C-Programm LINOP zur Auswertung von Filmdosimetern durch lineare Optimierung

Anwendungshandbuch

Fachbereich Strahlenhygiene
Institut für Strahlenhygiene

Peter Kragh

Bundesamt für Strahlenschutz

BfS-ISH-172/95

Bitte beziehen Sie sich beim Zitieren dieses Dokuments immer auf folgende URN:
urn:nbn:de:0221-2018012514503

Neuherberg, November 1995

C-Programm LINOP zur Auswertung von Filmdosimetern durch lineare Optimierung

Anwendungshandbuch

Fachbereich Strahlenhygiene
Institut für Strahlenhygiene

Peter Kragh

Zusammenfassung

Der optimale Meßwert für Filmdosimeter ergibt sich durch lineare Optimierung.

Zur Verwendung der linearen Optimierung in der Filmdosimetrie wurde das Programm Linop entwickelt. Es gestattet die Auswertung und Prüfung von Filmdosimetern und allen sonstigen mehrkomponentigen Dosimetern.

Das vorliegende Anwendungshandbuch zum Programm Linop enthält das Quellprogramm, eine Programmbeschreibung und eine Installations- und Anwendungsanweisung. Die Dateien mit Programmen und Beispielen können auf Anforderung zur Verfügung gestellt werden.

Summary

Linear programming results in an optimal measuring value for film dosemeters.

The Linop program was developed to be used for linear programming. The program permits the evaluation and control of film dosemeters and of all other multi-component dosemeters.

This user manual for the Linop program contains the source program, a description of the program and installation and use instructions. The data sets with programs and examples are available upon request.

Inhaltsverzeichnis

1.	**Einführung**	**4**
1.1	Linearkombinationsverfahren	
1.2	Lineare Optimierung	
1.3	Minimierung des Fehlers	
2.	**Installation**	**6**
2.1	Ausführbare Programme	
2.1.1	Unix	
2.1.2	Dos	
2.2	Quelltext der C-Programme	
2.3	Beispiel einer Datei mit einer Ansprechmatrix	
2.4	Beispiel einer Datei mit Einzelmesswerten	
3.	**Anwendung**	**8**
3.1	Aufruf des Programms	
3.2	Bestimmung der Eigenschaften der Ansprechmatrix	
3.3	Auswertung von Messungen	
3.3.1	Manuelle Eingabe der Einzelmesswerte	
3.3.2	Eingabe der Einzelmesswerte über eine Datei	
3.4	Fehlermeldungen	
4.	**Programmbeschreibung**	**16**
4.1.	Auswahl der Speicherbelegungsfunktion	
4.2.	Einbinden von Programm Simfeh	
4.3.	Parameter zur Steuerung von Testausgabe	
4.4.	Berechnung der Eigenschaften der Ansprechmatrix	
4.4.1	Einlesen der Ansprechmatrix und Reservierung von Speicherplatz	
4.4.2	Durchfuehrung der Berechnung der Eigenschaften der Ansprechmatrix	
4.5.	Auswertung von Messergebnissen	
4.5.1	Einlesen der Ansprechmatrix und Reservierung von Speicherplatz	
4.5.2	Auswertung von Messergebnissen aus einer Datei	
4.5.2.1	Öffnen der Datei mit Messdaten	
4.5.2.2	Bearbeitung der Datensätze mit Messdaten	
4.5.2.2.1	Einlesen eines Datensatzes mit Messdaten	
4.5.2.2.2	Berechnung von Dosiswerten und Spektren	
4.5.3	Auswertung einer einzelnen Messung, Eingabe mit dem Aufruf	
4.5.3.1	Übertragung der Einzelmesswerte	
4.5.3.2	Berechnung von Dosiswerten und Spektren	
5.	**Literaturverzeichnis**	**19**
6.	**Quelltext des C-Programms**	**20**
7.	**Anhang 1: Simfeh: C-Funktionen für das Simplexverfahren unter Berücksichtigung von Fehlern**	**23**
8.	**Anhang 2: Simplex: C-Funktionen für das Simplexverfahren**	**47**

1. Einführung

Ein Filmdosimeter ermöglicht neben der Bestimmung eines Personendosiswertes in gewissem Umfang eine Analyse des Energiespektrums von Photonenstrahlung. Das wird durch Filter ermöglicht, die eine unterschiedliche Schwärzung des Films hinter den Filtern zur Folge haben. Das Filmdosimeter besteht aus mehreren den einzelnen Filtern zuzuordnenden Komponenten. Jede einzelne Komponente ist ein Dosimeter, für das ein Dosiswert, der Einzelmeßwert, ermittelt wird. Durch Auswertung der Einzelmeßwerte wird ein Personendosiswert errechnet. Die Ermittlung der Personendosis zerfällt also in die beiden Schritte:

a. Ermittlung der Einzelmeßwerte
Die Einzelmeßwerte sind energie- und winkelabhängig, d.h. die Energieabhängigkeit (bzw. Winkelabhängigkeit) wird im "Fehler der Einzelmeßwerte" nicht berücksichtigt.

b. Auswertung
Durch die Auswertung werden aus den Einzelmeßwerten ein Personendosiswert und das Energiespektrum berechnet. Da im allgemeinen die mit den Einzelmeßwerten vorliegende Information zu einer genauen Bestimmung des Energiespektrums nicht ausreicht, tritt bei der Ermittlung der Personendosis aus den Einzelmeßwerten ein weiterer Fehler auf, der hier als "Auswertefehler" bezeichnet werden soll.

Zur Auswertung wird gegenwärtig das "Filteranalytische Verfahren" [1] verwendet. Die Meßgenauigkeit der Filmdosimetrie ist dabei nur schwer zu ermitteln, da das Filteranalytische Verfahren eine Ermittlung des Auswertefehlers nicht vorsieht.

Genauere Ergebnisse ermöglicht das Verfahren der linearen Optimierung [3], daß erstmals von Bergmann und Chanourdie [2] zur Auswertung von Filmdosimetern verwendet wurde. In der bereits früher vorgestellten [7] und hier verwendeten Form ermöglicht die lineare Optimierung die Berechnung der mit den gemessenen Einzelmeßwerten verträglichen Energiespektren mit dem minimalen und dem maximalen Dosiswert [5], [6]. Aus diesen beiden Extremwerten werden ein Personendosiswert und der Auswertefehler ermittelt. Der Auswertefehler ergibt sich aus den Abweichungen der berechneten Extremwerte von dem Personendosiswert. Durch Minimierung des Auswertefehlers ergibt sich der optimale Wert der Personendosis.

Zusätzlich wird die Prüfung von Dosimetern durchgeführt. Dabei erfolgen die Bestimmung des maximalen Auswertefehlers, der zugehörigen Einzelmeßwerte und der Spektren der Extremwerte der Dosis.

Neben dem eigentlichen Verfahren der linearen Optimierung wird noch das "Linearkombinationsverfahren" [4] verwendet. Dabei wird ebenfalls eine lineare Optimierung durchgeführt. Man erhält aber nicht mehr den optimalen Personendosiswert. Die beiden Verfahren können folgendermaßen gegeneinander abgegrenzt werden:

1.1 Linearkombinationsverfahren:

- Rechenaufwand für die Auswertung: klein
Die lineare Optimierung erfolgt nur einmal nach der Kalibrierung. Dabei werden die Koeffizienten einer linearen Funktion der Einzelmeßwerte zur Bestimmung der Personendosis bestimmt. Bei der Auswertung einer Messung muß dann nur noch der Wert dieser linearen Funktion ermittelt werden.

- Genauigkeit der Auswertung: im allgemeinen geringer
Der relative Auswertefehler wird im Zuge der linearen Optimierung auch nur einmal mit der Kalibrierung bestimmt. Damit ist der relative Auswertefehler unabhängig vom Energiespektrum der Strahlung für alle Auswertungen gleich groß. Da der Fehler für alle Bestrahlungsbedingungen herhalten muß, ist er gleich dem größten möglichen Fehler. Damit erhält man nicht den optimalen Dosiswert.

- Analyse der Bestrahlungsbedingungen: nein
Eine Analyse des Energiespektrums der Strahlung ist nicht vorgesehen.

1.2 Lineare Optimierung:

- Rechenaufwand für die Auswertung: größer
Die lineare Optimierung erfolgt bei jeder Auswertung einer Messung. Trotzdem fällt der Rechenaufwand bei den heutigen Preisen von Rechenzeit nicht mehr ins Gewicht.

- Genauigkeit der Auswertung: optimal
Der Auswertefehler wird im Zuge der linearen Optimierung mit jeder Auswertung neu bestimmt. Der Fehler ist abhängig von den Einzelmeßwerten und im allgemeinen wesentlich kleiner als der größtmögliche Fehler. Die errechneten Dosiswerte sind optimal.

- Analyse der Bestrahlungsbedingungen: ja
Das für die gemessene Verteilung der Einzelmeßwerte mögliche Energiespektrum der Strahlung wird für die beiden Extremwerte der Dosis ermittelt.

1.3 Minimierung des Fehlers:

Das Ziel der linearen Optimierung ist die Minimierung des Auswertefehlers. Zur Durchführung des Verfahrens müssen aus Verfahrensgründen die "Fehler der Einzelmeßwerte" als Variable eingeführt werden. Die Rechnung wird am einfachsten, wenn nur diejenigen Lösungen zugelassen werden, für die die "Fehler der Einzelmeßwerte" verschwinden [5]. In vielen Fällen (in der Praxis in über 50 % der Fälle) erhält man dann aber überhaupt keine Lösung. Im allgemeinen müssen deshalb neben dem Auswertefehler auch "Fehler der Einzelmeßwerte" zugelassen und gegebenenfalls minimiert werden.

Die nächsteinfache Möglichkeit der linearen Optimierung besteht in der Einführung von Gewichtsfaktoren für die Fehler der Einzelmesswerte und den Auswertefehler. Die Gewichtsfaktoren müssen auf die gemessenen Einzelmeßwerte abgestimmt werden [2], wenn unwahrscheinliche Lösungen vermieden werden sollen. Das kann einen manuellen Eingriff bei jeder einzelnen Auswertung erforderlich machen.

Diese Schwierigkeit wird vermieden, wenn die Optimierung in drei Schritten durchgeführt wird [7]:
a. Berechnung der Minima der "Fehler der Einzelmeßwerte" unabhängig vom Dosiswert
b. Abschätzung der für die weitere Rechnung zu verwendenden "Fehler der Einzelmeßwerte" unter Berücksichtigung sonstiger Kenntnisse (die abgeschätzten "Fehler der Einzelmeßwerte" müssen ein wenig größer als die zuvor ermittelten Minima sein)
c. Minimierung des Auswertefehlers unter Berücksichtigung der mit Schritt b. ermittelten abgeschätzten Fehler der Einzelmeßwerte, die bei diesem Rechengang konstant bleiben

Das Programm "Linop" führt die lineare Optimierung nach dem letztgenannten Verfahren durch. Dabei zeigt sich, daß bei Benutzung eines PC der Rechenzeitbedarf in der Größenordnung von 0.1 Sekunden liegt, so daß eine Anwendung in der Routine einer Meßstelle möglich erscheint.

Vor Anwendung des Programms muß die Ansprechmatrix ermittelt werden. Die Ansprechmatrix beschreibt das Ansprechvermögen der Komponenten des Dosimeters für eine Auswahl von Energiespektren, aus denen sich die in der Anwendung zu erwartenden Energiespektren zusammensetzen lassen.

2. Installation

Die folgenden Dateien stehen zur Verfügung:
- linop.txt
- linop
- linop.exe
- linop.c
- simplex.cun
- simfeh.cun
- m.txt
- me.txt

Neben der vorliegenden Programmbeschreibung (linop.txt) sind darin enthalten:

2.1 Ausführbare Programme

Die Laufzeit des Programms ist im wesentlichen proportional der Größe der Ansprechmatrix. Für die Auswertung einer Messung werden bei einer Ansprechmatrix mit L Elementen auf einem PC (Intel 486) unter DOS etwa: T = L * 0.2 msec benötigt.

2.1.1 Unix

Die ausführbare Datei linop wurde unter Unix (IBM) erstellt.

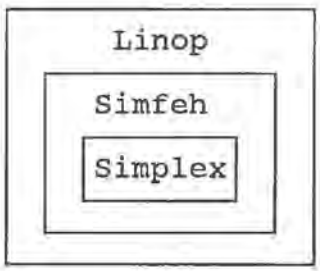
2.1.2 DOS

Die ausführbare Datei linop.exe wurde unter DOS mit dem Microsoft Compiler C600 erstellt. Das Speichermodell ist "medium". Eine Alternative unter DOS mit mehr Speicherplatz ist das Speichermodell "huge", für das der Eintrag im Quellprogramm linop.c verändert werden sollte, siehe Abschnitt 4.1.

2.2 Quelltext der C-Programme

Der Quelltext ist in den Dateien:
- linop.c
- simfeh.cun
- simplex.cun

enthalten. Bei erneuter Übersetzung muß nur die Datei linop.c aufgerufen werden. Die übrigen Dateien werden durch linop.c mit eingebunden, sie sind in den Anhängen 1 und 2 beschrieben. Die Beziehung der Programme Linop, Simfeh und Simplex kann durch das folgende Diagramm veranschaulicht werden:



Simplex:
Die Datei simplex.cun enthält Funktionen zur Durchführung der linearen Optimierung nach dem Simplexverfahren.

Simfeh:
Die Datei simfeh.cun enthält unter Verwendung von simplex.cun Funktionen zur
- Bestimmung der Eigenschaften der Ansprechmatrix
- Auswertung von Messungen
- Erstellung der bei fehlerbehafteten Daten für die lineare Optimierung benötigten Gleichungssysteme

Linop:
Die Datei linop.c enthält Anweisungen für:
- Kommunikation mit dem Anwender
- Aufruf der Funktionen von simfeh.cun

2.3 Beispiel einer Datei mit einer Ansprechmatrix

Vor dem Aufruf von linop muß eine Datei mit der Ansprechmatrix bereitgestellt werden. Mit der Datei m.txt ist ein Beispiel angegeben, das auch in den folgenden Beispielen für Ergebnisse verwendet wird. Die Datei mit der Ansprechmatrix muß im Textmode (nur Text ohne Steuerzeichen) vorliegen. Die Datei m.txt lautet beispielsweise (Einzelheiten sind beschrieben in Abschnitt 4.4.1):

m.txt:

```
3 6
0.00001 0.00001 0.00001
0.00001 0.00001 0.00001
0.77    0.1     0.0
0.09    0.57    0.0
3.37    1.64    0.5
1.12    1.05    0.93
0.15    1.49    1.78
0.12    0.2     0.59
```

2.4 Beispiel einer Datei mit Einzelmesswerten

Meßergebnisse können vom Programm linop aus einer Datei im Textmode gelesen werden. Die Datei me.txt enthält ein Beispiel einer Datei mit 6 Messungen mit je 3 Einzelmeßwerten, wobei in diesem Fall die 6 Messungen den Ansprechvektoren der Ansprechmatrix aus m.txt entsprechen (Einzelheiten sind beschrieben im Abschnitt 4.5.2.1.):

me.txt:

```
1   0.77   0.1    0.0
2   0.09   0.57   0.0
3   3.37   1.64   0.5
4   1.12   1.05   0.93
5   0.15   1.49   1.78
6   0.12   0.2    0.59
```

3. Anwendung

Das Programm linop ermöglicht im wesentlichen 2 verschiedene Rechnungen:
- Bestimmung der Eigenschaften der Ansprechmatrix
- Auswertung von Messungen

Zur Beschreibung der Ergebnisse des Programms soll für die in Abschnitt 2 angegebenen Daten die Ausgabe auf dem Standardausgabegerät angegeben und kommentiert werden.

