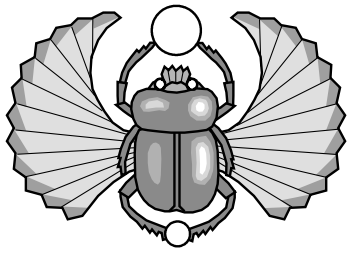
Multithread Version of P4 Including Parallelization of the VSOP87 Subroutine

Hans Jelitto
Hamburg, November 2015

Abstract

This report briefly describes and provides the Fortran source code of a parallelized version of the P4 program, based on “OpenMP,” which includes parallelization of the implemented VSOP87 subroutine. The P4 program is applied and described in detail in “Planetary Correlation of the Giza Pyramids – P4 Program Description.” The new version enables the use of multiple cores in the processor unit to increase the processing speed. This multithread program, called P4-4, as well as the single-thread program, P4, allow for the reproduction of the astronomical calculations concerning the Giza Pyramids and the planets of our solar system, published in the book *Pyramiden und Planeten* and related publications of the author. The focus of this report, the parallelization, is solely a programming issue.



Basic Aspects of the P4-4 Program

The following text provides some basic information and contains the text of the footnote on page 82 of “Planetary Correlation of the Giza Pyramids” [1] (p4-manual-06-2015.pdf). For both programs, P4 and P4-4, the Fortran 95 standard and the free-source form are used.

To obtain higher processing speed, some “hot spots” in the P4 program were parallelized with the application programming interface (API) “OpenMP.” The modified subroutine “VSOP87X” was renamed as “VSOP87Y.” For the compilation of the source code with GFortran (GCC), we use the following command: gfortran -fopenmp -O2 p4-4.f95. The subroutine has been adapted according to four threads, because the used processor has two cores with hyper-threading (Intel Core i5-3210M, 2.5 GHz, 8 GB, dual-channel, Turbo Boost not active). Therefore, the corresponding P4-4 file names have an additional “4.” Now, the calculated *combined* CPU time is longer than the runtime of P4. So, the execution time, especially of the TYMT test, is determined not only with the subroutine “CPU_time,” but also with “date_and_time.” TYMT (“Ten thousand Years Mercury Transit”) is a kind of benchmark test to measure the program performance, 64-bit version (see page 16 in [1]). Because of the parallelization, its runtime decreases from 46.0 s to approximately 22.4 s, and with a small terminal window of three lines to 20.0 s.

Although the parallelization yields a clear improvement concerning the processing speed, the modification has more of a technical meaning than a practical reason, because for normal use the single-thread program is fast enough. Moreover, with a single-core processor, the program would even decelerate. Thus, the original P4 source code is given in the appendix of Ref. [1]. The parallelized source code and code files (P4-4, listed in Table 1) are included in the P4 program package (download). Therefore, the multi-thread source code given here and in the p4-4.pdf file are identical (see Table 1). Compared with the p4-4.pdf file, the descriptive text was extended and some references and links were added. This parallelized version might be relevant for anyone interested in program optimization. Of course, the results are the same compared with those of the single-thread P4 program. (Up until now, no deviation has been found.) Both program versions, based primarily on the works of P. Bretagnon, G. Francou [2, 3], and J. Meeus [4, 5], can be used to reproduce the corresponding astronomical calculations in Refs. 1, 6–8.

Table 1: *The four files, representing the parallelized version of the P4 program (download from the author's homepage; program package: p4-program-06-2015.zip).*

File	Brief description
`p4-4.f95`	Fortran source code (parallelized version)
`p4-4.pdf`	Fortran source code in PDF-format (plus descriptive text)
`p4-4-64`	Executable program file for a 64-bit system
`p4-4-64.sh`	Shell-script that clears the screen display and starts `p4-4-64`

In the parallelized subroutine VSOP87Y, the program lines for calculating the planetary velocities are removed, because the velocities are not needed in P4, so this program part becomes more compact. Furthermore, the subroutine is checked only for the theory versions VSOP87A and VSOP87C. In any case, it should be easy to modify the subroutine accordingly if necessary.

The parallelized and the adapted program passages in the source code can be located by searching for the term “threads.” In the case of more than four threads, only the subroutine VSOP87Y has

to be modified. The application of OpenMP is relatively easy. OpenMP is currently available only for the programming languages Fortran, C, and C++. (An implementation of the OpenMP standard for Java is called “JaMP.”) Additional information about the p4-4 program itself can be found in the following Fortran source code.

Meanwhile, a VSOP version with improved accuracy (VSOP2013/TOP2013) [9] is available. However, the precision of VSOP87 is better than 1 arc second for the inner planets Mercury to Mars within the time span 2000 BC to 6000 AD [3, p. 311]. A deviation of 1 arc second, measured in radians, indicates a relative error of 0.0005 %. Therefore, the precision of VSOP87 is by far sufficient for our purpose (0.07 % [1, p. 30]), and the new VSOP version probably would not yield any significant change in the results. Nevertheless, it would be a good test of our findings if the calculations were performed independently and based, for example, on VSOP2013 or on a theory other than VSOP.

References

[1] Jelitto, H.: Planetary Correlation of the Giza Pyramids–P4 Program Description. 2nd edition, Hamburg (June 2015), DOI: 10.13140/RG.2.1.5135.2164 (full text on RG)

[2] Bretagnon, P.: Théorie du mouvement de l'ensemble des planétes–Solutions VSOP82. Astronomy and Astrophysics 114 (1982) 278–288, ADS-pdf

[3] Bretagnon, P., and Francou, G.: Planetary theories in rectangular and spherical variables–VSOP87 solutions. Astronomy and Astrophysics 202 (1988) 309–315, ADS-pdf

[4] Meeus, J.: Astronomical Algorithms. First Engl. Edition, Willmann-Bell, Inc., Richmond, Virginia (1991) W-Bell

[5] Meeus, J.: Transits. Willmann-Bell, Inc., Richmond, Virginia (1989) W-Bell

[6] Jelitto, H.: Pyramiden und Planeten – Ein vermeintlicher Meßfehler und ein neues Gesamtbild der Pyramiden von Giza. Wissenschaft & Technik Verlag, Berlin (1999), ISBN: 3-89685-507-7, more info

[7] Jelitto, H.: Gespiegelte Planeten – Die Anordnung der Pyramiden von Gizeh. Argo-Verlag, Marktoberdorf, Magazin 2000plus Spezial 6/156 (2000) 12–22 (full text on RG)

[8] Jelitto, H.: Geometrie und Anordnung der drei großen Pyramiden von Giza – Teil II: Chefren- und Mykerinos-Pyramide sowie Gesamtbild. Grenzgebiete der Wissenschaft, Resch Verlag, Innsbruck, GW 44/2 (1995) 99–120 (full text on RG)

[9] Simon, J.-L., Francou, G., Fienga, and A., Manche, H.: New analytical planetary theories VSOP2013 and TOP2013. Astronomy and Astrophysics 557, A49 (2013) A&A-pdf

