061 × xp(10.875 × th) + 747.0937 × xp(13.34 × th)
```

```
                                + 2 × asq(11, 1) × xp(13.595 × th) + z × xp((13.595 - u) × th));
            vdop[t,0] := sqrt(xi[t] × xi[t] - 82.98 / th) / 299776;
            z := vdw[t] := w + lpg[t] + .304 × x + log(y);
            y := eldp[t] := 7.762 + v + .3619 × x;
            pn[t,1] := 1;
            output9(1, ‹zzd.5d,3bd.4d,2b4zd,2(b-zd.3d),2bzd.d,3(3b-zd.3d),/›,
                       tau[t], -th, -5040 / th, v, lpg[t], xi[t], lk[t], y, z)
        end t;
        if 25 < tma then output0(1, ‹0›);
        go to ion
    end pre;

ion:    begin
        check(jma, 1, 12, 5);
        for j := 1 step 1 until jma do
        begin
            elj[j] := 0;
            epsj[j] := 6;
            read(list[j])
        end j;
newel:  read(el);
        if el < 0
            then
            begin
                output0(1, ‹//›);
                for j := 0 step 1 until jma do output1(1, ‹2b4d,3b›, elj[j]);
                output0(1, ‹//›);
                for t := 1 step 1 until tma do
                begin
                    for j := 0 step 1 until jma do output1(1, ‹-zd.3d,2b›, log(zeta[t,j]));
                    output0(1, ‹/›)
                end t;
                s := 51;
                go to jump[-el]
            end;
        check(nma, 1, 7, 5);
        for t := 1 step 1 until tma do sn[t] := 0;
        for n := 1 step 1 until nma do
        begin
            read(s, go);
            for t := 1 step 1 until tma do q[t] := go;
            for i := 1 step 1 until s do
            begin
                check(k, 0, 8, 5);
                read(gpr, x, y);
```

```
                if i = 1 then chin[n] := x;
                if k ≠ 0
                    then
                    begin
                        for j := 1 step 1 until k do read(g[j], chi[j]);
                        for t := 1 step 1 until tma do
                        begin
                            th := -theta[t];
                            q[t] := q[t] + sum(j, k, g[j] × xp(chi[j] × th)) + gpr × xp(th × x) × asq(y, n)
                        end t
                    end k
            end i;
            for t := 1 step 1 until tma do
            begin
                sn[t] := sn[t] + pn[t,n] × q[t];
                if n < nma then pn[t,n+1] := pn[t,n] × saha[t] × xp((n × dchi[t] - chin[n]) × theta[t])
            end t
        end n;
        for n := 1 step 1 until nma do
            for j := 1 step 1 until jma do
            begin
                if el+n-1 = list[j]
                    then
                    begin
                        elj[j] := list[j];
                        list[j] := 0;
                        x := 83.81 / my[el/100];
                        chij[j] := chin[n];
                        for t := 1 step 1 until tma do
                        begin
                            zeta[t,j] := pn[t,n] / sn[t];
                            vdop[t,j] := sqrt(xi[t] × xi[t] + x / theta[t]) / 299776
                        end t
                    end
            end nj;
        go to newel
    end ion;

eps:    begin
        check(nma, 1, jma, 1);
        for n := 1 step 1 until nma do
        begin
            read(el, x);
            for j := 1 step 1 until jma do
            begin
```

```
                if elj[j] = el then epsj[j] := x
            end j
        end n;
        go to lida
    end eps;

lida:   begin
        check(a, 0, 7, 1);
        check(bma, 1, 10, 1);
        single := bma = 1;
        for i := 1 step 1 until 3 do
        begin
            boa[i] := rest(a, 2) = 1;
            a := entier(a / 2)
        end i;
        read(mu, mlam, wm);
        for i := 1 step 1 until 6 do
        begin
            wa[i] := if mu < 0 then wflx[i] else wint[i];
            ta[i] := if mu < 0 then tflx[i] else tint[i] × mu
        end i;
        for b := 1 step 1 until bma do read(elb[b], mult[b], lam[b], chib[b], lgf[b], lc6[b], lc4[b], rad[b]);
        x := sum(b, bma, lam[b]) / bma;
        if 5 < abs(x - mlam) then text(3);
        if mlam < lamc[1] ∨ abs(lamc[lma]) < mlam then text(4);
        for b := 1 step 1 until bma do
        begin
            hy[b] := elb[b] = 100;
            x := min(x, lam[b]);
            for j := 0 step 1 until jma do
            begin
                if elb[b] = elj[j] then go to fi
            end j;
            text(5);
fi:         bj[b] := j
        end b;
        lamax := 0;
        for b := 1 step 1 until bma do
        begin
            dla[b] := 1000 × (lam[b] - x);
            lamax := max(lamax, dla[b])
        end b;
        dldop := 1000 × mlam × vdop[mt,bj[1]];
        lamax := lamax + 3 × dldop;
        if -, single ∨ hy[b] ∨ wm < .1 then boa[2] := boa[3] := false;
```