3.1 Aufruf des Programms

Syntax:	linop test Ansprechmatrix [Einzelmeßwerte]
Parameter:	
- test:	Parameter zur Steuerung der Ausgabe auf das Standardausgabegerät
- Ansprechmatrix:	Name der Datei mit der Ansprechmatrix
- Einzelmeßwerte:	Name der Datei mit Einzelmesswerten oder Einzelmesswerte einer Messung

Durch den Wert von test kann die Menge der ausgegebenen Daten bestimmt werden. Insbesondere erfolgt für test:

0:	Ausgabe der beiden Extremwerte der Dosis (nur für die Auswertung von Messungen)
1:	Ausgabe von Spektren und Dosiswerten
>1:	Ausgabe von Spektren, Dosiswerten und Zwischenergebnissen

Wenn das Ergebnis in einer Datei (z.B. test.dat) abgespeichert werden soll, so kann das durch Umleiten der Standardausgabe erreicht werden. Beispiel: linop 1 m.txt > test.dat .

3.2 Bestimmung der Eigenschaften der Ansprechmatrix

Für die in 2.3 angegebene Ansprechmatrix m.txt erhält man mit test = 1, d.h. mit dem Aufruf:
linop 1 m.txt
das in Ausdruck 1 dargestellte Ergebnis.

In Ausdruck 1 sind einzelne Zeilen mit eingeklammerten Nummern versehen und werden unter bezug auf diese Nummern kommentiert:

(1) Die Zeilen (1) bis (6) geben den Inhalt der Datei mit der Ansprechmatrix wieder.

(2) Bezeichnung der Datei, auf der sich die verwendete Ansprechmatrix befindet.

(3) m ist die Anzahl der Einzelmesswerte für jede Messung
n ist die Anzahl der Ansprechvektoren, d.h. die Anzahl der für die Kalibrierung durchgeführten Messungen

(4) vom Anwender geschätzter absoluter Fehler der Einzelmesswerte ϵ^a_k, k = 1 bis m

(5) vom Anwender geschätzter relativer Fehler der Einzelmesswerte ϵ^r_k, k = 1 bis m
In diesem Fall wurden unrealistisch kleine Werte für die Fehler eingesetzt. Das ist zur Berechnung der Koeffizienten für das Linearkombinationsverfahren vorteilhaft. Eine Prüfung mit größeren Fehlern führt auch zu einem Anwachsen der Quotienten der Extremwerte (14).

Ausdruck 1:

```
Programm linop                                  den 24. 4. 1995

Eigenschaften der Ansprechmatrix
********************************


(1)  Ansprechmatrix
     ==============
(2)  Ansprechmatrix auf der Datei:  m.txt
(3)  m, n:  3   6
(4)  Absolute Fehler:   0.00001  0.00001  0.00001
(5)  Relative Fehler:   0.00001  0.00001  0.00001

(6)  Ansprechmatrix:
     Lfd.Nr.  Ansprechvektoren
       1      0.770  0.100  0.000
       2      0.090  0.570  0.000
       3      3.370  1.640  0.500
       4      1.120  1.050  0.930
       5      0.150  1.490  1.780
       6      0.120  0.200  0.590


(8)  Lineare Optimierung (Eigenschaften der Ansprechnatrix)
     ======================================================
(10) Einzelmesswerte:   3.370  1.640  0.500
(11) Fehler         :   0.00002  0.00002  0.00002
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      3
     1.000
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     4.026  1.874  0.847
(14) Extremwerte der Dosis:    1.000    6.747


(15) Linearkombinationsverfahren
     ===========================
(16) Koeffizienten:  1.093  1.582  0.936
(17) Extremwerte der Dosis:   1.000,   6.747

(18) Dosiswerte für die Vektoren der Ansprechmatrix
     (Linearkombinationsverfahren):
     Lfd.Nr.  Dosiswerte
       1       1.000
       2       1.000
       3       6.747
       4       3.756
       5       4.188
       6       1.000

(19) Extremwerte der Vektoren der Ansprechmatrix
     (Linearkombinationsverfahren):   1.000,   6.747
```

(6) Ansprechmatrix $a_{i,k}$, i = 1 bis n, k = 1 bis m
Die Ansprechmatrix besteht aus n Ansprechvektoren. Zu jeder Messung bei der Kalibrierung gehört ein Ansprechvektor. Die einzelnen Ansprechvektoren werden bei der Ausgabe laufend durchnummeriert. Bei allen späteren Angaben werden Ansprechvektoren mit dieser Nummer bezeichnet.

Zu einer Spalte der dargestellten Matrix gehört eine Gleichung des zu lösenden Gleichungssystems. Im Unterschied zu dieser Darstellung werden normalerweise die Koeffizienten einer Gleichung als Zeilen einer Matrix dargestellt, weshalb die dargestellte Matrix als transponierte Matrix bezeichnet wird, d.h. Zeilen und Spalten sind vertauscht. Eine Zeile wird als Spalte und umgekehrt eine Spalte als Zeile dargestellt.

(8) Die Zeilen (8) bis (14) geben das Ergebnis der Prüfung der Eigenschaften der Ansprechmatrix wieder.

(10) Einzelmeßwerte, für die der Quotient aus den Extremwerten der Dosis den größten Wert annimmt.

(11) Berücksichtigte Fehler der Einzelmesswerte ϵ_k:

$$\epsilon_k = \epsilon^a_k + \epsilon^r_k * f_k , \qquad k = 1 \text{ bis } m.$$

Da zur Bestimmung der Eigenschaften der Ansprechmatrix Einzelmeßwerte f_k zunächst nicht vorliegen (die Einzelmeßwerte (10) sollen erst errechnet werden), werden Einzelmeßwerte aus der Ansprechmatrix abgeschätzt:

$$f_k = \Sigma\ a_{i,k} \ / n , \qquad k = 1 \text{ bis } m.$$

Damit ergibt sich für den zu berücksichtigenden Fehler der Einzelmeßwerte:

$$\epsilon_k = \epsilon^a_k + \epsilon^r_k * \sum_{i=1}^{n} a_{i,k} \ / n , \qquad k = 1 \text{ bis } m.$$

(12) Spektrum mit dem Minimum des Dosiswertes für die Einzelmesswerte (10)
Das Spektrum setzt sich zusammen aus bis zu m Ansprechvektoren, die durch die unter (6) eingeführte laufende Nummer bezeichnet werden, mit den entsprechenden Dosisanteilen.
In diesem Fall ist es:
Ansprechvektor 3 mit dem Dosisanteil 1.

(13) Spektrum mit dem Maximum des Dosiswertes für die Einzelmesswerte (10)
In diesem Fall setzt es sich zusammen aus:
Ansprechvektor 1 mit dem Dosisanteil 4.026,
Ansprechvektor 2 mit dem Dosisanteil 1.874,
Ansprechvektor 6 mit dem Dosisanteil 0.847.

(14) Extremwerte der Dosis
Sie ergeben sich aus den Dosisanteilen unter (12) und unter (13):
Minimaler Dosiswert: Summe der Dosisanteile unter (12)
Maximaler Dosiswert: Summe der Dosisanteile unter (13)

(15) Die Zeilen (15) bis (19) geben das Ergebnis der Rechnungen zum Linearkombinationsverfahren wieder.

(16) Koeffizienten für das Linearkombinationsverfahren c_k
Für die Einzelmeßwerte f_k ergibt sich damit ein Dosiswert:

$$D = \sum_{k=1}^{m} c_k * f_k .$$

(17) Durch lineare Optimierung errechnete Extremwerte der Dosis für das Linearkombinationsverfahren
Damit wird erkennbar, daß die Koeffizienten (16) konservativ sind, denn die bei der Berechnung verwendeten Ansprechvektoren gehören alle zum Dosiswert 1. Wenn keine konservativen Dosiswerte berechnet werden sollen, können alle Koeffizienten mit dem gleichen Faktor kleiner 1 multipliziert werden.

(18) Dosiswerte, die sich mit Hilfe der Koeffizienten (16) für die Ansprechvektoren ergeben

(19) Extremwerte von (18)
Die Ausgabe ist als Prüfung gedacht. Die Werte müssen für kleine Fehler (11) mit den Werten (14) übereinstimmen.

3.3 Auswertung von Messungen

3.3.1 Manuelle Eingabe der Einzelmesswerte

Für die in 2.3 angegebene Ansprechmatrix m.txt erhält man zur Auswertung der Einzelmeßwerte { 1, 1, 1 } mit test = 1, d.h. mit dem Aufruf:
linop 1 m.txt 1 1 1
das in Ausdruck 2 dargestellte Ergebnis.

Die Bedeutung der einzelnen Zeilen entspricht weitgehend den Kommentaren von Abschnitt 3.2. Die folgenden Abweichungen ergeben sich aus der geänderten Aufgabenstellung:

(9) Die Zeilen (9) bis (14) geben das Ergebnis der Auswertung für vom Anwender eingegebene Einzelmeßwerte wieder.

(10) Vom Anwender eingegebene Einzelmesswerte f_k, k = 1 bis m

(11) Berücksichtigte Fehler der Einzelmesswerte ϵ_k:

$$\epsilon_k = \epsilon^a_k + \epsilon^r_k * f_k , \quad k = 1 \text{ bis } m$$

Ausdruck 2:

```
Programm linop                                  den 25. 4. 1995

Auswertung von Messungen
************************

(1)  Ansprechmatrix
     ==============
(2)  Ansprechmatrix auf der Datei:  m.txt
(3)  m, n:  3   6
(4)  Absolute Fehler:   0.00001  0.00001  0.00001
(5)  Relative Fehler:   0.00001  0.00001  0.00001

(6)  Ansprechmatrix:
     Lfd.Nr.  Ansprechvektoren
       1      0.770  0.100  0.000
       2      0.090  0.570  0.000
       3      3.370  1.640  0.500
       4      1.120  1.050  0.930
       5      0.150  1.490  1.780
       6      0.120  0.200  0.590



(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:   1.000  1.000  1.000
(11) Fehler         :   0.00002  0.00002  0.00002
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      3      5      6
     0.264  0.308  0.542
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     0.918  0.999  1.695
(14) Extremwerte der Dosis:    1.114    3.611
```

Für die gleichen Daten mit test = 0, d.h. mit dem Aufruf:
linop 0 m.txt 1 1 1
werden nur die Extremwerte der Dosis ausgegeben:

Ausdruck 3:

```
  1.114    3.611
```

3.3.2 Eingabe der Einzelmesswerte über eine Datei

Für die in 2.3 angegebene Ansprechmatrix m.txt erhält man zur Auswertung der Datei mit Einzelmeßwerten me.txt mit test = 1, d.h. mit dem Aufruf:
linop 1 m.txt me.txt
das in Ausdruck 4 dargestellte Ergebnis.

Dieser Ausdruck enthält zusätzlich zu dem Abschnitt 3.3.1 die Zeilen:

(0) Name der Datei mit den Messungn

(7) Nummer der Messung auf der Datei (0)

Es wurden hier zum zusätzlichen Test der Ansprechmatrix die Elemente der Ansprechvektoren der Ansprechmatrix als Einzelmeßwerte von Messungen in die Datei me.txt eingegeben. Man erkennt, daß die Messungen 1, 2, 5 und 6 zu einer eindeutigen Lösung (dem entsprechenden Ansprechvektor) führen.

Der Ansprechvektor 4 ist an keiner der Lösungen beteiligt, auch nicht bei den Extremwerten von Messung 4. Daraus kann geschlossen werden, daß der Ansprechvektor 4 für die Auswertung überflüssig ist, er kann aus der Ansprechmatrix entfernt werden.

Beim Linearkombinationsverfahren muß für jede Kombination von Einzelmesswerten der größte Dosisfaktor, in diesem Fall 6.747 zur Bestimmung der Meßunsicherheit berücksichtigt werden.

Ausdruck 4:

```
Programm linop                              den 25. 4. 1995

Auswertung von Messungen
************************
(0) Einzelmesswerte auf der Datei: me.txt


(1)  Ansprechmatrix
     ==============
(2)  Ansprechmatrix auf der Datei:  m.txt
(3)  m, n:  3   6
(4)  Absolute Fehler:   0.00001  0.00001  0.00001
(5)  Relative Fehler:   0.00001  0.00001  0.00001

(6)  Ansprechmatrix:
     Lfd.Nr.  Ansprechvektoren
       1      0.770  0.100  0.000
       2      0.090  0.570  0.000
       3      3.370  1.640  0.500
       4      1.120  1.050  0.930
       5      0.150  1.490  1.780
       6      0.120  0.200  0.590
```

```
(7)  Nummer der Messung: 1
(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:   0.770  0.100  0
(11) Fehler          :  0.00002  0.00001  0.00001
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      3
     1.000  0.000
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     1.000  0.000  0.000
(14) Extremwerte der Dosis:    1.000    1.000


(7)  Nummer der Messung: 2
(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:   0.090  0.570  0
(11) Fehler          :  0.00001  0.00002  0.00001
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      2      3      5
     1.000  0.000  0.000
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     0.000  1.000  0.000
(14) Extremwerte der Dosis:    1.000    1.000


(7)  Nummer der Messung: 3
(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:   3.370  1.640  0.500
(11) Fehler          :  0.00004  0.00003  0.00002
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      3
     1.000
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     4.026  1.874  0.847
(14) Extremwerte der Dosis:    1.000    6.747


(7)  Nummer der Messung: 4
(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:   1.120  1.050  0.930
(11) Fehler          :  0.00002  0.00002  0.00002
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      3      5      6
     0.306  0.322  0.346
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     1.080  1.100  1.576
(14) Extremwerte der Dosis:    0.974    3.756
```

```
(7)  Nummer der Messung: 5
(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:    0.150  1.490  1.780
(11) Fehler          :   0.00001  0.00002  0.00003
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      5
     1.000
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      2      5      6
     0.000  1.000  0.000
(14) Extremwerte der Dosis:     1.000     1.000



(7)  Nummer der Messung: 6
(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)
     ================================================
(10) Einzelmesswerte:    0.120  0.200  0.590
(11) Fehler          :   0.00001  0.00001  0.00002
(12) Minimum, Dosisanteile der Ansprechvektoren, Spektrum:
      5      6
     0.000  1.000
(13) Maximum, Dosisanteile der Ansprechvektoren, Spektrum:
      1      2      6
     0.000  0.000  1.000
(14) Extremwerte der Dosis:     1.000     1.000
```

Für die gleichen Daten mit test = 0, d.h. mit dem Aufruf:
linop 0 m.txt me.txt
werden nur die Extremwerte der Dosis ausgegeben:

Ausdruck 5:

```
1.000    1.000
1.000    1.000
1.000    6.747
0.974    3.756
1.000    1.000
1.000    1.000
```

3.4 Fehlermeldungen

Fehlermeldungen werden auf das Standardausgabegerät ausgegeben.