Use of P4-4 Program and Copyrights

```
!-----------------------------------------------------------------------!
!                                                                       !
!                          P4-4 (Fortran 95)                            !
!                                                                       !
!              PLANETENKORRELATION DER PYRAMIDEN VON GIZA                !
!                                                                       !
!                Parallelisierte Version fuer 4 Threads                 !
!              Programmlogik and Ergebnisse identisch zu P4             !
!                                                                       !
!                                  =                                    !
!                                =   =                                  !
!                              =       =                                !
!                            =    P 4    =                   =         !
!                          =    Programm   =               =   =       !
!                        =   zur Berechnung  =           =       =      !
!                      =   der Planetenposi-   =       =           =    !
!                    =   tionen und zur Bestim-  =   =               =  !
! =                =   mung des Zeitpunktes, der   =                     !
!   =            =   durch die Pyramidenanordnung    =                  !
!     =        =   bzw. Kammeranordnung vorgegeben     =               !
!       =    =   ist. Grundlage sind Messungen namhaf-   =             !
!         =   ter Aegyptologen sowie die planetarische    =            !
!           =   Theorie VSOP87 von Bretagnon und Francou    =          !
!             =  (IMCCE, Paris). Das Programm ist eine viel-  =        !
!               =  seitige Weiterentwicklung des Programms P3.  = = =   !
!                 = = = = = = = = = = = = = = = = = = = = = = = = =     !
! = = = = = = = = = =                                                   !
!                                                                       !
!                                                                       !
!                 Hans Jelitto, Hamburg, 6. Juni 2015                   !
!                                                                       !
!                                                                       !
!                           Kurzbeschreibung                            !
!                                                                       !
!         Das Programm P4-4 berechnet fuer lange Zeitraeume die         !
!         Positionen der Planeten unseres Sonnensystems und er-         !
!         moeglicht einen praezisen Vergleich mit der Anordnung         !
!         der Giza-Pyramiden bzw. der Kammeranordnung innerhalb         !
!         der Cheops-Pyramide. Weiterhin berechnet es die Pha-          !
!         sen der Merkur- und Venustransite vor der Sonne und           !
!         bestimmt Zeitpunkte von "linearen" Planetenkonstella-         !
!         tionen (Syzygium) im Zusammenhang mit den Pyramiden.          !
!         Verschiedene Theorievarianten und eine Vielzahl von           !
!         Optionen ermoeglichen Quervergleiche. Es reproduziert         !
!         die astronomischen Berechnungen in den zwei Buechern:         !
!                                                                       !
!         1. "PYRAMIDEN UND PLANETEN - Ein vermeintlicher Mess-         !
!         fehler und ein neues Gesamtbild der Pyramiden von             !
!         Giza", Wissenschaft und Technik Verlag, Berlin (1999),        !
!         ISBN 3-89685-507-7                                            !
!                                                                       !
!         2. Buch 2 (in Vorbereitung)                                   !
!                                                                       !
!                  ---------------------------------                    !
!                  *         COPYRIGHTS UND        *                    !
!                  *    VERWENDUNG DES PROGRAMMS   *                    !
!                  ---------------------------------                    !
!                                                                       !
!         Bezogen auf das Copyright von H. Jelitto stehen das           !
```



```
!         Programm P4-4 und die uebrigen Programmteile, mit Aus-        !
!         nahme der Datei "p4-manual-06-2015.pdf" und ihren vor-        !
!         hergehenden Versionen, fuer wissenschaftliche, private,       !
!         Ausbildungs- und paedagogische Zwecke zur freien Ver-         !
!         fuegung, solange der Name des Urhebers ordnungsgemaess        !
!         genannt wird, und duerfen nicht fuer kommerzielle             !
!         Zwecke irgendeiner Art verwendet werden. Kommerzielle         !
!         Nutzung bedarf der schriftlichen Genehmigung. Fuer die        !
!         anderen Programmteile (A. bis C.), die im Folgenden           !
!         aufgezaehlt sind, ist zu pruefen, ob eine Genehmigung         !
!         der Urheber bzw. Copyright-Inhaber erforderlich ist.          !
!                                                                       !
!         (Informationen zur Nutzung und zum Copyright der Datei        !
!         "p4-manual-06-2015.pdf" stehen zu Anfang jener Datei.)        !
!                                                                       !
!         Das Programm P4-4 wird in der Hoffnung zur Verfuegung         !
!         gestellt, dass es fuer andere nuetzlich ist, jedoch           !
!         ohne irgendeine Art von Garantie oder Gewaehrleistung.        !
!                                                                       !
!         Die folgenden Angaben (A. bis D.) beziehen sich ent-          !
!         sprechend auf das Programm P4-4, die vorherige Version        !
!         P3 und alle zugehoerigen, unten aufgefuehrten Dateien.        !
!                                                                       !
!      A. Unterprogramm VSOP87Y (basierend auf der Theorie              !
!         "Variations Seculaires des Orbites Planetaires") und          !
!         zugehoerige Datenfiles: P. Bretagnon und G. Francou,          !
!         Institut de mecanique celeste et de calcul des                !
!         ephemerides (IMCCE), 77 Avenue Denfert-Rochereau,             !
!         F-75014 Paris, France.                                        !
!                                                                       !
!      B. Programmpaket FITEX (bestehend aus 4 Unterprogrammen          !
!         im hinteren Programmteil): KIT, Karlsruhe Institute of        !
!         Technology (zuvor: FZK, Forschungszentrum Karlsruhe in        !
!         der Helmholtz-Gemeinschaft), Institut fuer Kernphysik,        !
!         Postfach 3640, D-76021 Karlsruhe. FITEX wurde von             !
!         G.W. Schweimer um 1972 entwickelt und erstmals ver-           !
!         oeffentlicht in: H.J. Gils: "The Karlsruhe Code MODINA        !
!         for Model Independent Analysis of Elastic Scattering          !
!         of Spinless Particles." KfK 3063, Nov. 1980, Kernfor-         !
!         schungszentrum Karlsruhe (KfK), Zyklotron Laboratorium,       !
!         and KfK 3063, 1. Supplement, Dec. 1983.                       !
!                                                                       !
!      C. Umrechnung von "terrestrial time" (TT) in "universal          !
!         time" (UT) mittels  delta-T = TT - UT: Fred Espenak,          !
!         und Jean Meeus, NASA Eclipse Web Site, Polynomial             !
!         expressions for DELTA-T.                                      !
!                                                                       !
!      D. Das Hauptprogramm P4-4 und die uebrigen Programmteile,        !
!         einschliesslich der Modifikation des Unterprogramms           !
!         VSOP87 (--> "VSOP87Y"): (c) 2014, 2015 Hans Jelitto,          !
!         Ewaldsweg 12, D-20537 Hamburg, Germany.                       !
!                                                                       !
!                        --------- Danksagung --------                 !
!                                                                       !
!         Das Unterprogramm jdedate zur Umrechnung von JDE in           !
!         ein Kalenderdatum basiert auf einem Algorithmus aus dem       !
!         Buch von Jean Meeus: "Astronomical Algorithms", 1991,         !
!         Willmann-Bell, Inc., P.O.Box 35025, Richmond, Virginia        !
!         23235, USA, S.63. Dafuer und fuer die Auflistung der          !
```

```
!         gekuerzten Reihen der VSOP87D-Parameter gilt mein herz-     !
!         licher Dank! Ebenfalls war das Buch "Transits" von          !
!         J. Meeus (derselbe Verlag) als Basis und zum Testen         !
!         der Transitberechnungen aeusserst hilfreich.                !
!                                                                     !
!                    ----------------------------                     !
!                                                                     !
!         Zum Programm P4-4 gehoeren nachfolgende 30 Dateien:         !
!                                                                     !
!         Datei                    Kurzbeschreibung                   !
!      ---------------------------------------------------------      !
!         p4-4.f95  . . . FORTRAN-95-Quellcode (dieser Text)          !
!         p4-4-64 . . . . Ausfuehrbare Datei fuer 64-bit-System       !
!         p4-4-64.sh  . . loescht Bildschirm und startet p4-4-64      !
!         p4-manual-06-2015.pdf: Bedienungsanleitung zu P4, P4-4      !
!                          und Uebersicht der Planetenkorrelation     !
!         README  . . . . Kurzinformation zur Theorie VSOP87          !
!         vsop87.doc  . . Ausfuehrlichere Information zur Theo-       !
!                          rie "Planetary Solutions VSOP87"           !
!         out.txt . . . . Ergebnis-Datei. Wenn diese nicht be-        !
!                          reits existiert, wird sie bei entspre-     !
!                          chender Option vom Programm erstellt.      !
!                                                                     !
!         inedit.t  . . . Datei zum Editieren der Eingabeparame-      !
!                          ter --> Parametersatz fuer "inparm.t"      !
!         inparm.t  . . . Input gemaess Schnellstart-Optionen         !
!         inpdata.t . . . Parameter f. FITEX, Kammer-Koordinaten      !
!                          in der Cheops-P. und Pyramiden-Koord.      !
!         inserie.t . . . Transitserien fuer Merkur und Venus         !
!         invsop1.t . . . VSOP87D, gekuerzt, Meeus: Astr. Alg.        !
!         invsop3.t . . . Polynomdarstellung der Bahnelemente,        !
!                          berechn. aus VSOP82, Meeus: Astr. Alg.     !
!                                                                     !
!                          VSOP87A, kart. Koord. (Ekl. J2000.0)       !
!         VSOP87A.mer . . Merkur          (Diese und die folgen-      !
!         VSOP87A.ven . . Venus           den Dateien enthalten       !
!         VSOP87A.ear . . Erde            die Parameter zur           !
!         VSOP87A.mar . . Mars            VSOP87-Theorie voll-        !
!         VSOP87A.jup . . Jupiter         staendig.)                  !
!         VSOP87A.sat . . Saturn                                      !
!         VSOP87A.ura . . Uranus                                      !
!         VSOP87A.nep . . Neptun                                      !
!         VSOP87A.emb . . Erde-Mond-Schwerpunktsystem                 !
!                                                                     !
!                          VSOP87C, kart. Koord. (Ekl. d. Epoche)     !
!         VSOP87C.mer . . Merkur                                      !
!         VSOP87C.ven . . Venus                                       !
!         VSOP87C.ear . . Erde                                        !
!         VSOP87C.mar . . Mars                                        !
!         VSOP87C.jup . . Jupiter                                     !
!         VSOP87C.sat . . Saturn                                      !
!         VSOP87C.ura . . Uranus                                      !
!         VSOP87C.nep . . Neptun                                      !
!      ---------------------------------------------------------      !
!                                                                     !
!         Die VSOP87-Dateien wurden 2007 erneut aus dem Internet      !
!         heruntergeladen. Sie sind vom April 2005. Gross- und        !
!         Kleinschreibung sind zu beachten.                           !
!                                                                     !
```



```
!      ---------------------------------------------------------      !
!                        DIE VERSCHIEDENEN OPTIONEN                   !
!      ---------------------------------------------------------      !
!                                                                     !
!                  -->  Neue Optionen und Ergaenzungen:               !
!                                                                     !
!              Gegenueber der Programmversion P3, die fuer das        !
!              Buch "Pyramiden und Planeten" verwendet wurde,         !
!              kommen hier die folgenden Ergaenzungen hinzu:          !
!                                                                     !
!        >     a)  Zu typischen Parameterkombinationen gibt es        !
!        >         eine Reihe von Schnellstart-Optionen (1..15)      !
!        >         und zusaetzlich eine Info-Option (111).           !
!        >     b)  Verborgene Optionen: Ebenfalls Schnellstart-      !
!        >         optionen - aber nicht im Eingabe-Menue ange-      !
!        >         zeigt - existieren fuer die Resultate in den      !
!        >         Tabellen 39 bis 51 des Buches "Pyramiden und      !
!        >         Planeten" und fuer die Tabellen 17 bis 36 des     !
!        >         Buches 2, das sich in Vorbereitung befindet.      !
!        >         Die Tabelle 39 zum Beispiel besitzt drei Ab-      !
!        >         schnitte, die sich mit den Zahlen 390, 391        !
!        >         und 392 aufrufen lassen, zusammengesetzt aus      !
!        >         39 und 0 bis 2. Das heisst, alle verborgenen      !
!        >         Optionen bestehen aus drei Ziffern!               !
!        >     c)  Spezialoption -803: Diese erzeugt die Liste       !
!        >         der JDE-Nummern und Transit-Serien in einer       !
!        >         neuen Datei "inser-2.t". Wenn gewuenscht kann     !
!        >         diese Datei dann "inserie.t" ersetzen (im         !
!        >         Allgemeinen nicht erforderlich).                  !
!        >     d)  Optional: Programmstart mit einer Input-Datei     !
!        >         "inedit.t", in der die Parameter manuell edi-     !
!        >         tiert werden koennen (Option 999).                !
!        >     e)  Koordinaten der drei Kammern der Cheops-Pyra-     !
!        >         mide zum Positionsvergleich mit den Planeten.     !
!        >     f)  Positionsvorgabe durch die Kammermittelpunkte,    !
!        >         bzw. die Mittelpunkte der Ost- und Westwaende     !
!        >         der Kammern.                                      !
!        >     g)  Sechs verschiedene moegliche Zuordnungen der      !
!        >         Planeten Erde, Venus und Merkur zu den drei       !
!        >         Kammern in der Cheops-Pyramide.                   !
!        >     h)  Perihelzeiten beim Merkur, Zeitpunkte nahe        !
!        >         der Periheldurchgaenge und freier Zeitpunkt.      !
!        >     i)  Automatische Erkennung und Markierung der         !
!        >         Planetenkonstellationen 1 bis 14 bei Verwen-      !
!        >         dung beliebiger Optionen.                         !
!        >     j)  Uebertragung der Positionen von Merkur bis        !
!        >         Neptun ins Pyramidengelaende auf Basis der        !
!        >         Pyramiden- bzw. Kammeranordnung (bei 3D-Be-       !
!        >         rechnung mit FITEX, Einzelberechnung, Konst.      !
!        >         1 bis 14). Geographische Koord. (GPS) nur bei     !
!        >         Konst. 12, alle Etappen, fuer Merkur bis Mars.    !
!        >     k)  Kombination VSOP87-Kurzversion und -Voll-         !
!        >         version: Konstellationen, die mit der Kurz-       !
!        >         version gefunden wurden, werden automatisch       !
!        >         mit der Vollversion nachberechnet. Darueber       !
!        >         hinaus: "Zeitintervall um Aphel bzw. um Peri-     !
!        >         hel" auch fuer die Vollversion VSOP87 (sinn-      !
!        >         voll wegen schnellerer Mikroprozessoren und       !
!        >         der Programmoptimierung).                         !
```

```
!       >    l)  Ausser den beiden Optionen "Blick aus Richtung       !
!       >        ekl. Nordpol" und "ekl. Suedpol" sind jetzt          !
!       >        beide Optionen kombiniert moeglich.                  !
240 !   >    m)  Zeitraeume werden nicht mehr mit der k-Nummer        !
!       >        des Aphel- bzw. Periheldurchgangs des Merkurs        !
!       >        angegeben, sondern mit der eher gebraeuchli-         !
!       >        chen Jahreszahl.                                     !
!       >    n)  Die Berechnungen mit VSOP87 wurde auf den Zeit-      !
245 !   >        raum 13000 v.Chr. bis 17000 n.Chr. begrenzt.         !
!       >        Ausnahme: "Orbital Elements" und Loesung der         !
!       >        Keplerschen Gl., 30000 v.Chr. bis 30000 n.Chr.       !
!       >    o)  Syzygium: Merkur bis Erde bzw. Merkur bis Mars       !
!       >        in Konjunktion, d.h. 4 bzw. 5 Himmelskoerper         !
250 !   >        des Sonnensystems in einer Reihe: Sonne, Mer-        !
!       >        kur, Venus, Erde und optional auch Mars.             !
!       >    p)  Zusaetzlich werden Merkur- und Venustransite         !
!       >        vor der Sonnenscheibe registriert (VSOP87C).         !
!       >    q)  Zum Testen der Transit-Berechnung kann man           !
255 !   >        sich lueckenlos alle Transite von Merkur und         !
!       >        Venus anzeigen lassen, was einen Vergleich           !
!       >        mit Tabellen aus der Literatur bzw. aus dem          !
!       >        Internet ermoeglicht. In diesem Fall werden          !
!       >        Datum und Uhrzeit der Konjunktion, aufsteigen-       !
260 !   >        der bzw. absteigender Knoten und die Nummer          !
!       >        der jeweiligen Transitserie angegeben.               !
!       >    r)  Als Zeitpunkt fuer den Planetentransit gibt          !
!       >        es erstens das Kriterium "gleiche ekliptikale        !
!       >        Laengen", zweitens "minimale Separation zwi-         !
265 !   >        schen Sonne und Planet" (ohne Beruecksichti-         !
!       >        gung der Lichtlaufzeit) und drittens "Beginn,        !
!       >        Mitte und Ende des Transits", d.h. die genau-        !
!       >        en Kontaktzeitpunkte bzw. Phasen.                    !
!       >    s)  Bei der Phasenbestimmung gibt es die Option,         !
270 !   >        zusaetzlich die Positionswinkel des Planeten         !
!       >        waehrend der Phasen in Bezug auf die scheinba-       !
!       >        re Bewegungsrichtung der Sonne zu berechnen.         !
!       >        Hierbei ist eine Zeilenlaenge auf dem Monitor        !
!       >        von mindestens 148 Zeichen erforderlich.             !
275 !   >    t)  Fuer die Transitphasen gibt es die zwei Zeit-        !
!       >        systeme "terrestrial (dynamical) time" (TT)          !
!       >        und "universal time" (UT). Die Umrechnung mit        !
!       >        delta-T = TT - UT wird ueber analytische Glei-       !
!       >        chungen erreicht (F. Espenak und J. Meeus,           !
280 !   >        siehe NASA Eclipse Web Site).                        !
!       >    u)  Fuer die Angabe der Transitphasen von Merkur         !
!       >        und Venus wurde eine Datumsberechnung von            !
!       >        J. Meeus integriert. Hierbei gibt es die auto-       !
!       >        matische Kalenderwahl (julianischer bzw. gre-        !
285 !   >        gorianischer Kalender) oder es wird der grego-       !
!       >        rianische Kalender fuer alle Zeiten verwendet.       !
!       >        Die Datumsberechnung wurde derart modifiziert,       !
!       >        dass sie jetzt auch fuer negative JDE gilt.          !
!       >    v)  Die Berechnung der dezimalen Jahreszahl wurde        !
290 !   >        insofern verbessert, dass sie jetzt durch 2          !
!       >        lineare Funktionen dargestellt wird, die je-         !
!       >        weils fuer den Zeitraum des julianischen und         !
!       >        des gregorianischen Kalenders stehen (abhaen-        !
!       >        gig von der Kalenderwahl).                           !
295 !   >    w)  Die Option fuer die Programm-Ausgabe "Drucken"       !
```



```
!       >        im Programm "P3" wurde durch "in Datei" er-          !
!       >        setzt. Hierbei werden die Ergebnisse gleich-         !
!       >        zeitig auf den Bildschirm und in die Datei           !
!       >        "out.txt" geschrieben. Um die Resultate dauer-       !
300 !   >        haft zu speichern, muss die Datei "out.txt"          !
!       >        nach dem Programmlauf umbenannt werden. Sonst        !
!       >        kann sie beim naechsten Programmlauf ungewollt       !
!       >        ueberschrieben werden.                               !
!       >    x)  Ebenfalls wurde zur Anzeige der Ergebnisse           !
305 !   >        ein neues Format ergaenzt (special), das fuer        !
!       >        eine Konstellation (z.B. 12) einige spezielle        !
!       >        Parameter ausgibt. Damit lassen sich die we-         !
!       >        sentlichen Tabellen aus dem Buch 2, z.B. mit         !
!       >        den verborgenen Optionen (siehe oben Punkt b),       !
310 !   >        relativ einfach reproduzieren.                       !
!       >    y)  Optimierung der Rechengeschwindigkeit, unter         !
!       >        anderem durch Modifikation des Daten-Aufrufs         !
!       >        und Parallelisierung (neuer Name: VSOP87Y).          !
!       >    z)  Verbesserung der Programm-Ausgabe, z.B. durch        !
315 !   >        ausfuehrlichere Kopfzeilen, jetzt in Englisch.       !
!       >        Am Ende des Programmlaufs wird die benoetigte        !
!       >        Rechenzeit angegeben (CPU-time).                     !
!   ------------------------------------------------------------     !
!                                                                    !
320 !                       Optionen insgesamt:                      !
!                                                                    !
!   ---------- Schnellstart-Optionen: -------------------------     !
!          1-15    -->  Die wesentlichen astr. Berechnungen          !
!          111     -->  Information zu Autoren u. Copyrights         !
325 !          390-512 -->  Tabellen 39-51 aus "Pyram. u. Plan."     !
!          170-381 -->  Tabellen 17-33 und 35-38 aus Buch 2          !
!                       (Das Buch ist in Vorbereitung.)              !
!          999     -->  Input aus "inedit.t" (editierbar)            !
!         -803     -->  Erzeugung der Datei "inser-2.t"              !
330 !          (0)     -->  Parameter einzeln eingeben               !
!                                                                    !
!   ---------- Planetenpositionen: ----------------------------     !
!          1.  Anordnung der 3 Pyramiden                             !
!          2.  Anordnung der 3 Kammern der Cheops-Pyramide           !
335 !          3.  Konjunktionen (Transit, Syzygium)                 !
!                                                                    !
!   ---------- VSOP87-Version: --------------------------------     !
!          1.  Kombination von Kurz- u. Vollversion VSOP87           !
!          2.  VSOP87 Kurzversion (Buch von J. Meeus)                !
340 !          3.  Keplersche Gleichung mit VSOP82 (Meeus)           !
!          4.  VSOP87 Vollversion (IMCCE, Internet)                  !
!                                                                    !
!   ---------- Koordinatensystem in VSOP87: -------------------     !
!          1.  Ekliptik der Epoche (VSOP87C)                         !
345 !          2.  J2000.0 (VSOP87A, Vollv. und Kepl. Gl.)           !
!                                                                    !
!   ---------- Umfang der Progranmm-Ausgabe: ------------------     !
!          1.  normal (eine Zeile pro Konstellation)                 !
!          2.  detailliert (mehrere Zeilen pro Konstell.)            !
350 !                                                                !
!   ---------- Zuordnung: Planeten <-> Kammern: ---------------     !
!          1.-6.  Sechs moegl. Zuordnungen von Erde, Venus           !
!              und Merkur zu Koenigs-, Koeniginnen- und              !
!              Felsenkammer: 1. E-V-M (Standard), 2. E-M-V,          !
```

```
355  !               3. V-E-M,  4. V-M-E,  5. M-E-V,  6. M-V-E.          !
     !                                                                   !
     !     ---------- Zeitpunkte: ------------------------------------   !
     !           1.  Apheldurchgang des Merkurs                          !
     !           2.  Periheldurchgang des Merkurs                        !
360  !           3.  Aequidistante Abfolge von Zeitpunkten in            !
     !               Zeitintervallen, die jeweils den Aphel-             !
     !               durchgang des Merkurs enthalten                     !
     !           4.  Aequidistante Abfolge von Zeitpunkten ana-          !
     !               log um den Periheldurchgang des Merkurs             !
365  !           5.  Zeitpunkt voellig frei und Minimierung der          !
     !               Abweichung zwischen Pyramiden und Planeten-         !
     !               anordnung durch Variation des Zeitpunkts            !
     !                                                                   !
     !     ---------- "Sonnenposition": ------------------------------   !
370  !           1.  genau suedlich Mykerinos-Pyramide (1D)              !
     !           2.  genau suedlich Chefren-Pyramide (1D)                !
     !           3.  unbestimmt (2D und 3D)                              !
     !                                                                   !
     !     ---------- Berechnung ("Sonnenposition" unbestimmt): ------   !
375  !           1.  2-dimensional, Projektion auf Hauptebene            !
     !           2.  3-dimensional, durch lineares Gleichungs-           !
     !               system und Uebertragung der Loesung                 !
     !           3.  3-dimensional, Koordinatentransformation            !
     !               mit Fit-Programm FITEX                              !
380  !                                                                   !
     !     ---------- Referenzsystem bei 2D-Berechnung: --------------   !
     !           1.  Ekliptikales System                                 !
     !           2.  Merkurbahn-System, Transformtion A, B oder          !
     !               C (Gerade "Sonne - Merkur-Aphel" = x-Achse,         !
385  !               Merkurbahn def. xy-Ebene, Ekl. d. Epoche)           !
     !           3.  Venusbahn-System, Transformation A, (Pro-           !
     !               jektion "Aphel - Merkur" genau auf x-Achse,         !
     !               Venusbahn def. xy-Ebene, Ekl. der Epoche)           !
     !                                                                   !
390  !     ---------- "Polaritaet" bei Projektion (2D): --------------   !
     !           1.  Blick vom ekliptikalen Nordpol                      !
     !           2.  Blick vom ekliptikalen Suedpol                      !
     !           3.  Beide Optionen 1. oder 2.                           !
     !                                                                   !
395  !     ---------- Vorgegebene Hoehenlagen (3D): ------------------   !
     !           1.  Grundflaechen der Pyramiden                         !
     !           2.  Schwerpunkte   "      "                             !
     !           3.  Spitzen        "      "                             !
     !                                                                   !
400  !     ---------- Kammerpos. in Cheops-P. (3D, z-Koord.): --------   !
     !           1.  Ostwaende der Kammern                               !
     !           2.  Mitte      "     "                                  !
     !           3.  Westwaende "     "                                  !
     !                                                                   !
405  !     ---------- Zeitpunkt-Eingabe: ----------------------------   !
     !           1.  Angabe der Konstellation (Nr. 1 bis 14)             !
     !           2.  Jahr bzw. Jahresintervall (von ... bis ...)         !
     !           3.  Aphel- bzw. Periheldurchgang (k-Nummer)             !
     !           4.  Julian Ephemeris Day (JDE)                          !
410  !                                                                   !
     !     ---------- Planeten in Konjunktion: -----------------------   !
     !           1.  Alle Merkur-Transite in einem Zeitintervall         !
     !           2.  Alle Venus-Transite   "   "         "             !
```



```
     !           3.  Merkur bis Erde in einer Reihe (Syzygium)           !
415  !           4.  Merkur bis Mars  "   "     "   (    "   )          !
     !           5.  Syzygium (Pkt. 3./4.) mit simultanem Transit        !
     !                                                                   !
     !     ---------- Transit-Bestimmung: --------------------------   !
     !           1.  Transite: gleiche eklipt. Laenge Planet/Erde        !
420  !           2.  Transite: minimale Separation Planet/Sonne,         !
     !               1./2.: ohne Beruecksicht. der Lichtlaufzeit         !
     !           3.  Phasen und minimale Separation von der Erde         !
     !               aus gesehen, Lichtlaufzeit beruecksichtigt          !
     !           4.  Phasen wie in 3. und Positionswinkel                !
425  !                                                                   !
     !     ---------- Kalendersystem: --------------------------------   !
     !           1.  Automatische Wahl des Kalenders                     !
     !               (Greg. < 4712 BC < Julian. < 1582 AD < Greg.)       !
     !           2.  Gregorianischer Kalender fuer alle Zeiten           !
430  !                                                                   !
     !     ---------- Zeitsysteme: -----------------------------------   !
     !           1.  "terrestrial dynamical time" (TT) bzw. JDE          !
     !           2.  "universal time" (UT), basierend auf delta-T        !
     !               (NASA Eclipse Web Site).                            !
435  !                                                                   !
     !     ---------- Ausgabegeraet: ---------------------------------   !
     !           1.  Monitor                                             !
     !           2.  Monitor + Datei auf Festplatte ("out.txt")          !
     !           3.  Spezial-Programmausgabe (auf Mon. + Datei)          !
440  !           4.  Programm-Abbruch                                    !
     !                                                                   !
     !     ---------------------------------------------------------     !
     !                                                                   !
     !                          Anmerkungen:                             !
445  !                                                                   !
     !   Die letztere Aufzaehlung (Optionen insgesamt) wurde der         !
     !   Uebersichtlichkeit halber etwas vereinfacht. Sie entspricht     !
     !   nicht immer dem Eingabe-Menue, das beim Programmstart mit       !
     !   "detailed options (0)" abgefragt wird. Ausserdem sind nicht     !
450  !   alle Kombinationen der Optionen durchfuehrbar. Solche, die      !
     !   nicht erlaubt sind, werden beim Programmstart gar nicht zur     !
     !   Auswahl gestellt. Das Programm ist gegen inkorrekte Eingabe     !
     !   weitestgehend abgesichert. Eine Kontrolle entfaellt nur, wenn   !
     !   die Input-Parameter in der Datei "inedit.t" manuell editiert    !
455  !   werden und der Programmstart mit der Option 999 erfolgt.        !
     !                                                                   !
     !   Anstelle des FORTRANN-77-Compilers (IBM Professional For-       !
     !   tran Compiler, Version 1.0, Ryan McFarland) wird jetzt un-      !
     !   ter Ubuntu Linux der GNU-Compiler "gfortran" verwendet,         !
460  !   der den vollen Sprachumfang von Fortran 95 sowie Teile von      !
     !   Fortran 2003 und Fortran 2008 enthaelt. Das feste Zeilen-       !
     !   format wurde (im Prinzip) durch das freie Format ersetzt.       !
     !                                                                   !
     !   Zum Programmpaket FITEX:                                        !
465  !   Alle Real-Konstanten wurden mit Exponent "D" versehen, eben-    !
     !   falls Funktionen wie DSQRT usw. eingefuehrt, sowie REAL(8)      !
     !   und INTEGER(4). EPS wurde von 1.D-5 auf 1.D-8 gesetzt. (An-     !
     !   passung an Fortran-95-Standard.)                                !
     !                                                                   !
470  !   Zum Unterprogramm VSOP87 bzw. VSOP87Y:                          !
     !   Die VSOP87-Routine wurde dahingehend modifiziert, dass die      !
     !   umfangreichen Dateien der VSOP87-Theorie nur einmal gelesen     !
```

```
      !     und im Rechenspeicher in ein Array geschrieben werden. Da-      !
      !     rueber hinaus wurde das Unterprogramm auf der Basis des API     !
475   !     "OpenMP" parallelisiert, so dass 4 Threads gleichzeitig be-     !
      !     arbeitet werden koennen. (Fortran-95-Standard)                  !
      !                                                                     !
      !     Bei den Konstellationen 13, 14, sowie den "quick start          !
      !     options" 371 und 372 wird automatisch auch die jeweilige        !
480   !     Merkur-Aphelposition berechnet, da sich hierbei der Merkur      !
      !     nicht im Aphel seiner Bahn befindet. Dies geschieht jedoch      !
      !     nur bei Verwendung bestimmter Optionen, wie z.B. 3D/FITEX.      !
      !                                                                     !
      !     Dieses Quellprogramm enthaelt auch Code-Abschnitte, die de-     !
485   !     aktiviert wurden (durch "!c", "!f", "!h" bzw. "!t") und fuer    !
      !     spezielle Zwecke gedacht sind. Das Aktivieren einiger Zeilen    !
      !     durch Entfernen von z.B. "!h" am jeweiligen Zeilenanfang be-    !
      !     wirkt das Einsortieren der Genauigkeiten Fpos in ein Array      !
      !     (--> Histogramm: Fpos(0...5%) in Schritten von 0.05%).          !
490   !                                                                     !
      !     Groessere Stellenanzahl in der Ergebnisausgabe (siehe "!f"):    !
      !     Fuer einige Programmlaeufe koennen mehr Dezimalstellen ange-    !
      !     zeigt werden. Dafuer sind entspechende Format-Statements zu     !
      !     ersetzen. Schnellstart-Optionen 4, 9: s. Ende des Hauptpro-     !
495   !     gramms; 3, 8: s. Ende des Unterprogramms "plako" (durch Akti-   !
      !     vieren bzw. Deaktivieren entsprechender Formatzeilen).          !
      !                                                                     !