```
            go to cont
        end lida;

cont:   begin
            real array ka,la[1:3];
            z := 28541.7 / mlam;
            search(j, mlam, l, abs(lamc[l]), lma);
            if j+1 < lma ∧ 0 < lamc[j+1]
                then n := 3
                else
                    if j = 1
                        then n := 2
                        else
                            if 0 < lamc[j-1]
                                then
                                begin
                                    n := 3;
                                    j := j - 1
                                end
                                else n := 2;
            for l := 1 step 1 until n do la[l] := abs(lamc[j-1+l]);
            for t := 1 step 1 until tma do
            begin
                blam[t] := 1 / (exp(z × theta[t]) - 1);
                for l := 1 step 1 until n do ka[l] := lkl[t,j-1+l];
                etac[t] := xp(lk[t] - ipol(mlam, la, ka, n))
            end t;
            conti := true;
            fc := flux(0);
            conti := false;
            go to kapgam
        end cont;

kapgam: begin
            real n, m, anm;
            for b := 1 step 1 until bma do
            begin
                u := lam[b];
                v := chib[b] + 12394.7 / u;
                j := bj[b];
                m := lc6[b];
                n := lc4[b];
                epsb[b] := epsj[j];
                w := -32.3014 - log(1 + he) + lgf[b] + epsb[b] + (if hy[b] then 2.4053 + 2 × log(m) else log(u));
                if hy[b]
```

```
            then anm := .06479 × (n × n × n / m / m + m × m × m / n / n) / sqrt(n × n - m × m)
            else
                if m < 10
                    then
                    begin
                        z := rest(elb[b], 100) + 1;
                        x := z / (chij[j] - v);
                        y := z / (chij[j] - chib[b]);
                        m := lc6[b] := 29.7286 - m - log(x × x - y × y)
                    end;
        if rad[b] < 0 then rad[b] := 2.21₁₀7 / u / u;
        for t := 1 step 1 until tma do
        begin
            th := theta[t];
            eta[t,b] := zeta[t,j] × xp(w + lk[t] - v × th) / blam[t]
                    × (if hy[b] then th × xp(lpe[t]) else 1 / vdop[t,j]);
            if hy[b]
                then alpha[t,b] := anm × sqrt(th) × (4.1007 - lpe[t] / 2 - .86859 × ln(n) - .65144 × ln(th))
                else alpha[t,b] := ((if n < 0 then 0 else xp(-2 × n / 3 + eldp[t])) + rad[b] + xp(-.4 × m + vdw[t]))
                                × u / 37.671₁₀10 / vdop[t,j]
        end t
    end b;
    if boa[3]
        then
        begin comment coarse analysis;
            x := blam[1];
            for it := 1, 2, 3, 4 do new(-log((fc - x) / (flux(0) - x) - 1));
            new(ucg(wm))
        end coarse analysis;
    it := 0;
    go to equiv
end kapgam;

equiv:  begin
        wcal := 0;
        pma := 0;
        n := if single then +10 else -10;
        for n := n, 2 × n while rmin < dep(v) do v := n;
        dl := min(0, v);
        ddl := max(10, abs(v / 8));
```

```
simp:       y := dep(dl);
            u := 4 × dep(dl + ddl);
            v := dep(dl + 2 × ddl);
            w := 4 × dep(dl + 3 × ddl);
            x := dep(dl + 4 × ddl);
            z := (-y + u - 6 × v + w - x) / 15;
            y := y + u + 2 × v + w + x;
            x := x + v + w;
            if abs(z) > rmin ∧ ddl > 19
                then
                begin
                    ddl := ddl / 2;
                    go to simp
                end;
            wcal := wcal + ddl × (y + z) / 3;
            dl := dl + 4 × ddl;
            if abs(z) < rmin/32 then ddl := 2 × ddl;
            if dl < lamax ∨ 6×rmin < x then go to simp;
            go to fin
        end equiv;

fin:    begin
            if single then wcal := 2 × wcal;
            prot;
            y := ln(wm);
            z := ln(wcal);
            if boa[2] ∧ it < 5 ∧ 2×del < abs(y - z)
                then
                begin comment iteration;
                    order(z, epsb[1], lwit, epit, n, it);
                    new(if it = 1 then ucg(wm) - ucg(wcal) else ipol(y, lwit, epit, it) - epsb[1]);
                    go to equiv
                end iteration;
            if boa[1] then prof;
            go to lida
        end fin;

out: end of program;
```

Universelle Daten:

```
del:    .01;

mtau:   .11;

tflx:   .038,   .154,   .335,   .793,   1.467,  3.890;

wflx:   .1615,  .1346,  .2973,  .1872,  .1906,  .0288;

tint:   .1808,  .7524,  1.1808, 1.7524, 2.5858, 5.4142;

wint:   .3699,  .2623,  .1360,  .0965,  .1155,  .0198;

my:     4.00,   6.94,   9.02,   10.82,  12.01,  14.01,  16.00,
        19.00,  20.18,  23.00,  24.32,  26.97,  28.06,  30.98,
        32.07,  35.46,  39.94,  39.10,  40.08,  45.10,  47.90,
        50.95,  52.01,  54.93,  55.85,  58.94,  58.69,  63.54,
        65.38,  69.72,  72.60,  74.91,  78.96,  79.91,  83.70,
        85.48,  87.63,  88.92,  91.22,  92.91,  95.95,  99.00,
        101.70, 102.90, 106.70, 107.88, 112.40, 114.80, 118.70,
        121.80, 127.60, 126.90, 131.30, 132.90, 137.30, 138.90,
        140.10, 140.90, 144.30, 147.00, 150.40, 152.00, 156.90,
        159.20, 162.50, 164.90, 167.20, 169.40, 173.00, 175.00,
        178.60, 180.90, 183.90, 186.30, 190.20, 193.10, 195.20,
        197.20, 200.60, 204.40, 207.20, 209.00, 210.00, 211.00,
        222.00, 223.00, 226.05, 226.00, 232.10, 231.00, 238.10;
```