Das Programm wird mit der Funktion exit (status) verlassen, der Wert von status ist:
0: Das Programm wurde ordungsgemäß verlassen.
1: Es ist ein Fehler aufgetreten, das Programm wurde vorzeitig abgebrochen.

4. Programmbeschreibung

Das Programm besteht aus dem Hauptprogramm und der Funktion aufruf zur Ausgabe der Aufrufanweisung. Im folgenden werden die einzelnen Abschnitte des Hauptprogramms beschrieben. Die Überschriften dieses Abschnitts entsprechen im Wortlaut den Kommentarzeilen des Programms linop.

4.1. Auswahl der Speicherbelegungsfunktion

In der Datei simfeh.cun wird die Zeichenkette "Speicher" durch den Namen einer Speicherbelegungsfunktion ersetzt. Im allgemeinen kann die Funktion calloc verwendet werden. Unter DOS steht damit jedoch nicht genug Speicher zur Verfügung, wenn die Anzahl L der Matrixelemente der Ansprechmatrix groß wird.

Unter DOS gibt es bei Verwendung von der Funktion calloc und dem Speichermodell compact oder huge die Einschränkung:

L < 1600 beim Aufruf von sifmax
L < 3600 beim Aufruf von sif

Unter DOS kann durch Verwendung der Funktion halloc (anstatt calloc) und dem Speichermodell huge der zur Verfügung stehende Speicherplatz erweitert werden. Bei Verwendung von halloc gilt dann nur noch die Einschränkung:

L < 6700 beim Aufruf von sifmax
L < 14700 beim Aufruf von sif

4.2. Einbinden von Programm Simfeh

Die Datei simfeh.cun enthält die verwendeten Funktionen:
- sif: Extremwerte
- sifmax: Größter Abstand der Extremwerte
- ressif: Einlesen der Ansprechmatrix, Reservierung von Speicherplatz

4.3. Parameter zur Steuerung von Testausgabe

Die Standardausgabe von Daten durch die einzelnen Funktionen erfolgt in Abhängigkeit vom Wert der Variablen "test" im allgemeinen dann, wenn test > = 0. In der Regel wird der Wert von test beim Aufruf einer Funktion um 1 erniedrigt, so daß sich durch den Wert von test in der Hauptfunktion der Umfang der ausgegeben Daten steuern läßt.

Zur Berechnung von Dosiswerten ohne Angaben über das Spektrum ist test = 0, einschließlich Angaben über das Spektrum ist test = 1.

Zur Berechnung der Eigenschaften der Ansprechmatrix ohne Testausgaben ist test = 1.

Wenn Zwischenergebnisse ausgegeben werden sollen, muß test > 1 eingegeben werden.

4.4. Berechnung der Eigenschaften der Ansprechmatrix

Je nach Art der mit dem Aufruf eingegebenen Parameter wird entweder 4.4. oder 4.5. durchgeführt:

Die Berechnung der Eigenschaften der Ansprechmatrix wird durchgeführt, wenn beim Aufruf keine Meßwerte angegeben werden:
Syntax: linop test Ansprechmatrix
Beispiel: linop 1 m.txt

4.4.1 Einlesen der Ansprechmatrix und Reservierung von Speicherplatz

Angaben über die Fehlerabschätzung und die Ansprechmatrix befinden sich auf einer beim Aufruf des Programms anzugebenden Datei im Textmode, die mit Hilfe eines Editors bereitgestellt werden kann:

1. Zeile: Anzahl der Einzelmeßwerte: m
Anzahl der Ansprechvektoren: n
2. Zeile: absoluter Fehleranteil für die m Komponenten der Messung
Der absolute Fehleranteil muß immer > 0 angegeben werden.
3. Zeile: relativer Fehleranteil für die m Komponenten der Messung
Der relative Fehleranteil muß immer ≥ 0 angegeben werden.

Die folgenden Zeilen enthalten je einen Ansprechvektor, d.h. die Werte des Ansprechvermögens der Komponenten des Dosimeters.

Beispiel für m = 2 und n = 5:

```
2 5
 0.01    0.01
 0.03    0.03
 0.77    0.0
 3.37    0.5
 1.12    0.93
 0.15    1.78
 0.12    0.59
```

Für die Auswertung von Messungen entsprechen die in dem Beispiel angegebenen absoluten und relativen Fehler den in der Praxis mindestens erforderlichen Fehlern. Zur Untersuchung der Eigenschaften der Ansprechmatrix kann mit verschieden großen Fehlern experimentiert werden, solange die angegebenen Einschränkungen beachtet werden.

Die Menge der Ansprechvektoren bildet die transponierte Ansprechmatrix.

4.4.2 Durchfuehrung der Berechnung der Eigenschaften der Ansprechmatrix

Aufruf der Funktion sifmax

4.5. Auswertung von Messergebnissen

Die Auswertung wird durchgeführt, wenn beim Aufruf Einzelmeßwerte angegeben werden:
Syntax: linop test Ansprechmatrix Einzelmeßwerte
Für Einzelmeßwerte können alternativ
- der Name einer Datei mit den Ergebnissen mehrerer Messungen
- die Einzelmesswerte einer Messung
angegeben werden.

Beispiel 1: linop 1 m.txt me.txt
Beispiel 2: linop 1 m.txt 1 1 1

Beim Beispiel 2 gibt es 3 Einzelmeßwerte zu jeder Messung.

Je nach Art der mit dem Aufruf eingegebenen Parameter wird entweder 4.5.2 oder 4.5.3 durchgeführt:

4.5.1 Einlesen der Ansprechmatrix und Reservierung von Speicherplatz

siehe 4.4.1

4.5.2 Auswertung von Messergebnissen aus einer Datei

Zum Aufruf des Programms siehe Beispiel 1 in Abschnitt 4.5.

4.5.2.1 Öffnen der Datei mit Messdaten

Messdaten (für eine oder mehrere Messungen) befinden sich auf einer beim Aufruf des Programms anzugebenden Datei im Textmode. Jede Zeile der Datei entspricht einer Messung und enhält die bei der Messung festgelegte Nummer der Messung und m Einzelmesswerte. Beispiel einer Zeile für m = 5:
1017 0.5 1.0 1.0 0.3 0.0

4.5.2.2 Bearbeitung der Datensätze mit Messdaten

4.5.2.2.1 Einlesen eines Datensatzes mit Messdaten

Die Daten müssen im Textmode vorliegen und werden auf dem in der Datei simfeh.cun definierten Speicherplatz messw abgelegt.

4.5.2.2.2 Berechnung von Dosiswerten und Spektren

Eine Ausgabe von Daten erfolgt nur dann, wenn die Funktion sif ordnungsgemäß abgeschlossen werden konnte.

4.5.3 Auswertung einer einzelnen Messung, Eingabe mit dem Aufruf

Messdaten für eine Messung werden beim Aufruf des Programms eingegeben. Zum Aufruf des Programms siehe Beispiel 2 in Abschnitt 4.5.

4.5.3.1 Übertragung der Einzelmesswerte

Die Daten werden auf dem in der Datei simfeh.cun definierten Speicherplatz messw abgelegt.

4.5.3.2 Berechnung von Dosiswerten und Spektren

Eine Ausgabe von Daten erfolgt nur dann, wenn die Funktion sif ordnungsgemäß abgeschlossen werden konnte.

5. Literaturverzeichnis

1. Deutsche Norm DIN 6816: Filmdosimetrie nach dem Filteranalytischen Verfahren zur Strahlenschutzüberwachung. Beuth Verlag, Berlin und Köln 1984

2. Bergmann, F.; Chanourdie, J. C.: Reconnaissance assistée des rayonnements a partir des données d'un dosimètre photographique. Radioprotection 8(1973) 189-205

3. I. N. Bronstein und K. A. Semendjajew, Taschenbuch der Mathematik, 21. Auflage, Verlag Harri Deutsch, Thun und Frankfurt/Main (1988)

4. Ambrosi, P.: Auswertung vonFilmdosimetern mittels Linearkombination der korrigierten Anzeige. Physikalisch-Technische Bundesanstalt, PTB-Bericht PTB-Dos-17, Braunschweig 1988

5. Kragh, P.; Nitschke, J.: Auswertung von Filmdosimetern mittels Linearer Optimierung. Kerntechnik 57(1992) No. 5, 306-311

6. Ambrosi, P.; Böhm, J.; Kragh, P.; Ritzenhoff, K. H.; The influence of the evaluation procedure of individual dosemeters on their performance. Physikalisch-Technische Bundesanstalt, PTB-Mitteilungen 102, Braunschweig 1992, 283-288

7. Kragh, P.; Schmidt, H.: Einsatz des Simplexverfahrens zur Auswertung von Filmdosimetern. Kerntechnik 59(1994) No. 3, 105-108

6. Quelltext des C-Programms

/* Datei: linop.c, 25. 4. 95 */

```
/* 1. Auswahl der Speicherbelegungsfunktion: ******************************/
   #define speicher calloc(
/* #define speicher halloc((long) */

/* 2. Einbinden von Programm Simfeh ***************************************/
#include "simfeh.cun"
```

/* Auswertung von Dosimetern durch lineare Optimierung */

```
int aufruf(void);

void main(int argc, char *argv []) {
static long   nummer;
static double *a;
static int    test, i;


if(argc < 3) { aufruf(); exit(1); }


/* 3. Parameter zur Steuerung von Testausgabe *****************************/
test = atoi(argv [1]);
if(test < 0) { printf("\r\nFehler: test < 0"); aufruf(); exit(1); }
if(test > 0) {
   printf("Programm linop                              ");
   datum();
}

/* 4. Berechnung der Eigenschaften der Ansprechmatrix *********************/
if(argc == 3) {

   if(test > 0) {
      printf("\r\n\nEigenschaften der Ansprechmatrix");
         printf("\r\n*******************************");
   }

   /* 4.1 Einlesen der Ansprechmatrix und Reservierung von Speicherplatz */
   if(ressif(test, 0, argv [2])) {
      printf("\r\nAbbruch in ressif, 0");
      exit(1);
   }

   /* 4.2 Durchfuehrung der Berechnung der Eigenschaften der Ansprechmat.*/
   if(sifmax (test)) { printf("\r\nAbbruch in sifmax"); exit (1); }
   exit (0);
}
```

```
/* 5. Auswertung von Messergebnissen ************************************/

if(test > 0) {
   printf("\r\n\nAuswertung von Messungen");
   printf("\r\n************************");
   if(argc == 4)
      printf("\r\n(0) Einzelmesswerte auf der Datei: %s", argv [3]);
}

/* 5.1 Einlesen der Ansprechmatrix und Reservierung von Speicherplatz ****/
if(ressif(test, 1, argv [2])) { printf("\r\nAbbruch in ressif, 1"); exit(1); }

/* 5.2 Auswertung von Messergebnissen aus einer Datei ********************/
if(argc == 4) {

   /* 5.2.1 Oeffnen der Datei mit Messdaten ******************************/
   if((str = fopen(argv [3], "rb")) == 0) {
      printf("\r\nEine Datei mit Messwerten wurde nicht gefunden");
      exit(1);
   }

   /* 5.2.2 Bearbeitung der Datensaetze mit Messdaten ********************/
   while(feof(str) == 0) {

      /* 5.2.2.1 Einlesen eines Datensatzes mit Messdaten ****************/
      fscanf(str, "%ld", &nummer); if(feof(str)) break;
      i = 0; a = messw; while(i++ < malt) fscanf(str, "%lf", a++);
      if(test > 0) printf("\r\n\n\n(7)  Nummer der Messung: %ld", nummer);

      /* 5.2.2.2 Berechnung von Dosiswerten und Spektren *****************/
      if(sif (test)) { printf("\r\nFehler bei der Optimierung"); exit (1); }

   }

   fclose (str);
   exit(0);
}

/* 5.3 Auswertung einer einzelnen Messung, Eingabe mit dem Aufruf *******/
if(argc == (3 + malt)) {

   /* 5.3.1 Übertragung der Messwerte ************************************/
   i = 0; a = messw; while(i < malt) { *a++ = atof(argv [i + 3]); i++; }
   if(test > 0) printf("\r\n\n");

   /* 5.3.2 Berechnung von Dosiswerten und Spektren **********************/
   if(sif (test)) { printf("\r\nFehler bei der Optimierung"); exit (1); }

   exit(0);
}

aufruf (); exit(1);
}
```

/* Ausgabe einer Aufrufanweisung für das Programm linop */

```
int aufruf(void) {
   printf("\r\nfalsche Parameter");
   printf("\r\nAufruf des Programms:");
   printf(" linop test Ansprechmatrix [Einzelmeßwerte]");
   printf("\r\n\ntest: Parameter zur Bestimmung der Standardausgabe");
   printf("\r\n   0:        Ausgabe der beiden Extremwerte der Dosis");
   printf("\r\n             (nur für die Auswertung von Messungen)");
   printf("\r\n   1:        Ausgabe von Spektren und Dosiswerten");
   printf("\r\n   >1:       Ausgabe von Spektren, Dosiswerten und Zwischenergeb-
nissen");
   printf("\r\n\nAnsprechmatrix: Bezeichnung der Datei mit der Ansprechmatrix");
   printf("\r\n\nEinzelmesswerte (optional): ");
   printf("\r\n   entweder: Name der Datei mit Einzelmeßwerten");
   printf("\r\n   oder:     Einzelmeßwerte für eine Messung");
return(0);
}
```

Anhang 1

Simfeh: C-Funktionen für das Simplexverfahren unter Berücksichtigung von Fehlern

Inhaltsverzeichnis

1. Zusammenfassung
2. Gleichungssysteme
3. Globale Variable
4. Allgemeine Variable
5. Beschreibung der C-Funktionen
6. Quellenverzeichnis
7. Quelltext der C-Funktionen

1. Zusammenfassung

Das Programm Simfeh enthält, aufbauend auf dem Programm Simplex [1] weitere Funktionen in der Programmiersprache C zur linearen Optimierung mit Hilfe des Simplexverfahrens [2] insbesondere zur Berücksichtigung von Fehlern für die Absolutglieder des Gleichungssystems.

Dabei wird zunächst durch lineare Optimierung das kleinste Vielfache von vorgegebenen Fehlern bestimmt, für das eine Lösung existiert. Für die durch Multiplikation mit dem erhaltenen Faktor enstehenden Fehler wird dann die ursprüngliche Optimierungsaufgabe gelöst.

Außerdem werden die Absolutglieder bestimmt, für die der Abstand der beiden Extremwerte der Zielfunktion den größten Wert annimmt.

Für das Linearkombinationsverfahren werden die optimalen Koeffizienten bestimmt.

Die Programme entsprechen dem ANSI C Standard und wurden unter DOS und unter UNIX erprobt.