      !     Um bei Verwendung der Compiler-Option "-Wuninitialized" bzw.    !
      !     "-Wall" Warnmeldungen zu vermeiden, wurden einige Variablen     !
500   !     zusaetzlich vorab initialisiert und mit "pre-init." markiert.   !
      !                                                                     !
      !---------------------------------------------------------------------!

      !-----Module-----------------------------------------------------------
505         module base !    GRUNDLEGENDE VARIABLEN UND KONSTANTEN
            save        ! (Laengen in Metern, Zeiten in julian. Tagen)

            integer(4) :: lmax(15),jp(12,6),il(3)
            real(8)    :: xyr(37),re(78),pyr(40)
510         real(8)    :: ax,ay,az,bx,by,bz,cx,cy,cz,ao,ai,at

            real(8), parameter :: pi   = 3.1415926535897932d0, &
              pidg = pi/180.d0,   zjd0 = 2451545.d0, &
              gdpi = 180.d0/pi,   c    = 299792458.d0, &
515           tcen = 36525.d0,    AE   = 149597870610.d0, &
              tmil = 365250.d0,   z0   = 0.d0, &

      !       ("Allen's Astrophys. Q.", R-Sonne: 695508 km bzw. 958,966",
      !       Sonnenradius in "Transits", Meeus: 695990 km bzw. 959,63")
520           R0   = 695508000.d0, & ! R-Sonne (Brown/Christensen-Alsgaard)
              R3a  = 6378136.6d0,  R3p = 6356751.9d0, & ! R-Erde, IERS 2003
              pmer = 2451590.257d0, & ! Erste Merkur-Perihelzeit nach J2000
              ymer = 87.96934963d0  ! Merkur-Umlaufzeit: Perihel -> Perihel

525         real(8), dimension(2), parameter :: &
      !       Radien: Merkur 3,3629", Venus 8,41", Venusradius mit knapp
      !       50 km Atmosphaere (ohne Atm. 6051000 m)
              Ra   = (/ 2439000.d0, 6099500.d0 /), & ! Radien (Mer., Ven.)
              tsid = (/  87.9693d0, 224.7008d0 /), & ! T-siderisch ( ", ")
530           tsyn = (/ 115.8775d0, 583.9214d0 /), & ! T-synodisch ( ", ")
```



```
      !       Theoretischer Massstabsfaktor (Planetenpositionen : Pyramiden-
              zthe = (/ 9.7073d7, 2.3614d9 /) !     bzw. Kammerpositionen)

535         real(8), dimension(14), parameter :: &
      !       Nummern des Merkur-Apheldurchgangs der Konstellationen 1-14
              akon = (/-38912.d0,  -23134.d0,  -7356.d0,   8422.d0, &
                        24200.d0,  -24130.d0,  -8352.d0,   7426.d0, &
                        23204.d0,   38982.d0,  -4781.d0,   4519.d0, &
540                     39313.9134336d0,  -20240.1362451d0 /)
      !c                39313.91342804d0, -20240.136249887d0 /)
      !                 (alte Werte, Konst. 13, 14, manuell und
      !                 iterativ mit P3 bestimmt)
            end module
545
            module astro
            save
      !     Parameter der VSOP87-Kurzversion nach Meeus
            real(8)    :: par1(3,69,6,12)
550   !     Parameter der VSOP87-Vollversion
            real(8)    :: par2(3,2052,0:5,3,9)
            integer(4) :: it2(0:5,3,9),in2(0:5,3,9),iv2(9)
      !     zur Berechnung mittels Keplerscher Gleichung
            real(8)    :: par3(4,6,8,2)
555   !     zur Bestimmung der Transit-Serie
            real(8), parameter :: t13BC = -3027093.d0, t17AD = 7930183.d0
            real(8), dimension(2), parameter :: cc=(/16802.20d0,88756.13d0/)
            integer(4), dimension(4), parameter :: jj = (/-150,154,-6,19/)
            integer(4), dimension(2), parameter :: ji = (/15,7/)
560         real(8)    :: ser(-180:170,2),ase(-180:170),zstart
            integer(4) :: ise(-180:170),isflag,ismax
            end module

            program P4_4
565   !-----Hauptprogramm----------------------------------------------------

      !     -----Deklarationen und Initialisierungen
            use base; use astro
            implicit double precision (a-h,o-z)
570         dimension :: res(12),rp(3,4),md(0:9),pan(5),sd(2),zjda(4)
            dimension :: df(6),diff(9),r(6),rku(3),rk(12)
            dimension :: x(7),e(7),iw(100),f(9),y(9),z(9),w(1000)
            dimension :: x0(7),iw0(4),w0(3),zmem(78),inum(0:4)
            dimension :: ida(7),da(7),id5(5,7),da5(5,7)
575         dimension :: xx(5),yy(5),test(10),ort(0:9,4),rcm(3),acm(3)
            dimension :: iw1(8),iw2(8) ! (threads)
      !h    dimension :: ihis(100)  !h
            character(1) :: t1(3),tra(2),tr,dp,ts,sl
            character(2) :: dd,dn,ds,dss,kon
580         character(3) :: dk,pla(0:9)
            character(5) :: dmo,dmo5(5)
            character(7) :: emp
            character(8) :: str,str2,str3
            character(10) :: plan(0:9),zdate,ztime,zzone ! (threads)
585         character(20) :: dummy
            character(23) :: text(0:9),tt(2)
            character(49) :: titab
            data diff/0.d0,12.19d0,21.41d0,0.d0,-34.784d0,145.d0,60.4d0, &
                      168.d0,21.41d0/,pla/'Sun','Mer','Ven','Ear','Mar', &
590                 'Jup','Sat','Ura','Nep','E-M'/
```

```
      data titab/'body            x[m]      y[m]      z[m]    dr[m]'/
      data tt/   '  (pyramid positions)  ','  (chamber positions)  '/
      data text/'                       ','  of the "planets"     ', &
             7*'                        ','       barycenter   -->'/
595   data plan/'Sun       ','Mercury   ','Venus     ','Earth     ', &
                'Mars      ','Jupiter   ','Saturn    ','Uranus    ', &
                'Neptune   ','Earth-Moon'/
      data str/'    --- '/,str2/'   --   '/,str3/'    --  '/
      data emp/'   --- '/,dn/'  '/,ds/' *'/,dss/' <'/,dp/':'/
600   data zjde0/0.d0/,ifitrun/0/,zjdelim/0.d0/,izmin/0/ ! pre-init.
      data i0/0/,ic/0/,ierr/0/ ! pre-init.

!-----Input-Daten und Programmstart
      call inputdata(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
605    itran,isep,iuniv,ical,ika,iaph,iamax,step,ison,ihi,irb,ijd, &
       zmin,zmax,ak,zjde1,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop0,iout)
      if (iout==4) then; write(6,*); go to 500; endif
      call cpu_time(zia)
      call date_and_time(zdate,ztime,zzone,iw1) ! (--> threads)
610   write(6,'(/''  <P4-4> Computation started ...'')')

! . . Die Input-Parameter werden in die Datei "inedit.t" geschrieben.
!     Man kann sie dann gegebenenfalls manuell an geeigneter Stelle in
!     "inparm.t" (Liste der Schnellstart-Optionen) einfuegen, wobei
615 !     allerdings im Unterprogramm "inputdata" die Schnellstart-
!     Optionen angepasst werden muessen.
      if (iop0/=999 .and.iout/=1) then
        call inputfile(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
         itran,isep,iuniv,ical,ika,iaph,iamax,step,ison,ihi,irb,ijd, &
620      zmin,zmax,ak,zjde1,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop,2,iout)
      endif

! . . Parameter fuer Spezial-Output (Konst. 12) --> is12 = 1
      is12 = 0
625   if (((ipla==1 .and.iaph==1).or.(ipla==2 .and. &
          iaph==2 .and.ika==1)).and.imod<=2 .and. &
          ikomb==0 .and.iuniv==1 .and.io==2 .and. &
          ison==5 .and.ijd==12 .and.iout==3) is12 = 1

630 ! . . Erstellung weiterer Parameter
      if (iout==1) then
        ix = 6
      else
        ix = 1; open(unit=ix,file='out.txt')
635     write(6,'(9x,''Output file: "out.txt"'')')
      endif
   10 write(6,*); kmin = 0; kmax = 0
      if (ipla/=3) then
        if (ijd>=1 .and.ijd<=14) then
640       ak = akon(ijd); if (ipla==2 .and.iek==1) ak = ak - 1.d0
          call ephim(0,iaph,ipla,ical,ak,iak,zjde1,zjahr,delt)
        endif
        if (ijd==15 .and.imod==2 .and.iaph<=2) &
          call ephim(0,iaph,ipla,ical,ak,iak,zjde1,zjahr,delt)
645   endif
      if (ipla==3 .or.(ipla/=3 .and.ijd==15 .and. &
        (imod/=2 .or.(imod==2 .and.(iaph==3 .or.iaph==4))))) then
        call ephim(2,iaph,ipla,ical,ak,kmin,zjdemin,zmin,delt)
        call ephim(2,iaph,ipla,ical,ak,kmax,zjdemax,zmax,delt)
```



```
650     if (ipla==3) izmin = idint(zmin)
      endif

! . . Parameter fuer Transit-Pruefung
      if (ilin==1) then
655     itransit=1; il(1)=1; il(2)=3; il(3)=2
      elseif (ilin==2) then
        itransit=2; il(1)=2; il(2)=3; il(3)=1
      else
        itransit=0; il(1)=1; il(2)=4; il(3)=1
660   endif

!-----Einlesen der Startwerte und Parameter fuer FITEX
!     sowie der Koordinaten der Pyramiden bzw. Kammern
      j0 = 0
665   if (ipla==1) j0 = 18
      if (ipla==3) e(1) = 1.d-6
      if (ipla==1 .or.ipla==2) then
        open(unit=10,file='inpdata.t')
        do i=1,8+j0; read(10,*); enddo
670     read(10,*) dummy,(x0(i),i=1,7)
        read(10,*) dummy,(e(i),i=1,7)
        read(10,*)
        read(10,*) dummy,(iw0(i),i=1,4)
        read(10,*) dummy,(w0(i),i=1,3)
675     read(10,*)
        read(10,*) dummy,iter
        read(10,*); read(10,*)
!       Indizes von rp: 1. Pyr./Kammern, 2. Koordinaten und "Hoehe"
        read(10,*) dummy,(rp(1,i),i=1,4)
680     read(10,*) dummy,(rp(2,i),i=1,4)
        read(10,*) dummy,(rp(3,i),i=1,4)
        read(10,*)
        if ((ison==2 .or.ipla==2).and.is12==0) then
          read(10,*) dummy,diff(2),diff(3)
685     else
          read(10,*)
        endif
        do i=1,22-j0; read(10,*); enddo
        do i=1,4; read(10,*) dummy,zjda(i); enddo
690     close(10)
        if (ipla==2 .and.imod/=3) call chambers(ika,rp)
      endif

!-----Einlesen der Transitserien zum Festlegen der Startnummer(n)
695   if (ilin<=2) then
        do i=-180,170
          ase(i) = z0; ise(i) = i0
          if (.not.(iop0==-803 .and.ilin==2)) ser(i,1) = z0
          ser(i,2) = z0
700     enddo
        if (iop0/=-803) then
          open(unit=10,file='inserie.t')
          do i=1,5; read(10,*); enddo
          do i=-150,150,5; read(10,*)idummy,(ser(i+j,1),j=0,4); enddo
705       do i=1,4; read(10,*); enddo
          do i=-10,15,5; read(10,*)idummy,(ser(i+j,2),j=0,4); enddo
          close(10)
        endif
```

```
            ismax = -10000; zstart = 99.99d0
      endif

!-----Weitere Initialisierungen
      do i=0,4; inum(i) = i0; enddo
      isflag = i0; iflag1 = i0; iflag2 = i0
      ifl = i0; ipos = i0; nfit = 7; mfit = 9
      ipar = i0; if (isep==4) ipar = 2
      indx = 1; iekk = iek; prec = z0
      lu = 10; delt = z0; step = step/24.d0
      diff1 = diff(2); diff2 = diff(3)
      zamax = dfloat(iamax); zjdevor = -1.d10
      do i=0,9; md(i) = 1; enddo
!h    do i=1,100; ihis(i) = i0; enddo  !h

!     Initialisierung zur Berechnung fuer die Datei "inserie.t",
!     (--> "inser-2.t", danach manuelles Kopieren nach "inserie.t")
      if (iop0==-803) then
        if (ilin==1) is = -177  ! fuer Merkur, Jahre -18000 bis 18000
        if (ilin==2) is = -6    ! fuer  Venus, Jahre -30000 bis 30000
      endif

! . . Berechnung des Zeitsprungs fuer die Option "Linearkonstell.";
!     "tsprung" ist ein Zeitintervall in Tagen, das nach dem Ablauf
!     einer Konjunktion von Venus und Erde uebersprungen wird. Dieses
!     darf nicht zu gross sein, um alle Ereignisse zu erfassen.
!     Das erste Ereignis im Intervall der Jahre -13000 bis 17000 geht
!     verloren fuer tsy = 577 Tage (tsprung = 557 Tage, dwin = 5 Grad),
!     d.h. "tsprung" waere zu gross. Darueber hinaus ergab sich je-
!     weils als groesster zulaessiger Wert fuer tsy (Version Kepl.):
!
!       dwin    tsy   tsprung        dwin    tsy   tsprung
!      [Grad]  [Tage]  [Tage]       [Grad]  [Tage]  [Tage]
!     ------------------------     ------------------------
!         5     576     557           20     577     510
!        10     578     543           45     578     430 (not used)
!        15     578     527           90     575     286 (not used)
!     ------------------------     ------------------------
!
!     Die Gleichung fuer tsprung (siehe unten) ist sinnvoll, da alle
!     tsy-Werte etwa gleich gross sind, was auch fuer die Optionen
!     "Kurzv." und "Kombi." gilt. Zur Sicherheit wurde tsy = 570 Tage
!     festgelegt (synodische Umlaufzeit der Venus: 583.9 Tage).
      if (ipla==3 .and.ison==5) step = 1.d0
      dwi0 = dwi;  tsy = 570.d0    ! (fuer Syzygien)
      if (ilin==1) tsy = 115.7d0   ! (Merkur, optim.)
      if (ilin==2) tsy = 582.7d0   ! (Venus, optim.)
      if (ipla==3 .and.ikomb==i0) dwi = dwi + 1.d0; dwin = dwi
      if (ilin<=2) tsprung = tsy
      if (ilin>=3) tsprung = dnint(tsy*(1.d0-dwin/180.d0))
      if (tsprung<1.d0) tsprung = 1.d0

! . . Blickrichtung von der suedlichen ekliptikalen Hemisphaere
      if (iek==2 .and.ipla/=3) then
        diff1 = -diff1; diff2 = -diff2
        do i=1,9; diff(i) = -diff(i); enddo
      endif
      if (ipla==3) go to 20
```



```
!-----Pyramidenabstaende und Winkel
!     Indizes von "pyr":
!         1 bis 5: leer
!        6: leer     7: pdx      8: pdy      9: pdz     10: leer
!       11: pax     12: pbx     13: pcx     14: pay     15: pby
!       16: pcy     17: paz     18: pbz     19: pcz     20: leer
!       21: pa      22: pb      23: pc      24: pb/pa oder pbx/pax
!       25: pc/pa oder pby/pay  26: pc/pb oder pby/pbx  27: alpha
!       28: beta    29: gamma   30: leer    31: alpha1  32: alpha2
!       33: alpha3  34: pax/2   35: pay/2   36: pbx/2   37: pby/2
!       38: (pax+pbx)/2   39: (pay+pby)/2   40: leer
!     Indizes 11-19 und 21-29 bei "pyr" und "xyr" entsprechen sich.
!
! . . Anpassung der Koordinaten fuer Grundflaeche, Schwerpunkt und
!     Spitze der Pyramiden bzw. Ostwand, Mitte und Westwand der
!     Kammern.
      if (ihi==2) then
        cm = 0.25d0
        if (ipla==2) cm = 0.5d0
        do i=1,3; rp(i,4) = rp(i,4) * cm; enddo
      endif
      if (ihi==2 .or.ihi==3) then
        do i=1,3; rp(i,3) = rp(i,3) + rp(i,4); enddo
      endif
! . . Abstaende der Pyramiden bzw. Kammern und weitere Groessen.
      pyr(11) = rp(2,1)-rp(3,1)
      pyr(12) = rp(1,1)-rp(3,1)
      pyr(14) = rp(2,2)-rp(3,2)
      pyr(15) = rp(1,2)-rp(3,2)
      pyr(17) = rp(2,3)-rp(3,3)
      pyr(18) = rp(1,3)-rp(3,3)
      pyr(13) = pyr(12)-pyr(11)
      pyr(16) = pyr(15)-pyr(14)
      pax = pyr(11);  pay = pyr(14);  paz = z0
      pbx = pyr(12);  pby = pyr(15);  pbz = z0
      pcx = pyr(13);  pcy = pyr(16);  pcz = z0
      if (ison==3) then
        pyr(31) = - datan(pyr(14)/pyr(11))
        pyr(32) = - datan(pyr(15)/pyr(12))
        pyr(33) = - datan(pyr(16)/pyr(13))
        pyr(34) = pyr(11)*0.5d0;  pyr(35) = pyr(14)*0.5d0
        pyr(36) = pyr(12)*0.5d0;  pyr(37) = pyr(15)*0.5d0
        pyr(38) = (pyr(11)+pyr(12))*0.5d0
        pyr(39) = (pyr(14)+pyr(15))*0.5d0
      endif
!     Koordinaten des gemeinsamen Zentrums "rcm" der drei Pyramiden
!     bzw. Kammern und mittlerer Abstand zu den Pyramiden bzw. Kammern
!     "dmi" (zur Fehlerberechnung von "Sonnen-","Planeten- und Aphel-
!     positionen" in Giza in den Subroutinen "sonpos", "aphelko" und
!     "plako")
      do i=1,3; rcm(i) = (rp(1,i) + rp(2,i) + rp(3,i))/3.d0; enddo
      do i=1,3
        acm(i) = dsqrt((rp(i,1)-rcm(1))**2 + (rp(i,2)-rcm(2))**2 &
                       + (rp(i,3)-rcm(3))**2)
      enddo
      dmi = (acm(1) + acm(2) + acm(3))/3.d0
!c    do i=1,8
!c      write(6,'(5f12.6)') (pyr(5*(i-1)+j),j=1,5)
!c    enddo
```

```
      ! . . Zusaetze zur 3-dim. Berechnung
            if (ison>=4) then
              pyr(19) = pyr(18) - pyr(17)
 830          paz = pyr(17); pbz = pyr(18); pcz = pyr(19)
      !c      write(6,'('' x: '',3f12.3)') (pyr(i),i=11,13)
      !c      write(6,'('' y: '',3f12.3)') (pyr(i),i=14,16)
      !c      write(6,'('' z: '',3f12.3)') (pyr(i),i=17,19)
      ! . . . Erzeugung eines Vektors pd, der auf pa und pb senkrecht steht.
 835          pdx = pby * paz - pay * pbz
              pdy = pax * pbz - pbx * paz
              pdz = pbx * pay - pax * pby
              aba = dsqrt(pax*pax + pay*pay + paz*paz)
              abb = dsqrt(pbx*pbx + pby*pby + pbz*pbz)
 840          abd = dsqrt(pdx*pdx + pdy*pdy + pdz*pdz)
              dfakt = (abb + aba) * 0.5d0/abd
              pyr(7) = pdx * dfakt
              pyr(8) = pdy * dfakt
              pyr(9) = pdz * dfakt
 845  ! . . . Modellwerte fuer FITEX
              if (ison==5) then
                z(1) = z0;   z(2) = z0;   z(3) = z0
                z(4) = pax;  z(5) = pay;  z(6) = paz
                z(7) = pbx;  z(8) = pby;  z(9) = pbz
 850          endif
            endif
      ! . . Laengen, Laengenverhaeltnisse, Winkel
            if (ison<=2) then
              pyr(24) = pbx/pax
 855          pyr(25) = pby/pay
              pyr(26) = pby/pbx; if (iek==2) pyr(26) = -pyr(26)
            else
              pyr(21) = dsqrt(pax*pax + pay*pay + paz*paz)
              pyr(22) = dsqrt(pbx*pbx + pby*pby + pbz*pbz)
 860          pyr(23) = dsqrt(pcx*pcx + pcy*pcy + pcz*pcz)
              pyr(24) = pyr(22)/pyr(21)
              pyr(25) = pyr(23)/pyr(21)
              pyr(26) = pyr(23)/pyr(22)
              pyr(27) = dacos((pax*pbx+pay*pby+paz*pbz)/(pyr(21)*pyr(22)))
 865          pyr(28) = dacos((pax*pcx+pay*pcy+paz*pcz)/(pyr(21)*pyr(23)))
              pyr(29) = dacos((pbx*pcx+pby*pcy+pbz*pcz)/(pyr(22)*pyr(23)))
            endif

      !-----Einlesen aller Parameter der VSOP87D-Kurzversion (Meeus)
 870     20 if (imod==1) then
              open(unit=10,file='invsop1.t')
              read(10,*)
              do n=1,12
                read(10,*); read(10,*) lmax(n)
 875            read(10,*) (jp(n,j),j=1,lmax(n))
                do m=1,lmax(n)
                  read(10,*)
                  do j=1,jp(n,m)
                    read(10,*) idummy,(par1(i,j,m,n),i=1,3)
 880              enddo
                enddo
              enddo
              close(10)
            endif
 885
```



```
      !-----Bahnparameter als Polynome 3. Grades aus VSOP82 (Meeus)
            if (io==2 .or.irb/=1 .or.imod==3 .or.ipla==3) then
              open(unit=10,file='invsop3.t')
              do ll=1,2
 890            do n=1,3; read(10,*); enddo
                do k=1,8
                  do n=1,2; read(10,*); enddo
                  do j=1,6; read(10,*) (par3(i,j,k,ll),i=1,4); enddo
                enddo
 895          enddo
              close(10)
            endif

      !-----Titelzeilen
 900        do iu=ix,6,5
              call titel1(iaph,ijd,iu,ison,ipla,ilin,isep,nurtr, &
                iuniv,is12,iop0)
              call titel2(iu,imod,ivers,irb,ipla, &
                ison,ihi,iek,ijd,ika,iaph,ilin,ical,ak,zjde1,zjahr,delt, &
 905            dwi,dwikomb,dwi0,dwi2,dwi3,iamax,step,ikomb,zmin,zmax)
      ! . . . Tabellenkopf
              call tabe(iaph,imod,iek,iu,io,ison,ipla,ilin,itran,is12, &
                iop0,iout)
            enddo
 910        if (iaph==5) go to 200
            if (ipla==3) go to 300

      !     Anmerkung: In jedem Programmlauf wird nur eine
      !     der drei folgenden Hauptschleifen verwendet.
 915
      !=======================================================================
      !------------------------ 1. Hauptschleife ----------------------------
      !=======================================================================

 920  !-----1. Hauptschleife (Pyramiden- und Kammerpositionen---------------
      !     sowie Aphel- und Perihelzeitpunkte des Merkur)
            k = kmin
       100  zk = dfloat(k)
            if (imod==2 .and.ijd==15 .and.iaph<=2) zk = ak
 925        isw = 1; if (iaph<=2 .and.iout==3) isw = 2
            jmax = i0; ncount = i0

      !.....JDE-Zeitpunkt (Merkur im und ausserhalb des Aphels)
       120  zjde = zjde1
 930        if (ijd==15 .or.iaph==3 .or.iaph==4) then
              ik = k
              if (isw==1 .or.(isw==2 .and.iaph<=2)) then
                if (ijd==15 .and.(imod/=2 .or. &
                  (imod==2 .and.(iaph==3 .or.iaph==4)))) ak = zk
 935            if (ijd==15) then
                  call ephim(i0,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
                else
                  call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
                endif
 940          else
                acount = dfloat(ncount)
                if (ijd==15) then
                  ak = zk + step * (acount - zamax * 0.5d0)/ymer
                  call ephim(i0,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
```

```
            else
              zjde = zjde1 + step * (acount - zamax * 0.5d0)
              call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
            endif
          endif
        endif
        if (ijd==i0) call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
        ik = idnint(ak)
        time = (zjde - zjd0)/tcen
        tau  = (zjde - zjd0)/tmil
        if (ison==5) then
          do i=1,4; iw(i) = iw0(i); enddo
          do i=1,3; w(i) = w0(i); enddo
          do i=1,7; x(i) = x0(i); enddo
          do i=4,6; x(i) = x(i) * pidg; enddo
        endif
        inum(1) = inum(1) + 1

!.....Variante 1 (VSOP87D, Kurzversion aus "Meeus", mult. threads)
        if (imod==1) then
          !$omp parallel do default(shared) private(i,resu)
            do i=1,9; call vsop1(i,tau,resu); re(i) = resu; enddo
          !$omp end parallel do
        endif

!.....Variante 2 (VSOP87A/C, Vollversion)
  140 if (imod==2) then
          do i=1,3; ii = 3*(i-1)
            call vsop2(zjde,ivers,i,md,ix,prec,lu,r,ierr,rku)
            do j=1,3; re(ii+j) = rku(j); enddo
          enddo
        endif

!.....Variante 3 (Keplersche Gleichung, Polynome 3. Grades nach VSOP82)
        if (io==2 .or.irb/=1 .or.imod==3) then
          immax = 3
          if (io==2) immax = 4
          do i=1,immax; ii = 6*i
            call vsop3(lv,i,ix,ir,time,res)
            if (ir/=i0) go to 500
            re(25+ii) = res(1);  re(28+ii) = res(5)
            re(26+ii) = res(2);  re(29+ii) = res(4)
            re(27+ii) = res(3);  re(30+ii) = res(6)
            if (imod==3 .and.i<=4) re(3*i-2) = res(11)
          enddo
        endif

!.....Koordinaten-Transformation und Bestimmung von F-pos
        if (irb>=2 .or.imod/=3) call kartko(ison)
        if (irb>=2) call transfo(irb,rku)
        if (irb>=2 .or.imod/=3) &
          call relpos(ipla,ison,ijd,iek,iekk,ika)

!.....Korrelation der Positionen pruefen, Output
        ic   = i0
        err3 = z0
        err4 = z0
        dif1 = re(1) - re(4); call reduz(dif1,i0,i0)
        dif2 = re(1) - re(7); call reduz(dif2,i0,i0)
```



```
        if (ison<=2) then
          err1 = dif1 - diff1; call reduz(err1,i0,i0)
          err2 = dif2 - diff2; call reduz(err2,i0,i0)
          if (iek==3) then
            err3 = dif1 + diff1; call reduz(err3,i0,i0)
            err4 = dif2 + diff2; call reduz(err4,i0,i0)
          endif
!.......Hauptbedingung pruefen (ison = 1, 2). . . . . .
          if ((dabs(err1)<=dwi.and.dabs(err2)<=dwi).or.ijd/=15 &
            .or.(iek==3 .and.dabs(err3)<=dwi.and.dabs(err4)<=dwi) &
            .or.(ijd==15 .and.imod==2 .and.ikomb==i0)) then
            if (ikomb==1 .and.imod==1) then
              imod = 2; dwi = dwikomb; go to 140
            endif
            if (iek==3) then
              iekk = 1
              if (dabs(err3)<=dwi.and.dabs(err4)<=dwi) iekk = 2
            endif
            inum(2) = inum(2) + 1; ic = 1
!           Resultat Output
            call konst(ik,kon)
            dd = dn; if (iek==2 .or.iekk==2) dd = ds
            do iu=ix,6,5
              if (imod/=3) then
                if (iek==3 .and.iekk==1) then
                  write(iu,56)kon,ik,zjde,zjahr,re(1), &
                    dif1,dif2,err1,err2,dd
                elseif (iek==3 .and.iekk==2) then
                  write(iu,56)kon,ik,zjde,zjahr,re(1), &
                    dif1,dif2,err3,err4,dd
                else
                  write(iu,55)kon,ik,zjde,zjahr,re(1), &
                    dif1,dif2,err1,err2,xyr(36)
                endif
              else
                if (iek==3 .and.iekk==2) then
                  write(iu,56)kon,ik,zjde,zjahr,re(1), &
                    dif1,dif2,err3,err4,dd
                else
                  write(iu,56)kon,ik,zjde,zjahr,re(1), &
                    dif1,dif2,err1,err2,dd
                endif
              endif
            enddo
          endif
        else
          if ((iaph==3 .or.iaph==4).and.isw==1 .and.ijd==15) then
            ifl = i0; if (xyr(36)<=dwi2) ifl = 1
          endif
!.......Hauptbedingung pruefen (ison = 3, 4, 5) . . . . . .
          if (((isw==1 .or.(isw==2 .and.iaph<=2)).and. &
             (xyr(36)<=dwi.or.ijd/=15 .or. &
             (imod==2 .and.ikomb==i0.and.iaph<=2))).or. &
             (isw==2 .and.((ifl==1 .and.xyr(36)<=dwi3.and. &
             ijd==15).or.ijd/=15))) then
            if (ikomb==1 .and.imod==1) then
              imod = 2; dwi = dwikomb; go to 140
            endif
            inum(2) = inum(2) + 1
```

```
      !         Sonnenposition
                call sonpos(ison,iek,ix,rp(3,1),rp(3,2),rp(3,3),rcm,dmi, &
1065                          iter,iw,ke,mfit,nfit,f,x,e,w,y,z)
                ic = 1; dd = dn
                if (iek==2) dd = ds
                do isun=1,4; ort(i0,isun) = xyr(30+isun); enddo
      !         Resultat Output
1070            if (isw==1) then
                  call konst(ik,kon)
                  do iu=ix,6,5
                    if (ison==5) then
                      if (ipla==2) then
1075                    write(iu,184)kon,ik,zjahr,dif1,dif2,ke,iw(3), &
                          (xyr(30+i),i=1,4),dd,xyr(36)
                      else
                        write(iu,165)kon,ik,zjahr,dif1,dif2,ke,iw(3), &
                          (xyr(30+i),i=1,4),dd,xyr(36)
1080                    endif
                    elseif (ison==3) then
                      write(iu,67)kon,ik,zjahr,re(1),dif1,dif2, &
                        xyr(31),xyr(32),emp,xyr(34),dd,xyr(36)
                    else
1085                  if (ipla==2) then
                        write(iu,85)kon,ik,zjahr,re(1),dif1,dif2, &
                          (xyr(30+i),i=1,4),dd,xyr(36)
                      else
                        write(iu,65)kon,ik,zjahr,re(1),dif1,dif2, &
1090                      (xyr(30+i),i=1,4),dd,xyr(36)
                      endif
                    endif
                  enddo
                else
1095              if (((xyr(36)<=dwi2.or.iaph<=2).and.ijd==15).or. &
                      ijd/=15 .or.imod==2) then
                    if (iout==3) then
                      call konst(ik,kon)
                      delh = delt * 24.d0
1100                  call reduz(x(5),1,i0)
                      if (ipla==1) then
                        xma = xyr(35) * 1.d-7
                        sonne = -datan((xyr(33)-rp(3,3))/xyr(31))*gdpi
                      else
1105                    xma = xyr(35) * 1.d-9
                        dxr = xyr(31)-rp(3,1); dyr = xyr(32)-rp(3,2)
                        sonne = -datan(dyr/dxr)*gdpi
                        if (dxr*dcos(sonne*pidg)>0.d0) sonne = sonne + 180.d0
                        call reduz(sonne,i0,i0)
1110                  endif
                      do iu=ix,6,5
                        if (iaph==3 .or.iaph==4) then
                          if (ipla==2) then
                            write(iu,275)zjde,delh,x(5)*gdpi,xma, &
1115                          sonne,(xyr(30+i),i=1,4),dd,xyr(36)
                          else
                            write(iu,255)zjde,delh,x(5)*gdpi,xma, &
                              sonne,(xyr(30+i),i=1,4),dd,xyr(36)
                          endif
1120                    elseif (iaph<=2) then
                          if (ipla==2) then
```



```
                            write(iu,276)kon,ik,zjahr,x(5)*gdpi,xma, &
                              sonne,(xyr(30+i),i=1,4),dd,xyr(36)
                          else
1125                        write(iu,256)kon,ik,zjahr,x(5)*gdpi,xma, &
                              sonne,(xyr(30+i),i=1,4),dd,xyr(36)
                          endif
                        endif
                      enddo
1130                else
      !               Pruefung zur Signifikanz --> dk
                      dk = '   '
                      zf = dabs((xyr(35)-zthe(ipla))/zthe(ipla))
                      if (zf<=2.d-2 .and.xyr(36)>0.5d0) dk = 'M  '
1135                  if (zf>2.d-2 .and.xyr(36)<=0.5d0) dk = 'F  '
                      if (zf<=2.d-2 .and.xyr(36)<=0.5d0) dk = 'FM '
                      if (zf<=1.d-3 .and.xyr(36)<=0.1d0) dk = '>>>'
                      do iu=ix,6,5
                        if (ison==5) then
1140                      if (ipla==2) then
                            write(iu,386)dk,ik,zjde,xyr(35),ke,iw(3), &
                              (xyr(30+i),i=1,4),dd,xyr(36)
                          else
                            write(iu,366)dk,ik,zjde,xyr(35),ke,iw(3), &
1145                          (xyr(30+i),i=1,4),dd,xyr(36)
                          endif
                        elseif (ison==3) then
                          write(iu,367)dk,ik,zjde,xyr(35),ncount-iamax/2, &
                            xyr(31),xyr(32),emp,xyr(34),dd,xyr(36)
1150                    else
                          if (ipla==2) then
                            write(iu,384)dk,ik,zjde,xyr(35),ncount-iamax/2, &
                              (xyr(30+i),i=1,4),dd,xyr(36)
                          else
1155                        write(iu,365)dk,ik,zjde,xyr(35),ncount-iamax/2, &
                              (xyr(30+i),i=1,4),dd,xyr(36)
                          endif
                        endif
                      enddo
1160                endif
                  endif
                endif
      !h          call histogramm(xyr(36),ihis)  !h
              endif
1165        endif

      !.....Weiterer Output
            do iu=ix,6,5
              if (ic==1 .and.imod/=3 .and.io==2 .and.is12==0) then
1170            call linie(iu,2)
                write(iu,57) (re(i),i=1,9)
                do i=1,3; t1(i) = ' '; if (xyr(3+i)<z0) t1(i) = '-'; enddo
                write(iu,54) (xyr(i),i=1,3),t1(1),dabs(xyr(4)), &
                  t1(2),dabs(xyr(5)),t1(3),dabs(xyr(6)),(xyr(i),i=7,9)
1175            write(iu,'(1x,6f9.6,f22.8,'' %'')') xyr(11),xyr(12), &
                  xyr(14),xyr(15),xyr(17),xyr(18),xyr(36)
                call linie(iu,2)
              endif
              if (is12/=0) call linie(iu,1)
1180          if (is12==0 .and.ic==1.and.imod==3.and.io==2) call linie(iu,2)
```

```
            if (ic==1 .and.io==2 .and.is12==0) then
              if (imod/=3) then
                if (ivers==3) then
                  write(iu,'('' ascending node (M/V/E/Ma): '',2f12.6, &
1185                & ''      ---   '',f12.6)')re(34),re(40),re(52)
                else
                  write(iu,'('' ascending node (M/V/E/Ma): '',4f12.6)') &
                   (re(28+6*i),i=1,4)
                endif
1190            write(iu,'('' inclination i  (M/V/E/Ma): '',4f12.6)') &
                  (re(29+6*i),i=1,4)
                write(iu,'('' perihelion pi  (M/V/E/Ma): '',4f12.6)') &
                  (re(30+6*i),i=1,4)
                if (imod/=3 .and.irb/=1) &
1195            write(iu,'('' ang. par. (omega, i, tau): '',3f12.6)') &
                  ao*gdpi,ai*gdpi,at*gdpi
                if (ison==5) then
                  write(iu,'('' transl. X1, X2, X3; del-t: '',3f12.6, &
                    & f9.3,'' days'')') (x(i),i=1,3),delt
1200              do i=4,6; call reduz(x(i),1,i0); enddo
                  write(iu,'('' Euler angl. X4, X5, X6; M: '',3f12.6, &
                    & f13.0)') (x(i)*gdpi,i=4,6),xyr(35)
      !c          write(6,'('' X7: '', f12.6)') x(7)
                endif
1205            else
                  do i=5,8; ii = 6*i
                    call vsop3(lv,i,ix,ir,time,res); if (ir/=i0) go to 500
                    re(25+ii) = res(1);  re(28+ii) = res(5)
                    re(26+ii) = res(2);  re(29+ii) = res(4)
1210                re(27+ii) = res(3);  re(30+ii) = res(6)
                  enddo
                  call elements(iu,ivers,pla)
                endif
                if ((ison==3 .and.ijd>=1 .and.ijd<=10).or.ison==4) write(iu,&
1215              & '('' scale factor M          : '',f13.0)')xyr(35)
                call linie(iu,1)
              endif
            enddo

1220  !.....Output: Koordinaten aller Planeten einschliesslich Neptun und
      !     des Schwerpunktsystems Erde-Mond, letzteres nur fuer VSOP87A,
      !     sowie transformierte "planetarische" Koordinaten in Giza
            if ((imo4==1 .and.iaph<=2 .and.is12==0 .and.io==2) &
                .or.is12/=0) then
1225          call plako(diff,ipla,ijd,ik,ison,ipos, &
               rcm,x,y,ort,rp,dd,dn,dss,pla,plan,emp,text,tt,titab, &
               is12,dmi,zjda,zjde,ivers,md,ix,prec,lu,r,ierr,rku)
            endif

1230  ! . . Ruecksprung fuer Aphel-Umgebung
            if (ikomb==1 .and.imod==2) then
              imod = 1; dwi = dwi0
            endif
            if (iaph==3 .or.iaph==4) then
1235          ncount = ncount + 1
              if (ncount>jmax) then
                ncount = i0
                if (isw==1) then
                  if (ijd==15 .and.ifl==i0) go to 190
```



```
1240                isw = 2; jmax = iamax; go to 120
                  endif
                else
                  go to 120
                endif
1245          endif

      ! . . Standardrueecksprung
        190 k = k + 1
            if (k<=kmax) go to 100
1250
      !.....Aphelposition der Merkurbahn fuer Konstellation 13 bzw. 14,
      !     sowie "quick start option" 371 und 372
            if (ipla/=3) call aphelko(imod,ivers,iaph,ipla, &
               ison,ijd,io,iop0,ix,rp(3,4),x,y,rcm,dmi)
1255
      !-----Ende der 1. Hauptschleife (Pyramiden- und Kammerpositionen)------
            go to 400

      !=======================================================================
1260  !---------------------- 2. Hauptschleife -------------------------------
      !=======================================================================

      !-----2. Hauptschleife (freier Zeitpunkt und Minimierung von Fpos------
      !     fuer Pyramiden- und Kammeranordnung, Tabelle 51 in "Pyramiden
1265  !     und Planeten" und Tabelle 20 (?) im zweiten Buch)
        200 zjde = zjdemin
            dfe = 0.3d0; eep = e(1)
            irestart = i0; x36 = z0
      !     VORSICHT: "zfact" und "zstep" nicht zu gross waehlen, weil sonst
1270  !     beim Rueckspung (s.u.) Konstellationen verloren gehen. Standard-
      !     werte fuer Pyramiden: 0.5/ 1.0 und fuer die Kammern: 0.1/ 0.2
            if (ipla==1) then
              zfact = 0.5d0; zstep = 1.d0
            else
1275  !       (optimiert fuer alle Kammerzuordnungen)
              zfact = 0.1d0; zstep = 0.2d0
            endif

      !.....Startparameter fuer "fitmin"
1280    220 ifitrun = i0; itin = i0
            imodus = 1; iflag = i0
            ke = 1; indx = 1; nu = i0
            ddx1 = 1.d0; ddx2 = 1.d0
            do i=1,10; test(i) = z0; enddo
1285        do i=1,5
              xx(i) = z0; yy(i) = z0
            enddo
            xx(1) = zjde; go to 250
        240 call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
1290    250 tau = (zjde - zjd0)/tmil
            if (ison==5) then
              do i=1,4; iw(i) = iw0(i); enddo
              do i=1,3; w(i) = w0(i); enddo
              do i=1,7; x(i) = x0(i); enddo
1295          do i=4,6; x(i) = x(i) * pidg; enddo
            endif
            inum(1) = inum(1) + 1
```

```
      !.....Variante 1 (VSOP87D, Kurzversion aus "Meeus", mult. threads)
      if (imod==1) then
        !$omp parallel do default(shared) private(i,resu)
          do i=1,9; call vsop1(i,tau,resu); re(i) = resu; enddo
        !$omp end parallel do
      endif

      !.....Variante 2 (VSOP87A/C, Vollversion)
      if (imod==2) then
        do i=1,3; ii = 3*(i-1)
          call vsop2(zjde,ivers,i,md,ix,prec,lu,r,ierr,rku)
          do j=1,3; re(ii+j) = rku(j); enddo
        enddo
      endif

      !.....Koordinaten-Transformation und Bestimmung von F-pos
      call kartko(ison)
      call relpos(ipla,ison,ijd,iek,iekk,ika)
      if (ison==5) yy(indx) = xyr(36)

      ! . . zjde so lange erhoehen, bis relativer Fehler nicht mehr steigt.
      !c    write(6,'('' zjde,irestart,xyr(36),dwi,imod = '',f18.7,i3, &
      !c      & 2f9.3,i3)') zjde,irestart,xyr(36),dwi,imod
      if (xyr(36)>10.d0) imod = 1
      if (irestart==1) then
        if (xyr(36)>x36) then
          go to 290
        else
          zjdelim = zjde
        endif
      endif; irestart = i0

      ! . . Bedingung zum Aufruf von fitmin pruefen
      if (xyr(36)>dwi.and.ifitrun==i0) go to 290
      if (ikomb==1) imod = 2

      ! . . Minimierung des relativen Fehlers F-pos mit "fitmin"
      ifitrun = 1; imodus = 1
      if (ddx1<dfe.or.ddx2<dfe) imodus = 2
      call fitmin(imod,imodus,iaph,ke,xx,yy,eep,step,nu,iflag, &
             ddx1,ddx2,test,itin,indx,ix)
      zjde = xx(indx)
      if (ke==1) go to 240
      irestart = 1

      ! . . verhindert, dass fitmin endlos ins vorherige Minimum faellt
      if (dabs(zjde-zjdevor)<=0.1d0) then
         zjde = zjdelim; go to 290
      endif; zjdevor = zjde

      !.....Hauptbedingung pruefen (ison = 5) . . . . . . . . . . . . . . . .
      if (xyr(36)>=dwikomb) go to 290
      inum(2) = inum(2) + 1

      ! . . Sonnenposition und Output
      call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
      call konst(iak,kon)
      call sonpos(ison,iek,ix,rp(3,1),rp(3,2),rp(3,3), &
                  rcm,dmi,iter,iw,ke,mfit,nfit,f,x,e,w,y,z)
```



```
        dd = dn
        if (iek==2 .or.iekk==2) dd = ds
        xma = xyr(35) * 1.d-9
        if (ipla==1) xma = xyr(35) * 1.d-7
        call reduz(x(5),1,i0)
        do iu=ix,6,5
          if (iout==3) then
            if (ipla==1) then
              write(iu,405)kon,iak,zjahr,delt,x(5)*gdpi,xma, &
                (xyr(30+i),i=1,3),dd,xyr(36)
            else
              write(iu,406)kon,iak,zjahr,delt,x(5)*gdpi,xma, &
                (xyr(30+i),i=1,3),dd,xyr(36)
            endif
          else
            if (ipla==1) then
              write(iu,407)kon,iak,zjde,zjahr,ke,iw(3), &
                (xyr(30+i),i=1,4),dd,xyr(36)
            else
              write(iu,408)kon,iak,zjde,zjahr,ke,iw(3), &
                (xyr(30+i),i=1,4),dd,xyr(36)
            endif
          endif
        enddo
      !h    call histogramm(xyr(36),ihis)  !h

      ! . . Standardruecksprung
      290 zjump = xyr(36)*zfact + zstep
        zjde = zjde + zjump; x36 = xyr(36)
        if (zjde<=zjdemax) go to 220

      !-----Ende der 2. Hauptschleife (freier Zeitpunkt)--------------------
        go to 400

      !=========================================================================
      !------------------------ 3. Hauptschleife -------------------------------
      !=========================================================================

      !-----3. Hauptschleife (Suche von Linearkonstellationen)---------------
      !     Syzygium von Sonne, Merkur, Venus, Erde und Mars,
      !     sowie Bestimmung der Transite von Merkur und Venus.

      !     "zfact" und "zstep" wie in 2. Hauptschleife (nicht zu gross)
        300 zfact = 0.025d0 * (1.d0 + (21.d0-dwi)/20.d0)
        if (dwi>=21.d0) zfact = 0.025d0
        zstep = 0.01d0
        sz = (1.d0 + 10.d0*zfact)
        zjde = zjdemin
        dfd = 5.d0; dfe = 0.5d0
        izp = 1; icv = 0
        310 zjdestep = zjde
        if (ilin==2 .and.inum(0)>1 .and.iop0/=-803) dfd = 0.02d0
        call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
        ik = idnint(ak)
        inum(0) = inum(0) + 1
        if (ilin>=3) itransit = i0
        do i=1,2; tra(i) = ' '; enddo
        if (ison==5) ifitrun = i0
        if (ilin<=2) ifitrun = 1
```

```
      !.....Startparameter fuer "fitmin", "sekante" und "ringfit"
        320 if (ison==5) then
              iflag = i0; ke = 1
1420          indx = 1; nu = i0
              ddx1 = dfd; ddx2 = ddx1; itin = i0
              do i=1,10; test(i) = z0; enddo
              do i=1,5
                xx(i) = z0
1425            yy(i) = z0
              enddo
              xx(1) = zjde
            endif
            go to 340
1430    330 zjde = xx(indx)
            call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
        340 time = (zjde - zjd0)/tcen
            tau  = (zjde - zjd0)/tmil
            inum(1) = inum(1) + 1
1435
      !.....Variante 1 (VSOP87D, Kurzversion aus "Meeus", mult. threads)
            if (imod==1) then
              !$omp parallel do default(shared) private(i,resu)
                do i=1,12; call vsop1(i,tau,resu); re(i) = resu; enddo
1440          !$omp end parallel do
              if (ilin<=2) then
                call kartko(ison)
                do i=1,9; rk(i) = xyr(i); enddo
              endif
1445        endif

      !.....Variante 2 (VSOP87A/C, Vollversion)
        350 if (imod==2) then
              do i=il(1),il(2),il(3); ii = 3*(i-1)
1450            call vsop2(zjde,ivers,i,md,ix,prec,lu,r,ierr,rku)
                do j=1,3
                  re(ii+j) = rku(j)
                  if (ilin<=2) rk(ii+j) = r(j)
                enddo
1455          enddo
            endif

      !.....Variante 3 (Keplersche Gleichung, Polynome 3. Grades nach VSOP82)
            if (imod==3) then
1460          do i=1,4; ii = 6*i
                call vsop3(lv,i,ix,ir,time,res)
                if (ir/=i0) go to 500
                re(25+ii) = res(1);  re(28+ii) = res(5)
                re(26+ii) = res(2);  re(29+ii) = res(4)
1465            re(27+ii) = res(3);  re(30+ii) = res(6)
                if (i<=4) re(3*i-2) = res(11)
              enddo
            endif

1470  !.....Korrelation der Positionen pruefen
            ic = i0; iwo = i0
            df(1) = re(1)-re(4);   df(2) = re(1)-re(7)
            df(3) = re(1)-re(10);  df(4) = re(4)-re(7)
            df(5) = re(4)-re(10);  df(6) = re(7)-re(10)
1475        do i=1,6; call reduz(df(i),i0,i0); enddo
```



```
            if (ilin==3) difm = dmax1(dabs(df(1)),dabs(df(2)),dabs(df(4)))
            if (ilin==4) difm = dmax1(dabs(df(1)),dabs(df(2)),dabs(df(3)), &
                dabs(df(4)),dabs(df(5)),dabs(df(6)))
            if (isep==1) then
1480          if (itransit==1) difm = df(2)
              if (itransit==2) difm = df(4)
            else
              if (itransit==1 .or.itransit==2) then
                call sepa(itransit,2,rk,sep1)
1485            difm = dabs(sep1)
              endif
            endif
            if (ison==5) yy(indx) = difm
      ! . . Test-Ausdruck (--> !t)
1490  !t    difr = re(7)-re(1)
      !t    call reduz(difr,i0,i0)
      !t    do iu=ix,6,5; write(iu,'(''imod,ifit,dt,Le-Lm,jde,difm = '',2i2,&
      !t     &f5.1,f6.1,f18.7,f13.7)')imod,ifitrun,step,difr,zjde,difm;enddo

1495  !.....Hauptbedingung pruefen. . . . . .
            if (difm>dwi.and.ifitrun/=1) go to 370
      ! . . Rucksprung fuer ikomb = 1
            if (ikomb==1 .and.imod==1 .and.ilin>=3) then
              ifitrun = 1; imod = 2
1500          dwi = dwikomb
              go to 350
            endif

      ! . . Minimierung des Gesamtwinkels difm mit "fitmin" fuer ison = 5
1505  !     (Das heisst, "ison" hat hier eine andere Funktion und bedeutet
      !     Minimumsuche.)
            if (ison==5) then
              ifitrun = 1; step = 1.d0
              if (ilin>=3 .and.itransit==i0) then
1510            call fitmin(imod,1,iaph,ke,xx,yy,e(1),step,nu, &
                iflag,ddx1,ddx2,test,itin,indx,ix); zjde = xx(indx)
              endif
              if (itransit==1 .or.itransit==2) then
                if (isep==1) then
1515              xj2 = xx(indx); yy2 = yy(indx); indx = 2
                  call ringfit(xj1,xj2,xj3,yy1,yy2,yy3, &
                    1.d-6,1.d-2,nu,50,ix,ke)
                  xx(2) = xj2; zjde = xj2
                else
1520              eep = e(1)
                  if (ikomb==1 .and.imod==1 .and.isep>=3) eep=1.d2*e(1)
                  imodus = 1
                  if (ddx1<dfe.or.ddx2<dfe) imodus = 2
                   call fitmin(imod,imodus,iaph,ke,xx,yy,eep,dfd,nu, &
1525                 iflag,ddx1,ddx2,test,itin,indx,ix)
                     zjde = xx(indx)
                endif
              endif
              if (ke==1 .or.(isep==1 .and.ke==5)) go to 330
1530        endif

      ! . . Spezialtest fuer ikomb = 0 (imod = 1, 3)
      !     Anmerkung: Aufgrund der Zeitschritte (1 Tag) ist es moeglich,
      !     dass das Minimum des Winkelintervalls (difm) fuer die eklipti-
```

```
!     kalen Laengen der Planeten genau zwischen zwei Zeitpunkten er-
!     reicht wird. Falls die Schwelle (dwi0) so knapp unterschritten
!     wird, dass sie an den Zeitpunkten davor und danach schon wieder
!     ueberschritten wird, wuerde das Ereignis verloren gehen. Des-
!     halb wird die Schwelle (dwi) zuvor um 1 Grad erhoeht, dann das
!     Winkelintervall minimiert und anschliessend geprueft, ob die
!     urspruengliche Schwelle (dwi0) unterschritten wurde.
      if (ikomb==i0.and.ilin>=3) then
        if (difm<=dwi0) go to 360
        go to 370
      endif
! . . Gegebenenfalls Sprung von der oberen zur unteren Konjunktion.
!     Bei Minimierung der Winkelseparation (isep 2,3,4) wuerden ab
!     einem gewissen Zeitpunkt nur noch obere Konjunktionen berech-
!     net werden. Das wird durch die folgende if-Abfrage behoben.
  360 if (isep>=2 .and.((itransit==1 .and.dabs(df(2))>170.d0) &
          .or.(itransit==2 .and.dabs(df(4))>170.d0))) then
        zjde = zjde + tsy*0.5d0
        go to 320
      endif
      if (ikomb/=1 .or.(ikomb==1 .and.(difm<=dwikomb.or. &
          ilin<=2))) then
        if (itransit==i0.and.nurtr==1) inum(2) = inum(2) + 1
        ic = 1
        if (ic==1 .and.icv==0 .and.ison/=5 .and.ilin>=3) then
          inum(3) = inum(3) + 1
          do iu=ix,6,5
            write(iu,'(i12,''. syzygy'')') inum(3)
          enddo
        endif
        call konst(ik,kon)
! . . . Pruefen des Transits (nur bei imod = 1, 2)
        if (itran==1 .and.ison==5) then
          if (itransit==i0.or.ilin<=2) call memo(zjde,zjahr, &
              delt,df(1),df(2),df(3),difm,zmem,iak,imem)
          if (itransit==1 .or.itransit==2) then
            call transit(itransit,ikomb,imod,ipla,ilin,iaph,ivers, &
             isep,ical,iuniv,tr,sep1,itt,sep,zjde,id5,da5,dmo5, &
             zjahr,rk,md,ddx1,ddx2,dfd,test,itin,is,irs,ix,pan,sd,sl,&
             iop0,inum)
            tra(itransit) = tr
          endif
! . . . . Ereignis mit Transit und Output
          if ((ilin>=3 .and.itransit==2).or. &
              (ilin<=2 .and.tr/=' ')) then
            if (ikomb==1 .and.imod==1 .and.ilin<=2) then
              imod = 2; go to 320
            endif
            if (nurtr==1 .or.(nurtr==2 .and. &
              (tra(1)/=' '.or.tra(2)/=' '))) then
              if (ilin<=2 .or.nurtr==2) inum(2) = inum(2) + 1
              iwo = 1
              if (ilin>=3) then
                do iu=ix,6,5
                  if (dabs(zmem(5))<1.d-4) then
                    zmem(5) = dabs(zmem(5))
                    write(iu,456)kon,' ',tra(1),tra(2),imem, &
                      (zmem(i),i=1,7)
                  elseif (dabs(zmem(6))<1.d-4) then
```



```
                    zmem(6) = dabs(zmem(6))
                    write(iu,457)kon,' ',tra(1),tra(2),imem, &
                      (zmem(i),i=1,7)
                  else
                    write(iu,455)kon,' ',tra(1),tra(2),imem, &
                      (zmem(i),i=1,7)
                  endif
                enddo
              else
                if (iop0==-803 .and.(zjahr<=-13000.d0 .or. &
                    zjahr>=17000.d0)) go to 390; ts = ' '
                if (tra(ilin)/='M'.and.tra(ilin)/='V') ts=tra(ilin)
                if (iuniv==2) call delta_T(zjde)
                call jdedate(zjde,ical,ida,da,dmo)
                if (ida(3)>=izmin) then
                 do iu=ix,6,5
                   if (izp<=3) call zwizeile(iu,io,zmem(1), &
                     ilin,imod,isep,ical,izp)
                   if ((isep<=3 .and. zmem(1)<-1566122.5d0).or. &
                       (isep==4 .and.(zmem(1)<-1931365.5d0 .or. &
                                      zmem(1)>=5373484.5d0))) then
                    if (isep<=2) then
                      write(iu,458)kon,ts,imem,da(7),dmo,ida(3), &
                      (ida(i),dp,i=4,5),ida(6),(zmem(i),i=3,6),sep,irs
                    else
                     if (isep==3) then
                      if (itt==3) &
                       write(iu,459)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                       ((id5(l,i),dp,i=4,5),id5(l,6),l=1,5),sep,sl,irs
                      if (itt==2) &
                       write(iu,461)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                       ((id5(l,i),dp,i=4,5),id5(l,6),str2,l=1,3,2), &
                       (id5(5,i),dp,i=4,5),id5(5,6),sep,sl,irs
                      if (itt==1) &
                       write(iu,471)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                       str2,str2,(id5(3,i),dp,i=4,5),id5(3,6), &
                       str2,str2,sep,sl,irs
                     else
                      if (itt==3) &
                       write(iu,659)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                       ((id5(l,i),dp,i=4,5),id5(l,6),l=1,5),sep,sl, &
                       (pan(i),i=1,5),sd(1),sd(2),irs
                      if (itt==2) &
                       write(iu,661)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                       ((id5(l,i),dp,i=4,5),id5(l,6),str2,l=1,3,2), &
                       (id5(5,i),dp,i=4,5),id5(5,6),sep,sl,pan(1), &
                       str3,pan(3),str3,pan(5),sd(1),sd(2),irs
                      if (itt==1) &
                       write(iu,671)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                       str2,str2,(id5(3,i),dp,i=4,5),id5(3,6), &
                       str2,str2,sep,sl,str3,str3,pan(3), &
                       str3,str3,sd(1),sd(2),irs
                      endif
                      if (itt==i0.and.iu==6) inum(2) = inum(2) - 1
                     endif
                    else
                     if (isep<=2) then
                       write(iu,558)kon,ts,imem,da(7),dmo,ida(3), &
                       (ida(i),dp,i=4,5),ida(6),(zmem(i),i=3,6),sep,irs
```

```
                      else
                       if (isep==3) then
1655                    if (itt==3) &
                         write(iu,559)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                         ((id5(l,i),dp,i=4,5),id5(l,6),l=1,5),sep,sl,irs
                        if (itt==2) &
                         write(iu,561)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1660                     ((id5(l,i),dp,i=4,5),id5(l,6),str2,l=1,3,2), &
                         (id5(5,i),dp,i=4,5),id5(5,6),sep,sl,irs
                        if (itt==1) &
                         write(iu,571)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                         str2,str2,(id5(3,i),dp,i=4,5),id5(3,6), &
1665                     str2,str2,sep,sl,irs
                       else
                        if (itt==3) &
                         write(iu,759)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                         ((id5(l,i),dp,i=4,5),id5(l,6),l=1,5),sep,sl, &
1670                     (pan(i),i=1,5),sd(1),sd(2),irs
                        if (itt==2) &
                         write(iu,761)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                         ((id5(l,i),dp,i=4,5),id5(l,6),str2,l=1,3,2), &
                         (id5(5,i),dp,i=4,5),id5(5,6),sep,sl,pan(1), &
1675                     str3,pan(3),str3,pan(5),sd(1),sd(2),irs
                        if (itt==1) &
                         write(iu,771)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
                         str2,str2,(id5(3,i),dp,i=4,5),id5(3,6), &
                         str2,str2,sep,sl,str3,str3,pan(3), &
1680                     str3,str3,sd(1),sd(2),irs
                       endif
                       if (itt==i0.and.iu==6) inum(2) = inum(2) - 1
                      endif
                      if (isep<=2 .and.iu==6) then
1685                   if (ts=='m'.or.ts=='v') inum(3) = inum(3) + 1
                       if (ts=='C'.or.ts=='c') inum(4) = inum(4) + 1
                      endif
                     enddo
1690                else
                      ic = i0; iwo = i0; inum(2) = inum(2) - 1
                    endif
                   endif
                  endif
1695              endif
                  if (itransit==i0.or.ilin<=2) zjde0 = zjde
      !t          read(*,*)  !t
      ! . . . Ereignis ohne Transit-Pruefung (z.B. imod = 3), Output
              else
1700            do iu=ix,6,5
                  if (dabs(df(2))<1.d-4) then
                    write(iu,456)kon,' ',tra(1),tra(2),ik, &
                      zjde,zjahr,delt,(df(i),i=1,3),difm
                  elseif (dabs(df(3))<1.d-4) then
1705                write(iu,457)kon,' ',tra(1),tra(2),ik, &
                      zjde,zjahr,delt,(df(i),i=1,3),difm
                  else
                    write(iu,455)kon,' ',tra(1),tra(2),ik, &
                      zjde,zjahr,delt,(df(i),i=1,3),difm
1710              endif
                enddo
```



```
                call memo(zjde,zjahr,delt,df(1),df(2),df(3),difm,zmem, &
                  iak,imem)
              endif
1715        endif
      ! . . Ruecksprung fuer Transit-Pruefung
        370 if (itran==1 .and.ison==5 .and.ilin>=3) then
              if (itransit/=i0) zjde = zjde0
              if (ison==5 .and.ic==1 .and.ilin>=3) itransit=itransit+1
1720          if (itransit==i0.or.itransit>2 .or.ilin<=2) go to 380
              go to 320
            endif

      ! . . Bedingung fuer Zeitsprung zur Verkuerzung der Rechenzeit
1725    380 if (ilin>=3 .and.dwin<=21.d0) then
              iflag2 = iflag1; iflag1=i0; if (dabs(df(4))<=dwin) iflag1=1
            endif; ifitrun = i0

      ! . . Weiterer Output
1730        do iu=ix,6,5
              if (((ilin<=2 .and.(tra(1)/=' '.or.tra(2)/=' ').and. &
                  ((isep<=2 .or.(isep>=3 .and.itt/=0)).and. &
                  ida(3)>=izmin)).or.(ic==1 .and.ilin>=3)).and. &
                  io==2 .and.iwo==1) then
1735            if (imod/=3) then
                  call linie(iu,2+ipar); write(iu,57) (zmem(i),i=11,19)
                  write(iu,57) (zmem(i),i=20,22); call linie(iu,2)
                endif
                if (ic==1 .and.imod==3 .and.io==2) call linie(iu,2)
1740            immin = 1; if (imod==3) immin = 5
                immax = 4; if (ilin>=3) immax = 8
                if (immin<=immax) then
                  do i=immin,immax; ii = 6*i
                    call vsop3(lv,i,ix,ir,time,res); if (ir/=i0) go to 500
1745                zmem(25+ii) = res(1);  zmem(28+ii) = res(5)
                    zmem(26+ii) = res(2);  zmem(29+ii) = res(4)
                    zmem(27+ii) = res(3);  zmem(30+ii) = res(6)
                  enddo
                endif
1750            if (ilin<=2) then
                  if (ivers==3) then
                    write(iu,'('' ascending node (M/V/E/Ma): '',2f12.6, &
                     & ''      ---   '',f12.6)')zmem(34),zmem(40),zmem(52)
                  else
1755                write(iu,'('' ascending node (M/V/E/Ma): '',4f12.6)') &
                      (zmem(28+6*i),i=1,4)
                  endif
                  write(iu,'('' inclination i  (M/V/E/Ma): '',4f12.6)') &
                    (zmem(29+6*i),i=1,4)
1760              write(iu,'('' perihelion pi  (M/V/E/Ma): '',4f12.6)') &
                    (zmem(30+6*i),i=1,4)
                else
                  do i=31,78; re(i) = zmem(i); enddo
                  call elements(iu,ivers,pla)
1765            endif
                call linie(iu,1+ipar)
              endif
            enddo
        390 if (ikomb==1 .and.imod==2) then; imod = 1; dwi = dwi0; endif
1770
```

```
      ! . . Bedingter groesserer Zeitsprung
            if (ilin<=2 .or.(dwin<=21.d0 .and.((iflag2==1 .and.iflag1==i0) &
               .or.(ison==5 .and.ifitrun==i0.and.(ke==i0.or.ke==3))))) then
              zjde = zjde + tsprung; iflag1 = i0
1775        else
              zjde = zjdestep
              if (ison==5 .or.(ison/=5 .and.dabs(difm)>dwin*sz)) then
                stepl = difm*zfact + zstep
                if (ic==1) stepl = 0.9d0*ymer
1780            zjde = zjde + stepl
              else
                zjde = zjde + step
              endif
            endif; icv = ic
1785        if (zjde<=zjdemax) go to 310
      ! . . Ergaenzung (Tabellenkopf fuer Transit-Test mit inum(2)=0)
            if (ilin<=2 .and.inum(2)==0) then
              do iu=ix,6,5
                call zwizeile(iu,io,zmem(1),ilin,imod,isep,ical,izp)
1790          enddo
            endif

      !-----Ende der 3. Hauptschleife (Linearkonstellation, Transit)---------

1795  !=====================================================================
      !---------------------- Ende der Hauptschleifen ----------------------
      !=====================================================================

        400 do iu=ix,6,5; if (io/=2) call linie(iu,1+ipar); enddo
1800
      ! . . Ruecksprung bei Option -803 und Speichern von "inser-2.t"
            if (iop0==-803) then
              if (ilin==1) then
                ilin = 2; zmin = -30000; zmax = 30000
1805            go to 10
              endif
              call save_ser
            endif

1810  !-----Endzeilen
            call cpu_time(zib)
            call date_and_time(zdate,ztime,zzone,iw2) ! (--> threads)
            call comtime(1,zia,zib,iw1,iw2,ihour,imin,sec)    ! ( " )
            call comtime(2,zia,zib,iw1,iw2,ihour2,imin2,sec2) ! ( " )
1815        do iu=ix,6,5
              call endzeile(ipla,imod,ilin,iaph,isep,ison,ijd,ipos,iu, &
                inum,ihour,imin,sec,ihour2,imin2,sec2,is12,iop0) ! (threads)
      !h      if (ipla<=2.and.imod<=2.and.ison>=3) then                !h
      !h        write(iu,'(7x,a24,a33)') 'Frequency of deviations ', &  !h
1820  !h          & ' Fpos(0 to 5%) in steps of 0.05%:'                 !h
      !h        call linie(iu,1)                                        !h
      !h        do i=0,4                                                !h
      !h          write(iu,'(2(3x,10i3))') (ihis(j+i*20),j=1,20)        !h
      !h        enddo; call linie(iu,1); write(iu,*)                    !h
1825  !h      endif                                                     !h
            close(iu)
          enddo
        500 continue
```



```
1830  !-----Ende des Hauptprogramms-----------------------------------------
        stop
         54 format(1x,3f9.6,3(a1,f7.6),3f9.6)
         55 format(1x,a2,i7,f14.5,f10.3,f8.3,4f8.3,f6.1)
         56 format(1x,a2,i7,f15.5,f11.3,f9.3,4f8.3,a2)
1835     57 format(1x,3(f9.4,f8.4,f9.6))
         65 format(1x,a2,i7,f10.3,3f8.3,3f7.1,f5.1,a2,f7.3)
         67 format(1x,a2,i7,f10.3,3f8.3,2f7.1,a7,f5.1,a2,f7.3)
         85 format(1x,a2,i7,f10.3,3f8.3,3f7.2,f5.2,a2,f7.3)
        165 format(1x,a2,i7,f10.3,2f8.3,i3,i4,3f7.1,f6.1,a2,f7.3)
1840    184 format(1x,a2,i7,f10.3,2f8.3,i3,i4,3f7.2,f6.2,a2,f7.3)
        255 format(1x,f14.5,f7.1,f7.2,f7.3,f7.2,3f7.1,f6.1,a2,f7.3)
        256 format(1x,a2,i7,f10.3,f8.2,f7.3,f7.2,4f7.1,a2,f7.3)
        275 format(1x,f14.5,f7.1,f7.2,f7.3,f7.2,3f7.2,f6.2,a2,f7.3)
        276 format(1x,a2,i7,f10.3,f8.2,f7.3,f8.2,3f7.2,f6.2,a2,f7.3)
1845    365 format(1x,a3,i8,f13.3,f12.0,i6,1x,3f7.1,f5.1,a2,f7.3)
        366 format(1x,a3,i8,f13.3,f12.0,i2,i4,3f7.1,f6.1,a2,f7.3)
        367 format(1x,a3,i8,f13.3,f12.0,i6,1x,2f7.1,a7,f5.1,a2,f7.3)
        384 format(1x,a3,i8,f13.3,f12.0,i6,1x,3f7.2,f5.2,a2,f7.3)
        386 format(1x,a3,i8,f13.3,f12.0,i2,i4,3f7.2,f6.2,a2,f7.3)
1850    405 format(1x,a2,i7,f11.3,f8.3,f9.3,f9.4,f8.1,2f7.1,1x,a2,f7.3)
        406 format(1x,a2,i7,f11.3,f8.3,f9.3,f9.4,f8.2,2f7.2,1x,a2,f7.3)
        407 format(1x,a2,i7,f15.5,f11.3,i3,i4,f8.1,2f7.1,f6.2,a2,f6.3)
        408 format(1x,a2,i7,f15.5,f11.3,i3,i4,f8.2,2f7.2,f6.2,a2,f6.3)
        455 format(1x,a2,3a1,i7,f15.5,f11.3,5f8.3)
1855    456 format(1x,a2,3a1,i7,f15.5,f11.3,2f8.3,f6.1,f10.3,f8.3)
        457 format(1x,a2,3a1,i7,f15.5,f11.3,3f8.3,f6.1,f10.3)
        458 format(1x,a2,a1,i7,f5.0,a5,i6,i3,2(a1,i2),4f8.3,f7.1,i5)
        459 format(1x,a2,a1,f4.0,a5,i6,i3,2(a1,i2),4(i4,2(a1,i2)),f7.1,a1,i4)
        461 format(1x,a2,a1,f4.0,a5,i6,i3,2(a1,i2),2(a10,i4,2(a1,i2)), &
1860              f7.1,a1,i4)
        471 format(1x,a2,a1,f4.0,a5,i6,1x,a8,2x,a8,i4,2(a1,i2),2(2x,a8), &
                  f7.1,a1,i4)
        558 format(1x,a2,a1,i7,f5.0,a5,i5,i4,2(a1,i2),4f8.3,f7.1,i4)
        559 format(1x,a2,a1,f4.0,a5,i5,5(i4,2(a1,i2)),f7.1,a1,i3)
1865    561 format(1x,a2,a1,f4.0,a5,i5,i4,2(a1,i2),2(a10,i4,2(a1,i2)), &
                  f7.1,a1,i3)
        571 format(1x,a2,a1,f4.0,a5,i5,2a10,i4,2(a1,i2),2a10,f7.1,a1,i3)
        659 format(1x,a2,a1,f4.0,a5,i6,5(i4,a1,i2,a1,i2),f7.1,1x,a1, &
                  3x,5f8.2,4x,2f8.2,i6)
1870    661 format(1x,a2,a1,f4.0,a5,i6,i4,2(a1,i2),2(a10,i4,2(a1,i2)), &
                  f7.1,1x,a1,3x,f8.2,a8,f8.2,a8,f8.2,4x,2f8.2,i6)
        671 format(1x,a2,a1,f4.0,a5,i6,2a10,i4,2(a1,i2),2a10,f7.1,1x,a1, &
                  3x,2a8,f8.2,2a8,4x,2f8.2,i6)
        759 format(1x,a2,a1,f4.0,a5,i5,1x,5(i4,a1,i2,a1,i2),f7.1,1x,a1, &
1875              3x,5f8.2,4x,2f8.2,i6)
        761 format(1x,a2,a1,f4.0,a5,i5,1x,i4,2(a1,i2),2(a10,i4,2(a1,i2)), &
                  f7.1,1x,a1,3x,f8.2,a8,f8.2,a8,f8.2,4x,2f8.2,i6)
        771 format(1x,a2,a1,f4.0,a5,i5,1x,2a10,i4,2(a1,i2),2a10,f7.1,1x,a1, &
                  3x,2a8,f8.2,2a8,4x,2f8.2,i6)
1880
      ! . . Ausgabe einer groesseren Stellenanzahl zur Feinabstimmung bzw.
      !     Minimierung von F[%] fuer die Schnellstart-Optionen 4 und 9.
      !     Dies wurde verwendet fuer Buch 1.
      !     Suche in der Umgebung des Merkur-Aphels bzw. Merkur-Perihels
1885  !f255 format(1x,f14.5,f8.2,f7.2,f8.4,f6.1,3f7.1,f5.1,a2/65x,f14.8)
      !f275 format(1x,f14.5,f8.2,f7.2,f7.3,f7.2,3f7.2,f5.1,a2/65x,f14.8)
            end program P4_4
```

```
      subroutine inputdata(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
1890    itran,isep,iuniv,ical,ika,iaph,iamax,step,ison,ihi,irb,ijd, &
        zmin,zmax,ak,zjde1,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop0,iout)
!-----Inputdaten und Programmstart-----------------------------------
      implicit double precision (a-h,o-z)
      character(36) :: com
1895  data ita/0/ ! pre-init.
      iy = 6; ipla = 1; itran = 1
      io = 0; ire = 0; z0 = 0.d0
      write(iy,'(///27x,27(''-''))')
      write(iy,'(30x,''PLANETARY CORRELATION'')')
1900  write(iy,'(29x,''Program P4-4, June 2015'')')
      write(iy,'(27x,27(''-''))')

! . . Schnellstart-Menue
      write(iy,'(/7x,a16,9x,a18,7x,a16/5x,70a1/5(6x,2(a19,6x),a18/), &
1905  & 5x,70a1)') &
     'pyramids of Giza','chambers, Great P.','transits, syzygy', &
      ('-',i=1,70), &
     '3D Mer. at aph. (1)','3D Mer. at per. (6)','Mercury tr.   (11)',&
     '2D Mer. at aph. (2)','Keplers equ.    (7)','Venus tr.     (12)',&
1910 'const. 12, 3088 (3)','const. 12, 3088 (8)','syzygy, 3 pl. (13)',&
     '1.5 days, 3088  (4)','1.5 days, 3088  (9)','syzygy, 4 pl. (14)',&
     'near aphelion   (5)','F minimized    (10)','TYMT-test     (15)',&
      ('-',i=1,70)
      do
1915    do
          write(iy,'(8x,a10,3x,a20,3x,a26)',advance='no')'info (111)',&
           'detailed options (0)','(0..15 or book options) : '
          read(*,*,iostat=iox) iop0
          if (iox==0) exit
1920      call emes(ire,com,dm)
        enddo; iop=iop0
        if (iop==0) then; write(iy,*); go to 10; endif
        if (iop==111) then; call info; iout=4; return; endif

1925  ! . . . Verborgene Optionen fuer Tabellen aus beiden oben genannten
      !       Buechern, s.a. im Programmkopf unter "Neue Optionen, b)"
        if ((iop>=0 .and.iop<=15).or. &
      !       1. "Pyramiden und Planeten", Tab. 39-51
          (iop>=390 .and.iop<=392).or.(iop>=400 .and.iop<=402).or. &
1930      (iop>=410 .and.iop<=432).or.(iop>=440 .and.iop<=442).or. &
          iop==450 .or. &
          (iop>=460 .and.iop<=461).or.(iop>=470 .and.iop<=471).or. &
          (iop>=480 .and.iop<=481).or.(iop>=490 .and.iop<=492).or. &
          (iop>=500 .and.iop<=502).or.(iop>=510 .and.iop<=512).or. &
1935      (iop>=517 .and.iop<=519).or. &
      !       2. Buch 2, Tab. 17-38 ausser 34
          iop==170 .or. &
          iop==180 .or.(iop>=190 .and.iop<=192).or.iop==200 .or. &
          iop==210 .or.iop==220 .or.iop==230 .or.iop==240 .or. &
1940      iop==250 .or.iop==260 .or.iop==270 .or.iop==280 .or. &
          iop==290 .or.iop==300 .or.iop==310 .or.iop==320 .or. &
          iop==330 .or.iop==350 .or.iop==351 .or.iop==360 .or. &
          iop==361 .or.(iop>=370 .and.iop<=372).or.iop==380 .or. &
          iop==381 .or.iop==385 .or.iop==999 .or.iop==-803) exit
1945    ire = 1; call emes(ire,com,dm)
      enddo
```



```
! . . Auswertung der eingegebenen Option
      if (iop<0 .or.iop>15) then
1950    id = mod(iop,10); ita = (iop-id)/10

!       Buch 1 (Parameter fuer Datei "inparm.t")
        if (ita==39) iop = 16 + id
        if (ita==40) iop = 19 + id
1955    if (ita==41 .or.ita==42) then
          if (id<=6) iop = 22 + id
          if (id==7) iop = 3
          if (id>=8) iop = 21 + id
        endif
1960    if (ita==43) iop = 31 + id
        if (ita==44) iop = 23 + 3*id
        if (ita==45) iop = 2
        if (ita==46 .or.ita==47) iop = 34 + id
        if (ita==48) iop = 36 + id
1965    if (ita==49 .and.id==0) iop = 3
        if (ita==49 .and.id>=1) iop = 28 + id
        if (ita==50 .and.id==0) iop = 1
        if (ita==50 .and.id>=1) iop = 37 + id
        if (ita==51 .and.id<=2) iop = 40 + id
1970    if (ita==51 .and.id>=7) iop = 64 + id

!       Buch 2 (Parameter fuer Datei "inparm.t")
        if (ita==17 .or.ita==18) iop = 26 + ita
        if (ita==19) iop = 45 + id
1975    if (ita==20 .or.ita==21) iop = 28 + ita
        if (ita==22) iop = 50
        if (ita==23 .or.ita==24) iop = 28 + ita
        if (ita==25) iop = 8
        if (ita==26) iop = 3
1980    if (ita==27 .or.ita==28) iop = 26 + ita
        if (ita==29) iop = 14
        if (ita>=30 .and.ita<=33) iop = 25 + ita
        if (ita==35) iop = 59 + id
        if (ita==36) iop = 61 + id
1985    if (ita==37) iop = 63 + id ! Bei iop0=371, 372 s.a. "aphelko".
        if (ita==38 .and.id<=1) iop = 66 + id
        if (ita==38 .and.id==5) iop = 68
        if (iop0==-803) iop = 69 ! Erzeugung der Datei "inser-2.t"
      endif
1990
! . . Einlesen der Parameter aus "inparm.t"
      call inputfile(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
        itran,isep,iuniv,ical,ika,iaph,iamax,step,ison,ihi,irb,ijd, &
        zmin,zmax,ak,zjde1,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop,1,iout)
1995  return

!.....Menues fuer Einzeleingabe der Parameter.........................
! . . Planetenpositionen
   10 do
2000    write(iy,'(''  Constell.  pyr.(1),  chamb.(2),  lin.(3)  : ''&
          & )',advance='no')
        read(*,*,iostat=iox) ipla
        if (ipla>=1 .and.ipla<=3 .and.iox==0) exit
        call emes(ire,com,dm)
2005  enddo
```

```
      ! . . Linearkonstellation, Transite
            ilin = 4
            if (ipla==3) then
2010          do
               write(iy,'('' Tr. Mer.(1), Ven.(2), 3-co.(3), 4-co.(4)  : ''&
                 & )',advance='no')
               read(*,*,iostat=iox) ilin
               if (ilin>=1 .and.ilin<=4 .and.iox==0) exit
2015           call emes(ire,com,dm)
              enddo
            endif

      ! . . VSOP, Theorie-Variante
2020  !     Es erfolgt hier eine Aenderung des Parameters 'imod' (s.u.).
      !     Eingabe :  VSOP87  Kombi.(1), Kurzv.(2), Kepl.(3), Vollv.(4)
      !     intern  :  VSOP87  Kurzv.(1), Vollv.(2), Kepl.(3)
            ikomb = 0
            do
2025          if (ipla/=3) then
               write(iy,'('' VSOP87     combi. (1), short version (2),''/ &
                        &''         Kepl. equ. (3),  full version (4)  : ''&
                 & )',advance='no')
               read(*,*,iostat=iox) imod
2030           if (imod>=1 .and.imod<=4 .and.iox==0) exit
              else
               if (ilin>=3) then
                write(iy,'('' VSOP87  combi.(1), short v.(2), '', &
                  & ''Kepl.(3)  : '')',advance='no')
2035            read(*,*,iostat=iox) imod
                if (imod>=1 .and.imod<=3 .and.iox==0) exit
               else
                 write(iy,'('' VSOP87-version  full v.(1),  '', &
                  & ''short v.(2)  : '')',advance='no')
2040            read(*,*,iostat=iox) imod
                if (imod>=1 .and.imod<=2 .and.iox==0) exit
               endif
              endif
              call emes(ire,com,dm)
2045        enddo
      !     Aendern des Parameters "imod"
      !     (imo4 wird eingefuehrt, da imod wechselt, falls ikomb = 1 ist.)
            imo4 = 0
            if (imod==1) ikomb = 1
2050        if (imod==2) imod = 1
            if (imod==4) then
              imod = 2; imo4 = 1
            endif

2055  ! . . Version von VSOP87
      !     (Bei Transits u. J2000: geringe Abw. zu Meeus => keine Option
      !     bzw. ipla <= 2.)
            lv = 1; ivers = 3
            if (imod/=1 .or.(imod==1 .and.ikomb==1 .and.ipla<=2)) then
2060          do
               write(iy,'('' System   ecl. of epoch (1),  J2000.0 (2)  : ''&
                 & )',advance='no')
               read(*,*,iostat=iox) lv
               if ((lv==1 .or.lv==2).and.iox==0) exit
2065           call emes(ire,com,dm)
```



```
              enddo
              if (lv==2) ivers = 1
            endif

2070  ! . . Merkur- und Venustransite vor Sonne pruefen bei VSOP-Vollversion
      !     (Diese Option wird nicht mehr abgefragt, da nach Optimierung der
      !     VSOP87-Routine der Geschwindigkeitsvorteil durch Weglassen der
      !     Transit-Pruefung nur noch gering ist.)
      !c    if (ipla==3.and.ikomb==1.and.ilin>=3) then
2075  !c      do
      !c       write(iy,'('' Check planetary transit  yes (1), no (2)  : ''&
      !c         & )',advance='no')
      !c       read(*,*,iostat=iox) itran
      !c       if ((itran==1.or.itran==2).and.iox==0) exit
2080  !c       call emes(ire,com,dm)
      !c      enddo
      !c      if (itran==2) io = 1
      !c    endif

2085  ! . . Transit-Pruefung bei gleicher ekl. Laenge, minimaler Separation
      !     oder Berechnung der Phasen, optional mit Positionswinkeln
            isep = 1
            if (itran==1 .and.ilin<=2) then
              do
2090           write(iy,'('' Date  equ.L.(1), nearest (2), phases (3)''/ &
                        & ''            phases and position angles (4)  : ''&
                 & )',advance='no')
               read(*,*,iostat=iox) isep
               if (isep>=1 .and.isep<=4 .and.iox==0) exit
2095           call emes(ire,com,dm)
              enddo
            endif

      ! . . Julian/Gregorian calendar: Automatic choice of calender or
2100  !     only Gregorian calendar
            ical = 0
            do
              write(iy,'('' Calendar  only Greg. (1), Jul./Greg. (2)  : ''&
                & )',advance='no')
              read(*,*,iostat=iox) ical
2105          if ((ical==1 .or.ical==2).and.iox==0) exit
              call emes(ire,com,dm)
            enddo

2110  ! . . Terrestrial Time bzw. Universal Time
            iuniv = 1
            if (itran==1 .and.ilin<=2 .and.isep>=3) then
              do
               write(iy,'('' Time system         JDE/ TT (1),  UT (2)  : ''&
2115             & )',advance='no')
               read(*,*,iostat=iox) iuniv
               if ((iuniv==1 .or.iuniv==2).and.iox==0) exit
               call emes(ire,com,dm)
              enddo
2120        endif

      ! . . Zuordnung der Planeten Erde (E), Venus (V) und Merkur (M) zu
      !     Koenigs-, Koeniginnen- und Felsenkammer in dieser Reihenfolge
            ika = 0
```

```
      if (ipla==2 .and.imod/=3) then
        do
         write(iy,'('' Planets  E-V-M (1), E-M-V (2), V-E-M (3),''/ &
                & ''           V-M-E (4), M-E-V (5), M-V-E (6)  : ''&
           & )',advance='no')
         read(*,*,iostat=iox) ika
         if (ika>=1 .and.ika<=6 .and.iox==0) exit
         call emes(ire,com,dm)
        enddo
      endif

  ! . . Zeitpunkte im/um Aphel bzw. Perihel oder freier Zeitpunkt
      iaph = 1
      iamax = 0
      step = 24.d0
      if (ipla/=3) then
       do
        if (imod<=2 .and.ikomb==0 .and.imo4==0) then
         write(iy,'('' Passage aph./per. area of aph./per. free''/ &
                & ''          (1)  (2)          (3)  (4)   (5)  : ''&
           & )',advance='no')
         read(*,*,iostat=iox) iaph
         if (iaph>=1 .and.iaph<=5 .and.iox==0) exit
        elseif (imod<=2 .and.ikomb==1 .and.imo4==0) then
         write(iy,'('' Passage   aph. (1),  per. (2),  free (5)  : ''&
           & )',advance='no')
         read(*,*,iostat=iox) iaph
         if ((iaph==1 .or.iaph==2 .or.iaph==5).and.iox==0) exit
        elseif (imod<=2 .and.ikomb==0 .and.imo4==1) then
         write(iy,'('' Passage    aph./ per.  area of aph./ per.''/ &