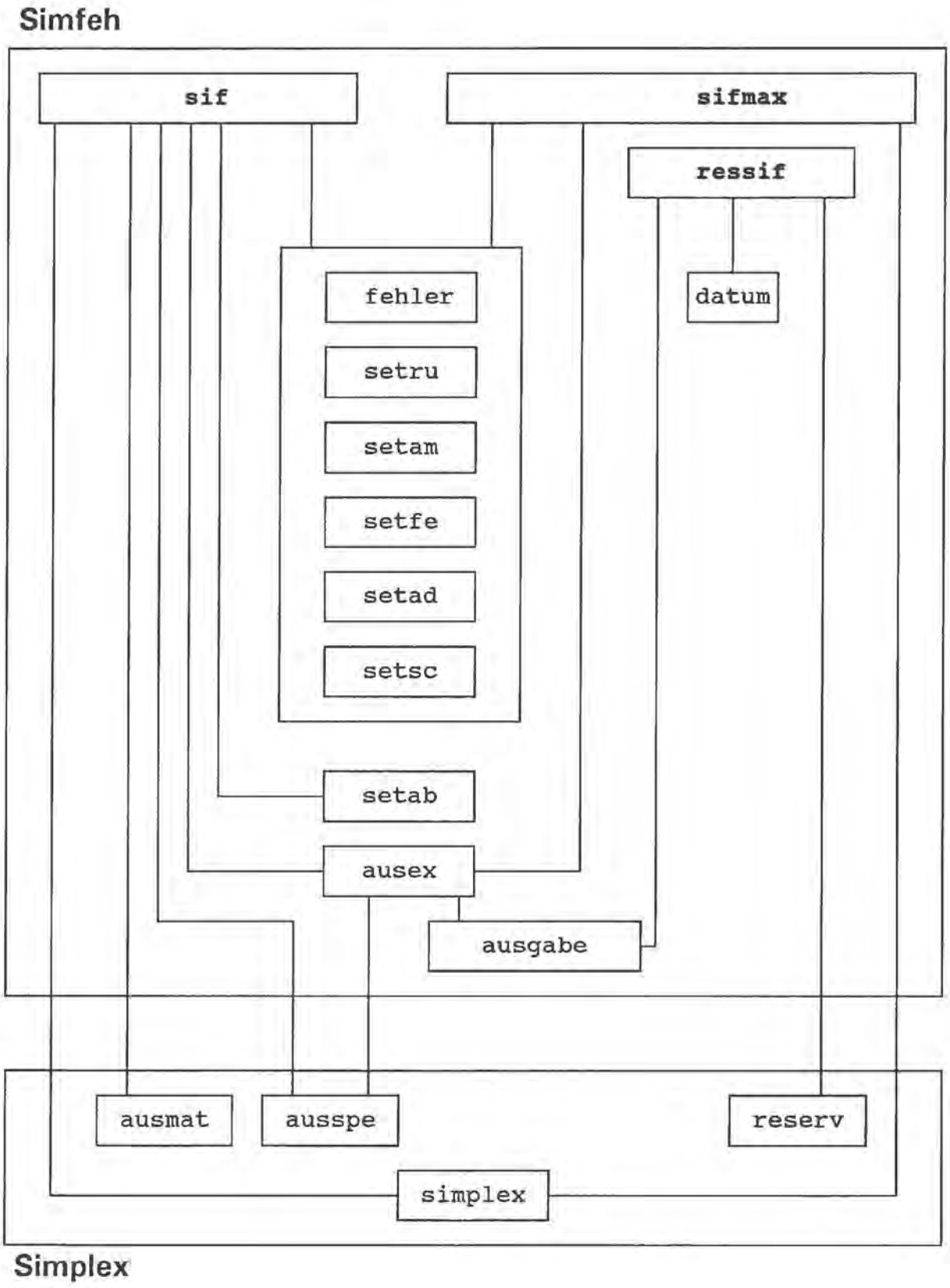
Für die Anwendung genügt der Aufruf der Funktionen:

- sif: Extremwerte
- sifmax: Größter Abstand der Extremwerte
- ressif: Einlesen der Ansprechmatrix, Reservierung von Speicherplatz

Daneben gibt es die intern verwendeten Funktionen:

- setru: Zurücksetzen der Matrix
- fehler: Fehler berechnen
- setam: Setzen der Ansprechmatrix
- setfe: Setzen der Fehlerelemente
- setad: Setzen der Adressen der Schlupfelemente
- setsc: Setzen der Schlupfelemente
- setab: Setzen der Absolutglieder
- ausgabe: Ausgabe der Zeichenkette "t" und der i double-Werte "a"
- ausex: Ausgabe der beiden Extremwerte und von deren Spektren
- datum: Ausgabe des Systemdatums

Die Beziehung der Funktionen wird durch das folgende Diagramm veranschaulicht, wobei die Funktionen von oben nach unten aufgerufen werden.



2. Gleichungsysteme

Im Programm Simplex [1] wird die Lösung wird die Lösung eines linearen Gleichungssystems durchgeführt mit den Unbekannten x_i und vorgegebenen Kostanten (den Elementen der Ansprechmatrix $a_{i,k}$ und den Absolutgliedern f_k):

$$(1) \quad \sum_{i=1}^{n} a_{i,k} * x_i = f_k , \qquad k = 1 \text{ bis } m$$

zur Bestimmung der beiden Extremwerte einer Zielfunktion:

$$(2) \quad D = \sum_{i=1}^{n} x_i ,$$

die mit D_{min} und mit D_{max} bezeichnet werden.

Für vorgegebene vorgegebene Kostanten $a_{i,k}$ und f_k muß eine Lösung von (1) nicht immer existieren. Eine Lösbarkeit kann durch Erweiterung der Ansprechmatrix (Vergrößerung der Anzahl der Zeilen und der Spalten) zur Aufnahme von Fehlervariablen erreicht werden. Im folgenden werden Fehler ϵ_k der Absolutglieder f_k zugelassen. Durch Wahl genügend großer Werte der Fehler ϵ_k soll immer mindestens eine Lösung gefunden werden können.

2.1 Extremwerte

Die Einführung von Fehlern ϵ_k der Absolutglieder f_k führt zu dem Gleichungssystem:

$$(3) \quad \left| \sum_{i=1}^{n} a_{i,k} * x_i - f_k \right| \leq f * \epsilon_k , \qquad k = 1 \text{ bis } m$$

Die Größen $a_{i,k}$, f_k und ϵ_k sind im folgenden Konstante, x_i und f sind zu bestimmende Variable, deren Werte mit der Funktion sif in zwei Schritten bestimmt wird. Eine Lösung existiert immer, wenn $\epsilon_k > 0$.

Schritt 1: Bestimmung des kleinstmöglichen Fehler

Es wird eine Lösung (x_i und f) gesucht unter Minimierung der Zielfunktion:

$$(4) \quad Z = f .$$

Der erhaltene Minimalwert von f wird um 0.5 vergrößert. Ist f dann kleiner als 1, so wird f auf 1 vergrößert. Der erhaltene Wert von f ist für die weitere Rechnung konstant.

Die kleinstmöglichen Fehler werden aus ϵ_k durch Multiplikation mit f errechnet.

Schritt 2: Berechnung der Extremwerte der Zielfunktion für die erhaltenen Fehler
Lösungen (x_i) werden gesucht unter Minimierung (bzw. Maximierung) der Zielfunktion (2).

Vor Anwendung des Simplexalgorithmus wird zur Vermeidung der Betragsfunktion das Ungleichungssystem (3) durch Einführung zusätzlicher Fehlervariabler e_k umgewandelt in:

$$(5)\qquad \begin{aligned} e_k &\leq 2 * f * \epsilon_k , && k = 1 \text{ bis } m \\ \sum_{i=1}^{n} a_{i,k} * x_i + e_k &= f_k + f * \epsilon_k , && k = 1 \text{ bis } m \end{aligned}$$

Zur Vermeidung der Ungleichungen werden in (5) zusätzliche Schlupfvariable s_k eingeführt:

$$(6)\qquad \begin{aligned} e_k + s_k &= 2 * f * \epsilon_k , && k = 1 \text{ bis } m \\ \sum_{i=1}^{n} a_{i,k} * x_i + e_k &= f_k + f * \epsilon_k , && k = 1 \text{ bis } m \end{aligned}$$

Das Gleichungssystem (6) dient der Optimierung der Zielfunktion (2) mit Schritt 2. Zur Optimierung der Zielfunktion (4) im Schritt 1 ist f Variable und erscheint deshalb auch auf der linken Seite des Gleichheitszeichens:

$$(7)\qquad \begin{aligned} -2 * \epsilon_k * f + e_k + s_k &= 0 , && k = 1 \text{ bis } m \\ -\epsilon_k * f + \sum_{i=1}^{n} a_{i,k} * x_i + e_k &= f_k , && k = 1 \text{ bis } m \end{aligned}$$

Zur Lösung des Hilfsproblems des Simplexverfahrens [2] durch die Funktion sif werden in (7) und (8) weitere m Schlupfvariable eingeführt.

2.2 Größter Abstand der Extremwerte

Mit der Funktion sifmax werden die Absolutglieder f_i bestimmt, für die der Quotient D_{max} / D_{min} den größten Wert annimmt. Dafür werden zwei Lösungen $\{x_i\}$ und $\{y_i\}$ gesucht mit:

$$(8)\qquad \begin{aligned} D_{min} &= \sum_{i=1}^{n} y_i = 1 , \\ D_{max} &= \sum_{i=1}^{n} x_i . \end{aligned}$$

maximal wird. Das führt zu dem Gleichungssystem:

$$(9)\qquad \begin{aligned} \sum_{i=1}^{n} y_i &= 1 \\ \left| \sum_{i=1}^{n} a_{i,k} * x_i - \sum_{i=1}^{n} a_{i,k} * y_i \right| &\leq \epsilon_k , && k = 1 \text{ bis } m \end{aligned}$$

mit der Zielfunktion D_{max} (8), für die das Maximum gesucht wird. Wie in 2.1 werden Fehler- und Schlupfvariable eingeführt, damit ergibt sich aus (9):

$$(10)\qquad \begin{aligned} \sum_{i=1}^{n} y_i &= 1 \\ e_k + s_k &= 2 * \epsilon_k , && k = 1 \text{ bis } m \\ \sum_{i=1}^{n} a_{i,k} * x_i - \sum_{i=1}^{n} a_{i,k} * y_i + e_k &= \epsilon_k , && k = 1 \text{ bis } m \end{aligned}$$

Die Anzahl der Anteile y_i und x_i ungleich 0 in beiden Lösungen von (10) ist im allgemeinen gleich m+1.

2.3 Linearkombinationsverfahren

Für das Linearkombinationsverfahren werden mit der Funktion sifmax die Koffizienten eines linearen Ausdrucks zur Abschätzung der Zielfunktion (2) bestimmt. Dabei wird der Umstand ausgenutzt, daß dabei im wesentlichen das zu 2.2 duale lineare Optimierungsproblem vorliegt. Deshalb müssen von allen Spalten der Ansprechmatrix ($a_{i,k}$) nur noch diejenigen berücksichtigt werden, für die die Lösung von (10) Anteile x_i oder y_i ungleich 0 enthält. Damit ergibt sich aus der Lösung von (10) das folgende lineare Gleichungssystem mit im allgemeinen m + 1 Gleichungen zur Bestimmung der Koeffizienten c_k und von f des Linearkombinationsverfahrens:

$$(11)\qquad \begin{aligned} \sum_{k=1}^{m} a_{i,k} * c_k &= 1 && \text{für alle i mit } x_i \text{ ungleich 0} \\ \sum_{k=1}^{m} a_{i,k} * c_k - f &= 0 && \text{für alle i mit } y_i \text{ ungleich 0} \end{aligned}$$

In der Lösung von (11) entspricht der Wert von f dem Quotienten D_{max} / D_{min} aus Abschnitt 2.2 bei Vernachlässigung der Fehler ϵ_k.

Das lineare Gleichungssystem (11) ist im allgemeinen eindeutig lösbar. Hier wird es in ein lineares Optimierungsproblem umgewandelt, damit die vorhandenen Funktionen verwendet werden können. Es werden die Variablen u_k und v_k eingeführt:

$$(12)\qquad c_k = u_k - v_k \;, \quad u_k \geq 0 \;, \quad v_k \geq 0 \;.$$

Damit ergibt sich aus (11) das Gleichungssystem:

$$(13)\qquad \begin{aligned} \sum_{k=1}^{m} a_{i,k} * u_k - \sum_{k=1}^{m} a_{i,k} * v_k &= 1 && \text{für alle i mit } x_i \text{ ungleich 0} \\ \sum_{k=1}^{m} a_{i,k} * u_k - \sum_{k=1}^{m} a_{i,k} * v_k - f &= 0 && \text{für alle i mit } y_i \text{ ungleich 0} \end{aligned}$$

Da die Lösung eindeutig ist, wird eine Zielfunktion nicht benötigt. Es muß nur das Hilfsproblem (mit der dafür üblichen Zielfunktion) gelöst werden.

3. Globale Variable

Die globalen Variablen des Programms Simplex behalten Namen und Bedeutung.

double kama: **Ansprechmatrix**
kama enthält die Elemente der Ansprechmatrix, das sind die Koeffizienten $a_{i,k}$ aus Gleichung (1) ohne Berücksichtigung von Fehler- und Schlupfvariablen. Die Funktionen erwarten die Speicherung der Matrix in der Reihenfolge der Spalten:

Spalte 1, Spalte 2, ... Spalte nalt

und innerhalb der Spalten in der Reihenfolge der Zeilen:

Zeile 1, Zeile 2, ... Zeile malt.

Die Matrix "am" des Programms Simplex enthält die Elemente der Ansprechmatrix "kama" und darüber hinaus weitere Elemente.

int nalt: **Anzahl der Spalten der Ansprechmatrix**

int malt: **Anzahl der Zeilen der Ansprechmatrix**

int nneu0: **Anzahl der Unbekannten**
nneu0 entspricht nneu (im Programm Simplex) bei der Reservierung von am.

double *messw: **Messwerte**
Die Werte von *messw sind die Absolutglieder f_k (1).

double *e0: **Fehler**
Die Werte von *e0 sind die Fehler ϵ_k (3) der Absolutglieder f_k. Wenn der Wert des Faktors f von ϵ_k (3) schon vorliegt, entsprechen die Werte von *e0 den Fehlern f * ϵ_k.

double *ea0: **Absolute Fehler**

double *er0: **Relative Fehler**
Aus den absoluten und relativen Fehler werden Fehler berechnet, im allgemeinen ist:
*e0 = *ea0 + *er0 * *messw.
Wenn der Wert von f vorliegt, wird *e0 noch mit f muliplizierd.

double *lkvk: **Koeffizienten für das Linearkombinationsverfahren**

spektrum *anfangsw: **Spektrum von Zwischenergebnissen**

spektrum *minw: **Spektrum zum Minimalwert der Zielfunktion**

spektrum *maxw: **Spektrum zum Maximalwert der Zielfunktion**

4. Allgemeine Variable

int test:
Die Standardausgabe von Daten durch die einzelnen Funktionen erfolgt in Abhängigkeit vom Wert der Variablen "test" im allgemeinen dann, wenn test > = 0. In der Regel wird der Wert von test beim Aufruf einer Funktion um 1 erniedrigt, so daß sich durch den Wert von test in der Hauptfunktion der Umfang der ausgegeben Daten steuern läßt.