                & ''             (1)   (2)          (3)   (4)  : ''&
           & )',advance='no')
         read(*,*,iostat=iox) iaph
         if (iaph>=1 .and.iaph<=4 .and.iox==0) exit
        else
         write(iy,'('' Passage    aphelion (1),  perihelion (2)  : ''&
           & )',advance='no')
         read(*,*,iostat=iox) iaph
         if ((iaph==1 .or.iaph==2).and.iox==0) exit
        endif
        call emes(ire,com,dm)
       enddo
       if (iaph==3 .or.iaph==4) then
        do
         write(iy,'('' Steps per Mercury passage : '')',advance='no')
         read(*,*,iostat=iox) iamax
         if (iamax>0 .and.iamax<=200000 .and.iox==0) exit
         call emes(ire,com,dm)
        enddo
        do
         write(iy,'('' Step width  (hours, real) : '')',advance='no')
         read(*,*,iostat=iox) step
         if (step>z0.and.iox==0) exit
         call emes(ire,com,dm)
        enddo
        if (imod==2) io = 1
       endif
      endif
```



```
  ! . . Sonnenposition
      ison = 1
      if (ipla/=3) then
       do
        if (ipla==1 .and.iaph<=2) then
         if (imod<=2) then
         write(iy,'('' Sun pos.  Myk.(1),  Chefr.(2),  free (3)  : ''&
          & )',advance='no')
         else
         write(iy,'('' Sun pos. south of  Myk.(1),  Chefr.(2)  : ''&
          & )',advance='no')
         endif
         read(*,*,iostat=iox) ison
        else
         if (imod<=2) ison = 3
        endif
        if (((imod<=2 .and.ison>=1 .and.ison<=3).or. &
            (imod==3 .and.(ison==1 .or.ison==2))).and.iox==0) exit
        call emes(ire,com,dm)
       enddo
      endif

  ! . . Freie Sonnenposition, Berechnung 2- oder 3-dimensional
      if (iaph==5) ison = 5
      if (ison==3) then
       do
        if (ipla==1) then
         write(iy,'('' Sun     2D (1), 3D/SLE (2), 3D/FITEX (3)  : ''&
           & )',advance='no')
        else
         write(iy,'('' Sun (three-dim.):    SLE (2),  FITEX (3)  : ''&
           & )',advance='no')
        endif
        read(*,*,iostat=iox) ison2
        if (((ipla==1 .and.ison2>=1 .and.ison2<=3).or. &
            (ipla==2 .and.(ison2==2 .or.ison2==3))).and.iox==0) exit
        call emes(ire,com,dm)
       enddo
       if (ison2==2) ison = 4
       if (ison2==3) ison = 5
      endif

  ! . . Hoehenlage der Pyramiden-Grundflaechen bzw. der -Schwerpunkte
      ihi = 0
      if (ipla/=3 .and.ison>=4) then
       do
        if (ipla==1) then
         write(iy,'('' z-coord.    base (1),  C-M (2),  top (3)  : ''&
           & )',advance='no')
        else
         write(iy,'('' Wall      east (1), middle (2), west (3)  : ''&
           & )',advance='no')
        endif
        read(*,*,iostat=iox) ihi
        if (ihi>=1 .and.ihi<=3 .and.iox==0) exit
        call emes(ire,com,dm)
       enddo
      endif
```

```
      ! . . Grundebene Ekliptik, Merkur- oder Venusbahn
            irb = 1
2245        if (ipla/=3 .and.imod<=2 .and.ison==1) then
             do
              write(iy,'('' Coord.  ecl.(1),  Mer.(2-4),  Ven.(5)  : '' &
                & )',advance='no')
              read(*,*,iostat=iox) irb
2250          if (irb>=1 .and.irb<=5 .and.iox==0) exit
              call emes(ire,com,dm)
             enddo
            endif

2255  ! . . Angabe bzw. Berechnung von JDE
            ijd = 15
            if (ipla/=3 .and.ikomb==0 .and.iaph/=5) then
             do
              if (imod==2 .and.iaph<=2) then
2260           write(iy,'('' Constell. (1..14),  k-No. (15),  JDE (0)  : ''&
                & )',advance='no')
              else
               write(iy,'('' Constell. (1..14),  years (15),  JDE (0)  : ''&
                & )',advance='no')
2265          endif
              read(*,*,iostat=iox) ijd
              if (ijd>=0 .and.ijd<=15 .and.iox==0) exit
              call emes(ire,com,dm)
             enddo
2270        endif
            ak = z0
            zmin = z0
            zmax = z0
            if (ijd==15) then
2275          if (imod==2 .and.iaph<=2 .and.ipla/=3) then
                do
                 write(iy,'('' k (real): '')',advance='no')
                 call pcheck(1,ak,2,dm,imod,ire)
                 if (ire==0) exit
2280            enddo
              else
                do
                 write(iy,'('' from  year (real): '')',advance='no')
                 call pcheck(1,zmin,1,dm,imod,ire)
2285             if (ire==0) exit
                enddo
                do
                 write(iy,'('' until year (real): '')',advance='no')
                 call pcheck(1,zmax,1,dm,imod,ire)
2290             if (zmin>=zmax.and.ire==0) then
                  call emes(ire,com,dm)
                  ire = 1
                 endif
                 if (ire==0) exit
2295            enddo
              endif
            endif
            if (ipla==3) then
             step = z0
             if (ilin>=3 .and.ikomb==0) then
              do
```



```
               write(iy,'('' Step width [hrs] (min.-search 0.) (real)  : ''&
                & )',advance='no')
               read(*,*,iostat=iox) step
2305           if (step>=z0.and.iox==0) exit
               call emes(ire,com,dm)
              enddo
             endif
            endif
2310        if (step==z0) ison = 5
            if (ipla==3 .and.step/=z0) io = 1
            zjde1 = z0
            if (ijd==0) then
              do
2315            write(iy,'('' JDE (real) : '')',advance='no')
                call pcheck(1,zjde1,3,dm,imod,ire)
                if (ire==0) exit
              enddo
            endif
2320
      ! . . Winkelintervall bzw. relativer Fehler
            dwi = z0
            dwi2 = z0
            dwi3 = z0
2325        dwikomb = z0; dm = 99.99d0
            if (ipla/=3 .and.ijd==15 .and.(imod/=2 .or. &
               (imod==2 .and.(iaph==3 .or.iaph==4)))) then
             if (ikomb==0 .and.iaph/=5) then
              do
2330            if (ison<=2) then
                 if (imod/=3) dm = 10.d0
                 write(iy,'('' Tolerance ecl. long. Venus, Earth (real)'', &
                   & ''  : '')',advance='no')
                else
2335             write(iy,'('' Max. F-pos at aphelion/ per.  (real) [%]'', &
                   & ''  : '')',advance='no')
                endif
                call pcheck(2,dwi,1,dm,imod,ire)
                if (ire==0) exit
2340            enddo
              else
               do
                if (ison<=2) then
                 if (imod/=3) dm = 10.d0
2345             write(iy,'('' Tolerance ecl. long. VSOP short   (real)'', &
                   & ''  : '')',advance='no')
                else
                 if (iaph/=5 .or.(iaph==5 .and.ikomb==1)) then
                  write(iy,'('' Max. F-pos    VSOP short ver. (real) [%]'', &
2350                & ''  : '')',advance='no')
                 else
                  write(iy,'('' Max. F-pos, VSOP short, start fitmin [%]'', &
                    & ''  : '')',advance='no')
                 endif
2355            endif
                call pcheck(2,dwi,1,dm,imod,ire)
                if (ire==0) exit
               enddo
               do
2360            if (ison<=2) then
```

```
            if (imod/=3) dm = 10.d0
            write(iy,'('' "  "  "  VSOP full (real)'', &
              & ''  : '')',advance='no')
           else
2365        if (iaph/=5 .or.(iaph==5 .and.ikomb==1)) then
            write(iy,'('' "  "  VSOP full ver. (real) [%]'', &
              & ''  : '')',advance='no')
           else
            write(iy,'('' "  "  VSOP short, final range [%]'', &
2370          & ''  : '')',advance='no')
           endif
          endif
          call pcheck(2,dwikomb,1,dm,imod,ire)
          if (ire==0) exit
2375     enddo
         endif
         if (iaph==3 .or.iaph==4) then
          do
           write(iy,'('' "  "  consider without printing [%]'', &
2380        & ''  : '')',advance='no')
           call pcheck(2,dwi2,1,dm,imod,ire)
           if (ire==0) exit
          enddo
          do
2385       write(iy,'('' "  "  print beyond aphelion/per.[%]'', &
            & ''  : '')',advance='no')
           call pcheck(2,dwi3,1,dm,imod,ire)
           if (ire==0) exit
          enddo
2390     endif
        endif
        if (ipla==3 .and.ilin>=3) then
         if (ikomb==0) then
          do
2395       write(iy,'('' Ang. range of eclipt. longitude (real)'', &
             & ''  : '')',advance='no')
           call pcheck(2,dwi,1,dm,imod,ire)
           if (ire==0) exit
          enddo
2400     else
          do
           write(iy,'('' Ecl. angular range, VSOP short v. (real)'', &
             & ''  : '')',advance='no')
           call pcheck(2,dwi,1,dm,imod,ire)
2405       if (ire==0) exit
          enddo
          do
           write(iy,'('' "  "  "  , VSOP full v. (real)'', &
             & ''  : '')',advance='no')
2410       call pcheck(2,dwikomb,1,dm,imod,ire)
           if (ire==0) exit
          enddo
         endif
        endif
2415
    ! . . Dreier- oder Viererkonjunktion nur mit Transit
        nurtr = 1
        if (ipla==3 .and.ilin>=3 .and.ison==5 .and.imod/=3 &
            .and.itran==1) then
```



```
2420        do
             write(iy,'('' All conjunctions (1), only transits (2)'', &
               & ''  : '')',advance='no')
             read(*,*,iostat=iox) nurtr
             if ((nurtr==1 .or.nurtr==2).and.iox==0) exit
2425         call emes(ire,com,dm)
            enddo
           endif

    ! . . Blickrichtung auf die Planetenbahnen
2430       iek = 1
           if (ipla/=3) then
            do
             if (ison<=2 .and.(ijd==15 .or.ijd==0)) then
              if ((imod==2 .and.iaph<=2).or.ijd==0) then
2435           write(iy,'('' View from ecliptic North (1), South (2)'', &
                 & ''  : '')',advance='no')
               read(*,*,iostat=iox) iek
               if (iek>=1 .and.iek<=2 .and.iox==0) exit
              else
2440           write(iy,'('' View from eclipt. N (1), S (2), N/S (3)'', &
                 & ''  : '')',advance='no')
               read(*,*,iostat=iox) iek
               if (iek>=1 .and.iek<=3 .and.iox==0) exit
              endif
2445          call emes(ire,com,dm)
             else
              iek = 1
              if ((ijd>=6 .and.ijd<=11).or.ijd==13 .or.ijd==14) iek=2; exit
             endif
2450        enddo
           endif

    ! . . Ausgabe
           if (io==0) then
2455         io = 2; if (iaph==5) io = 1
             if (imo4==0 .and.iaph/=5) then
               do
                write(iy,'('' Output normal (1), extended (2)'', &
                  & ''  : '')',advance='no')
2460            read(*,*,iostat=iox) io
                if ((io/=2 .or.io==2).and.iox==0) exit
                call emes(ire,com,dm)
               enddo
             endif
2465       endif

    ! . . Ausgabegeraet
           do
             if (imod<=2 .and.ipla<=2 .and.ison==5) then
2470           write(iy,'('' Mon.(1), file (2), special (3), exit (4)'', &
                 & ''  : '')',advance='no')
               read(*,*,iostat=iox) iout
               if (iout>=1 .and.iout<=4 .and.iox==0) exit
             else
2475           write(iy,'('' Monitor (1), mon. + file (2), exit (4)'', &
                 & ''  : '')',advance='no')
               read(*,*,iostat=iox) iout
               if ((iout==1 .or.iout==2 .or.iout==4).and.iox==0) exit
```

```
            endif
            call emes(ire,com,dm)
          enddo
          end subroutine

          subroutine inputfile(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
           itran,isep,iuniv,ical,ika,iaph,iamax,step,ison,ihi,irb,ijd, &
           zmin,zmax,ak,zjde1,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop,irw,iout)
      !-----Einlesen der Inputdaten bei Schnellstart-------------------------
      !     irw=1: lesen aus "inparm.t",  irw=2: schreiben in "inedit.t"
      !     Mit Hilfe von inedit.t kann inparm.t manuell editiert werden.
          implicit double precision (a-h,o-z)
          if (irw==1) then
            if (iop/=999) then
              open(unit=10,file='inparm.t')
              do i=1,10*iop+1; read(10,*); enddo
            else
              open(unit=10,file='inedit.t')
              do i=1,24; read(10,*); enddo
            endif
            read(10,*) ipla,ilin,imod,imo4,ikomb
            read(10,*) lv,ivers,itran,isep,iuniv
            read(10,*) ical,ika,iaph,iamax,step
            read(10,*) ison,ihi,irb,ijd
            read(10,*) zmin,zmax,ak,zjde1
            read(10,*) dwi,dwikomb,dwi2,dwi3
            read(10,*) nurtr,iek,io,iout
          elseif (irw==2) then
            open(unit=10,file='inedit.t')
            do i=1,34; read(10,*); enddo
            write(10,'(5i3)') ipla,ilin,imod,imo4,ikomb
            write(10,'(5i3)') lv,ivers,itran,isep,iuniv
            write(10,'(3i3,i6,f10.5)') ical,ika,iaph,iamax,step
            write(10,'(3i3,i4)') ison,ihi,irb,ijd
            write(10,'(3f13.5,f15.5)') zmin,zmax,ak,zjde1
            write(10,'(4f8.3)') dwi,dwikomb,dwi2,dwi3
            write(10,'(4i3)') nurtr,iek,io,iout
            write(10,*) ('-',i=1,59)
            write(10,*) ('*',i=1,27),' END ',('*',i=1,27)
          endif
          close(10)
          end subroutine

          subroutine chambers(ig,rx)
      !-----Aenderung der Planeten-Kammer-Zuordnung--------------------------
      !     Reihenfolge Koenigs-, Koeniginnen- u. Felsenkammer mit Planeten:
      !     ig: 1. E-V-M, 2. E-M-V, 3. V-E-M, 4. V-M-E, 5. M-E-V, 6. M-V-E
          implicit double precision (a-h,o-z)
          dimension :: rx(3,4),x(5),y(5)
          if (ig==3 .or.ig==5) call pchange(1,1,2,rx,x,y,indx)
          if (ig==2 .or.ig==4 .or.ig==5) call pchange(1,2,3,rx,x,y,indx)
          if (ig==4) call pchange(1,1,2,rx,x,y,indx)
          if (ig==6) call pchange(1,1,3,rx,x,y,indx)
          end subroutine

          subroutine pchange(imodus,iz,jz,rxx,x,y,indx)
      !-----Vertauschen von Input-Zeilen oder Zahlen in "fitmin"-------------
          implicit double precision (a-h,o-z)
          dimension :: rxx(3,4),x(5),y(5)
```



```
          if (imodus==1) then; do i=1,4
            rpc=rxx(iz,i); rxx(iz,i)=rxx(jz,i); rxx(jz,i)=rpc; enddo
          elseif (imodus==2) then
            z=x(iz); x(iz)=x(jz); x(jz)=z; z=y(iz); y(iz)=y(jz); y(jz)=z
            if (indx==iz) then; indx = jz; return; endif
            if (indx==jz) indx = iz
          endif
          end subroutine

          subroutine pcheck(i,p,n,dm,imod,ire)
      !-----Read and check of input parameter p------------------------------
      !     modus i: read + check time (1), tolerance (2)
      !     time  n: year (1), k-number (2), JDE (3)
      !     p: input parameter, dm: maximum allowed value
      !     error code ire (ire = 0 means "no error".)
          implicit double precision (a-h,o-z)
          character(36) :: com
          ire = 0; read(*,*,iostat=iox) p; if (iox/=0) ire = 1
          if (i==1 .and.ire==0) then
            ire = 2
            if (imod/=3) then
              if (n==1 .and.(p<-13000.00001d0 .or.p>17000.00001d0)) then
                com = ' (-13 000. <= year <= 17 000.)       '
              elseif (n==2 .and.(p<-63000.001d0 .or.p>63000.001d0)) then
                com = ' (-63 000. <=  k  <=  63 000.)       '
              elseif (n==3 .and.(p<-3030000.1d0 .or.p>7940000.1d0)) then
                com = ' (-3 030 000. <= JDE <= 7 940 000.) '
              else
                ire = 0
              endif
            else
              if (n==1 .and.(p<-30000.00001d0 .or.p>30000.00001d0)) then
                com = ' (-30 000. <= year <= 30 000.)       '
              elseif (n==2 .and.(p<-133000.01d0 .or.p>117000.01d0)) then
                com = ' (-133 000. <=  k  <=  117 000.)     '
              elseif (n==3 .and.(p<-9240000.1d0 .or.p>12680000.1d0)) then
                com = ' (-9 240 000. <= JDE <= 12 680 000.)'
              else
                ire = 0
              endif
            endif
          elseif (i==2 .and.ire==0) then
            if (p<=0.d0) ire = 1; if (p>dm) ire = 3
          endif
          if (ire/=0) call emes(ire,com,dm)
          end subroutine

          subroutine emes(ire,com,dm)
      !-----Error message---------------------------------------------------
          implicit double precision (a-h,o-z)
          character(36) :: com
          iy = 6
          if (ire<=1) write(iy,'(/'' ----> incorrect input.''/)')
          if (ire==2) write(iy,'(/'' ----> incorrect input. '', &
            & a36/)')com
          if (ire==3) write(iy,'(/'' ----> number too large '', &
            & ''(max.'',f6.2,'').''/)') dm
          end subroutine
```

```
      subroutine konst(ik,kon)
!-----Automatische Erkennung der Planetenkonst. 1 bis 14 --> kon-------
!     Suchtoleranz (+/-) fuer Konst.: 53 Tage, fuer "->": 880 Tage
      use base, only : akon
      implicit double precision (a-h,o-z)
      character(2) :: kon,tkon(14)
      data tkon/' 1',' 2',' 3',' 4',' 5',' 6',' 7', &
                ' 8',' 9','10','11','12','13','14'/
      ye = 10.d0; kon = '  '
      ep = 0.6d0; ako = dfloat(ik)
      do i=1,14
        a1 = dabs(ako-akon(i))
        a2 = dabs(ako-(akon(i)-1.d0))
        if (a1<ye.or.a2<ye) kon = '->'
        if (a1<ep.or.a2<ep) kon = tkon(i)
      enddo
      end subroutine

      subroutine ephim(i,iaph,ipla,ical,ak,iak,day,year,delt)
!-----Julian Ephemeris Day and Year (Merkur im Aphel)------------------
!     Input ist "ak" (Nummer des Apheldurchgangs), "day" oder "year".
!     i = 0:  ak   --> day, year, delt
!     i = 1:  day  --> ak, iak, year, delt
!     i = 2:  year --> day, ak, iak
      implicit double precision (a-h,o-z)
      if (i==0) call akday(0,iaph,ipla,ak,iak,day)

! . . Neue Werte (Buch 2)
!     Diese Zahlen verbessern nur die Genauigkeit der dezimalen Jah-
!     reszahl auf +/- 0,5 Tage, aendern jedoch nichts an den bishe-
!     rigen astronomischen Berechnungen und Datumsberechnungen. Alle
!     durch 400 teilbaren Jahreszahlen, wie z.B. -1200.0 oder 2000.0,
!     entsprechen jetzt exakt dem 1. Januar, 12 Uhr. Das heisst, das
!     dezimale Jahr 2000.0 bedeutet die Standard-Epoche J2000.0.
      if (ical==2 .and.((i<=1 .and.day>=0.d0 .and.day<2299160.5d0) &
       .or.(i==2 .and.year>=-4712.d0 .and.year<1582.7854097d0))) then
        A = 365.25d0; B = 0.d0; C =-4712.d0  ! (Julian. Kal.)
      else
        A = 365.2425d0; B = 2451545.d0; C = 2000.d0  ! (Gregor. Kal.)
      endif
! . . Vorherige Werte (Programm P3, Buch 1)
!c      A = 365.248d0; B = 0.d0; C = -4711.9986d0  ! (Programm P3)

! . . Umrechnung der Daten
      if (i<=1) year = (day - B)/A + C
      if (i==1) call akday(1,iaph,ipla,ak,iak,day)
      if (i<=1) then
        call akday(0,iaph,ipla,dnint(ak),iak,aiday)
        delt = day - aiday
      else
        day = A * (year - C) + B
        call akday(1,iaph,ipla,ak,iak,day)
      endif
      end subroutine

      subroutine akday(j,iaph,ipla,ak,iak,day)
!-----Julian Ephemeris Day---------------------------------------------
!     j = 0:  ak  --> day
!     j = 1:  day --> ak,iak
```



```
!     ymer = Umlaufzeit des Merkur in Tagen
      use base, only : pmer,ymer
      implicit double precision (a-h,o-z)
      if (j==0) then
        aak = ak
        if (iaph==1 .or.iaph==3 .or.(iaph==5 .and.ipla==1)) &
          aak = aak - 0.5d0
        day = pmer + ymer * aak
      endif
      if (j==1) then
        ak = (day - pmer)/ymer
        if (iaph==1 .or.iaph==3 .or.(iaph==5 .and.ipla==1)) &
          ak = ak + 0.5d0
        iak = idnint(ak)
      endif
! . . Apheldurchgang der Erde
!c    day = 2451547.507d0 + 365.2596358d0 * (ak + 0.5d0) &
!c          + 1.58d-8 * (ak + 0.5d0)**2
      end subroutine

      subroutine delta_T(zjd)
!-----Umrechnung: Terrestrial Time --> Universal Time------------------
!     Gleichungen von Fred Espenak und Jean Meeus, entwickelt auf Ba-
!     sis des "Five Millennium Canon of Solar Eclipses", nach Artikeln
!     von Morrison/Stephenson (2004) und Stephenson/Houlden (1986).
!     (NASA Eclipse Web Site, Polynomial expressions for DELTA-T, 2005)
      implicit double precision (a-h,o-z)
      call ephim(1,1,1,1,ak,iak,zjd,y,delt)
      if (y>-500.d0 .and.y<=500.d0) then
         u = y/100.d0
         del = 10583.6d0 - 1014.41d0 * u + 33.78311d0  * u**2 &
               - 5.952053d0 * u**3 - 0.1798452d0 * u**4 &
               + 0.022174192d0 * u**5 + 0.0090316521d0 * u**6
      elseif (y>500.d0 .and.y<=1600.d0) then
         u = (y-1000.d0)/100.d0
         del = 1574.2d0 - 556.01d0 * u + 71.23472d0 * u**2 &
               + 0.319781d0 * u**3 - 0.8503463d0 * u**4 &
               - 0.005050998d0 * u**5 + 0.0083572073d0 * u**6
      elseif (y>1600.d0 .and.y<=1700.d0) then
         t = y - 1600.d0
         del = 120.d0 - 0.9808d0 * t - 0.01532d0 * t**2 &
               + t**3 / 7129.d0
      elseif (y>1700.d0 .and.y<=1800.d0) then
         t = y - 1700.d0
         del = 8.83d0 + 0.1603d0 * t - 0.0059285d0 * t**2 &
               + 0.00013336d0 * t**3 - t**4 / 1174000.d0
      elseif (y>1800.d0 .and.y<=1860.d0) then
         t = y - 1800.d0
         del = 13.72d0 - 0.332447d0 * t + 0.0068612d0 * t**2 &
               + 0.0041116d0 * t**3 - 0.00037436d0 * t**4 &
               + 0.0000121272d0 * t**5 - 0.0000001699d0 * t**6 &
               + 0.000000000875d0 * t**7
      elseif (y>1860.d0 .and.y<=1900.d0) then
         t = y - 1860.d0
         del = 7.62d0 + 0.5737d0 * t - 0.251754d0 * t**2 &
               + 0.01680668d0 * t**3 - 0.0004473624d0 * t**4 &
               + t**5/233174.d0
      elseif (y>1900.d0 .and.y<=1920.d0) then
         t = y - 1900.d0
```

```
2715           del = -2.79d0 + 1.494119d0 * t - 0.0598939d0 * t**2 &
                     + 0.0061966d0 * t**3 - 0.000197d0 * t**4
               elseif (y>1920.d0 .and.y<=1941.d0) then
                  t = y - 1920.d0
                  del = 21.20d0 + 0.84493d0 * t - 0.076100d0 * t**2 &
2720                 + 0.0020936d0 * t**3
               elseif (y>1941.d0 .and.y<=1961.d0) then
                  t = y - 1950.d0
                  del = 29.07d0 + 0.407d0 * t - t**2/233.d0 + t**3/2547.d0
               elseif (y>1961.d0 .and.y<=1986.d0) then
2725              t = y - 1975.d0
                  del = 45.45d0 + 1.067d0 * t - t**2/260.d0 - t**3/718.d0
               elseif (y>1986.d0 .and.y<=2005.d0) then
                  t = y - 2000.d0
                  del = 63.86d0 + 0.3345d0 * t - 0.060374d0 * t**2 &
2730                 + 0.0017275d0 * t**3 + 0.000651814d0 * t**4 &
                     + 0.000002373599d0 * t**5
               elseif (y>2005.d0 .and.y<=2050.d0) then
                  t = y - 2000.d0
                  del = 62.92d0 + 0.32217d0 * t + 0.005589d0 * t**2
2735           elseif (y>2050.d0 .and.y<=2150.d0) then
                  del = -20.d0 + 32.d0 * ((y-1820.d0)/100.d0)**2 &
                     - 0.5628d0 * (2150.d0 - y)
               else
                  u = (y - 1820.d0)/100.d0
2740              del = -20.d0 + 32.d0 * u**2
               endif
               zjd = zjd - del/86400.d0  ! DELTA-T (del) in Sekunden

         ! . . Alternativ: Jean Meeus, "Transits", S. 73, der wiederum fol-
2745     !     gende Referenz zitiert: L.V. Morrison, F.R. Stephenson, Sun
         !     and Planetary System, Vol. 96, Reidel, Dordrecht, 1982, S. 73
         !c    zjd = zjd - ((zjd-2382148.d0)**2/41048480.d0 - 15.d0)/86400.d0
               end subroutine

2750           subroutine jdedate(zjd,ical,ida,da,dmo)
         !-----Berechnung von Datum und Uhrzeit (TT)----------------------------
         !     Programm zur Umrechnung von Julian Day in ein Kalenderdatum.
         !     Es basiert auf einem Algorithmus aus dem Buch von Jean Meeus:
         !     "Astronomical Algorithms", Copyright: 1991, Willmann-Bell,
2755     !     Inc., P.O.Box 35025, Richmond, Virginia 23235, USA (S. 63).
         !     Anmerkung: Der Algorithmus wurde geringfuegig modifiziert,
         !     so dass er jetzt fuer beide Kalender auch fuer JDE < 0 gilt.
         !     Indizes:
         !     1: dez.Tag, 2: Mon., 3: Jahr, 4: Std, 5: Min, 6: Sek, 7: int.Tag
2760           implicit double precision (A-H,O-Z)
               dimension :: ida(7),da(7)
               character(5) :: monat(12),dmo
               data monat/' Jan.',' Feb.',' Mar.',' Apr.',' May ',' June', &
                          ' July',' Aug.',' Sep.',' Oct.',' Nov.',' Dec.'/
2765           Z = sdint(zjd + 0.5d0)
               F = zjd + 0.5d0 - Z
               if (z>=0.d0 .and.Z<2299161.d0 .and.ical==2) then
                 A = Z
               else
2770             alpha = sdint((Z - 1867216.25d0)/36524.25)
                 A = Z + 1.d0 + alpha - sdint(alpha*0.25d0)
               endif
               B = A + 1524.d0
```



```
               C = sdint((B - 122.1d0)/365.25d0)
2775           D = sdint(365.25d0 * C)
               E = sdint((B - D)/30.6001d0)
               da(1) = B - D - sdint(30.6001d0*E) + F + 5.d-9
               if (E<14.d0) then
                 da(2) = E - 1.d0
2780           else
                 if (E==14.d0 .or.E==15.d0) then
                   da(2) = E - 13.d0
                 else
                   da(2) = 999.d0
2785             endif
               endif
               M = idnint(da(2))
               if (M>2) then
                  da(3) = C - 4716.d0
2790           else
                 if (M==1 .or.M==2) then
                   da(3) = C - 4715.d0
                 else
                   da(3) = 9999999999999.d0
2795             endif
               endif
               st  = da(1) - sdint(da(1))
               dst = st*24.D0
               da(4) = sdint(dst)
2800           da(5) = (dst - sdint(dst))*60.D0
               da(6) = (da(5) - sdint(da(5)))*60.D0
               da(7) = sdint(da(1))                    ! day
               ida(3) = idnint(da(3))                  ! year
               ida(4) = idnint(da(4))                  ! hours
2805           ida(5) = idnint(da(5)-0.5d0+1.d-10)     ! minutes
               ida(6) = idnint(da(6))                  ! seconds
               imo    = idnint(da(2))                  ! month

         !     Geringfuegige Korrektur der Darstellung
2810     !     (Beispiel: Uhrzeit 13:44:60 wird zu 13:45:00)
               do i=6,5,-1
                 if (ida(i)>=60) then
                   ida(i) = ida(i) - 60
                   ida(i-1) = ida(i-1) + 1
2815             endif
               enddo
               if (ida(4)>=24) then
                 ida(4) = ida(4) - 24
                 da(1) = da(1) + 1.d0
2820             da(7) = sdint(da(1))
               endif
               if ((dabs(da(7)-32.d0)<=1.d-8.and.(imo==1.or.imo==3 &
                  .or.imo==5.or.imo==7.or.imo==8.or.imo==10.or.imo==12)).or. &
                   (dabs(da(7)-31.d0)<=1.d-8.and.(imo==4.or.imo==6.or.imo==9 &
2825               .or.imo==11)).or.(dabs(da(7)-30.d0)<=1.d-8.and.imo==2)) then
                 do k=30,32
                   q=dfloat(k)
                   if (dabs(da(7)-q)<=1.d-8) da(1) = da(1)+1.d0-q
                 enddo
2830             da(7) = sdint(da(1)); imo = imo + 1
                 if (imo==13) then
                   imo = 1
```

```
              da(3) = da(3) + 1.d0
              ida(3) = idnint(da(3))
2835        endif
          endif
          dmo = monat(imo)
          end subroutine

2840      double precision function sdint(x)
     !-----Step function-----------------------------------------------------
     !     replacing some integer-functions in the subroutine "jdedate"
     !     in order to expand the domain of definition for JDE < 0
          real(8) :: x
2845      sdint = dint(x)
          if (x<0.d0 .and.dmod(x,1.d0)/=0.d0) sdint = sdint - 1
          end function

          subroutine weekday(ZJD,wd)
2850 !-----Berechnung des Wochentages-----------------------------------------
          implicit double precision(a-h,o-z)
          character(10) :: wday(0:6),wd
          data wday/'    Sunday','    Monday','   Tuesday',' Wednesday', &
                    '  Thursday','    Friday','  Saturday'/
2855      wd = wday(idnint(dmod(dint(ZJD + 700000001.5d0),7.d0)))
          end subroutine

          subroutine vsop1(l,tau,resu)
     !-----Berechnung der ekliptikalen Koordinaten (VSOP87D-Kurzversion)----
2860      use base, only : gdpi,z0,lmax,jp; use astro, only : par1
          implicit double precision (a-h,o-z)
          resu = z0
          do j=1,lmax(l)
            sum0 = z0
2865        do i=1,jp(l,j)
              sum0 = sum0 + par1(1,i,j,l) * &
                     dcos(par1(2,i,j,l) + par1(3,i,j,l)*tau)
            enddo
            resu = resu + sum0*tau**(j-1)
2870      enddo
          resu = resu * 1.d-8
          if (l==1 .or.l==4 .or.l==7 .or.l==10) call reduz(resu,1,1)
          if (l/=3 .and.l/=6 .and.l/=9 .and.l/=12) resu = resu*gdpi
          end subroutine
2875
          subroutine vsop2(zjde,ivers,ibody,md,ix,prec,lu,r,ierr,rku)
     !-----Aufruf der VSOP-Subroutine (VSOP87A/C-Vollversionen)-------------
     !    (Index von rku  1: L,  2: B,  3: r)
          implicit double precision (a-h,o-z)
2880      dimension :: r(6),rku(3),md(0:9)
          character(11) :: afile(9),cfile(8)
          data afile/    'VSOP87A.mer','VSOP87A.ven','VSOP87A.ear', &
           'VSOP87A.mar','VSOP87A.jup','VSOP87A.sat','VSOP87A.ura', &
           'VSOP87A.nep','VSOP87A.emb'/
2885      data cfile/    'VSOP87C.mer','VSOP87C.ven','VSOP87C.ear', &
           'VSOP87C.mar','VSOP87C.jup','VSOP87C.sat','VSOP87C.ura', &
           'VSOP87C.nep'/
          if (md(ibody)==1) then
            if (ivers==1) open(unit=10,file=afile(ibody))
2890        if (ivers==3) open(unit=10,file=cfile(ibody))
          endif
```



```
          call VSOP87Y(zjde,ivers,ibody,prec,lu,r,ierr,md)
          if (md(ibody)==1) close(10)
          call kugelko(r(1),r(2),r(3),rku)
2895 !c   write(6,'(/''  x, y, z  = '',3f14.10)') (r(i),i=1,3)
     !c   write(6,'( ''  vx,vy,vz = '',3f14.10)') (r(i),i=4,6)
     !c   write(6,'( ''  L, B, r  = '',3f14.10)') (rku(i),i=1,3)
          do iu=ix,6,5
            if (ierr/=0) write(iu,'(''  In VSOP87Y: ierr = '',i2)')ierr
2900      enddo
          end subroutine

          subroutine vsop3(l,k,ix,ke,time,res)
     !-----Bahn-Elemente, abgeleitet aus VSOP82 (nach Meeus)----------------
2905 !    fuer J2000.0 und Ekliptik der Epoche; Berechnung der wahren
     !    Anomalie (ekliptikale Laenge) mit der Keplerschen Gleichung.
     !    (Index von res  1: L, 2: a, 3: e, 4: i, 5: Omega, 6: pi, 7: M,
     !                    8: omega, 9: E, 10: nue, 11: eklipt. Laenge)
          use base, only : pidg,gdpi
2910      use astro, only : par3
          implicit double precision (a-h,o-z)
          dimension :: res(12)
          u360 = 360.d0; ke = 0
          eps = 1.d-13
2915      do j=1,6
            resu = 0.d0
            do i=1,4
              resu = resu + par3(i,j,k,l)*time**(i-1)
              if (j==1 .or.j>=5) call reduz(resu,0,1)
2920          res(j) = resu
            enddo
          enddo
          res(7) = res(1) - res(6)
          if (res(7)<0.d0) res(7) = res(7) + u360
2925      res(8) = res(6) - res(5)
          if (res(8)<0.d0) res(8) = res(8) + u360

     ! . . Loesung der Keplerschen Gleichung (Resultat: zen)
          ii = 0
2930      E  = res(3)
          zm = res(7)*pidg
          ze = zm
          itmax = 100  ! Maximalzahl der Iterationen

2935      meth = 1  ! Drei iterative Methoden zur Auswahl (meth = 1..3)
          if (meth<3) then
            do
              if (meth==1) then
     !        1. Verfahren von Newton-Raphson (schnellste Methode)
2940            zen = ze + (zm + E*dsin(ze) - ze)/(1.d0 - E*dcos(ze))
              else
     !        2. Fixpunktverfahren (Keplersche Gleichung)
                zen = zm + E*dsin(ze)
              endif
2945          if (dabs(zen-ze)<eps) exit
              if (ii>itmax) then; ke = 2; go to 20; endif
              ii = ii+1
              ze = zen
            enddo
2950      else
```

```
      !     3. Sekantenverfahren (verwendet Sekantensteigung)
               ke = 1
               ze2 = zm
         10    fze2 = zm + E*dsin(ze2) - ze2
2955           call sekante(ze1,ze2,fze1,fze2,eps,0.1d0,ii,itmax,ix,ke)
               if (ke==1) go to 10
               if (ke==2) go to 20 ! ("Ringfit" hat hier keinen Zeitvorteil
               zen = ze2           ! gegenueber "sekante", da die Keplersche
            endif                  ! Gleichung deutlich weniger Rechenzeit
2960        go to 30               ! benoetigt als "Ringfit" selbst.)

      ! . . zu viele Iterationen
         20 do iu=ix,6,5
              write(iu,'(/''  ---->  error in "vsop3" '', &
2965            & ''(Keplers equation), ke ='',I2/)') ke
            enddo
            return
         30 res(9) = zen*gdpi
            if (res(9)<0.d0) res(9) = res(9) + u360
2970
      ! . . Berechnung der wahren Anomalie
            res(10) = 2.d0 * datan(dsqrt((1.d0 + E)/(1.d0 - E)) &
                      * dtan(zen*0.5d0))*gdpi
            if (res(10)<0.d0) res(10) = res(10) + u360
2975        res(11) = res(10) + res(6)
            if (res(11)>u360) res(11) = res(11) - u360
            end subroutine

            subroutine transit(ip,ikomb,imod,ipla,ilin,iap,ivers,isep, &
2980          ical,iuniv,tr,sepmin,itt,sep,zjde,id5,da5,dmo5,zjahr, &
              rk,md,ddx1,ddx2,dfd,test,itin,is,ires,ix,pan,sd,sl,iop0,inum)
      !-----Ueberpruefung der Transite von Merkur bzw. Venus-----------------
      !     Die berechneten Zeitpunkte sind optional dieselbe ekliptikale
      !     Laenge bei Erde und Merkur bzw. Venus, die minimale Separation
2985  !     oder die genauen Phasen. "M" bedeutet "normaler", "C" (geozen-
      !     trischer) zentr. Transit des Merkurs und "m"/"c", dass irgend-
      !     wo auf der Erde der Transit partiell/zentral erscheint. Analog
      !     stehen "V" und "v" fuer die Venus. Das Minuszeichen "-" bedeu-
      !     tet, dass der Planet die Sonne knapp verfehlt und dass der
2990  !     dichteste Abstand der "sichtbaren" Scheiben (Sonnen- und Plane-
      !     tenrand) nicht mehr als etwa 1 Prozent des scheinbaren Sonnen-
      !     radius' betraegt (verwendet nur bei Syzygy-Berechnungen). Die
      !     Planetenscheibe ist in diesem Fall natuerlich nicht sichtbar.
      !     Index (ip): 1 = Merkur, 2 = Venus
2995        use base
            implicit double precision (a-h,o-z)
            dimension :: zi(2),sd(2),tcorr(2),rem(78)
            dimension :: ida(7),da(7),id5(5,7),da5(5,7),pan(5)
            dimension :: r(6),rku(3),rk(12),md(0:9),inum(0:4)
3000        dimension :: xx(5),yy(5),xk(2),yk(2),test(10)
            character(5) :: dmo,dmo5(5)
            character(1) :: tr,tp(8),sl
            data tp/'M','m','V','v','-',' ','C','c'/
            data idr/0/,blim/0.d0/,ba/0.d0/,ang/0.d0/,shift/0.d0/! pre-init.
3005  ! . . Einige Konstanten
            T = (zjde-zjd0)/tcen
      !     Axel D. Wittmann: we = Schiefe der Ekliptik der Epoche
            we = (23.4458042d0 - 0.856033d0 * &
                  dsin(0.015306d0 * (T + 0.50747d0))) * pidg
```



```
3010        zi(1) = re(35); zi(2) = re(41)
            wfact = 3600.d0*gdpi; eps = 1.d-7
      !     (Der folgende Korrekturfaktor "tcorr" zur Berechnung
      !     der minimalen Separation ist nur eine Abschaetzung.)
            do j=1,2; tcorr(j) = tsyn(j)/tsid(j); enddo
3015        ee = dsqrt(R3a*R3a-R3p*R3p)/R3a
            R3 = R3p/(AE*dsqrt(1.d0-(ee*dsin(we))**2))
            a  = dasin(R0/(AE*re(9)))
            b3 = dasin(R3*re(3*ip)/(re(9)*(re(9)-re(3*ip))))
            bp = dasin(Ra(ip)/(AE*(re(9)-re(3*ip))))
3020        bmin1 = a-bp; bmin2 = a-bp-b3
            bmax1 = a+bp; bmax2 = a+bp+b3

      !.....OPTIONEN 1/ 2: gleiche eklipt. Laenge u. minimale Separation
            if (isep==1) then
3025          din = dcos(zi(ip)*pidg*tcorr(ip))
              dre = (re(3*ip-1)-re(8))*pidg
              ba  = din*datan(re(3*ip)*dsin(dre)/(re(9)-re(3*ip)*dcos(dre)))
              bap = dabs(ba)
            else
3030          bap = sepmin
            endif
            if (ikomb==1 .and.imod==1) bmax2 = bmax2*1.8d0
            bout = bmax2*1.01d0; tr = tp(6)
            if (bap<=bmin2) tr = tp(2*ip-1)
3035        if (bap>bmin2.and.bap<=bmax2) tr = tp(2*ip)
            if (bap>bmax2.and.bap<=bout.and.ilin>=3) tr = tp(5)
            if (isep<=2 .and.ilin<=2) then
             if (bap<=bp+b3) tr = tp(8)
             if (bap<=bp) tr = tp(7)
3040        endif
      !c    do iu=ix,6,5; write(iu,'(a15,a18,i3,5f8.5)')'ip,bmin2,bmin1,', &
      !c      'bmax1,bmax2,bap = ',ip,bmin2,bmin1,bmax1,bmax2,bap; enddo

      ! . . Min. Separation (sep) zw. Sonne und Planet in
3045  !     Bogensekunden. Bei "plus" passiert der Planet das
      !     das Sonnenzentrum noerdlich, bei "minus" suedlich.
            if (isep==1) then
              sep = ba*wfact
            else
3050          sep = bap*wfact
              if (re(3*ip-1)<0.d0) sep = -sep
            endif
            if (isep<=2) then
              if (tr==' '.or.ilin>=3) return; go to 60
3055        endif

      !.....OPTIONEN 3/ 4: Transitphasen ohne/mit Positionswinkeln
      !     (Beginn, Ende und minimale Separation des geozentrischen Tran-
      !     sits => Ein, drei oder fuenf Zeitpunkte werden berechnet.)
3060        if (bap>bmax2*1.005d0 .or.(ikomb==1 .and.imod==1)) then
              itt = 0
              return
            endif

3065  ! . . Weitere Parameter festlegen
            prec = z0; lu = 10
            itr = 1
            do j=1,78; rem(j) = re(j); enddo
```

```
            do j=1,5
3070          do k=1,7
                id5(j,k) = 0
                da5(j,k) = z0
              enddo
            enddo
3075        xj2 = zjde

      ! . . Mitte des Transits, minimale Separation mit Lichtlaufzeit
            if (itr==1) then
              idr = 3; ke = 1; indx = 1
3080          step = 5.d-2; iflag = 0
              ddx1 = dfd + 1.d0; nu = 0
              if (ilin<=2) ddx1 = 1; ddx2 = ddx1
              xx(1) = xj2; itin = 0; iex = 0
              do j=1,10; test(j) = z0; enddo
3085  !       Mittlere Laufzeit des Lichtes, optimierter Startwert [Tage]
              if (ip==1) del = 320.d0/86400.d0  ! Merkur
              if (ip==2) del = 150.d0/86400.d0  ! Venus
              if (imod==1) then; ept=3.d-14; else; ept=2.d-9; endif

3090  !       VSOP87-Berechnung mit Beruecksichtigung der Lichtlaufzeit
         10   if (imod==1) then
               call vsop1tr(ip,rk,(xj2-zjd0-del)/tmil,del,r3i,ept,inum,resu)
              else
               call vsop2tr(xj2-del,ivers,ip,md,ix,prec,lu,r,rk, &
3095             ierr,del,r3i,ept,inum,rku)
              endif
              if (iex==1) go to 20
      !       Bestimmung: auf- bzw. absteigender Knoten
              if (nu==1 .or.nu==2) then
3100            xk(nu) = xj2; yk(nu) = re(3*ip-1)
              endif
              if (nu==2) then
                sl = '/'; if ((yk(2)-yk(1))/(xk(2)-xk(1))<0.d0) sl = ' '
              endif
3105  !       Ende Knotenbestimmung
              call sepa(ip,2,rk,sep0i)
              yy(indx) = sep0i
              epv = 1.d-6; if (sep0i<30.d0) epv = 1.d-7
              call fitmin(imod,2,iap,ke,xx,yy,epv,step,nu,iflag, &
3110                ddx1,ddx2,test,itin,indx,ix)
              xj2 = xx(indx)
              if (ke==0 .and.isep==4 .and.iex==0) then
              iex = 1; go to 10; endif
              if (ke==1) go to 10
3115
      !       Art des (streifenden) Transits
         20   if (sep0i<=bmin2) then; tr=tp(2*ip-1); itt=3; endif
              if (sep0i>bmin2.and.sep0i<=bmin1) itt=3
              if (sep0i>bmin1.and.sep0i<=bmax1) itt=2
3120          if (sep0i>bmax1.and.sep0i<=bmax2) itt=1
              if (sep0i>bmax2) then; itt = 0; return; endif
              if (sep0i>bmin2.and.sep0i<=bmax2) then
                inum(3) = inum(3) + 1
                tr=tp(2*ip)
3125          endif
              sep = sep0i*wfact
              if (re(3*ip-1)<0.d0) sep = -sep
```



```
              xjdt = xj2; zjde = xj2
              if (iuniv==2) call delta_T(xjdt)
3130          call jdedate(xjdt,ical,ida,da,dmo)
              call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)

      !       Berechnung des Positionswinkels (minimale Separation)
              if (isep==4) call pos_angle(ip,zjde,rk,ang)
3135
      !       Radien (semidiameter) von Sonne und Merkur/Venus
              if (isep>=3 .and.ilin<=2) then
                sd(1) = dasin(R0/(AE*re(9))) * wfact
                sd(2) = dasin(Ra(ip)/(AE*r3i)) * wfact
3140  !         Kennzeichnung des zentralen Transits
                csep = (r3*re(3*ip)/re(9)+Ra(ip)/AE)*wfact/(re(9)-re(3*ip))
                if (dabs(sep)<csep) then
                  tr = tp(8)
                  if (dabs(sep)<sd(2)) tr = tp(7)
3145              inum(4) = inum(4) + 1
                endif
      !         Mit der zeitlichen Verschiebung "shift" (in julian. Tagen)
      !         wird der spaeter folgende Startpunkt fuer "ringfit" bzw.
      !         "sekante" moeglichst nahe an die Nullstelle verlegt.
3150            wu = 1.d0-(sep/sd(1))**2
                if (wu<1.d-2) wu = 1.d-2
                if (ip==1) shift = 0.115d0 * dsqrt(wu)
                if (ip==2) shift = 0.17d0 * dsqrt(wu)
              endif
3155        endif

            if (itr==1) then
              if (itt==1) itr = 6
              go to 50
3160        endif

      ! . . Vorbereitung zur naechsten Berechnung im selben Transit
         30 iis = 0; ke = 1
            itr = itr + 1
3165  !     Kontaktpunkt I
            if (itr==2) then
              idr = 1; blim = bmax1
              xj2 = zjde - shift
            endif
3170  !     Kontaktpunkt II
            if (itr==3) then
              if (itt==2) itr = 5
              idr = 2; blim = bmin1
              xj2 = zjde - shift
3175        endif
      !     Kontaktpunkt III
            if (itr==4) then
              idr = 4; blim = bmin1
              xj2 = zjde + shift
3180        endif
      !     Kontaktpunkt IV
            if (itr==5) then
              idr = 5; blim = bmax1
              xj2 = zjde + shift
3185        endif
```

```
      ! . . Berechnung der Kontaktzeiten I bis IV
            if (imod==1) then; ept=1.d-12; else; ept=2.d-7; endif
         40 tau = (xj2 - zjd0)/tmil
3190
      !     VSOP87D Kurzversion (imod=1), VSOP87C Vollversion (imod=2)
            if (imod==1) then
              call vsop1tr(ip,rk,tau,del,r3i,ept,inum,resu)
            else
3195          call vsop2tr(xj2,ivers,ip,md,ix,prec, &
                lu,r,rk,ierr,del,r3i,ept,inum,rku)
            endif
      !     "Sekante" wurde durch das etwas schnellere "ringfit" ersetzt.
            call sepa(ip,2,rk,sep0i)
3200        yy2 = sep0i-blim
            call ringfit(xj1,xj2,xj3,yy1,yy2,yy3,eps,1.d-3,iis,25,ix,ke)
            if (ke==1 .or.ke==5) go to 40
            if (ke==2) go to 60
            xjdt = xj2 + del
3205        if (iuniv==2) call delta_T(xjdt)
            call jdedate(xjdt,ical,ida,da,dmo)

      ! . . Berechnung des Positionswinkels (Planet am Sonnenrand)
            if (isep==4 .and.itr/=1) call pos_angle(ip,xj2,rk,ang)
3210
      ! . . Ruecksprung
         50 do k=1,7; id5(idr,k) = ida(k); da5(idr,k) = da(k); enddo
            dmo5(idr) = dmo
            pan(idr) = ang
3215        if (itr<=4) go to 30
            do j=1,78; re(j) = rem(j); enddo

      !.....Berechnung der Transitserie
         60 if (ikomb==0 .or.(ikomb==1 .and.imod==2)) &
3220          call tserie(ip,zjde,is,iop0,ires)
            end subroutine

            subroutine sepa(ip,iv,rk,sep0i)
      !-----Berechnung der Separation Sonne-Merkur bzw. Sonne-Venus----------
3225  !     Index ip:  1 = Merkur, 2 = Venus
            use base, only : pidg,re
            implicit double precision (a-h,o-z)
            dimension :: rk(12),rd(3)
            if (iv==1) then
3230  ! . . . 1. Variante - raeumliche Geometrie (Testvariante)
              cos0i = dsin(re(3*ip-1)*pidg) * dsin(re(8)*pidg) + &
                      dcos(re(3*ip-1)*pidg) * dcos(re(8)*pidg) * &
                      dcos((re(3*ip-2)-re(7))*pidg)
              sep0i = datan(re(3*ip)*dsqrt(1.d0-cos0i*cos0i)/ &
3235                        (re(9)-re(3*ip)*cos0i))
            else
      ! . . . 2. Variante - Vektoranalysis
              do j=1,3; rd(j) = rk(3*(ip-1)+j) - rk(6+j); enddo
              ab = -rk(7)*rd(1)-rk(8)*rd(2)-rk(9)*rd(3)
3240          a  = dsqrt(rk(7)**2 + rk(8)**2 + rk(9)**2)
              b  = dsqrt(rd(1)**2 + rd(2)**2 + rd(3)**2)
              sep0i = dacos(ab/(a*b))
            endif
            end subroutine
3245
```



```
            subroutine pos_angle(ip,xjd,rk,ang)
      !-----Positionswinkel des Planeten fuer beliebigen Zeitpunkt des Tran-
      !     sits in Bezug auf die Richtung zum Himmelsnordpol (y-Achse auf
      !     Sonnenscheibe) – vgl. scheinbare Bewegungsrichtung der Sonne.
3250  !     ip        : 1 fuer Merkur, 2 fuer Venus
      !     xjd       : Zeitpunkt der Ankunft des Lichtes auf der Erde
      !     rk(1..9)  : rechtwinklige heliozentrische Koordinaten
      !                 von Merkur, Venus und Erde (VSOP87C)
      !     eeps      : Stellung Erdachse gegen Ekliptik in jener Epoche
3255  !     rgeo(1..9): transformierte geozentrische Koordinaten von Sonne,
      !                 Merkur und Venus (rechtwinklig, dann sphaerisch)
      !     ang       : Positionswinkel des Planeten vor der Sonne
            use base, only : pidg,gdpi,zjd0,tcen
            implicit double precision (a-h,o-z)
3260        dimension :: rk(12),rgeo(9),rku(3),xx(3)
            do i=1,9; rgeo(i) = rk(i); enddo

      !.....Die Berechnung des Positionswinkels erfolgt in 4 Schritten.
      !     Schritte 1-3: Koordinatentransformation helio- zu geozentrisch.
3265
      !  1. Rotation um x-Achse um Winkel der Schiefe der Ekliptik (Epoche);
      !     Axel D. Wittmann: "On the variation of the obliquity of the
      !     ecliptic", Univ.-Sternwarte Goettingen, 1984, MitAG 62, S.203
            T = (xjd-zjd0)/tcen
3270        eeps = (23.4458042d0 - 0.856033d0 * &
                    dsin(0.015306d0 * (T + 0.50747d0))) * pidg
            call rotmat(1,-eeps,0.d0,0.d0,rgeo)

      !  2. Translation des heliozentrischen Koordinatenursprungs von der
3275  !     Sonne zur Erde. Das ergibt neue Koordinaten fuer Sonne und
      !     Merkur bzw. Venus.
            do i=1,3
              xx(i) = -rgeo(6+i); rgeo(6+i) = rgeo(3+i)
              rgeo(3+i) = rgeo(i); rgeo(i) = 0.d0
3280        enddo
            call translat(xx(1),xx(2),xx(3),rgeo)

      !  3. Umrechnung in sphaerische Koordinaten
      !     (Positionen von Sonne, Merkur und Venus)
3285        do i=0,2; ii = 3*i
              call kugelko(rgeo(ii+1),rgeo(ii+2),rgeo(ii+3),rku)
              do j=1,3; rgeo(ii+j) = rku(j); enddo
            enddo

3290  !  4. Berechnung des Positionswinkels nach Andre Danjon: "Astronomie
      !     Generale", S.36, Gl."3 bis". Siehe auch Jean Meeus: "Transits",
      !     S.15 ("kartesische" Koordinaten x und y in Bogensekunden).
            sdec = rgeo(2) * pidg
            dra  = (rgeo(3*ip+1)-rgeo(1)) * pidg
3295        ddec = (rgeo(3*ip+2)-rgeo(2)) * pidg
            tdra = dsin(sdec) * dtan(dra) * dtan(dra*0.5d0)
            zk = 206264.8062d0/(1.d0 + dsin(sdec) * tdra)
            x = -zk * (1.d0 - dtan(sdec)*dsin(ddec)) * dcos(sdec)*dtan(dra)
            y =  zk * (dsin(ddec) + dcos(sdec) * tdra)
3300        ang = datan(-x/y)*gdpi
            if (y*dcos(ang*pidg)<0.d0) ang = ang + 180.d0
            call reduz(ang,0,1)
            end subroutine
```

```
      subroutine tserie(ip,zjde,is,iop0,ires)
!-----Bestimmung der Transit-Serie-------------------------------------
!     Die Seriennummern entsprechen denen der "NASA Eclipse Web Site".
!     (Die Liste der Seriennummern "inserie.t" wird nur einmal verwen-
!     det, um die Startnummern, d.h. die Nummern zu bestimmen, die den
!     ersten gefundenen Transiten zugeordnet werden. Danach werden al-
!     le weiteren Seriennummern unabhaengig von der Liste berechnet.)
!     Index (ip):  1 = Merkur, 2 = Venus
      use astro, only : ser,ase,cc,t13BC,t17AD, &
        zstart,ise,ji,jj,isflag,ismax
      implicit double precision (a-h,o-z)
      if (dabs(zstart-99.99d0)<1.d-10) zstart = zjde
      if (iop0/=-803) then
        if (zjde<t13BC-365.d0 .or.zjde>t17AD+365.d0) then
          ires = 999; return
        endif

! . . . Seriennummer (is) fuer Startzeitpunkt suchen
        if (isflag==0) then
          do j=jj(2*ip-1),jj(2*ip)
            if (ser(j,ip)>zjde) then
              is = j
              isflag = 1
              exit
            endif
          enddo
        endif
      endif

! . . Aktuelle Seriennummer bestimmen
      kflag = 0
      do j=is-ji(ip),is
        zlim = dmax1(t13BC,zstart)
        if (zjde-zlim>cc(ip)+100.d0) then
          do k=jj(2*ip-1),is
            ise(k) = 1
          enddo
        endif
        a = (zjde-ser(j,ip))/cc(ip)
        x = dabs((a-dnint(a))*cc(ip))
        b = dabs(zjde-ase(j)-cc(ip))
!c      write(6,'(''a,x,b,ise(j),j,is,ismax ='',f9.3,f10.3,f16.6, &
!c       & i3,3i5)')a,x,b,ise(j),j,is,ismax
        if (x<=10.d0 .and.(b<=2.d0 .or.ise(j)==0)) then
          ires = j
          kflag = 1
          if (j>ismax) ismax = j
        endif
        if (j==is.and.kflag==1) go to 20
      enddo
      if (ismax==-10000 .or.is>ismax) ismax = is - 1
      is = ismax + 1
      ismax = is
      ser(is,ip) = zjde
      ires = is
   20 ase(ires) = zjde
      ise(ires) = 1
      end subroutine
```



```
      subroutine VSOP87Y(tdj,ivers,ibody,prec,lu,r,ierr,md)
!-----------------------------------------------------------------------
! >>
! >>  UPGRADE (by H. Jelitto): As proposed by Bretagnon and Francou
! >>  for rapidity of computation, the parameters in the VSOP87-files
! >>  are read only once at the first call for each planet. The main
! >>  data are copied into the 5-dimensional array "par2" for random
! >>  access, covering all planets of one VSOP87-version. For the
! >>  calculation of the transit phases (TYMT-test), this reduces the
! >>  computing time by a factor 20 to 30. Thus, the original subrou-
! >>  tine "VSOP87" is extended and renamed as "VSOP87X".
! >>
! >>  The new VSOP87X routine has been checked only for the use of
! >>  the theory versions VSOP87A and VSOP87C. Furthermore, the code
! >>  is converted to the Fortran 95 standard and the free source
! >>  form.
! >>
! >>  PARALLEL PROCESSING: To realize parallel processing, the
! >>  VSOP87X-subroutine is modified with the application programming
! >>  interface (API) "OpenMP". For the compilation, we use the com-
! >>  mand: "gfortran -fopenmp -O2 p4-4.f95". In this case, the run-
! >>  time is determined with the subroutine "CPU_time" and also with
! >>  "date_and_time". Here, VSOP87X is adapted to 4 threads and its
! >>  name is changed to "VSOP87Y". If more threads shall be used,
! >>  only VSOP87Y has to be modified. No other changes are necessary.
! >>  For the adaption of the code and for the differences to the
! >>  original p4.f95 code, search for the phrase "threads". Notice:
! >>  For optimization of the speed, the if-statement for comparison
! >>  with the parameter p in the inner do-loop has been removed.
! >>  This statement probably had an advantage in former times, when
! >>  the data were read from magnetic tape. Now, it would slow down
! >>  a bit the processing speed.
! >>
! >>  The following text belongs to the original VSOP87-subroutine.
! >>
!-----------------------------------------------------------------------
!
!     Reference : Bureau des Longitudes - PBGF9502
!
!     Object :
!
!     Substitution of time in VSOP87 solution written on a file.
!     The file corresponds to a version of VSOP87 theory and to a body.
!
!     Input :
!
!     tdj      julian date (real double precision).
!              time scale : dynamical time TDB.
!
!     ivers    version index (integer).
!              0: VSOP87 (initial solution).
!                 elliptic coordinates
!                 dynamical equinox and ecliptic J2000.
!              1: VSOP87A.
!                 rectangular coordinates
!                 heliocentric positions and velocities
!                 dynamical equinox and ecliptic J2000.
!              2: VSOP87B.
!                 spherical coordinates
```

```
      !                heliocentric positions and velocities
      !                dynamical equinox and ecliptic J2000.
3425  !             3: VSOP87C.
      !                rectangular coordinates
      !                heliocentric positions and velocities
      !                dynamical equinox and ecliptic of the date.
      !             4: VSOP87D.
3430  !                spherical coordinates
      !                heliocentric positions and velocities
      !                dynamical equinox and ecliptic of the date.
      !             5: VSOP87E.
      !                rectangular coordinates
3435  !                barycentric positions and velocities
      !                dynamical equinox and ecliptic J2000.
      !
      !     ibody    body index (integer).
      !              0: Sun (not used here in VSOP87Y)
3440  !              1: Mercury
      !              2: Venus
      !              3: Earth
      !              4: Mars
      !              5: Jupiter
3445  !              6: Saturn
      !              7: Uranus
      !              8: Neptune
      !              9: Earth-Moon barycenter
      !
3450  !     prec     relative precision (real double precision).
      !
      !              if prec is = 0 then the precision is the precision
      !                 p0 of the complete solution VSOP87.
      !                 Mercury    p0 =  0.6 10**-8
3455  !                 Venus      p0 =  2.5 10**-8
      !                 Earth      p0 =  2.5 10**-8
      !                 Mars       p0 = 10.0 10**-8
      !                 Jupiter    p0 = 35.0 10**-8
      !                 Saturn     p0 = 70.0 10**-8
3460  !                 Uranus     p0 =  8.0 10**-8
      !                 Neptune    p0 = 42.0 10**-8
      !
      !              if prec is not equal to 0, let us say in between p0 and
      !              10**-2, the precision is :
3465  !                 for the positions :
      !                 - prec*a0 au for the distances.
      !                 - prec rd for the other variables.
      !                 for the velocities :
      !                 - prec*a0 au/day for the distances.
3470  !                 - prec rd/day for the other variables.
      !                   a0 is semi-major axis of the body.
      !                 Mercury    a0 =  0.3871 au
      !                 Venus      a0 =  0.7233 au
      !                 Earth      a0 =  1.0000 au
3475  !                 Mars       a0 =  1.5237 au
      !                 Jupiter    a0 =  5.2026 au
      !                 Saturn     a0 =  9.5547 au
      !                 Uranus     a0 = 19.2181 au
      !                 Neptune    a0 = 30.1096 au
3480  !
      !     lu       logical unit index of the file (integer).
```



```
      !              The file corresponds to a version of VSOP87 theory and
      !              a body, and it must be defined and opened before the
      !              first call to subroutine VSOP87.
3485  !
      !     Output :
      !
      !     r(6)     array of the results (real double precision).
      !
3490  !              for elliptic coordinates :
      !                  1: semi-major axis (au)
      !                  2: mean longitude (rd)
      !                  3: k = e*cos(pi) (rd)
      !                  4: h = e*sin(pi) (rd)
3495  !                  5: q = sin(i/2)*cos(omega) (rd)
      !                  6: p = sin(i/2)*sin(omega) (rd)
      !                     e:     eccentricity
      !                     pi:    perihelion longitude
      !                     i:     inclination
3500  !                     omega: ascending node longitude
      !
      !              for rectangular coordinates :
      !                  1: position x (au)
      !                  2: position y (au)
3505  !                  3: position z (au)
      !                  4: velocity x (au/day)
      !                  5: velocity y (au/day)
      !                  6: velocity z (au/day)
      !
3510  !              for spherical coordinates :
      !                  1: longitude (rd)
      !                  2: latitude (rd)
      !                  3: radius (au)
      !                  4: longitude velocity (rd/day)
3515  !                  5: latitude velocity (rd/day)
      !                  6: radius velocity (au/day)
      !
      !     ierr     error index (integer).
      !                  0: no error.
3520  !                  1: file error (check up ivers index).
      !                  2: file error (check up ibody index).
      !                  3: precision error (check up prec parameter).
      !                  4: reading file error.
      !
3525  !-----------------------------------------------------------------------

      !     -------------------------------
      !     Declarations and initializations
      !     -------------------------------
3530        use astro, only : par2,it2,in2,iv2
            implicit double precision (a-h,o-z)
            character(7) :: bo,body(0:9)
            dimension :: r(6),t(-1:5),a0(0:9),md(0:9)
            data body/'SUN','MERCURY','VENUS','EARTH','MARS','JUPITER', &
3535                  'SATURN','URANUS','NEPTUNE','EMB'/
            data a0/0.01d0,0.3871d0,0.7233d0,1.d0,1.5237d0,5.2026d0, &
                    9.5547d0,19.2181d0,30.1096d0,1.d0/
            data dpi/6.28318530717958647692d0/
            data t/0.d0,1.d0,5*0.d0/
3540        data t2000/2451545.d0/
```

```
      data a1000/365250.d0/
      k=0
      ierr=3
      if (md(ibody)==1) then
3545    ideb=0
        do i=1,3; do j=0,5; it2(j,i,ibody) = -1; enddo; enddo
      endif
      do i=1,6; r(i)=0.d0; enddo
      t(1)=(tdj-t2000)/a1000
3550  do i=2,5; t(i)=t(1)*t(i-1); enddo
      if (prec<0.d0 .or.prec>1.d-2) return
      if (md(ibody)/=1) ierr = 0
      q=dmax1(3.d0,-dlog10(prec+1.d-50))

3555  !   ------------------------------------------------------------
      !   File reading, for each planet only at first call to VSOP87Y
      !   ------------------------------------------------------------
      if (md(ibody)==1) then
   10   read (lu,1001,end=20) iv,bo,ic,it,inn
3560    iv2(ibody) = iv
        it2(it,ic,ibody) = 1
        in2(it,ic,ibody) = inn
        if (ideb==0) then
          ideb=1; ierr=1
3565      if (iv/=ivers) return
          ierr=2
          if (bo/=body(ibody)) return
          ierr=0
        endif
3570    if (inn==0) go to 10
        do n=1,inn
          read (lu,1002) (par2(i,n,it,ic,ibody),i=1,3)
        enddo
        go to 10
3575 20 md(ibody) = 2
      endif

      !   ------------------------------------
      !   Computation of planetary coordinates
3580  !   ------------------------------------
      ic = 1; it = 0
      iv = iv2(ibody)
      if (iv==0) k=2
      if (iv==2 .or.iv==4) k=1
3585 25 r1 = 0.d0; r2 = 0.d0; r3 = 0.d0
      !   Fork --> 4 threads -----------------
      !$omp parallel sections default(shared) &
      !$omp private(n,a,b,c,inn,ith,itn) firstprivate(ic,ibody,t)
        !$omp section
3590 30   inn = in2(it,ic,ibody); if (inn==0) go to 50
          do n=1,inn,4
            a = par2(1,n,it,ic,ibody)
            b = par2(2,n,it,ic,ibody)
            c = par2(3,n,it,ic,ibody)
3595        r(ic) = r(ic) + a*dcos(b + c*t(1))*t(it)
          enddo
   50     if (it<=4) itn = it2(it+1,ic,ibody)
          if (it<=4 .and.itn/=-1) then; it = it+1; go to 30
          endif
```



```
3600      !$omp section
            ith = 0
   31       inn = in2(ith,ic,ibody); if (inn==0) go to 51
            do n=2,inn,4
              a = par2(1,n,ith,ic,ibody)
3605          b = par2(2,n,ith,ic,ibody)
              c = par2(3,n,ith,ic,ibody)
              r1 = r1 + a*dcos(b + c*t(1))*t(ith)
            enddo
   51       if (ith<=4) itn = it2(ith+1,ic,ibody)
3610        if (ith<=4 .and.itn/=-1) then; ith = ith+1; go to 31
            endif
          !$omp section
            ith = 0
   32       inn = in2(ith,ic,ibody); if (inn==0) go to 52
3615        do n=3,inn,4
              a = par2(1,n,ith,ic,ibody)
              b = par2(2,n,ith,ic,ibody)
              c = par2(3,n,ith,ic,ibody)
              r2 = r2 + a*dcos(b + c*t(1))*t(ith)
3620        enddo
   52       if (ith<=4) itn = it2(ith+1,ic,ibody)
            if (ith<=4 .and.itn/=-1) then; ith = ith+1; go to 32
            endif
          !$omp section
3625        ith = 0
   33       inn = in2(ith,ic,ibody); if (inn==0) go to 53
            do n=4,inn,4
              a = par2(1,n,ith,ic,ibody)
              b = par2(2,n,ith,ic,ibody)
3630          c = par2(3,n,ith,ic,ibody)
              r3 = r3 + a*dcos(b + c*t(1))*t(ith)
            enddo
   53       if (ith<=4) itn = it2(ith+1,ic,ibody)
            if (ith<=4 .and.itn/=-1) then; ith = ith+1; go to 33
3635        endif
        !$omp end parallel sections
      !   Join --> serial processing ---------
        r(ic) = r(ic) + r1 + r2 + r3 ! (results of threads)
        if (ic<3) then
3640      it = 0
          ic = ic + 1
          go to 25
        endif
        if (iv/=0) then
3645      do i=4,6; r(i)=r(i)/a1000; enddo
        endif
        if (k==0) return
        r(k)=dmod(r(k),dpi)
        if (r(k)<0.d0) r(k)=r(k)+dpi
3650    return

      !   -------
      !   Formats
      !   -------
3655   1001 format (17x,i1,4x,a7,12x,i1,17x,i1,i7)
       1002 format (79x,f18.11,f14.11,f20.11)
      end subroutine
```

```
      subroutine kartko(ison)
!-----Umwandlung in kartesische Koordinaten, re(1..9) --> xyr(1..9)----
!     mit Merkur bei x-Achse
!     Indizes von "re" :  1: Lm'  2: Bm   3: rm   4: Lv'  5: Bv
!                         6: rv   7: Le'  8: Be   9: re
!     Indizes von "xyr":  1: xm   2: ym   3: zm   4: xv   5: yv
!                         6: zv   7: xe   8: ye   9: ze  10: leer
      use base
      implicit double precision (a-h,o-z)
      rr = re(1)
      if (ison==2) rr = re(4)
      if (ison==0) rr = 0.d0
      do i=3,9,3
       xyr(i-2) = re(i)*dcos(re(i-1)*pidg)*dcos((re(i-2)-rr)*pidg)
       xyr(i-1) = re(i)*dcos(re(i-1)*pidg)*dsin((re(i-2)-rr)*pidg)
       xyr(i)   = re(i)*dsin(re(i-1)*pidg)
      enddo
      end subroutine

      subroutine relpos(ipla,ison,ijd,iek,iekk,ika)
!-----Vergleich der Positionen Pyramiden/Kammern mit Planeten,---------
!     daraus Bestimmung der Genauigkeit Fpos bzw. xyr(36) in Prozent
!     und der Polaritaet "iek" bzw. "iekk". Weitere Indizes von "xyr":
!        11: xv-xm   12: xe-xm   13: xe-xv   14: yv-ym   15: ye-ym
!        16: ye-yv   17: zv-zm   18: ze-zm   19: ze-zv   20: leer
!        21: v - m   22: e - m   23: e - v   24: q1       25: q2
!        26: q3      27: alpha'  28: beta'   29: gamma'   30: leer
!        31: x-Son   32: y-Son   33: z-Son   34: delta-s 35: M
!        36: Fpos, F'pos, F"pos
!     Indizes 11 - 19 und 21 - 29 bei "pyr" und "xyr" entsprechen sich.
      use base
      implicit double precision (a-h,o-z)

! . . Pyramidenabstaende
      xyr(11) = xyr(4)-xyr(1)
      xyr(12) = xyr(7)-xyr(1)
      xyr(13) = xyr(7)-xyr(4)
      xyr(14) = xyr(5)-xyr(2)
      xyr(15) = xyr(8)-xyr(2)
      xyr(16) = xyr(8)-xyr(5)
      xyr(17) = xyr(6)-xyr(3)
      xyr(18) = xyr(9)-xyr(3)
      xyr(19) = xyr(9)-xyr(6)
      ax = xyr(11);  ay = xyr(14)
      bx = xyr(12);  by = xyr(15)
      cx = xyr(13);  cy = xyr(16)
      if (ison==3) then
        az = z0; bz = z0; cz = z0
      else
        az = xyr(17); bz = xyr(18)
        cz = xyr(19)
      endif

! . . Feststellen der Polaritaet (Blickrichtung auf die Ekliptik)
!     gemaess Vorzeichen der z-Komponente des Vektorproduktes a x c.
      if (ijd==15 .or.ijd==0) then
        if (iek/=3) iek  = 1
        if (iek==3) iekk = 1
        ez = ax*cy-ay*cx
        if ((ipla==1 .and.ez>=z0).or.(ipla==2 .and. &
          ((ez<z0.and.(ika==1 .or.ika==4 .or.ika==5)).or. &
          (ez>=z0.and.(ika==2 .or.ika==3 .or.ika==6))))) then
          if (iek/=3) iek  = 2
          if (iek==3) iekk = 2
        endif
      endif

! . . Berechnung der rel. Abweichung [%] --> xyr(36)
!     Sonnenposition auf Nordsuedachse
      if (ison<=2) then
        xyr(24) = bx/ax; xyr(25) = by/ay; xyr(26) = by/bx
        s = 1.d0
        if (iek==3 .and.iekk==2) s = -1.d0
        dx1 = (xyr(24) - pyr(24))/pyr(24)
        dx2 = (xyr(25) - pyr(25))/pyr(25)
        dx3 = (xyr(26)-s*pyr(26))/pyr(26)
        xyr(36) = 100.d0 * dsqrt((dx1*dx1 + dx2*dx2 + dx3*dx3)/3.d0)
        return
      endif

!.....Relative Abweichung, Sonnenposition frei (2- und 3-dimensional)
!     Anmerkung: Bei Berechnung von F"pos (Sonnenpos. frei) laesst
!     sich statt der Strecken Mykerinos- Chefren-Pyramide u. Mykerinos-
!     Cheops-Pyramide auch ein anderes Streckenpaar verwenden, wie z.B.
!     Mykerinos- Chefren-Pyramide und Chefren- Cheops-Pyramide. F"pos
!     hat dann eventuell etwas andere Werte, aber die Minimierung von
!     F"pos liefert dieselben Zeitpunkte. Das heisst, die wesentlichen
!     Ergebnisse bleiben identisch.
      xyr(21) = dsqrt(ax*ax + ay*ay + az*az)
      xyr(22) = dsqrt(bx*bx + by*by + bz*bz)
      xyr(23) = dsqrt(cx*cx + cy*cy + cz*cz)
      xyr(24) = xyr(22)/xyr(21)
!c    xyr(25) = xyr(23)/xyr(21)
!c    xyr(26) = xyr(23)/xyr(22)
      xyr(27) = dacos((ax*bx + ay*by + az*bz)/(xyr(21) * xyr(22)))
!c    xyr(28) = dacos((ax*cx + ay*cy + az*cz)/(xyr(21) * xyr(23)))
!c    xyr(29) = dacos((bx*cx + by*cy + bz*cz)/(xyr(22) * xyr(23)))
      dx1 = (xyr(24)-pyr(24))/pyr(24)
      dx2 = xyr(27)-pyr(27)
      xyr(36) = 100.d0 * dsqrt((dx1*dx1 + dx2*dx2)*0.5d0)
      end subroutine

      subroutine sonpos(ison,iek,ix,xp3,yp3,zp3, &
        rcm,dmi,iter,iw,ke,m,n,f,x,e,w,y,z)
!-----Bestimmung von Sonnenposition und Massstab --> xyr(31 - 35)------
!     Indizes von xyr wie in relpos
      use base
      implicit double precision (a-h,o-z)
      dimension :: D(3,3),xsta(n),ysta(m),rcm(3)
      dimension :: x(n),e(n),iw(100),f(m),y(m),z(m),w(1000)

!.....Zweidimensionale Berechnung der Sonnenpos. (x- und y-Koord.)
!     Projektion der Planetenpositionen in die Ekliptikebene.
!     Zusammengehoerige Pyramiden- und Planetenabstaende werden paral-
!     lel ausgerichtet und in der Mitte zur Deckung gebracht. (Wegen
!     des gemeinsamen Massstabsfaktors "zmas" haben die entsprechenden
!     Strecken leicht unterschiedliche Laengen.)
      em = 1.d0
```

```
            if (iek==2) em = -1.d0
            if (ison<=3) then
              sax = (xyr(4)+xyr(1)) * .5d0
              say = (xyr(5)+xyr(2)) * .5d0
              sbx = (xyr(7)+xyr(1)) * .5d0
              sby = (xyr(8)+xyr(2)) * .5d0
              scx = (xyr(7)+xyr(4)) * .5d0
              scy = (xyr(8)+xyr(5)) * .5d0
              al1 = - em * pyr(31) - datan(ay/ax) + datan(say/sax)
              al2 = - em * pyr(32) - datan(by/bx) + datan(sby/sbx)
              al3 = - em * pyr(33) - datan(cy/cx) + datan(scy/scx)
              r1 = dsqrt(sax*sax + say*say)
              r2 = dsqrt(sbx*sbx + sby*sby)
              r3 = dsqrt(scx*scx + scy*scy)
      !       Massstabsfaktor
              zmas = (pyr(21)/xyr(21)+pyr(22)/xyr(22)+pyr(23)/xyr(23))/3.d0
              xso1 = - r1 * zmas * dcos(al1) + pyr(34)
              xso2 = - r2 * zmas * dcos(al2) + pyr(36)
              xso3 = - r3 * zmas * dcos(al3) + pyr(38)
              yso1 = - r1 * zmas * dsin(al1) + pyr(35) * em
              yso2 = - r2 * zmas * dsin(al2) + pyr(37) * em
              yso3 = - r3 * zmas * dsin(al3) + pyr(39) * em
              xyr(31) = (xso1 + xso2 + xso3)/3.d0
              xyr(32) = (yso1 + yso2 + yso3)/3.d0
              if (iek==2) xyr(32) = - xyr(32)
              xyr(33) = z0
      ! . . . Fehlerabschaetzung fuer die Sonnenposition
              xyr(34) = dsqrt((xyr(31)-rcm(1))**2 + (xyr(32)-rcm(2))**2) &
                * xyr(36) * 1.d-2
      ! . . . Massstabsfaktor (nur fuer "Sonne" suedlich der
      !       dritten Pyramide, zweidimensional gerechnet.)
              xyr(35)=AE*0.25d0*(dabs(xyr(11)/pyr(11))+dabs(xyr(12)/pyr(12))&
                              + dabs(xyr(14)/pyr(14))+dabs(xyr(15)/pyr(15)))
            endif

      !.....Dreidimensionale Berechnung (x-, y- und z-Koordinate)
      !     Loesung eines linearen inhomogenen Gleichungssystems bzgl. der
      !     Planetenpositionen und Uebertragung des Ergebnisses auf die
      !     Pyramidenpositionen.
      ! . . Erzeugung eines (schiefwinkligen) Vektordreibeins fuer die Pla-
      !     neten (mit Hilfe des Vektorproduktes). Die 3 Vektoren bilden
      !     dann die Spalten der Koeffizienten-Matrix.
            if (ison==4) then
              D(1,1) = ax
              D(2,1) = ay
              D(3,1) = az
              D(1,2) = bx
              D(2,2) = by
              D(3,2) = bz
              dx = by*az - ay*bz
              dy = ax*bz - bx*az
              dz = bx*ay - ax*by
              aba = dsqrt(ax*ax + ay*ay + az*az)
              abb = dsqrt(bx*bx + by*by + bz*bz)
              abd = dsqrt(dx*dx + dy*dy + dz*dz)
              dfakt = (aba + abb) * 0.5d0/abd
              D(1,3) = dx * dfakt
              D(2,3) = dy * dfakt
              D(3,3) = dz * dfakt
```



```
      ! . . . Inversion der Matrix D
              call invert(D)
      ! . . . Berechnung der Loesung mit x = Inv.(D) * (- Merkur-Koord.)
              x1 = - D(1,1) * xyr(1) - D(1,2) * xyr(2) - D(1,3) * xyr(3)
              x2 = - D(2,1) * xyr(1) - D(2,2) * xyr(2) - D(2,3) * xyr(3)
              x3 = - D(3,1) * xyr(1) - D(3,2) * xyr(2) - D(3,3) * xyr(3)
      ! . . . Koordinaten der Sonnenposition in Giza
              xyr(31) = x1 * pyr(11) + x2 * pyr(12) + x3 * pyr(7)
              xyr(32) = x1 * pyr(14) + x2 * pyr(15) + x3 * pyr(8)
              xyr(33) = x1 * pyr(17) + x2 * pyr(18) + x3 * pyr(9)
      ! . . . Massstabsfaktor
              xyr(35) = AE * dsqrt((xyr(12)**2 + xyr(15)**2 + xyr(18)**2)/ &
                                   (pyr(12)**2 + pyr(15)**2 + pyr(18)**2))
            endif

      !.....Dreidimensionale Berechnung (x-, y- und z-Koordinate)
      !     mit Hilfe des Fit-Programms FITEX. Die Konstellation der Plane-
      !     ten wird durch Translation, Rotation und Groessenaenderung mit
      !     der Anordnung der Pyramiden bzw. der Kammern in der Cheops-Pyra-
      !     mide zur Deckung gebracht. Anschliessend wird die resultierende
      !     Transformation auf die Sonnenposition (Koordinatenursprung)
      !     angewendet.
            if (ison==5) then
              istart = 0; ke = 0
              if (iter/=0) then; do iu=ix,6,5; write(iu,*); enddo; endif

      ! . . . Koordinatentransformation --> y(i)
              do
                do i=1,m; y(i) = xyr(i); enddo
                call translat(x(1),x(2),x(3),y)
                call rotmat(5,x(4),x(5),x(6),y)
                call mastab(x(7),y)
                if (istart==0) then
                  do i=1,n; xsta(i) = x(i); enddo
                  do i=1,m; ysta(i) = y(i); enddo
                endif

      ! . . . . Die Fehlerquadrate dabs(F)**2
                w(4) = z0
                do i=1,m
                  f(i) = y(i) - z(i)
                  w(4) = w(4) + f(i)*f(i)
                enddo
                istart = istart + 1

      ! . . . . Ausgabe der Iterationen (Aufruf von FITEX)
                do iu=ix,6,5
                 if (iter/=0) write(iu,152)iw(3),iw(4),w(3),w(4),(x(i),i=1,n)
                enddo
                call fitex(ke,m,n,f,x,e,w,iw)
                if (ke/=1) exit
              enddo

      ! . . . Ausgabe der Ergebnisse
              if (iter/=0) then
                do iu=ix,6,5
                  write(iu,153) ke,iw(3),iw(4),w(3),w(4)
                  j2 = n+n
                  write(iu,154) x,(w(4+j),j=1,j2)
```

```
3895              if (w(5)==z0) go to 10
                  j2=4+j2
      !c          do i=1,n
      !c            j1=j2+1; j2=j1+i-1
      !c            write(iu,154) (w(j),j=j1,j2)
3900  !c          enddo
         10       write(iu,*)
                  write(iu,'(''  start   x(1..'',i1,''):'',7F13.3)') &
                    n,(xsta(i),i=1,3),(xsta(i)*gdpi,i=4,6),xsta(7)
                  write(iu,'(''      "    y(1..'',i1,''):'',9f13.3)') &
3905                m,(ysta(i),i=1,m)
                  write(iu,'(''  results x(1..'',i1,''):'',7f13.3)') &
                    n,(x(i),i=1,3),(x(i)*gdpi,i=4,6),x(7)
                  write(iu,'(''      "    y(1..'',i1,''):'',9f13.3/)') &
                    m,(y(i),i=1,m)
3910            enddo
              endif

      ! . . . Berechnung der Sonnenposition im Pyramidengelaende mit Hilfe
      !       der gerade bestimmten Parameter x(1)..x(7) durch Transforma-
3915  !       tion des Koordinatenursprungs (Sonne)
              do i=1,m; y(i) = z0; enddo