5. Beschreibung der C-Funktionen

Die folgenden Dateien werden benötigt:
- simplex.cun

Die folgenden Standarddateien werden benötigt:
- time.h

Der Rückgabewert der Funktionen ist 0, wenn sie bis zum Ende durchlaufen werden.

5.1 Extremwerte

int sif (int test)

Globale Daten:
- mneu
- nneu
- am
- nalt
- malt
- nneu0
- e0
- anfangsw
- minw
- maxw

Rückgabe:

0	Die Berechnung wurde durchgeführt
1	Es ist ein Fehler aufgetreten

Beschreibung:

Die Berechnung der Extremwerte der Zielfunktion (2) und der zugehörigen Spektren wird durchgeführt.

Gliederung:

1.	Berechnung der Fehler e0
2.	Berechnung des kleinsten Fehlerfaktors f
2.1	Setzen der Matrix am
2.2	Berechnung des Spektrums anfangsw
3.	Aktualisieren der Fehler unter Berücksichtigung von f
4.	Minimalwert der Zielfunktion
4.1	Setzen der Matrix am
4.2	Berechnung des Spektrums minw
5.	Maximalwert der Zielfunktion
5.1	Setzen der Matrix am
5.2	Berechnung des Spektrums maxw
6.	Standardausgabe des Ergebnisses

5.2 Eigenschaften der Ansprechmatrix

int sifmax (int test)

Globale Daten:
- mneu
- nneu
- basis
- am
- nalt
- malt
- nneu0
- e0
- kama
- messw
- lkvk
- anfangsw
- minw
- maxw

Rückgabe:

0	Die Berechnung wurde durchgeführt
1	Es ist ein Fehler aufgetreten

Beschreibung:
Zur Beschreibung der Eigenschaften der Ansprechmatrix erfolgt:
- Berechnung der Absolutglieder f_i, für die der Quotient D_{max} / D_{min} den größten Wert annimmt und der zugehörigen Lösungen $\{x_i\}$ und $\{y_i\}$
- Berechnung der Koeffizienten für das Linearkombinationsverfahren

Gliederung:

1.	Größter Abstand der Extremwerte
1.1	Berechnung von Messwerten als Mittelwerten der Komponenten der Ansprechvektoren
1.2	Berechnung der Fehler ϵ_i
1.3	Setzen der Matrix am
1.4	Berechnung des Spektrums
1.5	Aufteilung des Spektrums anfangsw auf minw und maxw
1.6	Berechnung der mittleren Messwerte zu diesen Spektren
1.7	Standardausgabe der Spektren
2.	Berechnung der Koeffizienten für das Linearkombinationsverfahren
2.1	Setzen der Matrix
2.1.1	Setzen der Matrix für alle i mit y_i ungleich 0 (Minimum)
2.1.2	Setzen der Matrix für alle i mit x_i ungleich 0 (Maximum)
2.1.3	Setzen der Schlupfelemente
2.2	Berechnung des Spektrums
2.3	Berechnung der Koeffizienten
2.4	Standardausgabe der Koeffizienten
2.5	Berechnung der Extremwerte der Dosis für die Vektoren der Ansprechmatrix
2.6	Standardausgabe der Extremwerte der Dosis für die Vektoren der Ansprechmatrix

5.3 Einlesen der Ansprechmatrix, Reservierung von Speicherplatz

int ressif (int test, int fall, char *dateim)

Globale Daten:
- mneu
- nneu
- nalt
- malt
- nneu0
- e0
- ea0
- er0
- kama
- messw
- lkvk
- anfangsw
- minw
- maxw

Funktionsparameter:

- fall	Parameter zur Steuerung von Speicherplatzreservierung und Ausgabe von Daten	
	0	Größter Abstand der Extremwerte (Funktion sifmax)
	1	Extremwerte (Funktion sif)
- dateim	Name der Datei mit der Ansprechmatrix	

Rückgabe:

0	Die Berechnung wurde durchgeführt
1	Es ist ein Fehler aufgetreten

Beschreibung:

Einlesen der Daten der Ansprechmatrix und Reservierung von Speicherplatz

Gliederung:

1. Öffnen der Datei mit der Ansprechmatrix

2. Einlesen von malt und nalt

3. Berechnung von mneu und nneu (abhängig vom Wert von fall)

4. Reservierung von Speicherplatz
   Für die Zeichenkette "speicher" muß noch eine Funktionsbezeichnung eingesetzt werden:

- "halloc((long)	unter DOS für größere Ansprechmatritzen
- "calloc("	sonst

5. Einlesen von Fehlern und Ansprechmatrix

6. Standardausgabe der Daten

5.4 Zurücksetzen der Matrix

int setru (void)

Globale Daten:
- mneu
- am
- nneu0

Rückgabe:
0 Die Berechnung wurde durchgeführt

Beschreibung:
Zurücksetzen aller reservierten Elemente der Matrix am

5.5 Fehler berechnen

int fehler (void)

Globale Daten:
- malt
- e0
- ea0
- er0
- messw

Rückgabe:
0 Die Berechnung wurde durchgeführt

Beschreibung:
Berechnung des Fehlers: ϵ_i = *(ea0 + i) + *(er0 + i) * ϵ_i

5.6 Setzen der Ansprechmatrix

int setam (double f, double *amz)

Globale Daten:
- mneu
- nalt
- malt
- kama

Funktionsparameter:
- f Faktor für die Elemente der Ansprechmatrix kama in am
- amz Adresse des 1. Elementes in am, für das ein Element der Ansprechmatrix gesetzt werden soll

Rückgabe:
0 Die Berechnung wurde durchgeführt

Beschreibung:
Setzen der Elemente der Ansprechmatrix multipliziert mit dem Faktor f in der Matrix am

5.7 Setzen der Fehlerelemente

int setfe (double *amz)

Globale Daten:
- mneu
- malt

Funktionsparameter:
- amz Adresse des 1. Elementes in am, für das ein Fehlerelement gesetzt werden soll

Rückgabe:
0 Die Berechnung wurde durchgeführt

Beschreibung:
Setzen von malt Fehlerelementen in der Matrix am

5.8 Setzen der Adressen der Schlupfelemente

int setad (int spalte, int anzahl)

Globale Daten:
- basis

Funktionsparameter:
- spalte Höchste Nummer der Spalten, für die Schlupfelemente gesetzt werden sollen
- anzahl Anzahl der zu setzenden Schlupfelemente

Rückgabe:
0 Die Berechnung wurde durchgeführt

Beschreibung:
Abspeichern von anzahl Adressen von Schlupfelementen der Matrix am in basis

5.9 Setzen der Schlupfelemente

int setsc (int anzahl)

Globale Daten:
- mneu
- basis
- am

Funktionsparameter:
- anzahl Anzahl der zu setzenden Schlupfelemente

Rückgabe:
0 Die Berechnung wurde durchgeführt

Beschreibung:
Setzen von anzahl Schlupfelementen in der Matrix am, deren Adressen in basis vorhanden sein müssen

5.10 Setzen der Absolutglieder

int setab (double f)

Globale Daten:
- am
- malt
- e0
- messw

Funktionsparameter:

- f	Faktor des Fehlers ϵ_i

Rückgabe:

0	Die Berechnung wurde durchgeführt

Beschreibung:
Setzen der Absolutglieder in der Matrix am

5.11 Ausgabe der Zeichenkette "t" und der i double-Werte "a"

int ausgabe (char *t, double *a, int i)

Globale Daten:
- nalt
- malt
- e0
- messw
- minw
- maxw

Funktionsparameter:

- t	Auszugebender Text
- a	Adresse des Vektors
- i	Anzahl der auszugebenden Elemente

Rückgabe:

0	Die Ausgabe wurde durchgeführt

Beschreibung:
Ausgabe eines Vektors (double)

5.12 Ausgabe der beiden Extremwerte und von deren Spektren

int ausex (void)

Globale Daten:
- nalt
- malt
- e0
- messw
- minw
- maxw

Rückgabe:
0 Die Ausgabe wurde durchgeführt

Beschreibung:
Standardausgabe von:
- Einzelmesswerte
- Fehler der Einzelmesswerte
- Spektren für die beiden Extremwerte
- Extremwerte

5.13 Ausgabe des Systemdatums

int datum (void)

Rückgabe:
0 Die Ausgabe wurde durchgeführt

Beschreibung:
Ausgabe des Systemdatums auf Standardausgabe

6. Quellenverzeichnis

1. P.Kragh, C-Funktionen für das Simplexverfahren, 10. 4. 1995.

2. I. N. Bronstein und K. A. Semendjajew, Taschenbuch der Mathematik, 21. Auflage, Verlag Harri Deutsch, Thun und Frankfurt/Main (1988)

7. Quelltext der C-Funktionen

/* Datei: simfeh.cun, 26. 4. 95
C-Funktionen für das Simplexverfahren bei fehlerbehafteten Daten */

```
#include "simplex.cun"
#include <time.h>


FILE *str;

static int    nalt, malt, nneu0;
static double *e0, *ea0, *er0, *kama, *messw, *lkvk;
static spektrum *anfangsw, *minw, *maxw;


int sif      (int test);
int sifmax   (int test);
int ressif   (int test, int fall, char *dateim);
int fehler   (void);
int setru    (void);
int setam    (double f, double *amz);
int setad    (int spalte, int anzahl);
int setsc    (int anzahl);
int setab    (double f);
int setfe    (double *amz);
int ausgabe  (char *t, double *a, int i);
int ausex    (void);
int datum    (void);
```

/* Extremwerte */

```
int sif (int test) {
static double f, *a;
static int    i;

/* 1. Berechnung der Fehler e0 ****************************************/
   fehler();

/* 2. Berechnung des kleisten Fehlerfaktors f *************************/

   /* 2.1 Setzen der Matrix am ****************************************/
   nneu = nneu0;
   setru ();
   setam (1, am + 2 * (mneu + 1) + 1);
   setfe (am + (nalt + 2) * (mneu + 1) + malt);
   setad (nneu, mneu);
   setsc (mneu);
   a = am + mneu + 2; /***************************** Koeffizienten von f */
   i = 0; while(i < malt) { *a++ = - *(e0 + i); i++; }
   i = 0; while(i < malt) { *a++ = - 2 * *(e0 + i); i++; }
   setab (0);
```

```
   /* 2.2 Berechnung des Spektrums anfangsw ******************************/
   i = simplex (test - 1, anfangsw, 1, 1, nneu0 - malt + 1, nneu0, 1);
   if(i == 1) return (1);
   if(test > 1) { ausmat ("sif 1"); ausspe ("sif 1", anfangsw, mneu); }

/* 3. Aktualisierung der Fehler unter Berücksichtigung von f *************/
   f = - *am;
   f += 0.5; if(f < 1) f = 1;
   i = 0; while(i < malt) *(e0 + i++) *= f;


/* 4. Minimalwert der Zielfunktion ***************************************/

   /* 4.1 Setzen der Matrix am ***************************************/
   nneu = nneu0 - 1;
   setru ();
   setam  (1, am + mneu + 2);
   setfe (am + (nalt + 1) * (mneu + 1) + malt);
   setad (nneu, mneu);
   setsc (mneu);
   setab (1);

   /* 4.2 Berechnung des Spektrums minw **********************************/
   i = simplex (test - 1, minw, 1, nalt, nneu0 - malt, nneu0 - 1, 1);
   if(i == 1) return (1);


/* 5. Maximalwert der Zielfunktion ***************************************/

   /* 5.1 Setzen der Matrix am ***************************************/
   nneu = nneu0 - 1;
   setru ();
   setam (1, am + mneu + 2);
   setfe (am + (nalt + 1) * (mneu + 1) + malt);
   setad (nneu, mneu);
   setsc (mneu);
   setab (1);

   /* 5.2 Berechnung des Spektrums maxw **********************************/
   i = simplex (test - 1, maxw, 1, nalt, nneu0 - malt, nneu0 - 1, - 1);
   if(i == 1) return (1);

/* 6. Standardausgabe des Ergebnisses ************************************/
   if(test > 0) {
      printf("\r\n(9)  Lineare Optimierung (vorgegebene Einzelmesswerte)");
      printf("\r\n     ==============================================");
      ausex ();
   }
   if(test <= 0) printf("\r\n%8.3lf %8.3lf", minw->wert, maxw->wert);

return (0);
}
```