              call translat(x(1),x(2),x(3),y)
              call rotmat(5,x(4),x(5),x(6),y)
              call mastab(x(7),y)
3920          xyr(31) = y(1)
              xyr(32) = y(2)
              xyr(33) = y(3)
              xyr(35) = AE/x(7)
            endif
3925
            if (ison>=4) then
      !.......Korrektur der Koordinaten (1/4 Hoehe oder ganze Hoehe der
      !       3. Pyramide bzw. Positionskoordinaten der Felsenkammer)
              xyr(31) = xyr(31) + xp3
3930          xyr(32) = xyr(32) + yp3
              xyr(33) = xyr(33) + zp3

      ! . . . Fehlerabschaetzung fuer die Sonnenposition
      !c      if (ison==4) then
3935            dcm = dsqrt((xyr(31)-rcm(1))**2 + (xyr(32)-rcm(2))**2 &
                            + (xyr(33)-rcm(3))**2)
                qu = dcm
                if (dcm<dmi) qu = dmi * ((dcm/dmi)**2 + 1.d0)*0.5d0
                xyr(34) = qu * xyr(36) * 1.d-2
3940  !c      else
      !c        xyr(34) = dsqrt(w(4))
      !c      endif
            endif

3945        return
          152 format(5x,2i5,1p,9e13.5)
          153 format(3i5,1p,8e23.15)
          154 format('  ',1p,6e13.5)
            end subroutine
3950
            subroutine invert(a)
      !-----Inversion der 3x3-Matrix a, d.h. a -> inv(a)---------------------
            implicit double precision (a-h,o-z)
```



```
            dimension :: a(3,3),b(3,3)
3955  ! . . Die Kofaktoren
            b(1,1) =   a(2,2)*a(3,3) - a(3,2)*a(2,3)
            b(1,2) = - a(2,1)*a(3,3) + a(3,1)*a(2,3)
            b(1,3) =   a(2,1)*a(3,2) - a(3,1)*a(2,2)
            b(2,1) = - a(1,2)*a(3,3) + a(3,2)*a(1,3)
3960        b(2,2) =   a(1,1)*a(3,3) - a(3,1)*a(1,3)
            b(2,3) = - a(1,1)*a(3,2) + a(3,1)*a(1,2)
            b(3,1) =   a(1,2)*a(2,3) - a(2,2)*a(1,3)
            b(3,2) = - a(1,1)*a(2,3) + a(2,1)*a(1,3)
            b(3,3) =   a(1,1)*a(2,2) - a(2,1)*a(1,2)
3965  ! . . Kehrwert der Determinante und Transponieren
            dei = 1.d0/(a(1,1)*b(1,1) + a(1,2)*b(1,2) + a(1,3)*b(1,3))
            do i=1,3; do j=1,3; a(i,j) = b(j,i)*dei; enddo; enddo
            end subroutine

3970        subroutine rotmat(iachse,w1,w2,w3,a)
      !-----Erstellung der Dreh-Matrix und Multiplikation--------------------
      !     3 Vektoren fuer Merkur bis Erde: a(1..9) --> a(1..9)
      !     iachse = 1-3:  Drehung um x-, y- oder z-Achse (Winkel w1)
      !
3975  !                                    (  cos w1  sin w1   0  )
      !                       z.B.   Dz(w1) = ( -sin w1  cos w1   0  )
      !                                    (    0       0      1  )
      !
      !     iachse = 4:    Drehung um Knotenlinie (Winkel w1, w2)
3980  !     iachse = 5:    Drehung um beliebige Achse (Winkel w1, w2
      !                    und w3: die Eulerschen Winkel)
            implicit double precision (a-h,o-z)
            dimension :: a(9),b(9),D(3,3)
            z0 = 0.d0
3985        one = 1.d0
            s1 = dsin(w1); c1 = dcos(w1)
            if (iachse<=3) then
              do j=1,3; do i=1,3; D(i,j) = z0; enddo; enddo
              if (iachse==1) then
3990            D(1,1) = one           ! axis 1
                D(2,2) = c1
                D(2,3) = s1
                D(3,2) = - s1
                D(3,3) = c1
3995          else
                D(1,1) = c1
                if (iachse==2) then
                  D(1,3) = s1          ! axis 2
                  D(2,2) = one
4000              D(3,1) = - s1
                  D(3,3) = c1
                else
                  D(1,2) = s1          ! axis 3
                  D(2,1) = - s1
4005              D(2,2) = c1
                  D(3,3) = one
                endif
              endif
            else
4010          s2 = dsin(w2); c2 = dcos(w2)
              if (iachse==4) then
                D(1,1) = - s1 * s1 * (one - c2) + one    ! axis 4
```

```
              D(1,2) =   s1 * c1 * (one - c2)
              D(1,3) = - s1 * s2
              D(2,1) =   s1 * c1 * (one - c2)
              D(2,2) = - c1 * c1 * (one - c2) + one
              D(2,3) =   c1 * s2
            else
              s3 = dsin(w3); c3 = dcos(w3)
              D(1,1) =   c1 * c3 - s1 * c2 * s3        ! axis 5
              D(1,2) =   s1 * c3 + c1 * c2 * s3
              D(1,3) =   s2 * s3
              D(2,1) = - c1 * s3 - s1 * c2 * c3
              D(2,2) = - s1 * s3 + c1 * c2 * c3
              D(2,3) =   s2 * c3
            endif
            D(3,1) =   s1 * s2
            D(3,2) = - c1 * s2
            D(3,3) =   c2
          endif

      ! . . Ausfuehrung der Transformation (Merkur-, Venus- und Erdposition)
      !c    do i = 1,3; write(6,'(3f13.8)')(D(i,j),j=1,3); enddo
            do i=1,9; b(i) = z0; enddo
            do k=0,6,3
              do i=1,3
                do j=1,3
                  b(k+i) = b(k+i) + D(i,j)*a(j+k)
                enddo
              enddo
            enddo
            do i=1,9; a(i) = b(i); enddo
      !c    write(6,'(a12,3f13.8)') '  Mercury : ',(a(j),j=1,3)
      !c    write(6,'(a12,3f13.8)') '  Venus   : ',(a(j),j=4,6)
      !c    write(6,'(a12,3f13.8)') '  Earth   : ',(a(j),j=7,9)
            end subroutine

            subroutine translat(a1,a2,a3,a)
      !-----Translation der Positionen der 3 Planeten-----------------------
      !     3 Vektoren a(1..9) --> a(1..9)
            implicit double precision (a-h,o-z)
            dimension :: a(9)
            do i=1,7,3
              a(i) = a(i)+a1; a(i+1) = a(i+1)+a2
              a(i+2) = a(i+2)+a3
            enddo
            end subroutine

            subroutine mastab(zmas,a)
      !-----Massstabsaenderung----------------------------------------------
      !     3 Vektoren a(1..9) --> a(1..9)
            implicit double precision (a-h,o-z)
            dimension :: a(9)
            do i=1,9
              a(i) = zmas * a(i)
            enddo
            end subroutine

            subroutine transfo(irb,rku)
      !-----Transformation ins Merkurbahn-System (Venusbahn-System)----------
      !     re(1..9) --> re(1..9),  xyr(1..9) --> xyr(1..9)
```



```
      !     Die Transformationen A, B und C liefern dasselbe Ergebnis.
      !     Die Eingabewinkel ao, ai, at sind im Modul "base" gespeichert.
            use base
            implicit double precision (a-h,o-z)
            dimension :: xyt(9),rku(3)
            pi2 = pi * 2.d0
            if (irb>=2 .and.irb<=4) then
              ao = (re(34) - re(1))*pidg
            else
              ao = (re(40) - re(1))*pidg
            endif
            if (ao<z0) ao = ao + pi2
            if (ao>pi2) ao = ao - pi2
      !c    write(6,'(a10,f23.8)') '  re(4)   ',re(4)
      !c    write(6,'(a10,f23.8)') '  re(40)  ',re(40)
            if (irb>=2 .and.irb<=4) then
              ai = dabs(datan(xyr(3)/(xyr(1)*dsin(ao))))
            else
              rxy = dsqrt(xyr(4)*xyr(4) + xyr(5)*xyr(5))
              aov = (re(40) - re(4))*pidg
              ai  = dabs(datan(xyr(6)/(rxy*dsin(aov))))
            endif
            at = dasin(dsin(ao)/dsqrt(1.d0-(dsin(ai)*dcos(ao))**2))+ao-pi
            a1 = ao; a2 = ai
            a3 = at
      !c    write(6,'(a12,3f13.8)') '  Mercury : ',(xyr(j),j=1,3)
      !c    write(6,'(a12,3f13.8)') '  Venus   : ',(xyr(j+3),j=1,3)
      !c    write(6,'(a12,3f13.8)') '  Earth   : ',(xyr(j+6),j=1,3)
            do i=1,9; xyt(i) = xyr(i); enddo

      !.....Transformation A  -->  Dz(at) * K(ao,ai)
      !     (Reihenfolge der Matrizen von rechts nach links!)
            if (irb==2 .or.irb==5) then
      ! . . . Matrix K(ao,ai)
              call rotmat(4,a1,a2,z0,xyt)
      ! . . . Matrix Dz(at)
              if (irb==5) then
                at = datan(xyt(2)/xyt(1))
                a3 = at
              endif
              call rotmat(3,a3,z0,z0,xyt)
            endif

      !.....Transformation B  -->  Dz(at-ao) * Dx(ai) * Dz(ao)
            if (irb==3) then
      ! . . . Matrix Dz(ao)
              call rotmat(3,a1,z0,z0,xyt)
      ! . . . Matrix Dx(ai)
              call rotmat(1,a2,z0,z0,xyt)
      ! . . . Matrix Dz(at-ao)
              call rotmat(3,a3-a1,z0,z0,xyt)
            endif

      !.....Transformation C  -->  R(ao,ai,at-ao)
            if (irb==4) then
      ! . . . Matrix R(ao,ai,at-ao)
              call rotmat(5,a1,a2,a3-a1,xyt)
            endif
```

```
      ! . . Ruecktransformation in Kugelkoordinaten
            do i=1,9; xyr(i) = xyt(i); enddo
            do i=1,3
              k=3*(i-1)
4135          xy1 = xyr(k+1)
              xy2 = xyr(k+2)
              xy3 = xyr(k+3)
              call kugelko(xy1,xy2,xy3,rku)
              do j=1,3
4140            re(k+j) = rku(j)
              enddo
            enddo
            end subroutine

4145        subroutine kugelko(r1,r2,r3,rku)
      !-----Umrechnung in Kugelkoordinaten rku(1)..rku(3)--------------------
      !     (Index von rku  1: phi, 2: theta, 3: r)
            use base, only : gdpi
            implicit double precision (a-h,o-z)
4150        dimension :: rku(3)
            ra = dsqrt(r1*r1 + r2*r2)
            rku(1) = datan(r2/r1)*gdpi
            rku(2) = datan(r3/ra)*gdpi
            rku(3) = dsqrt(ra*ra + r3*r3)
4155        if (r1<0.d0) rku(1) = rku(1) + 180.d0
            if (rku(1)<0.d0) rku(1) = rku(1) + 360.d0
            end subroutine

            subroutine aphelko(imod,ivers,iaph,ipla, &
4160          ison,ijd,io,iop0,ix,dh3,x,y,rcm,dmi)
      !-----Berechnung der "Merkur-Aphelposition" in Giza--------------------
      !     fuer Konstell. 13, 14, sowie "quick start option" 371 und 372.
      !     Die Berechnung kann mit VSOP87A (ivers=1) und VSOP87C (ivers=3)
      !     durchgefuehrt werden. Die Ortsabweichungen im Pyramidengelaende
4165  !     zwischen beiden Versionen liegen fuer Konst. 13 bzw. 14 bei ca.
      !     10 cm und 5 mm, bei der "Schatten-Konstellation 12" bei ca. 4 mm.
      !     Sollte sich an den Zeitpunkten dieser Konstellationen etwas aen-
      !     dern, sind die astron. Aphelkoordinaten in "aphelm" anzupassen.
            use base
4170        implicit double precision (a-h,o-z)
            dimension :: aphelm(18),x(7),y(9),rcm(3)

      !.....Sphaerische ekliptikale Koordinaten L, B und r des Merkur-Aphels
      !     fuer Konst. 13 und 14 jeweils fuer J2000.0 und Ekl. der Epoche
4175  !     und fuer "Schatten-Konstellation 12" mit J2000.0 (Option 372)
      !     und Ekliptik der Epoche (Option 371).
      !
      ! . . A. Berechnung mit Gl. (7.1) --> Konst. 13:  JDE = 5909973.28368
      !                                     Konst. 14:  JDE =  671046.63581
4180  !                           Optionen 371 und 372:  JDE = 2849071.14941
      !     data aphelm/
      !       272.2596751d0, -5.4263369d0, 0.4672908784d0,   (K.13, VSOP87A)
      !        46.8137077d0, -6.4048699d0, 0.4670482474d0,   (K.13, VSOP87C)
      !       249.5729904d0, -1.9354192d0, 0.4662991040d0,   (K.14, VSOP87A)
4185  !       182.1787524d0, -1.3530604d0, 0.4662950222d0,.. (K.14, VSOP87C)
      !
      ! . . B. r(Mer.) optimiert --> Konst. 13 (VSOP87A): JDE = 5909973.264
      !        (r maximal fuer Aphel)          (VSOP87C): JDE = 5909973.255
      !                             Konst. 14 (VSOP87A/C): JDE =  671046.632
```



```
4190        data aphelm/272.2054713d0, -5.4229877d0, 0.4672909313d0, &
                         46.7345218d0, -6.4007584d0, 0.4670483641d0, &
                        249.5625348d0, -1.9341303d0, 0.4662991059d0, &
                        182.1682931d0, -1.3518259d0, 0.4662950244d0, &
                        258.9945271d0, -3.6947988d0, 0.4667842406d0, &
4195                    274.2350325d0, -3.8355115d0, 0.4667842399d0/
            if ((ijd==13 .or.ijd==14 .or.iop0==371 .or.iop0==372).and. &
             imod<=2 .and.ison==5 .and.iaph==1 .and.ipla==1 .and.io==2) then
              if (ijd==13 .and.ivers==1) j = 1
              if (ijd==13 .and.ivers/=1) j = 4
4200          if (ijd==14 .and.ivers==1) j = 7
              if (ijd==14 .and.ivers/=1) j = 10
              if (iop0==371) j = 16
              if (iop0==372) j = 13
              do i=4,6; re(i) = aphelm(j+i-4); enddo
4205  !       Umrechnung in kartesische Koordinaten
              call kartko(ison)
      !       Koordinatentransformation: Weltraum --> Pyramidengelaende
              do i=4,6; y(i) = xyr(i); enddo
              call translat(x(1),x(2),x(3),y)
4210          call rotmat(5,x(4),x(5),x(6),y)
              call mastab(x(7),y)
              y(6) = y(6) + dh3
      !       Fehler in Metern (dr)
              dcm = dsqrt((y(4)-rcm(1))**2 + (y(5)-rcm(2))**2 &
4215                    + (y(6)-rcm(3))**2)
              qu = dcm
              if (dcm<dmi) qu = dmi * ((dcm/dmi)**2 + 1.d0)*0.5d0
              dr = qu * xyr(36) * 1.d-2
      !       Ausgabe des Ergebnisses
4220          do iu=ix,6,5
                write(iu,'(''  Mercury aphelion coordinates [m]:'', &
                  & f13.2,2f10.2,f9.2)') y(4),y(5),y(6),dr
                call linie(iu,1)
              enddo
4225        endif
            end subroutine

            subroutine plako(diff,ipla,ijd,ik,ison,ipos, &
              rcm,x,y,ort,rp,dd,dn,dss,pla,plan,emp,text,tt,titab, &
4230          is12,dmi,zjda,zjde,ivers,md,ix,prec,lu,r,ierr,rku)
      !-----Koordinaten fuer Merkur bis Neptun-------------------------------
      !     und Berechnung der "Planetenpositionen" im Giza-Gelaende fuer
      !     Konst. 1-14 mit ison = 5 (FITEX) und imod = 2 (VSOP87-Vollv.).
      !     Zusaetzlich:
4235  !     Spezialausgabe fuer Konst. 12 mit iuniv = 1 (TT) und iout = 3
      !     (spezial). In diesem Fall sind nur noch folgende Parameter
      !     variierbar: ipla (Pyr.- oder Kammerpositionen), imod (VSOP87
      !     Voll- oder Kurzv.), ivers (VSOP87A oder VSOP87C, bei Vollv.)
      !     und ihi (z-Koordinate)
4240        use base
            implicit double precision (a-h,o-z)
            dimension :: diff(9),r(6),rku(3),md(0:9),x(7),y(9),rcm(3)
            dimension :: ort(0:9,4),rp(3,4),zjda(4)
            character(2) :: dd,dn,dss
4245        character(3) :: pla(0:9),line
            character(7) :: emp
            character(10) :: plan(0:9)
            character(18) :: date(4)
```

```
      character(23) :: text(0:9),tt(2)
4250  character(49) :: titab
      data date/'date of chambers: ','date of syzygy:   ', &
                'date of transit:  ','date of pyramids: '/
      data line/'---'/

4255  ! . . Tabellenkopf
      do iu=ix,6,5
        if (is12==0) then
          write(iu,*); call linie(iu,1)
          write(iu,*)'pla.   x[AU]     y[AU]     z[AU]      L       ',&
4260        'B       r[AU]     Lm-L     dev.'
          call linie(iu,2)
        else
          write(iu,'(/27x,''Celestial positions in Giza'')')
          call linie(iu,1)
4265      write(iu,*)' body          x[m]      y[m]     z[m]', &
            '   dr[m]       latitude N    longitude E'
        endif
      enddo

4270  !.....Positionen von Merkur bis Neptun und Sonne im Pyramiden-
      !     gelaende und im System innerhalb der Cheops-Pyramide (nur
      !     VSOP87-Vollversion)
      icm = 1; imax = 8; if (ivers==1) imax = 9
      if (is12/=0) imax = 4
4275  icmax = 1; if (is12/=0) icmax = 4
     10 if (is12/=0) then
        zjde = zjda(icm)
        do iu=ix,6,5
          call linie(iu,2)
4280      write(iu,'(4x,a18,''JDE ='',f14.5)') date(icm),zjda(icm)
          call linie(iu,2)
        enddo
      endif
      if (is12/=0 .and.icm==1) then
4285    if (ipla==1) then
          call geoko(ort(0,1),-ort(0,2),ipla,iB1,zB2,iL1,zL2)
        else
          call geoko(ort(0,1),ort(0,3),ipla,iB1,zB2,iL1,zL2)
        endif
4290    do iu=ix,6,5
          write(iu,102) plan(0),(ort(0,j),j=1,4),iB1,zB2,iL1,zL2
        enddo
      endif
      do 20 id=1,imax
4295    call vsop2(zjde,ivers,id,md,ix,prec,lu,r,ierr,rku)
        dif = re(1) - rku(1); call reduz(dif,0,0)
        err = dif-diff(id); call reduz(err,0,0)
        if (is12==0) then
          do iu=ix,6,5
4300        if (id/=4 .and.(id<=6 .or.id==9)) then
             write(iu,100)pla(id),(r(i),i=1,3),(rku(i),i=1,3),dif,err
            else
             write(iu,101)pla(id),(r(i),i=1,3),(rku(i),i=1,3),dif,emp
            endif
4305      enddo
        endif
```



```
      !....."Planetenpositionen" im Giza-Gelaende (kartesische Koord.)
            if (((ijd>=1 .and.ijd<=14).or.(ik==4519 .and.ipla==1) &
4310           .or.(ik==4518 .and.ipla==2)).and.ison==5) ipos = 1
            if (ipos==1) then
              if (id==1) then
                do j=1,3; y(j) = rku(j); enddo
              endif
4315          do j=1,3; re(j+3) = rku(j); enddo
              call kartko(ison)
              do j=4,6; y(j) = xyr(j); enddo
              call translat(x(1),x(2),x(3),y)
              call rotmat(5,x(4),x(5),x(6),y)
4320          call mastab(x(7),y)
              do j=1,3; ort(id,j) = y(3+j) + rp(3,j); enddo
      !       Genauigkeit der "Planetenpositionen"
              if (id<=3 .and.is12==0) then
                ort(id,4) = dsqrt((ort(id,1)-rp(4-id,1))**2 &
4325                            + (ort(id,2)-rp(4-id,2))**2 &
                                + (ort(id,3)-rp(4-id,3))**2)
              elseif (id==9 .and.is12==0) then
                ort(id,4) = dsqrt((ort(id,1)-rp(1,1))**2 &
                                + (ort(id,2)-rp(1,2))**2 &
4330                            + (ort(id,3)-rp(1,3))**2)
              else
                dcm = dsqrt((ort(id,1)-rcm(1))**2 &
                          + (ort(id,2)-rcm(2))**2 &
                          + (ort(id,3)-rcm(3))**2)
4335            qu = dcm
                if (dcm<dmi) qu = dmi * ((dcm/dmi)**2 + 1.d0)*0.5d0
                ort(id,4) = qu * xyr(36) * 1.d-2
              endif
      !       Geographische Koordinaten (Laenge und Breite) der
4340  !       transformierten Sonnen- und Planetenpositionen
              if (is12/=0) then
                if (ipla==1) then
                  call geoko(ort(id,1),-ort(id,2),ipla,iB1,zB2,iL1,zL2)
                else
4345              call geoko(ort(id,1),ort(id,3),ipla,iB1,zB2,iL1,zL2)
                endif
                do iu=ix,6,5
                 write(iu,102) plan(id),(ort(id,j),j=1,4),iB1,zB2,iL1,zL2
                enddo
4350          endif
            endif
         20 enddo
      !     Ruecksprung zum naechsten Planeten
            icm = icm + 1
            if (icm<=icmax) go to 10

      ! . . Weitere Ergebnis-Ausgabe
            if (ipos==1 .and.is12==0) then
              text(2) = tt(ipla)
4360          do iu=ix,6,5
                call linie(iu,1)
                write(iu,'(''  Celestial pos. in Giza'',4x,a49)')titab
                call linie(iu,2)
                write(iu,'(''  Local coordinates'',9x,''Sun       '', &
4365              & f10.2,2f10.2,f9.2)') (ort(0,j),j=1,4)
              enddo
```

```
            do i=1,imax
              dd = dn
              if ((i>=1 .and.i<=3).or.i==9) dd = dss
4370          do iu=ix,6,5
                write(iu,'(a23,5x,a10,3f10.2,f9.2,a2)') &
                text(i),plan(i),(ort(i,j),j=1,4),dd
              enddo
            enddo
4375      endif
          do iu=ix,6,5; call linie(iu,1); enddo

          return
        100 format(1x,a3,3f10.6,f9.4,f8.4,f10.6,2f9.4)
4380    101 format(1x,a3,3f10.6,f9.4,f8.4,f10.6,f9.4,1x,a7)
        102 format(2x,a10,f9.2,f10.2,f9.2,f7.2,i8,f9.4,i6,f8.4)
      ! . . Groessere Stellenanzahl fuer Schnellstart-Optionen 3 und 8
      !f100 format(2x,a3,f11.6,2f10.6/28x,f13.7,f11.7,f14.10/58x,f13.7,f8.3)
      !f101 format(2x,a3,f11.6,2f10.6/28x,f13.7,f11.7,f14.10/58x,f13.7,a8)
4385      end subroutine

          subroutine geoko(x,y,ipla,iB1,zB2,iL1,zL2)
      !-----Berechnung der geographischen Koordinaten-----------------------
      !     (iB1,zB2 und iL1,zL2, jeweils in Grad und Minuten)
4390      use base, only : pi,pidg,R3a,R3p
          implicit double precision (a-h,o-z)

      ! . . Erdumfang ueber Pole. Anstelle von Ue = 40008 km folgt
      !     Ellipsenumfang nach Srinivasa Ramanujan.
4395      zl = 3.d0*((R3a-R3p)/(R3a+R3p))**2
          Ue = pi*(R3a+R3p) * (1.d0 + zl/(10.d0 + dsqrt(4.d0-zl)))
      !     Geographische Position des Koordinatenursprungs (Pyr./Kam.)
          if (ipla==1) then
            zB0 = 29.972530d0  !  Zentrum der Mykerinos-Pyramide
4400        zL0 = 31.128243d0  !  (Pyramiden-Koordinaten)
          else
            zB0 = 29.979200d0  !  Mittelachse der Ostwand
            zL0 = 31.134276d0  !  der Koeniginnenkammer
          endif
4405
      ! . . Geographische Breite (zB)
          dBa = 360.d0 * x/Ue
          zBa = zB0 + dBa
          call geokar(zBa,ua,va)
4410      call geokar(zB0,u0,v0)
          xa  = dsqrt((ua-u0)**2 + (va-v0)**2)
          dB  = dBa * dabs(x/xa)
          zB  = zB0 + dB
          iB1 = idint(zB)
4415      zB2 = dmod(zB,1.d0)*60.d0

      ! . . Geographische Laenge (zL)
          zBm = 0.5d0*(zB + zB0)
          call geokar(zBm,um,vm)
4420      dL  = y/(pidg*um)
          zL  = zL0 + dL
          iL1 = idint(zL)
          zL2 = dmod(zL,1.d0)*60.d0
          end subroutine
4425
```



```
          subroutine geokar(B,u,v)
      !-----Abstand eines Punktes der geographischen Breite B---------------
      !     zur Erdachse (u) und zur Aequatorebene (v) (kartesische Koord.)
          use base, only : pidg,R3a,R3p
4430      implicit double precision (a-h,o-z)
          u = R3a/dsqrt(1.d0 + (dtan(B*pidg)*R3p/R3a)**2)
          v = R3p*dsqrt(1.d0 - (u/R3a)**2)
          end subroutine

4435      subroutine reduz(a,i,j)
      !-----Winkelreduzierung  a --> a  (z.B. 387 Grad --> 27 Grad)----------
      !     i = 0/1: dezimale Grad/ Bogenmass
      !     j = 0:   a --> -180...180 Grad
      !     j = 1:   a -->    0...360 Grad
4440      use base, only : pidg,gdpi
          implicit double precision (a-h,o-z)
          u360 = 360.d0
          z1 = 1.d0
          if (a<0.d0) z1 = -1.d0
4445      if (i/=0) a = a*gdpi
          ab = dabs(a)
          if (ab>u360) ab = dmod(ab,u360)
          if ((j==0 .and.ab>180.d0).or. &
              (j==1 .and.a<0.d0)) ab = ab - u360
4450      a = z1 * ab
          if (i/=0) a = a * pidg
          end subroutine

          subroutine memo(zz1,zz2,zz3,zz4,zz5,zz6,zz7,zmem,ik,imem)
4455  !-----Ergebnis-Parameter merken---------------------------------------
          use base, only : re
          implicit double precision (a-h,o-z)
          dimension :: zmem(78)
          zmem(1) = zz1
4460      zmem(2) = zz2
          zmem(3) = zz3
          zmem(4) = zz4
          zmem(5) = zz5
          zmem(6) = zz6
4465      zmem(7) = zz7
          do i=1,12; zmem(10+i) = re(i); enddo
          do i=31,78; zmem(i) = re(i); enddo
          imem = ik
          end subroutine
4470
          subroutine info
      !-----Information zu den Copyrights (aus der Datei "inpdata.t")--------
          character(70) :: itext(37)
          open(unit=10,file='inpdata.t')
4475      do i=1,105; read(10,*); enddo
          do i=1,37; read(10,*) itext(i); enddo
          close(10)
          write(6,'(///37(5x,a70/))') (itext(i),i=1,37)
          end subroutine
4480
          subroutine titel1(iaph,ijd,ia,ison,ipla, &
            ilin,isep,nurtr,iuniv,is12,iop0)
      !-----Haupttitel und Untertitel---------------------------------------
          implicit double precision (a-h,o-z)
```

```
      write(ia,*)
      if (iop0==350) then
        write(ia,'(20x,A20,A22)')'4 PLANETS IN A LINE ', &
             '(SYZYGY), 17. MAY 3088'
        go to 20
      elseif (iop0==351) then
        write(ia,'(17x,A16,A31)')'MERCURY TRANSIT ', &
             '(MIN. SEPARATION), 18. MAY 3088'
        go to 20
      elseif (iop0==360) then
        write(ia,'(18x,A14,A32)')'VENUS TRANSIT ', &
             '(MIN. SEPARATION), 18. DEC. 3089'
        go to 20
      elseif (iop0==361) then
        write(ia,'(19x,A20,A23)')'3 PLANETS IN A LINE ', &
             '(SYZYGY), 23. DEC. 3089'
        go to 20
      elseif (iop0==370) then
        write(ia,'(24x,A34)')'SEARCH FOR "SHADOW-CONSTELLATIONS"'
        go to 10
      elseif (iop0==371 .or.iop0==372) then
        write(ia,'(16x,A20,A29)')'PRECEDING "SHADOW-CO', &
             'NSTELLATION" 12, 22. MAY 3088'
        go to 20
      endif
      if (ipla==1) write(ia,*)'                PLANETS IN ', &
          'ALIGNMENT WITH THE PYRAMIDS OF GIZA'
      if (ipla==2) write(ia,*)'          PLANETS IN ALIGNME', &
          'NT WITH THE CHAMBERS OF THE CHEOPS PYRAMID'
      if (ipla==3) then
        if (ilin>=3) write(ia,'(28x,a11,a15)')'PLANETS IN ', &
          'A LINE (SYZYGY)'
        if (ilin==1) write(ia,'(31x,a19)')'TRANSITS OF MERCURY'
        if (ilin==2) write(ia,'(32x,a17)')'TRANSITS OF VENUS'
      endif
   10 if (ipla/=3 .and.is12==0) then
        if (iaph==1 .and.ijd/=13 .and.ijd/=14) &
          write(ia,'(30x,a21)')'(Mercury at aphelion)'
        if (iaph==2 .and.ijd/=13 .and.ijd/=14) &
          write(ia,'(29x,a23)')'(Mercury at perihelion)'
        if (iaph==3 .or.(iaph==1 .and.(ijd==13 .or.ijd==14))) &
          write(ia,'(29x,a23)')'(Mercury near aphelion)'
        if (iaph==4 .or.(iaph==2 .and.(ijd==13 .or.ijd==14))) &
          write(ia,'(28x,a25)')'(Mercury near perihelion)'
        if (iaph==5) write(ia,'(24x,a34)') &
           '(time not restricted, F minimized)'
      elseif (ipla/=3 .and.is12/=0) then
        if (ipla==1) write(ia,'(17x,a48)') &
          '(more positions - coordinate system of pyramids)'
        if (ipla==2) write(ia,'(17x,a48)') &
          '(more positions - coordinate system of chambers)'
      else
        if (isep==1) then
          if (ison/=5) then
            write(ia,'(14x,a21,a33)')'(eclipt. longitudes, ', &
              'all within an angular range, JDE)'
          else
            if (ilin>=3) then
              if (nurtr==1) then
```



```
                write(ia,'(13x,a18,a37)')'(angular range of ', &
                'eclipt. longitudes dL minimized, JDE)'
              else
                write(ia,'(5x,a18,a52)')'(angular range of ', &
                'eclipt. longitudes dL minimized, only transits, JDE)'
              endif
            else
              write(ia,'(11x,a18,a41)')'(equal eclipt. lon', &
                'gitudes for Earth und transit planet, TT)'
            endif
          endif
        elseif (isep==2) then
          write(ia,'(14x,a54)') &
            '(minimum separation, without travel time of light, TT)'
        else
          if (iuniv==1) then
            write(ia,'(17x,a48)') &
              '(geocentric transit phases, terrestrial time TT)'
          else
            write(ia,'(18x,a46)') &
              '(geocentric transit phases, universal time UT)'
          endif
        endif
      endif
    20 if (isep/=4) then
        write(ia,'(34x,a8,i4,a2)')'< option',iop0,' >'
      else
        write(ia,'(11x,a8,i4,a47)')'< option',iop0, &
        ' >  (monitor line width minimal 148 characters)'
      endif
      end subroutine

      subroutine titel2(ia,imod,ivers,irb,ipla, &
        ison,ihi,iek,ijd,ika,iaph,ilin,ical,ak,zjde1,zjahr,delt, &
        dwi,dwikomb,dwi0,dwi2,dwi3,iamax,step,ikomb,zmin,zmax)
!-----Zwei weitere Titelzeilen-----------------------------------------
      implicit double precision (a-h,o-z)
      dimension :: ida(7),da(7)
      character(5)  :: ca(2),dmo
      character(7)  :: cal(2)
      character(10) :: wd
      character(15) :: text0
      character(27) :: text1
      character(19) :: text2
      character(8)  :: text3(0:6)
      character(25) :: text4
      character(22) :: text5(2)
      data ca/' (c1)',' (c2)'/,cal/'Gregor.','Julian.'/
      data text3/ '         ',' E-V-M, ',' E-M-V, ', &
       ' V-E-M, ',' V-M-E, ',' M-E-V, ',' M-V-E, '/
      data text5/',  only Greg. calendar',',  Jul./Greg. calendar'/
      if (imod==1)  text1 = '  VSOP87D short ver.(Meeus)'
      if (imod==2 .and.ivers==1) text1='  VSOP87A (2005) full ver.,'
      if (imod==2 .and.ivers==3) text1='  VSOP87C (2005) full ver.,'
      if (imod==3)  text1 = '  "Keplers equation",      '
      if (ikomb==1 .and.ivers==1)text1='  VSOP87A, comb. search,   '
      if (ikomb==1 .and.ivers==3)text1='  VSOP87C, comb. search,   '
      if (ivers==1) text2 = '  standard J2000.0,'
      if (ivers==3) text2 = '  ecliptic of date,'
```

```
      if (ipla/=3) then
       if (irb==1) then
4605    if (ison==1) text4 = ' "Sun" south of Myker. P.'
        if (imod==3 .and.ipla==2) text4 ='"Sun" south of sub. cham.'
        if (ison==2) text4 = '"Sun" south of Chefren P.'
        if (ison==3) text4 = '"Sun position" free, 2D  '
        if (ipla==1) then
4610     if (ison==4 .and.ihi==1) text4 ='"Sun" free, 3D, base, SLE'
         if (ison==4 .and.ihi==2) text4 =' "Sun" free, 3D, C-M, SLE'
         if (ison==4 .and.ihi==3) text4 =' "Sun" free, 3D, top, SLE'
         if (ison==5 .and.ihi==1) text4 ='"Sun" free 3D base, FITEX'
         if (ison==5 .and.ihi==2) text4 ='"Sun" free 3D, C-M, FITEX'
4615     if (ison==5 .and.ihi==3) text4 ='"Sun" free 3D, top, FITEX'
        endif
        if (ipla==2) then
         if (ison==4 .and.ihi==1) text4 ='"Sun" free, 3D, East, SLE'
         if (ison==4 .and.ihi==2) text4 ='"Sun" free, 3D, mid., SLE'
4620     if (ison==4 .and.ihi==3) text4 ='"Sun" free, 3D, West, SLE'
         if (ison==5 .and.ihi==1) text4 ='"Sun" free 3D East, FITEX'
         if (ison==5 .and.ihi==2) text4 ='"Sun" free 3D mid., FITEX'
         if (ison==5 .and.ihi==3) text4 ='"Sun" free 3D West, FITEX'
        endif
4625   endif
       if (irb==2) text4 = '   ref. Mercury orbit (A)'
       if (irb==3) text4 = '   ref. Mercury orbit (B)'
       if (irb==4) text4 = '   ref. Mercury orbit (C)'
       if (irb==5) text4 = '    reference Venus orbit'
4630  else
       if (ilin==1) text4 = '     all Mercury transits'
       if (ilin==2) text4 = '       all Venus transits'
       if (ilin==3) text4 = 'linear c., Merc. to Earth'
       if (ilin==4) text4 = 'linear c. Mercury to Mars'
4635  endif
      write(ia,'(/a27,a19,a8,a25)') text1,text2,text3(ika),text4
      if (ipla/=3) then
       if (iek==1)  text0 ='  Ecl. north p/'
       if (iek==2)  text0 ='  Ecl. south p/'
4640   if (ison>=3 .or.iek==3) text0 ='  Ecl. N and S,'
      else;                    text0 ='  Period   (yea'
      endif
      if (ijd==15 .and.(imod/=2 .or.(imod==2 .and. &
         (iaph==3 .or.iaph==4)))) then
4645   if (ipla/=3) then
        if (ison<=2) then
         if (ikomb/=1) write(ia,'(a15,'' years'',f10.2, &
          & '' to'',f10.2,a5,''    angular range:'',f8.4,'' deg'')') &
          text0,zmin,zmax,ca(ical),dwi0
4650     if (ikomb==1) write(ia,'(a15,'' years'',f10.2, &
          & '' to'',f10.2,a5,'', angular r.:'',f6.2,''/'',f6.2, &
          & '' deg'')') text0,zmin,zmax,ca(ical),dwi,dwikomb
        else
         if (ikomb/=1 .and.iaph/=5) then
4655      write(ia,'(a15,'' years'',f10.2,'' to'',f10.2,a5, &
           &'',     tolerance F <='',f8.4,'' %'')') &
           text0,zmin,zmax,ca(ical),dwi0
         else
          write(ia,'(a15,'' years'',f10.2,'' to'',f10.2,a5, &
4660       & '', tolerance F <='',f6.2,''/'',f5.2,'' %'')') &
           text0,zmin,zmax,ca(ical),dwi,dwikomb
```



```
         endif
        endif
       else
4665    if (ilin>=3) then
         if (ikomb==1) write(ia,'(a15,''rs)'',f10.2, &
          & ''  to'',f10.2,a5,'',   angular r.:'',f6.2,''/'',f6.2, &
          & '' deg'')') text0,zmin,zmax,ca(ical),dwi,dwikomb
         if (ikomb/=1) write(ia,'(a15,''rs)'',f10.2,''  to'', &
4670      & f10.2,a5,3x,''   angular range:'',f8.4,'' deg'')') &
          text0,zmin,zmax,ca(ical),dwi0
        else
         write(ia,'(5x,a15,''rs) from'',f10.2,''  to'',f10.2,a22)') &
          text0,zmin,zmax,text5(ical)
4675     return
        endif
       endif
      else
       call ephim(1,iaph,ipla,ical,ak,iak,zjde1,zjahr,delt)
4680   if (ijd>=1 .and.ijd<=14) then
        write(ia,'(a15,''  constellation'',i3,'',   JDE ='', &
         & f15.5,'', year ='',f9.2,a5)')text0,ijd,zjde1,zjahr,ca(ical)
       else
        write(ia,'(a15,20x,''  JDE ='',f15.5,'', year ='',f9.2,a5)') &
4685      text0,zjde1,zjahr,ca(ical)
       endif
       if (iaph<=2) then
         call jdedate(zjde1,ical,ida,da,dmo)
         call weekday(zjde1,wd)
4690     k = 1
         if (zjde1>=0.d0 .and.zjde1<2299161.d0 .and.ical==2) k = 2
         if (zjde1>=1356183.d0 .and.zjde1<=5373484.d0) then
           write(ia,'(25x,''date ('',a7,'',TT) ='', &
             & f4.0,a5,i5,'','',i3,2('':'',i2),'','',A10)') &
4695         cal(k),da(7),dmo,(ida(i),i=3,6),wd
           return
         else
           write(ia,'(24x,''date ('',a7,'',TT) ='', &
             & f4.0,a5,i6,'','',i3,2('':'',i2),'','',A10)') &
4700         cal(k),da(7),dmo,(ida(i),i=3,6),wd
           return
         endif
       endif
      endif
4705  if (iaph==3 .or.iaph==4) then
        write(ia,'(''  Special search (interval), step number ='',i6, &
          & '',   step width ='',f7.3,'' hour(s)'')')iamax,24.d0*step
      endif
      if ((iaph==3 .or.iaph==4).and.ijd==15) then
4710    write(ia,'(''  Consider without printing by tolerance ='', &
          & f8.4)') dwi2
        write(ia,'(''  Print beyond aphelion (per.) by toler. ='', &
          & f8.4)') dwi3
      endif
4715  end subroutine

      subroutine tabe(iaph,imod,iek,ia,io, &
        ison,ipla,ilin,itran,is12,iop0,iout)
!-----Tabellenkopf------------------------------------------------------
4720  !     Bei Datumsberechnungen uebernimmt das Unterprogramm
```

```
      !     "zwischenzeile" die Tabellenueberschrift.
            implicit double precision (a-h,o-z)
            character(2) :: trs
            if (ilin>=3.) then
4725          write(ia,*)
              if (io==2 .and.imod/=3) call linie(ia,1)
            endif
            if (ipla==3) then
              trs = 'tr'
4730          if (itran==2 .or.ison/=5 .or.imod==3) trs = '  '
              if (ilin>=3) then
               if (ison==5) then
                write(ia,'('' co '',a2,''    k         JDE          year'', &
                  & ''   dt[days]  Lm-Lv   Lm-Le  Lm-Lma   dLmin'')')trs
4735           else
                write(ia,'('' co '',a2,''    k         JDE          year'', &
                  & ''   dt[days]  Lm-Lv   Lm-Le  Lm-Lma    dL'')')trs
               endif
              endif
4740        else
             if (ison<=2) then
               if (imod/=3 .and.iek/=3) then
                 write(ia,'('' con   k        JDE         year  '', &
                   & ''    Lm     Lm-Lv   Lm-Le    del1   del2   F[%]'')')
4745            else
                 write(ia,'('' con   k         JDE          year'', &
                   & ''        Lm     Lm-Lv   Lm-Le    del1   del2  P'')')
               endif
             else
               if (ison==3 .or.ison==4) then
                write(ia,'('' con    k      year     Lm     Lm'', &
                  & ''-Lv   Lm-Le  x-Sun  y-Sun  z-Sun  dr  P   F[%]'')')
                if (iaph==3 .or.iaph==4) then
                 write(ia,'(''      (  ~k       JDE           M'', &
4755               & ''        no.      "      "      "    "   "    "  )'')')
                endif
               endif
               if (ison==5) then
                if (iaph==3 .or.iaph==4 .or.iout/=3) then
4760             if (iaph/=5) then
                  write(ia,'('' con   k      year    Lm-Lv   Lm'', &
                    & ''-Le   e  it  x-Sun  y-Sun  z-Sun   dr  P   F[%]'')')
                 else
                  write(ia,'('' con   k         JDE          ye'', &
4765                & ''ar    e  it   x-Sun  y-Sun  z-Sun   dr  P  F[%]'')')
                 endif
                else
                 if (ipla==1) then
                  if (iaph/=5) then
4770               write(ia,'('' con   k      year      X5   M/1'', &
                    & ''0^7  h-Sun  x-Sun  y-Sun  z-Sun    dr  P   F[%]'')')
                   else
                    write(ia,'('' con   k       year   dt[days]  '', &
                     & ''  X5     M/10^7   x-Sun  y-Sun  z-Sun  P   F[%]'')')
4775               endif
                  else
                   if (iaph/=5) then
                    write(ia,'('' con   k      year      X5   M/1'', &
                     & ''0^9   h-Sun  x-Sun  y-Sun  z-Sun   dr  P   F[%]'')')
```



```
4780               else
                    write(ia,'('' con   k       year   dt[days]  '', &
                     & ''  X5     M/10^9   x-Sun  y-Sun  z-Sun  P   F[%]'')')
                   endif
                  endif
4785             endif
                 if (iaph==3 .or.iaph==4) then