/* Eigenschaften der Ansprechmatrix */

```
int sifmax (int test) {
static int    i, k, l, p;
static double *a, *b, z, dmi, dma, dmi0, dma0;
static spektrum *sp, *pi, *pa;

/* 1. Groesster Abstand der Extremwerte ********************************/
/* ********************************************************************/

   if(test > 1) {
      printf("\r\n\n\nLineare Optimierung, Testausdrücke");
      printf("\r\n==================================");
   }

/* 1.1 Berechnung von Messwerten als Mittelwerten der Komponenten der
       Ansprechvektoren ************************************************/
   i = 0; while(i < malt) *(messw + i++) = 0;
   k = 0; a = kama;
   while(k++ < nalt) {
      i = 0; while(i < malt) *(messw + i++) += *a++;
   }
   i = 0; while(i < malt) *(messw + i++) /= (double) nalt;

/* 1.2 Berechnen der Fehler e0 *****************************************/
   fehler();


/* 1.3 Setzen der Matrix am ********************************************/
   nneu = nneu0;
   setru ();
   setam  (1, am + mneu + 2);
   setam  (-1, am + (mneu + 1) * (nalt + 1) + 1);
   setfe (am + (mneu + 1) * (2 * nalt + 1) + malt);
   setad (nneu - 1, mneu - 1); *(basis + mneu) = nneu;
   setsc (mneu);
   a = am + (mneu + 1) * (nalt + 2) - 1; /*********** mneu-te Gleichung */
   i = 0; while(i < nalt) *(a + (mneu + 1) * i++) = 1;

   i = 0; a = am + 1; /************************ Absolutglieder setzen */
   while(i < malt) {
       *a = *(e0 + i);
       *(a + malt) = 2 * *(e0 + i);
       i++; a++;
   }
   *(am + mneu) = 1;
```

```
/* 1.4 Berechnung des Spektrums ***************************************/
   i = simplex (test - 1, anfangsw, 1, nalt, nneu - malt, nneu, -1);
   if(i == 1) return (1);

/* 1.5 Aufteilung des Spektrums anfangsw auf minw und maxw **************/
   pi = minw; pa = maxw; i = 0;
   while(i++ <= mneu) {
      pi->basisv = nneu + 1; pi++;
      pa->basisv = nneu + 1; pa++;
   }
   sp = anfangsw + 1; pi = minw; pa = maxw; i = 0; l = 0;
   pi->basisv = 0; pi->wert = 1;              pi++;
   pa->basisv = 0; pa->wert = anfangsw->wert; pa++;
   while(i++ < mneu) {
      p = sp->basisv;
      if(p <= (2 * nalt)) {
           if(p <= nalt) { pa->basisv = p; pa->wert = sp->wert; pa++;}
           if(p > nalt) {
              p -= nalt;      pi->basisv = p; pi->wert = sp->wert; pi++;
           }
      l++;
      }
      sp++;
   }

/* 1.6 Berechnung der mittleren Meßwerte zu diesen Spektren *************/
   i = 0; while(i < malt) *(messw + i++) = 0;
   sp = minw + 1; i = 0; z = 0;
   while(i++ < mneu) {
      p = sp->basisv;
      if(p <= nalt) {
           k = 0; a = kama + (malt * (p - 1)) ;
           while(k < malt) *(messw + k++) += sp->wert * *a++;
           z += sp->wert;
      }
      sp++;
   }
   i = 0; while(i < malt) *(messw + i++) /= z;

/* 1.7 Standardausgabe zu diesen Spektren *******************************/
   if(test > 0) {
      printf("\r\n\n(8)  Lineare Optimierung (Eigenschaften der Ansprechma-
trix)");
           printf("\r\n
==================================================");
      ausex ();
   }


/* 2. Berechnung der Koeffizienten für das Linearkombinationsverfahren ***/
/* ********************************************************************/

   if(test > 1) {
      printf("\r\n\n\nLinearkombinationsverfahren, Testausdrücke");
      printf("\r\n=======================================");
   }
```

```
/* 2.1 Setzen der Matrix ************************************************/
   if(l != (malt + 1)) {
      printf("\r\nKoeffizienten für das Linearkombinationsverfahren");
      printf("\r\nkönnen nicht berechnet werden");
      printf("\r\nVersuchen Sie es mit kleineren Fehlern!\r\n");
      return (1);
   }
   mneu = l; nneu = (2 * malt) + 1 + mneu;
   setru();

/* 2.1.1 Setzen der Matrix für alle i mit y(i) ungleich 0 (Minimum) ******/
   sp = minw + 1; i = 0; l = 0;
   while(i++ < mneu) {
      p = sp->basisv;
      if(p <= nalt) {
           l++; b = am + l;
           *b = 0; b += (mneu + 1); /****************** Absolutbetrag */
           k = 0; a = kama + (malt * (p - 1)) ; /************* Matrix */
           while(k++ < malt) { *b = *a++; b += (mneu + 1); }
           k = 0; a = kama + (malt * (p - 1)) ;
           while(k++ < malt) { *b = - *a++; b += (mneu + 1); }
           *b = -1; /****************************** Koeffizient von q */
      }
      sp++;
   }

/* 2.1.2 Setzen der Matrix für alle i mit x(i) ungleich 0 (Maximum) ******/
   sp = maxw + 1; i = 0;
   while(i++ < mneu) {
      p = sp->basisv;
      if(p <= nalt) {
           l++; b = am + l;
           *b = 1; b += (mneu + 1); /****************** Absolutbetrag */
           k = 0; a = kama + (malt * (p - 1)) ; /************* Matrix */
           while(k++ < malt) { *b = *a++; b += (mneu + 1); }
           k = 0; a = kama + (malt * (p - 1)) ;
           while(k++ < malt) { *b = - *a++; b += (mneu + 1); }
      }
      sp++;
   }

/* 2.1.3 Setzen der Schlupfelemente *************************************/
   setad (nneu, mneu);
   setsc (mneu);

   if(test > 1) ausmat("Linearkombination");

/* 2.2 Berechnung des Spektrums *****************************************/
   i = simplex (test - 1, minw, 1, -1, nneu - malt, nneu, 1);
   if(i == 1) return (1);
```

```
/* 2.3 Berechnung der Koeffizienten ************************************/
   i = 0; while(i < malt) *(lkvk + i++) = 0;
   i = 0; sp = minw + 1;
   while(i++ < malt) {
      k = sp->basisv;
      if(k <= malt) *(lkvk + k - 1) += sp->wert;;
      if(k > malt) *(lkvk + k - 1 - malt) -= sp->wert;;
   sp++;
   }
   dmi0 = 1; dma0 = (minw + malt + 1)->wert;

/* 2.4 Standardausgabe der Koeffizienten *******************************/
   if(test > 0) {
      printf("\r\n\n(15) Linearkombinationsverfahren");
           printf("\r\n     ==========================");
      printf("\r\n(16) Koeffizienten:");
      i = 0; while(i < malt) printf("%7.3lf", *(lkvk + i++));
      printf("\r\n(17) Extremwerte der Dosis: %7.3lf, %7.3lf", dmi0, dma0);
   }

/* 2.5 Berechnung der Extremwerte der Dosis für die Vektoren der
       Ansprechmatrix ****************************************************/
   dmi = dmi0; dma = dma0;
   if(test > 0) {
      printf("\r\n\n(18) Dosiswerte für die Vektoren der Ansprechmatrix ");
      printf("\r\n     (Linearkombinationsverfahren):");
      printf("\r\n     Lfd.Nr.  Dosiswerte");
   }
   k = 0; while(k < nalt) {
      i = 0; a = kama + (k * malt); z = 0;
      while(i < malt) { z += *(lkvk + i) * *a; i++; a++; }
      if(z < dmi) dmi = z;
      if(z > dma) dma = z;
      if(test > 0) printf("\r\n%8d     %7.3lf", k + 1, z);
      k++;
   }

/* 2.6 Standardausgabe der Extremwerte der Dosis für die Vektoren der
       Ansprechmatrix ****************************************************/
   if(test > 0) {
      printf("\r\n\n(19) Extremwerte der Vektoren der Ansprechmatrix\r\n   ");
      printf("  (Linearkombinationsverfahren): %7.3lf, %7.3lf\r\n", dmi, dma);
   }

return (0);
}
```

/* Einlesen der Ansprechmatrix, Reservierung von Speicherplatz */

```
/* ersetze speicher durch "calloc(" oder durch "halloc((long)" */

int ressif (int test, int fall, char *dateim) {
static int i, l;
static double *a;
static long il, ll;

/* 1. Öffnen der Datei mit der Ansprechmatrix ****************************/
   if((str = fopen(dateim, "rb")) == NULL) {
      printf("\r\nAnsprechmatrix nicht gefunden");
      return (1);
   }

/* 2. Einlesen von malt und nalt *****************************************/
   if(fscanf(str, "%d", &malt) != 1) {
      printf("\r\n1. Fehler der Ansprechmatrix"); return (1); }
   if(fscanf(str, "%d", &nalt) != 1) {
      printf("\r\n2. Fehler der Ansprechmatrix"); return (1); }
   if((malt < 1) || (nalt < 1)) {
      printf("\r\n\n3. Eingabefehler?"); return (1); }

/* 3. Berechnung von mneu und nneu ***************************************/
   if(fall == 1) { /********** Extremwerte, zum Aufruf der Funktion sif */
      mneu = 2 * malt;
      nneu = nalt + mneu + malt + 1;
   }
   if(fall == 0) { /*** Größter Abstand der Extremwerte, Funktion sifmax */
      mneu = 2 * malt + 1;
      nneu = 2 * nalt + mneu + malt;
   }
   nneu0 = nneu;

/* 4. Reservierung von Speicherplatz *************************************/
   if(reserv()) return(1);
   messw = (double *) malloc(sizeof(double) * malt);
   anfangsw = (spektrum *) malloc(sizeof(spektrum) * (mneu + 1));
   minw   = (spektrum *) malloc(sizeof(spektrum) * (mneu + 1));
   maxw   = (spektrum *) malloc(sizeof(spektrum) * (mneu + 1));
   e0     = (double *) malloc(sizeof(double) * malt);
   ea0    = (double *) malloc(sizeof(double) * malt);
   er0    = (double *) malloc(sizeof(double) * malt);
   lkvk = (double *) malloc(sizeof(double) * malt);
   kama = (double *) speicher malt * nalt, sizeof(double));
   if((messw == 0) || (anfangsw == 0) || (minw == 0) || (maxw == 0) ||
      (e0 == 0) || (ea0 == 0) || (er0 == 0) || (kama == 0))
      { printf("\r\n4. ressif, Kein Speicher"); return (1); }
```

```
/* 5. Einlesen von Fehlern und Ansprechmatrix ***************************/
   i = 0; a = ea0;
   while(i++ < malt) if(fscanf(str, "%lf", a++) != 1) {
      printf("\r\n5. Fehler der Ansprechmatrix"); return (1); }
   i = 0; a = er0;
   while(i++ < malt) if(fscanf(str, "%lf", a++) != 1) {
      printf("\r\n6. Fehler der Ansprechmatrix"); return (1); }
   il = 0; a = kama; ll = (long) malt * (long) nalt;
   while(il < ll) {
      il += 1;
      if(fscanf(str, "%lf", a++) != 1) {
           printf("\r\n7. Fehler der Ansprechmatrix");
           printf("\r\nil, ll: %ld %ld", il, ll);
           return (1);
      }
   }
   fclose (str);

/* 6. Standardausgabe der Daten *****************************************/
   if(test > 0) {
      printf("\r\n\n\n(1)  Ansprechmatrix\r\n     ==============");
      printf("\r\n(2)  Ansprechmatrix auf der Datei:  %s", dateim);
      printf("\r\n(3)  m, n:  %d   %d", malt, nalt);
      ausgabe ("(4)  Absolute Fehler", ea0, malt);
      ausgabe ("(5)  Relative Fehler", er0, malt);
      printf("\r\n\n(6)  Ansprechmatrix:");
      printf("\r\n     Lfd.Nr.  Ansprechvektoren");
      l = 0; a = kama;
      while(l++ < nalt) {
           printf("\r\n%8d    ", l);
           i = 0; while(i++ < malt) printf("%7.3lf", *a++);
      }
      printf("\r\n");
   }

return (0);
}
```

/* Zurücksetzen der Matrix */

```
int setru (void) {
static int i, l;
static double *a;
   l = (mneu + 1) * (nneu0 + 1); i = 0; a = am;
   while(i++ < l) *a++ = 0;
return (0);
}
```

/* Setzen der Ansprechmatrix */

```
int setam (double f, double *amz) {
static double *a, *b;
static int    i, k;
   i = 0; b = kama;
   while(i < nalt) {
      k = 0; a = amz + ((mneu + 1) * i); while(k++ < malt) *a++ = f * *b++;
      i++;
   }
return (0);
}
```

/* Setzen der Fehlerelemente */

```
int setfe (double *amz) {
static double *a;
static int    i;
   a = amz; i = 0;
   while(i++ < malt) { *a = 1; *(a + malt) = 1; a += mneu; }
return (0);
}
```

/* Setzen der Adressen der Schlupfelemente */

```
int setad (int spalte, int anzahl) {
static int    i, k, *iz;
   iz = basis; *iz++ = 0; i = spalte;
   k = 0; while(k++ < anzahl) *iz++ = i--;
return (0);
}
```

/* Setzen der Schlupfelemente */

```
int setsc (int anzahl) {
static int i;
   i = 1;
   while(i <= anzahl) { *(am + i + ((mneu + 1) * *(basis + i))) = 1; i++; }
return (0);
}
```

/* Absolutglieder setzen */

```
int setab (double f) {
static double *a;
static int    i;
   i = 0; a = am + 1;
   while(i < malt) {
       *a = *(messw + i) + f * *(e0 + i); /* Messwerte */
       *(a + malt) = 2 * f * *(e0 + i);         /* Fehler */
       i++; a++;
   }
return (0);
}
```

/* Fehler berechnen */

```
int fehler (void) {
static int i;
   i = 0;
   while(i < malt) {
      *(e0 + i) = *(ea0 + i) + *(er0 + i) * *(messw + i);
      i++;
   }
return (0); }
```

/* Ausgabe der Zeichenkette "t" und der i double-Werte "a" */

```
int ausgabe (char *t, double *a, int i) {
static int s, k;
static double *az, zw;
   printf("\r\n%s: ", t);
   if(i > 10) printf("\r\n");
   k = 0; az = a; zw = 0;
   while(k++ < i) {
      if(*az > zw) zw = *az;
      if(*az < -zw) zw = -(*az);
      az++;
   }
   k = 0; s = 0; az = a;
   while(k++ < i) {
      if(zw > 0.005) {
          if(*az) printf("%7.3lf", *az); else printf("  0    ");
      } else {
          if(*az) printf("%9.5lf", *az); else printf("  0      ");
      }

      az++;
      s += 1; if(s > 12) { s = 0; printf("\r\n"); }
   }
return (0); }
```

/* Ausgabe der beiden Extremwerte und von deren Spektren */

```
int ausex(void) {
static int i, k;
   ausgabe ("(10) Einzelmesswerte", messw, malt);
   ausgabe ("(11) Fehler          ", e0, malt);
   i = 0; k = 0;
   while(i <= malt) { if((minw + i)->basisv <= nalt) k = i; i++; }
   ausspe ("(12) Minimum, Dosisanteile der Ansprechvektoren", minw + 1, k - 1);
   i = 0; k = 0;
   while(i <= malt) { if((maxw + i)->basisv <= nalt) k = i; i++; }
   ausspe ("(13) Maximum, Dosisanteile der Ansprechvektoren", maxw + 1, k - 1);
   printf("\r\n(14) Extremwerte der Dosis: %8.3lf %8.3lf\r\n",
      minw->wert, maxw->wert);
return (0); }
```

/* Ausgabe des Systemdatums */

```
int datum (void) {
struct tm *nt;
time_t ac;
   time(&ac); nt = localtime(&ac);
   printf("den %d. %d. %d",
      (int) nt->tm_mday, (int) nt->tm_mon + 1, (int) nt->tm_year + 1900);
return (0); }
```

Anhang 2

Simplex: C-Funktionen für das Simplexverfahren

Inhaltsverzeichnis

1. Zusammenfassung
2. Globale Variable
3. Strukturen
4. Allgemeine Variable
5. Beschreibung der C-Funktionen für das Simplexverfahren
6. Literaturverzeichnis
7. Quelltext der C-Funktionen für das Simplexverfahren

1. Zusammenfassung

Das Programm Simplex enthält Funktionen in der Programmiersprache C zur linearen Optimierung mit Hilfe des Simplexverfahrens [1]. Die Programme entsprechen dem ANSI C Standard und wurden unter DOS und unter UNIX erprobt.

Ziel des Verfahrens ist die Lösung eines linearen Gleichungssystems mit den Unbekannten x_i und den vorgegebenen Kostanten $a_{i,k}$, f_k und α_i:

$$(1) \quad \sum_{i=1}^{n} a_{i,k} * x_i = f_k , \qquad k = 1 \text{ bis } m$$

unter gleichzeitiger Minimierung einer linearen Zielfunktion:

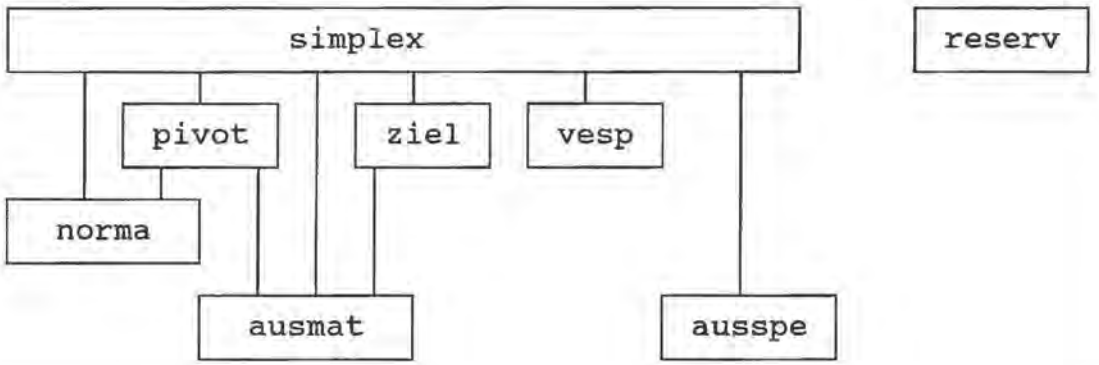
$$(2) \quad Z = \sum_{i=1}^{n} \alpha_i * x_i .$$

Für die Anwendung gedacht sind die Funktionen:
- reserv: Reservierung von Speicherplatz
- simplex: Lineare Optimierung
- ausmat: Ausgabe der transponierten Ansprechmatrix
- ausspe: Ausgabe der Lösung

Die übrigen Funktionen werden von der Funktion simplex aufgerufen:
- pivot: Suchen des Pivotelementes und Umrechnung
- vesp: Vergleich zweier Werte spektrum
- norma: Umrechnung der Matrix auf die Basisvariable zeile
- ziel: Zielfunktion setzen

Die Beziehung der Funktionen wird durch das folgende Diagramm veranschaulicht, wobei die Funktionen von oben nach unten aufgerufen werden.

Bild 1



2. Globale Variable

Zur Abspeicherung der Zielfunktion und der Koeffizienten des Gleichungssystems für m Gleichungen mit n Variablen wird eine Matrix mit (m + 1) Zeilen und (n + 1) Spalten gebildet:

$a_{i,k}$, $i = 0$ bis n, $k = 0$ bis m.

Die Koeffizienten der Zielfunktion sind in der 0-ten Zeile und die der k-ten Gleichung in der k-ten Zeile enthalten.

Zielfunktion: $Z = a_{0,0} + \sum_{i=0}^{n} a_{i,0} * x_i$.

Gleichungssystem: $a_{0,k} = \sum_{i=1}^{n} a_{i,k} * x_i$, $k = 1$ bis m.

Im allgemeinen ist $n > m$. Es wird nur nach Lösungen des Gleichungssystems unter Minimierung der Zielfunktion gesucht. Für diese Lösungen sind höchstens m Werte der n Variablen X_i von null verschieden. Diese m Variablen einer Lösung heißen Basisvariable. Die Basisvariablen werden beschrieben durch die zugehörigen Spaltennummern:

b_k, $k = 1$ bis m, wobei $1 \leq b_k \leq n$.

Mit Hilfe der Funktion "ausmat" werden Basis und transponierte Matrix (d.h. Zeilen und Spalten sind vertauscht) in der folgenden Anordnung ausgegeben:

Bild 2:

Basis:					
	b_1	b_2	.	.	b_m
transponierte Matrix:					
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	.	.	$a_{0,m}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	.	.	$a_{1,m}$
.	.	.	.	.	.
.	.	.	.	.	.
$a_{n,0}$	$a_{n,1}$	$a_{n,2}$	.	.	$a_{n,m}$

Die Bezeichnungen für die globalen Variablen im Programm sind:

int mneu: **Anzahl der Gleichungen**

int nneu: **Anzahl der Unbekannten**

double *am: Matrix, Absolutwerte und Zielfunktionskoeffizienten
Die Funktionen erwarten die Speicherung der Matrix in der Reihenfolge der Spalten:
Spalte 0, Spalte 1, ... Spalte nneu
und innerhalb der Spalten in der Reihenfolge der Zeilen:
Zeile 0, Zeile 1, ... Zeile mneu.

int *basis: Basisvariable
Zu einer zulässigen Lösung gehören mneu Basisvariable. Sie werden in Form der zugehörigen Spaltennummern in *basis abgelegt.

Ein Beispiel für die Darstellung der Werte der globalen Variablen durch die Funktionen "ausmat" und "ausspe" ist dargestellt in Bild 3. Weil die Matrix zum Zweck der Darstellung transponiert wurde, gehört die Zielfunktion zur Spalte 0, die 1. Gleichung zur Spalte 1 und entsprechendes gilt für die übrigen Gleichungen.

Bild 3:

```
Basis:
            2       1       7       4

transponierte Matrix:
-33.356   0.510 33.356   0.761   1.067
  0         0     1.000   0       0
  0       1.000   0       0       0
  3.904   0.711 -3.904  -0.089  -0.125
  0         0     0       0       1.000
  7.303   0.899 -7.303   0.834  -0.234
 57.297  -0.640-57.297  -1.306  -0.834
  0         0     0       1.000   0

Spektrum:
      0       1       2       4       7
 33.356  33.356   0.510   1.067   0.761
```

Die Basis bezeichnet die Basisvariablen X_2, X_1, X_7 und X_4.

Die Matrix befindet sich in kanonischer Form , d.h. alle Koeffizienten der Basisvariablen sind 0 außer dem Koeffizienten $a_{2,1}$ von X_2 in der 1. Gleichung, dem Koeffizienten $a_{1,2}$ von X_1 in der 2. Gleichung, dem Koeffizienten $a_{7,3}$ von X_7 in der 3. Gleichung und dem Koeffizienten $a_{4,4}$ von X_4 in der 4. Gleichung.

Die Zielfunktion (0. Spalte) ist umgerechnet auf die Basisvariablen, d.h. die Koeffizienten der Basisvarablen ($a_{1,0}$, $a_{2,0}$, $a_{4,0}$ und $a_{7,0}$) sind 0.

In dieser Darstellung kann eine Lösung aus der ersten Zeile der Matrix abgelesen werden. Sie ist (sortiert nach dem Index der Basisvariablen) angegeben unter der Überschrift "Spektrum:". Für X_1 = 33.356, X_2 = 0.51, X_4 = 1.067 und X_7 = 0.761 ist der Wert der Zielfunktion 33.356 (das negative Vorzeichen von 33.356 in der Matrix rührt daher, daß bei der Bestimmung des Minimalwertes einer Zielfunktion die Koeffizienten der Zielfunktion mit -1 multipliziert in der Matrix abgelegt werden). Diese Lösung gehört zum Minimum der Zielfunktion, was sich daran ablesen läßt, daß alle Koeffizieneten der Zielfunktion ($a_{1,0}$, $a_{2,0}$, ...$a_{7,0}$) nicht neagtiv sind.

3. Strukturen

Die Lösung des Optimierungsproblems wird abgelegt mit Hilfe der Struktur:
struct spektrum { int basisv; double wert; }
- basisv: Nummer der Variablen
- wert: Wert der Variablen

Die Lösung besteht aus mneu Wertepaaren spektrum { basisv, wert }.

4. Allgemeine Variable

int test:
Die Standardausgabe von Daten durch die einzelnen Funktionen erfolgt in Abhängigkeit vom Wert der Variablen "test" im allgemeinen dann, wenn test > = 0. In der Regel wird der Wert von test beim Aufruf einer Funktion um 1 erniedrigt, so daß sich durch den Wert von test in der Hauptfunktion der Umfang der ausgegeben Daten steuern läßt.

5. Beschreibung der C-Funktionen für das Simplexverfahren

Die folgenden Standarddateien werden benötigt:
- stdlib.h
- stdio.h

Der Rückgabewert der Funktionen ist 0, wenn sie bis zum Ende durchlaufen werden.

5.1 Simplex

int simplex (int test, spektrum *ew, int t1, int t2, int f1, int f2, double w)

Globale Daten:
- nneu
- mneu
- am (die Zielfunktion ist nicht gesetzt)
- basis

Funktionsparameter:
- t1, t2 Intervall der Spalten, für die die Koeffizienten der Zielfunktion des Hauptproblems ungleich 0 sein sollen
  Wenn t2 < t1, dann wird nur das Nebenproblem gelöst.
- f1, f2 Intervall der Spalten, für die die Koeffizienten der Zielfunktion des Hilfsproblems ungleich 0 sein sollen
  Der Wert von f2 entspricht der Anzahl der Variablen.
- w Wert der Koeffizienten der Zielfunktion des Hauptproblems
  w > 0 zur Bestimmung des Minimalwertes der Zielfunktion
  w < 0 zur Bestimmung des Maximalwertes der Zielfunktion

Erzeugte Daten:
- ew->wert Absolutbetrag der Zielfunktion
- (ew + i) i-te Komponente des Spektrums, i = 1, ...mneu

Rückgabe:
0 Ein Extremwert wurde gefunden
1 Ein Extremwert wurde nicht gefunden

Beschreibung:
Das Simplexverfahren durch Lösung des Hilfs- und des Hauptproblems wird für die Matrix "am" durchgeführt. Bei der Lösung des Hilfsproblems werden zur Ermöglichung einer kanonischen Darstellung der Matrix zunächst zusätzliche künstliche Variable eingeführt, die dann durch entsprechende Wahl der Zielfunktion zunächst wieder elemeniert werden müssen. Anschließend wird dann die Zielfunktion des eigentlichen Problems (Hauptproblem) eingeführt und dieses gelöst (wenn t2 ≥ t1).

5.2 Suchen des Pivotelementes und Umrechnung

int pivot(int test)

Globale Daten:
- nneu
- mneu
- am
- basis

Rüchgabe:

0	eine Pivotelement wurde gefunden
1	ein Pivotelement wurde nicht gefunden

Beschreibung:
Es wird ein Pivotelement gesucht. Im Erfolgsfall wird die zugehörige Variable als Basisvariable akzeptiert. Die Nummer der Pivotspalte wird in basis abgelegt. Die Matrix wird auf die neue Basisvariable umgerechnet.

5.3 Vergleich zweier Werte spektrum

int vesp (spektrum *a, spektrum *b)

Rückgabe:

-1	a->basisv < b->basisv
0	a->basisv = b->basisv
1	a->basisv > b->basisv

Beschreibung:
Vergleich der Werte von a->basisv und b->basisv zur Verwendung in der Sortierfunktion qsort

5.4 Umrechnung der Matrix auf die Basisvariable zeile

int norma (int zeile)

Globale Daten:
- nneu
- mneu
- am (die Zielfunktion ist nicht gesetzt)
- basis

Funktionsparameter:

- zeile	Zeile des Pivotelementes

Rückgabe:
0 Die Umrechnung wurde durchgeführt
1 Das durch zeile und basis vorgegebene Matrixelement (Pivotelement) ist gleich 0

Beschreibung:
Die angegebene Zeile legt mit Hilfe von basis ein Pivotelement fest. Für das Pivotelement wird die Matrix umgerechnet, so daß anschließend das Pivotelement gleich 1 und die übrigen Matrixelemente in der Pivotspalte gleich 0 sind.

5.5 Zielfunktion setzen

int ziel (int test, int m1, int m2, double w)

Globale Daten:
- nneu
- mneu
- am (die Zielfunktion ist nicht gesetzt)

Funktionsparameter:
- m1, m2 Intervall der Spalten, für die die Koeffizienten der Zielfunktion gleich w gesetzt werden
- w Wert der Koeffizienten der Zielfunktion
  w > 0 zur Bestimmung des Minimalwertes der Zielfunktion
  w < 0 zur Bestimmung des Maximalwertes der Zielfunktion

Rückgabe:
0 Die Rechnung wurde durchgeführt

Beschreibung:
Die Zielfunktion wird gesetzt.

5.6 Reservierung von Speicherplatz

int reserv (void)

Globale Daten:
- nneu
- mneu

Rückgabe:
0 Die Reservierung wurde durchgeführt

Beschreibung:
Es wird Speicherplatz für am und basis reserviert. In dem aufrufenden Programm muß mit Hilfe eines Makros die Speicherbelegungsfunktion festgelegt werden:

- #define speicher calloc(
  für UNIX und bei geringem Speicherbedarf für DOS
- #define speicher halloc((long)
  bei etwas mehr Speicherbedarf für DOS

5.7 Ausgabe der transponierten Ansprechmatrix

int ausmat (char *t)

Globale Daten:
- nneu
- mneu
- am (die Zielfunktion ist nicht gesetzt)
- basis

Funktionsparameter:

- t	Auszugebender Text

Rückgabe:

0	Die Ausgabe wurde durchgeführt

Beschreibung:
Basis und transponierte Matrix (Zeilen und Spalten sind vertauscht) werden ausgegeben.

5.8 Ausgabe des Spektrums

int ausspe (char *t, spektrum *ew, int m)

Funktionsparameter:

- t	Auszugebender Text
- ew	Spektrum
- m	Anzahl der auszugebenden Elemente von Spektrum

Rückgabe:

0	Die Ausgabe wurde durchgeführt

Beschreibung:
Von dem Spektrum ew werden m Elemente ausgegeben.

6. Literaturverzeichnis

1. I. N. Bronstein und K. A. Semendjajew, Taschenbuch der Mathematik, 21. Auflage, Verlag Harri Deutsch, Thun und Frankfurt/Main (1988)

7. Quelltext der C-Funktionen für das Simplexverfahren

/* Datei: simplex.cun, C-Funktionen für das Simplexverfahren, 25. 4. 95 */

```
#include <stdlib.h>
#include <stdio.h>
#include <malloc.h>


typedef struct spk {
   int basisv;
   double wert; } spektrum;


static int    *basis, nneu, mneu;
static double *am;


int pivot   (int test);
int simplex (int test, spektrum *ew, int t1, int t2, int f1, int f2, double w);
int vesp    (spektrum *a, spektrum *b);
int norma   (int zeile);
int ziel    (int test, int m1, int m2, double w);
int reserv  (void);
int ausmat  (char *t);
int ausspe  (char *t, spektrum *ew, int m);
```

/* Simplex */

```
int simplex (int test, spektrum *ew, int t1, int t2, int f1, int f2, double w) {
static int i, k, lauf, n1, *iz, mb;
static spektrum *ewz;


   /***************************************************** Hilfsproblem */
   nneu = f2;

   ziel(test - 1, f1, f2, 1);/***************** Setzen der Zielfunktion */
   if(test > 0) ausmat("simplex 1");

   i = 1; while(i <= mneu) norma(i++); /**** Umrechnen der Zielfunktion */

   lauf = 0; n1 = 3 * nneu; mb = f2; k = 0; /****************** Simplex */
   while ((mb >= f1) && (k == 0) && (lauf++ < n1)) {
      k = pivot(test - 1);
      i = 0; iz = basis; mb = 0;
      while(i++ <= mneu) { if(*iz > mb) mb = *iz; iz++; }
   }
   if(test > 0) ausmat("simplex 2");
   if(mb >= f1) return (1);


   /***************************************************** Hauptproblem */
   if(t2 >= t1) {
      nneu = f1 - 1;

      ziel(test - 1, t1, t2, w);/************** Setzen der Zielfunktion */
      if(test > 0) ausmat("simplex 3");

      i = 1; while(i <= mneu) norma(i++); /* Umrechnen der Zielfunktion */
      if(test > 0) ausmat("simplex 4");

      lauf = 0; n1 = 3 * nneu; k = 0; /*********************** Simplex */
      while ((k == 0) && (lauf++ < n1)) k = pivot(test - 1);
      if(test > 0) ausmat("simplex 5");
      if(k == 0) return (1);
   }

   /***************************************** Erzeugung von Spektrum */
   i = 0; ewz = ew;
   while(i <= mneu) {
      ewz->basisv = *(basis + i);
      ewz->wert = *(am + i);
      ewz++; i++;
   }
   ew->wert *= -w;
   qsort(ew, mneu + 1, sizeof(spektrum), vesp);
   if(test > 0) ausspe ("simplex 6", ew, mneu);

return (0); }
```

/* Suchen des Pivotelementes und Umrechnung */

```
int pivot(int test) {
static int i, zeile, spalte, m1, anfang;
static double *a, z, *c, u;

   m1 = mneu + 1;

   /****************************************** Suche der Pivotspalte */
   spalte = 0; a = am + m1; z = *a + 1; i = 1;
   while(i <= nneu) {
      if(*a < z) { z = *a; spalte = i; }
      i++; a += m1;
   }
   if(z >= 0) return (1);

   /****************************************** Suche der Pivotzeile */
   zeile = 0; a = am + 1; c = a + (m1 * spalte); anfang = 1; i = 1;
   while(i <= mneu) {
      if(*c > 0) {
           z = *a / *c;
           if(anfang) { u = z; zeile = i; anfang = 0; }
           if((anfang == 0) && (z < u)) { u = z; zeile = i; }
      }
      i++; a++; c++;
   }
   if(zeile == 0) return (1);

   /****************************************** Aktualisieren der Basis */
   *(basis + zeile) = spalte;
   norma(zeile);

   if(test > 0) ausmat ("pivot");

return (0); }
```

/* Vergleich zweier Werte spektrum */

```
int vesp (spektrum *a, spektrum *b) {

   if(a->basisv > b->basisv) return (1);
   if(a->basisv < b->basisv) return (-1);

return (0); }
```

/* Umrechnung der Matrix auf die Basisvariable zeile */

```
int norma (int zeile) {
static double p, *pz, *a, *zl, *zp;
static int    spalte, ml, i, l;

   if((zeile < 1) || (zeile > mneu)) return 1;

   ml = mneu + 1;
   spalte = *(basis + zeile);
   p = *(am + (ml * spalte) + zeile);
   if(p == 0) return 1;

   if(p != 1) { /* Umrechnung der Pivotzeile */
      a = am + zeile;
      i = 0; while(i++ <= nneu) { *a /= p; a += ml; }
   }

   l = 0; /* Umrechnung der übrigen Zeilen */
   while(l <= mneu) {
      if(l != zeile) {
           zp = am + zeile;
           zl = am + l;
           pz = zl + (ml * spalte); p = *pz;
           if(p != 0) {
              i = 0; while(i++ <= nneu) { *zl -= *zp * p; zl += ml; zp += ml; }
              *pz = 0;
           }
      }
      l++;
   }

return (0); }
```

/* Zielfunktion setzen */

```
int ziel (int test, int ml, int m2, double w) {
static int i;
static double *a;

   a = am; *a = 0; i = 0;
   while(i <= nneu) {
      *a = 0; if((i >= ml) && (i <= m2)) *a = w;
      a += (mneu + 1); i++;
   }

   if(test > 0) ausmat ("ziel");

return (0); }
```

/* Reservierung von Speicherplatz */

```
/* ersetze speicher durch "calloc(" oder durch "halloc((long)" */

int reserv (void) {

   am = (double *) speicher (mneu + 1) * (nneu + 1), sizeof(double));

   basis = (int *) malloc(sizeof(int) * (mneu + 1));
   if((basis == NULL) || (am == NULL)) {
      printf("\r\nreserv, Kein Speicher");
      return (1);
   }

return (0);
}
```

/* Ausgabe der transponierten Ansprechmatrix */

```
int ausmat (char *t) {
static int k, i, *b;
static double *a;

   printf("\r\n\n%s, Basis: \r\n   ", t);
   k = 1; b = basis + 1; while(k++ <= mneu) printf("%7d", *b++);

   printf("\r\n%s, transponierte Matrix:", t);
   k = 0; a = am;
   while(k++ <= nneu) {
      printf("\r\n");
      i = 0;
      while(i++ <= mneu) {
          if(*a) printf("%7.3lf", *a); else printf("  0    ");
          a++;
      }
   }

return (0); }
```

/* Ausgabe des Spektrums */

```
int ausspe (char *t, spektrum *ew, int m) {
static int k;
static spektrum *ewz;

   printf("\r\n%s, Spektrum: \r\n", t);
   k = 0; ewz = ew;
   while(k++ <= m) { printf("%7d", ewz->basisv); ewz++; }
   k = 0; ewz = ew; printf("\r\n   ");
   while(k++ <= m) {
      if(ewz->wert) printf("%7.3lf", ewz->wert); else printf("  0    ");
      ewz++;
   }

return (0); }
```

Liste der bisher erschienenen BfS-ISH-Berichte

BfS-ISH-140/89
Bayer, A.; Braun, H.; Dehos, R.; Frasch, G.; Haubelt, R.; Hoppe-Schönhammer, J.; Kaul, A.; Löbke, A.; Werner, M.
Erfassung, Dokumentation und strahlenhygienische Bewertung vorliegender Aktivitätsmeßdaten aus der Bundesrepublik Deutschland als Folge des Reaktorunfalles im Kernkraftwerk Tschernobyl.

BfS-ISH-141/90
Stamm-Meyer, A.; Stanek, H.; Bögl, K.W.
Biologische Indikatoren zum Nachweis von Strahlenexpositionen - Thymidinkonzentration im Humanserum als "biologisches Dosismeter"?

BfS-ISH-142/90
Burkhardt, J.; Lux, D.
Characterization of Critical Population Groups with Special Consumption Habits in Bavaria.

BfS-ISH-143/90
Roedler, H. D.; Pittelkow, E.
Strahlenexposition des Patienten bei der nuklearmedizinischen Anwendung markierter monoklonaler Antikörper.

BfS-ISH-144/90
Frasch, G. A.
Fehlbildungshäufigkeiten in Bayern 1968 - 1979 / Bericht im Rahmen des Strahlenbiologischen Umweltmonitorings Bayern.

BfS-ISH-145/90
Martignoni, K.
Spontane und Strahleninduzierte kongenitale Anomalien einschließlich Fehl- und Totgeburten.

BfS-ISH-146/90
Schaller, G.; Leising, Chr.; Krestel, R.; Wirth, E.
Cäsium- und Kalium-Aufnahme durch Pflanzen aus Böden.

BfS-ISH-147/90
Brachner, A.
Entwicklung der Säuglingssterblichkeit in Bayern (1972 - 1986).

BfS-ISH-148/90
Winkelmann, I.; Endrulat, H.-J.; Fouasnon, S.; Gesewsky, P.; Haubelt, R.; Klopfer, P.; Köhler, H.; Kohl, R.; Kucheida, D.; Leising, C.; Müller, M.-K.; Neumann, P.; Schmidt, H.; Vogl, K.; Weimer, S.; Wildermuth, H.; Winkler, S.; Wirth, E.; Wolff, S.
Radioactivity Measurements in the Federal Republic of Germany after the Chernobyl Accident.
(Unveränderter Nachdruck von ISH-116)

BfS-ISH-149/90
Hofmann, R.; Hendriks, W.; Schreiber, G. A.; Bögl, K. W.
BLood Amylase - A Biochemical Radiation Indicator?

BfS-ISH-150/91
Frasch, G.; Martignoni, K.
Verwertbarkeit und Zuverlässigkeit von Ergebnissen vorliegender epidemiologischer Untersuchungen für die Abschätzung des strahlenbedingten Krebsrisikos. III. Das strahlenbedingte Brustkrebsrisiko.

BfS-ISH-151/91
Martignoni, K. (unter Mitarbeit von *Elsasser, U.*)
Verwertbarkeit und Zuverlässigkeit von Ergebnissen vorliegender epidemiologischer Untersuchungen für die Abschätzung des strahlenbedingten Krebsrisikos. IV. Das strahlenbedingte Schilddrüsen-Krebsrisiko.

BfS-ISH-152/91
Hoeltz, J.; Hoeltz, A.; Potthoff, P. (Infratest Gesundheitsforschung, München); Brachner, A.; Grosche, B.; Hinz, G.; Kaul, A.; Martignoni, K.; Roedler, H.-D.; Schwarz, E.; Tsavachidis, C.
Schwangerschaften und Geburten nach dem Reaktorunfall in Tschernobyl.
Eine repräsentative Erhebung für die Bundesrepublik Deutschland und Berlin (West). Kurzfassung.

BfS-ISH-153/91
Brachner, A.; Grosche, B.
Risikofaktoren für bösartige Neubildungen.
Neuherberg, Juni 1991

BfS-ISH-154/91
Brachner, A.; Grosche, B.
Perinatale Risikofaktoren einschließlich Fehlbildungen.
Neuherberg, Oktober 1991

BfS-ISH-155/91
Römmelt, R.; Hiersche, L.; Wirth, E.
Untersuchungen über den Transfer von Caesium 137 und Strontium 90 in ausgewählten Belastungspfaden.
Abschlußbericht zum Forschungsvorhaben St.Sch. 1033.
Neuherberg, Dezember 1991

BfS-ISH-156/91
Poschner, J.; Schaller, G.; Wirth, E.
Verbesserung und Neuentwicklung von radioökologischen Modellen zur Berechnung der Strahlenexposition bei der Beseitigung von schwach radioaktiv kontami-

nierten Abfällen.
Abschlußbericht zum Forschungsvorhaben St.Sch. 1104.
Neuherberg, Dezember 1991

BfS-ISH-157/92
Hoeltz, J.; Hoeltz, A.; Potthoff, P.; Brachner, A.; Grosche, B.; Hinz, G.; Kaul, A.; Martignoni, K.; Roedler, H.-D. †; Schwarz, E.; Tsavachidis, C.
Schwangerschaften und Geburten nach dem Reaktorunfall in Tschernobyl.
Eine repräsentative Erhebung für die Bundesrepublik Deutschland und Berlin (West).
- Abschlußbericht -.
Neuherberg, September 1992

BfS-ISH-158/92
Lörch, Th.; Wittler, C.; Frieben, M.; Stephan, G.
Automatische Chromosomendosimetrie.
Neuherberg, Oktober 1992

BfS-ISH-159/92
Schmier, H.; König, K.; Aßmann, G.; Berg, D.
Ganzkörpermessungen an bayerischen Schulkindern.
Abschlußbericht . Juli 1992.
Neuherberg, Dezember 1992

BfS-ISH-160/93
Irl, C.; Schoetzau, A.; Steinhilber, B.; Grosche, B.; Jahraus, H.; van Santen, E.
Entwicklung der Säuglingssterblichkeit in Bayern 1972 bis 1990.
Neuherberg, März 1993

BfS-ISH-161/93
Dalheimer, A.; Henrichs, K. (Hrsg.)
Thorium, Probleme der Inkorporationsüberwachung. Anwendung, Messung, Interpretation.
Seminar in Kloster Scheyern/Bayern am 12. und 13. Oktober 1992, durchgeführt vom Institut für Strahlenhygiene des BfS.
Neuherberg, September 1993

BfS-ISH-162/93
Daten zur Umgebungs- und Umweltradioaktivität in der Bundesrepublik Deutschland in den Jahren 1990 bis 1992.
Bearbeitet vom Bundesamt für Strahlenschutz und den Leitstellen des Bundes.
Neuherberg, Oktober 1993

BfS-ISH-163/93
Steinmetz, M. (Hrsg.)
Arbeitsgespräch Terrestrisches solares UV-Monitoring am 2. Juni 1992 im Institut für Strahlenhygiene des Bundesamtes für Strahlenschutz.
Neuherberg, Oktober 1993

BfS-ISH-164/93
Poschner, J.; Schaller, G.
Richtwerte für die spezifische Aktivität von schwach radioaktiv kontaminierten Abfällen, die konventionell entsorgt werden.
Neuherberg, Dezember 1993

BfS-ISH-165/94
Schmitt-Hannig, A.; Thieme, M.
Forschungsprogramm Strahlenschutz, 1992 bis 1993.
Bericht über das vom Bundesamt für Strahlenschutz fachlich und verwaltungsmäßig begleitete Ressortforschungsprogramm Strahlenschutz des Bundesministeriums für Umwelt, Naturschutz und Reaktorsicherheit.
Neuherberg, Januar 1994

BfS-ISH-166/94
Burkart, W. (Hrsg.)
Erste deutsche Aktivitäten zur Validierung der radiologischen Lage im Südural.
Neuherberg, August 1994

BfS-ISH-167/94
Gödde, R.; Schmitt-Hannig, A.; Thieme, M.
Strahlenschutzforschung. Programmreport 1994.
Bericht über das vom Bundesamt für Strahlenschutz fachlich und verwaltungsmäßig begleitete Ressortforschungsprogramm Strahlenschutz des Bundesministeriums für Umwelt, Naturschutz und Reaktorsicherheit.
Neuherberg, Oktober 1994

BfS-ISH-168/94
Schoetzau, A.; van Santen, F.; Irl, C.; Grosche, B.
Angeborene Fehlbildungen und Säuglingssterblichkeit in Bayern nach dem Reaktorunfall in Tschernobyl.
Neuherberg, Dezember 1994

BfS-ISH-169/95
Poschner, J.; Schaller, G.
Richtwerte für die spezifische Aktivität von schwach radioaktiv kontaminierten Abfällen, die konventionell entsorgt werden.
Vollständig überarbeitete Version von BfS-ISH-164/93.
Neuherberg, Januar 1995

BfS-ISH-170/95
Angerstein, W.; Bauer, B.; Barth, I.
Daten über die Röntgendiagnostik in der ehemaligen DDR.
Neuherberg, März 1995

BfS-ISH-171/95
Schopka, H.-J.; Steinmetz, M. (Editors)
Environmental UV radiation and health effects. Proceedings of teh International Symposium, Munich-Neuherberg, Germany, May 4-6, 1993.
Neuherberg, Mai 1995

BfS-ISH-172/95
Kragh, P.
C-Programm LINOP zur Auswertung von Filmdosimetern durch lineare Optimierung.
Anwendungshandbuch.
Neuherberg, November 1995

Verantwortung für Mensch und Umwelt

Kontakt:
Bundesamt für Strahlenschutz
Postfach 10 01 49
38201 Salzgitter
Telefon: + 49 (0)3018 333-0
Telefax: + 49 (0)3018 333-1885
Internet: www.bfs.de
E-Mail: ePost@bfs.de
Gedruckt auf Recyclingpapier aus 100 % Altpapier.

Bundesamt für Strahlenschutz