                  if (iout==3) then
                   if (ipla==1) then
                    write(ia,'(''    (  JDE       dt[h]   X5   M/'', &
4790                & ''10^7  h-Sun    "      "      "    "   "    "  )'')')
                   else
                    write(ia,'(''    (  JDE       dt[h]   X5   M/'', &
                     & ''10^9  h-Sun    "      "      "    "   "    "  )'')')
                   endif
4795              else
                    write(ia,'(''      (  ~k       JDE           M'', &
                     & ''      "  "     "      "      "     "   "    "  )'')')
                  endif
                 endif
4800            endif
               endif
      !        (Output zum Vergleich mit den Pyramidenabstaenden)
               if (ilin>=3) then
4805            if (imod==3) then
                  call linie(ia,1)
                else
                  call linie(ia,io)
                endif
4810            if (io==2 .and.imod/=3 .and.is12==0) then
                 write(ia,'('' Lm      Bm      Rm      Lv      Bv     '',&
                   & ''  Rv      Le      Be      Re  '')')
                 if (ipla==3) write(ia,'('' Lma     Bma     Rma'')')
                 if (ipla/=3) then
4815             write(ia,'('' xm      ym      zm      xv      yv     '',&
                   & '' zv      xe      ye      ze   '')')
                 write(ia,'('' xv-xm   xe-xm   yv-ym   ye-ym   zv-zm  '',&
                   & '' ze-zm            rel. deviation'')')
                 endif
4820             call linie(ia,1)
               endif
               if (iop0==-803) write(ia,'(/23x,a35/31x,a19)') &
                'calculation of the file "inser-2.t"','--- please wait ---'
4825           end subroutine

               subroutine elements(ia,ivers,pla)
      !-----Ausgabe der Bahnelemente aller Planeten-------------------------
      !        im Rahmen der erweiterten Ergebnisausgabe
4830           use base, only : re
               implicit double precision (a-h,o-z)
               character(3) :: pla(0:9)
               write(ia,'('' pla.   mean long.   a [AU]   '', &
                 & ''eccentr.  asc.node    incl.     per.    per.[AU]'')')
4835           call linie(ia,2)
               do i=1,8
                 pd = re(26+6*i) * (1.d0-re(27+6*i))
                 if (ivers==3 .and.i==3) then
```

```
            write(ia,'(1x,a3,f13.5,2f10.5,a11,f9.5,f11.5,f10.5)')pla(i),&
4840          (re(24+6*i+j),j=1,3),'     ---    ',(re(24+6*i+j),j=5,6),pd
            else
             write(ia,'(1x,a3,f13.5,2f10.5,f11.5,f9.5,f11.5,f10.5)') &
               pla(i),(re(24+6*i+j),j=1,6),pd
            endif
4845      enddo
          end subroutine

          subroutine linie(ia,ib)
      !-----Linie, waagerecht-----------------------------------------------
4850      implicit double precision (a-h,o-z)
          if (ib==1) write(ia,'(1x,79a1)') ('=',i=1,79)
          if (ib==2) write(ia,'(1x,79a1)') ('-',i=1,79)
          if (ib==3) write(ia,'(1x,147a1)') ('=',i=1,147)
          if (ib==4) write(ia,'(1x,147a1)') ('-',i=1,147)
4855      end subroutine

          subroutine zwizeile(ia,io,zjde,ilin,imod,isep,ical,izp)
      !-----Tabellenueberschrift und Zwischenzeile bei Datumsausgaben--------
      !     Bei Transitbestimmungen werden abhaengig von der Wahl der
4860  !     Kalender-Option Zwischenzeilen eingefuegt, die den Uebergang
      !     von einem zum anderen Kalender kennzeichnen.
          implicit double precision (a-h,o-z)
          ipar = 0; if (isep==4) ipar = 2; is = isep; if (is==2) is = 1
          if (izp==1) then
4865        if (isep/=4) then
              write(ia,*)
            else
              write(ia,'(92x,''position angles [deg]'',13x, &
                & ''semidiameters ["]'')')
4870        endif
          endif
          if (izp==1) then
            if (ilin<=2 .and.io==2) call linie(ia,1+ipar)
            if (isep<=2) then
4875         write(ia,'('' co/p   k        date         time'', &
               & ''   dt[days]  Lm-Lv   Lm-Le  Lm-Lma  sep["]  S'')')
            elseif (isep==3) then
             write(ia,'('' co/p    date/  time:   I        I'', &
               & ''I      nearest     III       IV    sep["]a  S'')')
4880        else
             write(ia,'('' co/p    date/   time:   I        II     '', &
             &'' nearest     III       IV    sep["] a       P1      P2'',&
             &''     near.    P3      P4         s-Sun    s-pl.    S'')')
            endif
4885        if (imod/=3 .and.io/=2) then
              call linie(ia,1+ipar)
            else
              call linie(ia,io+ipar)
            endif
4890        if (io==2 .and.imod/=3) then
              write(ia,'(''   Lm       Bm       Rm       Lv       Bv   '',&
               & ''    Rv       Le       Be       Re  '')')
              write(ia,'(''   Lma      Bma      Rma'')')
              call linie(ia,1+ipar)
4895        endif
            if (ia==6) then
              izp=2; if (zjde>=0) izp=3; if (zjde>=2299161.d0) izp=4
```



```
            endif
          elseif (zjde>=0.d0 .and.izp==2 .and.ical==2) then
4900       select case (is)
            case(1); write(ia,'(1x,13(''-''),'' (Jul. cal.) '',53(''-''))')
            case(3); write(ia,'(1x,''----- (Jul. cal.) '',61(''-''))')
            case(4); write(ia,'(1x,''----- (Jul. cal.) '',129(''-''))')
           end select
4905       if (ia==6) izp = 3
          elseif (zjde>=2299161.d0 .and.izp==3 .and.ical==2) then
           select case (is)
            case(1);write(ia,'(1x,12(''-''),'' (Greg. cal.) '',53(''-''))')
            case(3);write(ia,'(1x,''---- (Greg. cal.) '',61(''-''))')
4910        case(4);write(ia,'(1x,''---- (Greg. cal.) '',129(''-''))')
           end select
           if (ia==6) izp = 4
          endif
          end subroutine
4915
          subroutine comtime(i,za,zb,iw1,iw2,ihour,imin,sec) ! (threads)
      !-----Bestimmung der Rechenzeit ---------------------------------------
      !     i = 1: CPU-time, i = 2: runtime
      !     Stopzeit zb - Startzeit za = Rechenzeit [hhh:mm:ss.sss]
4920      implicit double precision (a-h,o-z)
          dimension :: iw1(8),iw2(8)
          if (i==1) then
            t1 = za; t2 = zb
          else
4925        t1 = dfloat(iw1(5)*3600+iw1(6)*60+iw1(7))+dfloat(iw1(8))*1.d-3
            t2 = dfloat(iw2(5)*3600+iw2(6)*60+iw2(7))+dfloat(iw2(8))*1.d-3
          endif
          zt = t2-t1
          if (zt<0.d0) zt = zt + 86400.d0
4930      zih = dint(zt/3600.d0); ihour = idnint(zih)
          zm  = (zt-zih*3600.d0)/60.d0
          zim = dint(zm); imin = idnint(zim)
          sec = (zm-zim)*60.d0
          end subroutine
4935
          subroutine endzeile(ipla,imod,ilin,iaph,isep,ison,ijd,ipos, &
           ia,inum,ihour,imin,sec,ihour2,imin2,sec2,is12,iop0) ! (threads)
      !-----Endzeilen des Outputs--------------------------------------------
      !     Zusammenfassung: Anzahl gefundener Ereignisse, Erklaerung
4940  !     von Zeichen und Ausgabe der CPU-Zeit
          implicit double precision (a-h,o-z)
          dimension :: inum(0:4)
          character(37) :: te1
          character(8)  :: te2,te22
4945      character(1)  :: te3
          character(29) :: te4
          character(15) :: te5
          te1 = '                                      '
          te2 =  'CPU-time'; te3 = ':'; te5 = ' -- end of run.'
4950      te22 = 'run-time'; te4 = '("<" exact deviation dr)     '
          if ((imod/=3 .and.ison>=3).or.imod==3) then
            if (ipla==1) te1 = '(P: polarity, * view from ecl. south)'
            if (ipla==2) te1 = '(P: polarity, resp. view on ecliptic)'
          endif
4955      if (ilin<=2 .and.isep>=3) &
            te1 = '           ("/" means ascending node)'
```

```
          if (ijd==15 .and.iop0/=-803 .and.(imod/=2 .or.(imod==2 .and. &
              (iaph==3 .or.iaph==4 .or.ilin<=2)))) then
            write(ia,500)' Computed constellations:',inum(1),te1
4960        if (ilin<=2) then
              write(ia,501)' Tested planet. passages:',inum(0)
              write(ia,501)' Detected transits      :',inum(2)
              write(ia,502)' Centr./grazing transits:',inum(4),' /', &
                inum(3),te2,ihour,te3,imin,te3,sec
4965          else
              if (ipla/=3) then
                write(ia,503)' Detected constellations:',inum(2),te2, &
                  ihour,te3,imin,te3,sec
              else
4970            if (ison==5) then
                  inumber = inum(2)
                else
                  write(ia,501)' Detected constellations:',inum(2)
                  inumber = inum(3)
4975            endif
                write(ia,503)' Number of syzygies     :',inumber,te2, &
                  ihour,te3,imin,te3,sec
              endif
            endif
4980      else
            if (ipos==1 .and.is12==0 .and.iop0/=-803) then
              write(ia,504)te4,te2,ihour,te3,imin,te3,sec
            else
              if (iop0==-803) write(ia,'(41x,a38)') &
4985            'The file "inser-2.t" has been created.'
              write(ia,505)te2,ihour,te3,imin,te3,sec
            endif
          endif
          write(ia,506)te22,ihour2,te3,imin2,te3,sec2,te5
4990    500 format(1x,a25,i10,6x,a37)
        501 format(1x,a25,i10)
        502 format(1x,a25,i5,a2,i3,7x,a8,i3,a1,i2,a1,f6.3)
        503 format(1x,a25,i10,7x,a8,i3,a1,i2,a1,f6.3)
        504 format(14x,a29,a8,i3,a1,i2,a1,f6.3)
4995    505 format(43x,a8,i3,a1,i2,a1,f6.3)
        506 format(43x,a8,i3,a1,i2,a1,f6.3,a15/)
          end subroutine

      !h    subroutine histogramm(zz,ihis)  !h
5000  !-----Einsortieren der Genauigkeiten Fpos (zz) in ein Array------------
      !     fuer Pyramiden oder Kammern (ipla <= 2, imod <= 2, ison >= 3).
      !     Zur Nutzung muessen alle !h-Kommentarzeilen aktiviert werden.
      !h    implicit double precision (a-h,o-z)
      !h    dimension :: ihis(100)
5005  !h    i = idnint(zz*20.d0 + 0.5d0); if (i<=100) ihis(i) = ihis(i) + 1
      !h    end subroutine

          subroutine save_ser
      !-----Berechnung des Inhalts der Datei "inserie.t"----------------------
5010  !     Wenn die Datei "inserie.t" mit den julianischen Tagen (JDE)
      !     und den Nummern der Transit-Serien neu berechnet werden soll,
      !     erfolgt dies mit der Schnellstart-Option -803. Hiermit wird
      !     die neue Datei "inser-2.t" erzeugt. Falls gewuenscht kann
      !     diese - durch Umbenennung in "inserie.t" - die vorherige bzw.
5015  !     fehlende Datei "inserie.t" ersetzen. Die Verwendung dieser
```



```
      !     Option ist normalerweise nicht erforderlich.
          use astro, only : ser
          implicit double precision (a-h,o-z)
          open(unit=10,file='inser-2.t')
5020      write(10,'(9x,a21,a42/6x,a10,a58)')'Julian Ephemeris Day ', &
            'of each first transit in a series (S-No.),','to be used', &
            ' for the years -13000 BC to 17000 AD, VSOP87C full version'
          write(10,'(34x,a9)')'(Mercury)'
          write(10,'(a14,4(12x,a3))')'S-No.      JDE',('JDE',i=1,4)
5025      write(10,'(79a1)')('-',i=1,79)
          do i=-150,150,5
            write(10,'(I4,5f15.5)')i,(ser(i+j,1),j=0,4)  ! Serien, Merkur
          enddo
          write(10,'(79a1)')('-',i=1,79)
5030      write(10,'(35x,a7)')'(Venus)'
          write(10,'(a14,4(12x,a3))')'S-No.      JDE',('JDE',i=1,4)
          write(10,'(79a1)')('-',i=1,79)
          do i=-10,10,5
            write(10,'(I4,5f15.5)')i,(ser(i+j,2),j=0,4)  ! Serien, Venus
5035      enddo
          ser(19,2) = 1.d12
          write(10,'(I4,4f15.5,e15.1)')i,(ser(15+j,2),j=0,4) ! "     "
          write(10,'(79a1/)')('-',i=1,79)
          close(10)
5040      end subroutine

          subroutine vsop1tr(ip,rk,tau,del,r3i,eps,inum,resu)
      !-----Berechnung der ekliptikalen Koordinaten (Kurzversion VSOP87)-----
      !     Beruecksichtigung der Laufzeit des Lichtes, die bei Berechnung
5045  !     der Transitphasen eine Rolle spielt (siehe "vsop2tr")
      !     Index ip: 1 = Merkur, 2 = Venus
          use base
          implicit double precision (a-h,o-z)
          dimension :: rk(12),rd(3),inum(0:4)
5050      del = del/tmil  ! Laufzeit des Lichtes: Merkur/Venus --> Erde
          ist = 3*ip-2; ii = 3*(ip-1)
          do j=ist,ist+2
            call vsop1(j,tau,resu)
            re(j) = resu
5055      enddo
          call kartko(0)
          do j=ist,ist+2; rk(j) = xyr(j); enddo
          do
            tau1 = tau + del; inum(1) = inum(1) + 1
5060        do j=7,9; call vsop1(j,tau1,resu); re(j) = resu; enddo
            call kartko(0)
            do j=7,9; rk(j) = xyr(j); enddo
            do j=1,3; rd(j) = rk(ii+j) - rk(6+j); enddo
            r3i = dsqrt(rd(1)**2 + rd(2)**2 + rd(3)**2)
5065        del = r3i*AE/(c*86400.d0*tmil); tau2 = tau + del
            if (dabs(tau2-tau1)<eps) exit
          enddo
          del = del*tmil
          end subroutine
5070
          subroutine vsop2tr(xj2,ivers,ip,md, &
            ix,prec,lu,r,rk,ierr,del,r3i,eps,inum,rku)
      !-----Aufruf der VSOP87-Subroutine (Vollversion)-----------------------
      !     Beruecksichtigung der Laufzeit des Lichtes
```

```
!     Index von rku: 1 = L, 2 = B, 3 = r;  ip: 1 = Merkur, 2 = Venus
!     Input: Zeitpunkt "xj2", Output: Koordinaten der Planeten und
!     Laufzeit des Lichtes "del" vom Planet "ip" zur Erde
      use base, only : re,c,AE
      implicit double precision (a-h,o-z)
      dimension :: rk(12),rd(3),r(6),rku(3),md(0:9),inum(0:4)
      ii = 3*(ip-1)
      call vsop2(xj2,ivers,ip,md,ix,prec,lu,r,ierr,rku)
      do k=1,3
        re(ii+k) = rku(k)
        rk(ii+k) = r(k)
      enddo
      do
        xj3 = xj2 + del
        inum(1) = inum(1) + 1
        call vsop2(xj3,ivers,3,md,ix,prec,lu,r,ierr,rku)
        do k=1,3
          re(6+k) = rku(k)
          rk(6+k) = r(k)
        enddo
        do j=1,3
          rd(j) = rk(ii+j) - rk(6+j)
        enddo
        r3i = dsqrt(rd(1)**2 + rd(2)**2 + rd(3)**2)
        del = r3i*AE/(c*86400.d0)
        xj4 = xj2 + del
        if (dabs(xj4-xj3)<eps) exit
      enddo
      end subroutine

      subroutine fitmin(imod,imodus,iap,ke,x,y,ee1, &
        step,nu,iflag,ddx1,ddx2,test,itin,indx,ix)
!-----Minimum stetiger aber nicht ueberall diff.-barer Funktionen------
!     -->  Resultat = x(indx), indx = 1, 2 oder 3.
!
!     imodus = 1
!     Das Unterprogramm basiert auf einer Art ternaerem Suchen. Es
!     verwendet 3 Stuetzpunkte, um einen neuen Punkt zu finden und
!     einen alten durch diesen zu ersetzen. Dabei ruecken die Punkte
!     immer naeher zusammen, bis die Suchgenauigkeit (ee1) unter-
!     schritten wird. Das Minimum wird durch wiederholten Aufruf
!     von fitmin gefunden. Dieser Such-Algorithmus ist nicht beson-
!     ders schnell, konvergiert aber zuverlaessig und wird u.a. zur
!     Minimierung von "dL" bei Syzygien verwendet.
!
!     imodus = 2 (Spezialsuche)
!     Das Unterprogramm findet den Scheitelpunkt (Minimum) hyper-
!     bolischer Funktionen der Form: y = a * sqrt((x-b)**2 + c**2).
!     Dieser Algorithmus konvergiert deutlich schneller, findet
!     jedoch im konkreten Fall der Planetenbewegung die Loesung nur
!     dann, wenn sie zeitlich nicht zu weit entfernt liegt. Er dient
!     zur schnellen Berechnung der minimalen Separation des Transits.

      implicit double precision (a-h,o-z)
      dimension :: rx(3,4),x(5),y(5),test(10),d(3)
      ie  = 0
      ze  = 0.d0
      ee2 = 1.d-30
      zpa = 5.d0  ! zpa >= 2.d0
```



```
   10 iconv = 0
!c    do iu=ix,6,5; write(iu,'(''  nu,imod,imodus,indx,ddx1,ddx2 =''',&
!c       & i4,3i3,2f13.8)')nu,imod,imodus,indx,ddx1,ddx2
!c      write(iu,'(a12,3f18.8)') '  x(1..3) = ',(x(i),i=1,3)
!c      write(iu,'(a12,3f18.12/)')'  y(1..3) = ',(y(i),i=1,3); enddo
      nulim = 1
!.....Bestimmung der ersten drei x- und y-Werte
      if (iap==5 .and.imod==2) then
        nulim = 2
        if (nu==0) then
          indx = 1; go to 99
        endif
      endif
      if (nu<=nulim) then
        do i=1,2
          x(4-i) = x(3-i)
          y(4-i) = y(3-i)
        enddo
        x(1) = x(1) + step
        indx = 1; go to 99
      endif
      dy1 = y(2)-y(1); dy2 = y(3)-y(2)

! . . Pruefen auf numerisches Rauschen (im Minimum) und Konvergenz-
!     problem. Letzteres Problem entsteht eventuell beim Umschalten
!     von der VSOP87-Kurzversion zur -Vollversion.
      if (dy1>=ze.and.dy2<ze) then
        i1 = 0; if (ddx1+ddx2>1.d-3) i1 = 1
        i2 = 0; if (dabs(dy1)+dabs(dy2)>1.d-3) i2 = 1
!c      if (i1==0.and.i2==0) write(6,*)' -->  num. noise, nu =',nu
!c      if (i2==1) write(6,'(a23,i3)') ' -->  switch-pr.(dy), ',nu
!c      if (i1==1) write(6,'(a23,i3)') ' -->  switch-pr.(dx), ',nu
        if (i1==1 .or.i2==1) then
          iconv = 1; go to 20
        endif
        if (imodus==1) then; ke = 0; return; endif
      endif
   20 if (imodus==1) then
!.......Quasiternaeres Suchen (imodus = 1)
        if (dy1>=ze.and.dy2>=ze.and.iflag==0) then
          do i=1,2
            x(4-i) = x(3-i)                          ! way 1
            y(4-i) = y(3-i)
          enddo
          x(1) = x(1)+x(2)-x(3)
          if (dabs(x(1)-x(4))<1.d-8) then
            y(1) = y(4); go to 10
          endif
          indx = 1
        elseif ((dy1<ze.and.dy2<ze.and.iflag==0).or.iconv==1) then
          do i=1,2
            x(i) = x(1+i)                            ! way 2
            y(i) = y(1+i)
          enddo
          x(3) = x(3)+x(2)-x(1)
          if (dabs(x(3)-x(5))<1.d-8) then
            y(3) = y(5); go to 10
          endif
          indx = 3
```

```
            elseif ((dy1<ze.and.dy2>=ze).or.iflag==1) then
              select case (iflag)
5195          case(0)                                      ! way 3
                do i=1,2
                  x(3+i) = x(2*i-1); y(3+i) = y(2*i-1)
                enddo
                x(3) = (x(3)+(zpa-1.d0)*x(2))/zpa
5200            indx = 3; iflag = 1
              case(1)
                x(1) = (x(1)+(zpa-1.d0)*x(2))/zpa
                indx = 1; iflag = 0
              end select
5205        endif
          else
      !.......Suche mit hyperbolischem Fit (imodus = 2)
            a1 = x(1)-x(2); a3 = x(3)-x(2)
            b1 = (y(2)**2-y(1)**2)*a3
5210        b2 = (y(3)**2-y(2)**2)*a1
            if (dabs(b1+b2)<ee2) then; ke = 0; return; endif
            b = 0.5d0*(b1*a3+b2*a1)/(b1+b2) + x(2)
            d(1) = dabs(x(1)-b)
            d(2) = dabs(x(2)-b)
5215        d(3) = dabs(x(3)-b);          indx = 1
            if (d(2)>d(1).and.d(2)>d(3)) indx = 2
            if (d(3)>d(1).and.d(3)>d(2)) indx = 3
            x(indx) = b
            if (x(1)>x(2)) call pchange(2,1,2,rx,x,y,indx)
5220        if (x(2)>x(3)) call pchange(2,2,3,rx,x,y,indx)
            if (x(1)>x(2)) call pchange(2,1,2,rx,x,y,indx)
          endif
          ddx1 = dabs(x(2)-x(1))
          ddx2 = dabs(x(3)-x(2))
5225      ddx3 = dabs(x(3)-x(1))
          if (imodus==2) then
            do i=1,10
              if (dabs(ddx3-test(i))<1.d-7) ie = 1
            enddo
5230      endif
      !.....Hauptbedingung pruefen und Check auf Endlosschleife (ie=1)
          if (ddx1<=ee1.or.ddx2<=ee1.or.ie==1) then
      !c    do iu=ix,6,5; write(iu,'(''  nu,imod,imods,indx,dx1,dx2,ie'',&
      !c     & '' ='',i4,3i3,2f13.8,i3)') nu,imod,imodus,indx,ddx1,ddx2,ie
5235  !c    write(iu,'(a12,3f18.8/)') '  x(1..3) = ',(x(i),i=1,3); enddo
            ke = 0; return
          endif
          if (imodus==2) then
            itin = itin + 1; if (itin>10) itin = 1
5240        test(itin) = ddx3
          endif

         99 nu = nu + 1
      !c  write(6,'(a11,2i2,3f18.7)')' m,n,x1-3 =',imodus,nu,(x(i),i=1,3)
5245      if (nu<=100) return
          ke = 2
          do iu=ix,6,5
            write(iu,'(/''  ---->  error in "fitmin", ke ='',I2/)') ke
          enddo
5250      end subroutine
```



```
          subroutine ringfit(x1,x2,x3,y1,y2,y3,ep,step,nu,itmax,ix,ke)
      !-----Nullstellenbestimmung-----------------------------------------------
      !     Die Routine liefert fuer die Kreisfunktion, die durch (x1,y1),
5255  !     (x2,y2) und (x3,y3) verlaeuft, die naechstgelegene Nullstelle
      !     (neuer x2-Wert). Wie bei "sekante" ergibt wiederholtes Aufrufen
      !     von "ringfit" die Nullstelle einer stetig differenzierbaren Funk-
      !     tion. Abhaengig von den Optionen (ilin<=2, isep=3, 4 bzw. 1) ver-
      !     kuerzt sich die Rechenzeit um bis zu 3%, was wenig ist. Da die
5260  !     Grundidee und die Gleichungen jedoch auch eine gewisse Aesthetik
      !     besitzen, wurde diese Routine beibehalten. (Der Einsatz von
      !     "ringfit" ist nur sinnvoll, wenn die Berechnung der Ausgangs-
      !     funktion deutlich mehr Zeit erfordert als "ringfit" selbst.)
          implicit double precision (a-h,o-z)
5265      if (ke/=5) ke = 1; ep0 = 1.d-20
          if (nu<=1 .or.ke==5) then
            call sekante(x1,x2,y1,y2,ep,step,nu,itmax,ix,ke); return
          endif
          if (nu==2) then  ! Erzeugung des 3. Startpunktes
5270        x31 = x1; y31 = y1; x32 = x2; y32 = y2
            call sekante(x1,x2,y1,y2,ep,step,nu,itmax,ix,ke)
            if (x1==x31) then; x3 = x32; y3 = y32
                         else; x3 = x31; y3 = y31
            endif; return
5275      endif
          sh = x2  ! Verschiebung (x2) zum Ursprung
          x1 = x1-sh; x2 = 0.d0; x3 = x3-sh
      !c  do iu=ix,6,5; write(iu,'(a16,i3,6f10.6)') &
      !c    'nu, x123, y123 =',nu,x1,x2,x3,y1,y2,y3; enddo
5280      z1 = x1*x1 + y1*y1;  ya = y2-y1;  xa = -x1
          z2 =         y2*y2;  yb = y3-y2;  xb = x3
          z3 = x3*x3 + y3*y3;  yc = y1-y3;  xc = x1-x3
          xy = 0.5d0/(x1*yb + x3*ya)
          if (dabs(xy)<ep0) then
5285        x1 = x1+sh; x2 = sh; ke = 5; return
          endif
          x0 =  (z1*yb + z2*yc + z3*ya)*xy
          y0 = -(z1*xb + z2*xc + z3*xa)*xy
          wu = x0*x0 + (y2-y0)**2 - y0*y0
5290      if (wu<0.d0) then; ke = 4; go to 10; endif
          wu = dsqrt(wu)
          xx = x0 + wu; xx2 = x0 - wu  ! (2 Loesungen)
          if (dabs(xx)>dabs(xx2)) xx = xx2
          d1 = dabs(x1-xx); d2 = dabs(xx); d3 = dabs(x3-xx)
5295      if     (d3>d1.and.d3>d2) then; x3 = 0.d0; y3 = y2
          elseif (d1>d2.and.d1>d3) then; x1 = 0.d0; y1 = y2
          endif
          x1 = x1+sh; x2 = xx+sh; x3 = x3+sh; nu = nu+1
          if (dabs(x2-x1)<ep.or.dabs(x3-x2)<ep) then
5300  !c    do iu=ix,6,5; write(iu,'(a8,7x,a1,i3,3f14.10)') &
      !c      'nu, x123','=',nu,x1-sh,x2-sh,x3-sh; enddo
            ke = 0; return
          endif
          if (nu<=itmax) return
5305      ke = 2
         10 do iu=ix,6,5
            write(iu,'(/''  ---->  error in "ringfit", ke ='',I2/)') ke
          enddo
          end subroutine
5310
```

```
      subroutine sekante(x1,x2,y1,y2,ep,step,nu,itmax,ix,ke)
!-----Nullstellenbestimmung der Sekante------------------------------
!     Das Programm liefert die Nullstelle der linearen Funktion, die
!     durch (x1,y1) und (x2,y2) verlaeuft. Das Ergebnis wird als
!     neuer x2-Wert ausgegeben. Wiederholtes Aufrufen dieser Routine
!     liefert die Nullstelle (erster Ordnung) einer stetig differen-
!     zierbaren, nicht notwendigerweise linearen Funktion.
      implicit double precision (a-h,o-z)
      if (ke/=5) ke = 1
!c    do iu=ix,6,5; write(iu,'(a16,i3,2f16.6,2f12.6)') &
!c      'nu,x1,x2,y1,y2 =',nu,x1,x2,y1,y2; enddo
      nu = nu + 1
      if (nu<=1) then
        x1 = x2                     !---------------------------------
        y1 = y2                     !  In "sekante", "ringfit",
        x2 = x1 + step              !  "fitmin", and "vsop3" the
        return                      !  error code "ke" means:
      endif                         !
      if (y1==y2) then              !  ke=0:  no error
        ke = 3; go to 10            !  ke=1:  routine runs (internal)
      endif                         !  ke=2:  too many iterations
      x0 = x2-y2*(x2-x1)/(y2-y1)    !  ke=3:  division by zero
      if (dabs(y2)<dabs(y1)) then   !  ke=4:  root of negative number
        x1 = x2                     !  ke=5:  switchover to "sekante"
        y1 = y2                     !---------------------------------
      endif
      x2 = x0
      if (dabs(x2-x1)<ep.and.nu>2) then
!c      do iu=ix,6,5; write(iu,'(a16,i3,2f16.6)') &
!c        'nu,x1,x2       =',nu,x1,x2; enddo
        ke = 0; return
      endif
      if (nu<=itmax) return; ke = 2
   10 do iu=ix,6,5
        write(iu,'(/''  ---->  error in "sekante", ke ='',I2/)') ke
      enddo
      end subroutine

! >>    Update: The 4 subroutines of FITEX have been updated        <<
! >>            for Fortran 95 standard, double precision,          <<
! >>            and free source form.                               <<

!-----------------------------------------------------------------------
!     FITEX                                              M O D I N A 8 7
!-----------------------------------------------------------------------
!
!  PROGRAMM BESCHREIBUNG NR. 320 VON G. W. SCHWEIMER    (VERSION 1985)
!
!  CHISQUARE MINIMISING SUBROUTINE
!  SOLVES THE NONLINEAR LEAST SQUARES PROBLEM
!  USING A LEAST SQUARES INTERPOLATION BETWEEN VARIABLES AND FUNCTIONS
!  OR THE EXACT GRADIENT OF THE FUNCTIONS
!  CALLED SUBROUTINES: LILESQ(LINEAR LEAST SQUARES PROBLEM)
!                      INVATA(INVERSION OF A(TRANSPOSED)*A)
!                      FIT1(ONE DIMENSIONAL MINIMUM SEARCH)
!  CALLING SEQUENCE
!     KE=0
!     M=NUMBER OF FUNCTIONS, M GE N
!     N=NUMBER OF VARIABLES, N GE 1
```



```
!     DO 1 I=1,N
!     X(I)=STARTING VALUES OF THE VARIABLES
!   1 E(I)=ABSOLUTE SEARCH ACCURACIES FOR THE VARIABLES, E(I) NE 0
!     W(1)=FIRST STEP SIZE IN UNITS OF E(I), IF LE 1 W(1) = 100 BY
!          FITEX THE MAXIMUM ALLOWED STEP SIZE IS 2*W(1)
!     W(2)=METHOD OF APPROXIMATION, 0 FOR LEAST SQUARES INTERPOLATION
!                                   1 FOR EXACT GRADIENT OF THE FUNCTIONS
!     IW(1)=NUMBER OF POINTS TO BE REMEMBERED, IF LE N IW(1) = N+1
!     IW(2)=MAXIMUM NUMBER OF FUNCT. EVALUATIONS, IF EQ 0 IW(2)=2IW(1)
!           IF IW(2) LT 0 NO ACTION EXCEPT KE = 0
!     JA=4+MAX0(14,(N*(N+5))/2)+(M+N+1)*(IW(1)+1)
!   2 W(4)=0.
!     DO 3 I=1,M
!     F(I)=FUNCTION VALUES AT THE POINT X
!     IF(W(2)==0.) GO TO 3
!     W(JA+I+M*(J-1))= DF(I)/DX(J) FOR J=1,N
!   3 W(4)=W(4)+F(I)*F(I)
!     OPTIONAL WRITE(*,*) IW(3),IW(4),W(3),W(4),X,F
!     CALL FITEX(KE,M,N,F4,X4,E4,W4,IW)
!     IF(KE==1) GO TO 2
!     W(3)=ERROR RENORMALISATION FACTOR
!     W(4)=MINIMUM QUADRATIC SUM OF THE F(I)
!     X=MINIMUM POINT
!     F=FUNCTIONS AT THE MINIMUM POINT
!     KE=ERROR CODE   KE=0: WITHOUT ERRORS
!                     KE=2: USER INTERRUPT; RETURNS MINIMUM VALUES
!                           WITHOUT ERRORS. THE CURRENT POINT IS
!                           IGNORED. FOR NORMAL USER INTERRUPT SET
!                           IW(2)=IW(3).
!                     KE=3: MAXIMUM NUMBER OF FUNCTION EVALUATIONS
!                     KE=4: ROUNDING ERRORS
!                     KE=5: THE FUNCTIONS DO NOT DEPEND ON X(IW(4))
!                     KE=6: USELESS VARIABLES IN THE PREPARATORY CALLS,
!                           THE LABELS OF THE VARIABLES ARE IW(3),IW(4)
!                     KE=7: M LT N OR N LT 0 OR W(2)*(W(2)-1.) NE 0
!     W(4+I)=STANDARD ERRORS OF THE VARIABLES
!            THE ERROR CALCULATION ASSUMES LINEAR FUNCTIONS.
!            THE PROGRAM SHOWS THE LINEARITY BY THE KIND OF
!            PREDICTION IW(3)
!            IW(3)=0: LINEAR PREDICTION
!                 =1: STEP SIZE LIMITATION
!                 =2: ONE DIMENSIONAL SEARCH
!                 =3: RANDOM SEARCH
!            THE ERRORS ARE CORRECTLY CALCULATED IF THE LAST
!            N ITERATIONS WERE LINEAR, I.E. IW(3)=0.
!     W(4+N+I)=ERROR ENHANCEMENTS
!     W(4+N+N+I+(J*(J-1))/2)=ERROR CORRELATION BETW. X(I) AND X(J) I<J
!     IW(3): NUMBER OF FUNCTION EVALUATIONS
!     IW(4): NUMBER OF DEGREES OF FREEDOM
!  WORKING FIELD:  IW: LENGTH 4+K WITH K = IW(1)
!     W: LENGTH 4+MAX(14,(N*(N+5))/2)+(M+N+1)*(K+1)+M*N
!                  ADRESSES IN IW
!                      4+L: LABELS OF THE QUADRATIC SUMS
!                  ADRESSES IN W
!                      4+I: STANDARD ERROR OF X(I)
!                      4+N+I: ERROR ENHANCEMENT FOR X(I)
!                      FROM 4+N+N+1: MATRIX D AND ERROR CORRELATIONS
!                      FROM JS+1 MATRIX S; JS = 4+MAX0(14,(N*(N+5))/2)
!                      FROM JA+1: MATRIX A WITH JA = JS+(M+N+1)*(K+1)
```

```
      !  THE WORKING FIELDS CONTAIN ALL INFORMATION FOR THE CONTINUATION OF
5430  !  THE SEARCH. THIS ALLOWS A SEARCH WITHIN ANOTHER SEARCH JUST CHANGING
      !  THE WORKING FIELDS
      !
      !------------------------------------------------------------------------
           SUBROUTINE FITEX(KE,M,N,F,X,E,W,IW)
5435       IMPLICIT NONE
           INTEGER(4) :: KE,M,N,I,I1,I2,J,J1,J2,J3,JA,JD,JM,JS,K,KV
      ! >>   Sizes of IW and W are increased because of index overflow,
      ! >>   although FITEX ran correctly before. (The numbers 100 and 1000
      ! >>   are appropriate, if n = 7 and m = 9.)
5440       INTEGER(4) :: IW(100),L,LM,MF
           REAL(8) :: E(N),F(M),W(1000),X(N),EPS,S,T,U,V,BIG
           REAL(4) :: A
           INTEGER(2) :: IR
      ! >>   A and IR in the equivalence statement have still the original
5445  ! >>   single precision, since they are used to generate random numbers
      ! >>   and so the calculation is not changed.
           EQUIVALENCE (A,IR)
           DATA EPS/1.D-8/,BIG/7.D+75/
           DATA MF/0/,J/0/,LM/0/,JS/0/,JM/0/,JD/0/,JA/0/,J3/0/ ! pre-init.
5450       IF (IW(2)<0) GO TO 50
           JD = 4 + N + N
           JS = 4 + MAX0(14,(N*(N+5))/2)
           LM = M + N + 1
           IF (KE/=0) GO TO 2
5455       IF (IW(1)<=N) IW(1) = N + 1
           IF (IW(2)==0) IW(2) = 2*IW(1)
           IF (W(1)<=1.D0) W(1) = 100.D0
           IW(3) = 1
           K = IW(1)
5460       DO L = 1,K
             IW(L+4) = 1 + K - L
             W(JS+LM*L) = 7.D75
           ENDDO
           KE = 1
5465     2 K = IW(1)
           KV = K
           JA = JS + LM* (K+1)
           JM = JS + LM*IW(5) - LM
           J3 = JA - LM
5470       IF (KE==2) GO TO 52
           IF (M<N.OR.N<1 .OR.W(2)*(W(2)-1.D0)/=0.D0) GO TO 57
           IF (W(4)<=0.D0) GO TO 50
           L = IW(K+4)
           IF (W(JS+LM*L)==BIG) KV = L - 1
5475       DO I = 1,K
             J1 = JS + LM*IW(I+4)
             IF (W(4)<W(J1)) GO TO 4
           ENDDO
           GO TO 37
5480     4 IF ((W(2)==0.D0 .AND.I>MAX0(N+1,KV)).OR. &
               (W(2)==1.D0 .AND.I>1)) GO TO 37
           IF (KV<K) KV = KV + 1
           I1 = K + 4
           I2 = K - I
5485       IF (I2==0) GO TO 6
           DO J = 1,I2
             I1 = I1 - 1
```



```
             IW(I1+1) = IW(I1)
           ENDDO
5490       IW(I1) = L
           JM = JS + LM*IW(5) - LM
      !  NEW ROW
         6 J1 = JS + LM* (L-1)
           DO I = 1,N
5495         J1 = J1 + 1
             W(J1) = X(I)
           ENDDO
           DO I = 1,M
             J1 = J1 + 1
5500         W(J1) = F(I)
           ENDDO
           W(J1+1) = W(4)
      !  TEST MAXIMUM NUMBER OF FUNCTION EVALUATIONS
           IF (IW(3)>=IW(2)) GO TO 53
5505       IF (N==1) GO TO 42
      !  EXACT GRADIENTS OR END OF PREPARATORY FUNCTION EVALUATIONS
           IF (W(2)==1.D0 .OR.IW(3)>N+1) GO TO 15
      !  PREPARATORY FUNCTION EVALUATIONS
           MF = IW(3)
5510       IF (MF==1) GO TO 12
           X(MF-1) = W(3)
           J2 = JS + N
           S = 0.D0
           DO I = 1,M
5515         T = F(I) - W(J2+I)
             S = S + T*T
           ENDDO
           J = 2
           IF (S<EPS*EPS*W(JS+LM)) GO TO 55
5520       W(3) = S
           J1 = 2 + N + MF
           W(J1) = DSQRT(W(3))
           IF (MF<=2) GO TO 12
           I1 = N + 1
5525       DO J = 3,MF
             I2 = J2 + LM* (J-2)
             S = 0.D0
             DO I = 1,M
               S = S + (W(I2+I)-W(J2+I))* (F(I)-W(J2+I))
5530         ENDDO
             IF (DABS(W(J1)*W(I1+J)-DABS(S))<EPS*DABS(S)) GO TO 56
           ENDDO
        12 IF (MF==N+1) GO TO 15
           W(3) = X(MF)
5535       X(MF) = X(MF) + W(1)*E(MF)
           GO TO 100
      !  END OF PREPARATORY FUNCTION EVALUATIONS
      !  SUM OF INVERSES OF THE QUADRATIC SUMS
        15 S = 0.D0
5540       DO L = 1,KV
             T = W(JS+LM*L)
             S = S + 1.D0/ (T*T)
           ENDDO
           W(JA) = 1.D0/S
5545  !  CENTRE OF THE VARIABLES AND FUNCTIONS
           I1 = M + N
```

```
            DO I = 1,I1
              J1 = JS
              S = 0.D0
5550          DO L = 1,KV
                T = W(J1+LM)
                S = S + W(J1+I)/ (T*T)
                J1 = J1 + LM
              ENDDO
5555          W(J3+I) = S*W(JA)
            ENDDO
            IF (KE/=1) GO TO 60
            IF (W(2)==0.D0) GO TO 20
            J1 = JA - M - 1
5560        DO I = 1,M; W(J1+I) = F(I); ENDDO
            GO TO 23
      !   MATRIX A
         20 J1 = JA
            DO I = 1,N
5565          U = W(J3+I)
              DO J = 1,M
                J1 = J1 + 1
                J2 = JS
                S = 0.D0
5570            T = W(J3+N+J)
                DO L = 1,KV
                  V = W(J2+LM)
                  S = S + (W(J2+N+J)-T)* (W(J2+I)-U)/ (V*V)
                  J2 = J2 + LM
5575            ENDDO
                W(J1) = S*W(JA)
              ENDDO
            ENDDO
            IF (KE/=1) GO TO 62
5580    !   LINEAR LEAST SQUARES PROBLEM
         23 CALL LILESQ(M,N,IR,W(JA+1),W(JA-M),W(5),W(N+5))
            IF (IR<0) GO TO 54
            IF (IR==0) GO TO 24; GO TO 35
      !   MATRIX D
5585     24 J1 = JD
            DO I = 1,N
              T = W(J3+I)
              DO J = 1,I
                J1 = J1 + 1
5590            J2 = JS
                S = 0.D0
                U = W(J3+J)
                DO L = 1,KV
                  V = W(J2+LM)
5595              S = S + (W(J2+I)-T)* (W(J2+J)-U)/ (V*V)
                  J2 = J2 + LM
                ENDDO
                W(J1) = S*W(JA)
              ENDDO
5600        ENDDO
      !   NEW VARIABLES
            IF (W(2)==0.D0) GO TO 28
            DO I = 1,N; X(I) = W(JM+I) - W(I+4); ENDDO
            GO TO 31
5605     28 DO I = 1,N
```



```
              I2 = 1
              J1 = JD + (I*I-I)/2
              S = 0.D0
              DO J = 1,N
5610            J1 = J1 + I2
                IF (J>=I) I2 = J
                S = S + W(J1)*W(J+4)
              ENDDO
              X(I) = W(J3+I) - S
5615        ENDDO
      !   TEST OF CONVERGENCE
         31 A = 0.E0
            DO I = 1,N
              W(I+4) = X(I) - W(JM+I)
5620          A = AMAX1(A,SNGL(DABS(W(I+4)/E(I))))
            ENDDO
            IF (A<1.E0) GO TO 50
            IW(4) = 0
            W(3) = 1.D0
5625        IF (A<2.E0*W(1)) GO TO 33
      !   STEP SIZE LIMITATION
            IW(4) = 1
            W(3) = 2.D0*W(1)/A
         33 DO I = 1,N; X(I) = W(JM+I) + W(3)*W(I+4); ENDDO
5630        GO TO 100
      !   RANDOM PREDICTION
         35 DO I = 1,N
              A = SNGL(W(J3+I))
              X(I) = W(JM+I) + W(1)*E(I)* &
5635                (MOD(IABS(INT(IR,KIND=4)),200)-100)/100.D0
            ENDDO
            IW(4) = 3
            GO TO 100
      !   ONE DIMENSIONAL SEARCH
5640     37 IF (N==1) GO TO 43
            IF (IW(3)>=IW(2)) GO TO 53
            IF (IW(4)==2) GO TO 39
            IW(4) = 2
            DO I = 1,N; W(J3+I) = X(I) - W(JM+I); ENDDO
5645        IR = 3
            W(5) = IR
            IR = 20
            W(6) = IR
            W(8) = 0.5D0
5650        W(11) = 0.D0
            W(12) = 0.D0
            W(13) = 0.D0
            W(14) = 1.D0
            W(16) = W(JM+LM)
5655        W(17) = W(4)
            GO TO 40
         39 W(9) = W(4)
            CALL FIT1(KE,W(5),W(8))
         40 DO I = 1,N; X(I) = W(JM+I) + W(8)*W(J3+I); ENDDO
5660        IF (KE==3) KE = 2
            IF (KE==2) GO TO 53
            KE = 1
            W(3) = W(8)
            GO TO 100
```

```
!  ONLY ONE VARIABLE X
   42 IF (IW(3)>1) GO TO 43
      KE = 0
      W(10) = W(1)*E(1)
      W(11) = E(1)
      W(12) = 0.D0
   43 IR = INT(IW(2),KIND=2)
      W(6) = A
      W(8) = X(1)
      W(9) = W(4)
      CALL FIT1(KE,W(5),W(8))
      IW(4) = 2
      X(1) = W(8)
      IF (KE==1) GO TO 100
      IF (KE>0) KE = KE + 1
      W(3) = 0.D0
      W(5) = 0.D0
      IF (W(6)/=0.D0) GO TO 74
      W(5) = DSQRT(DABS((W(13)-W(15))/ ((W(16)-W(17))/ (W(13)-W(14))- &
      (W(17)-W(18))/ (W(14)-W(15)))))
      W(6) = 1.D0
      W(7) = 1.D0
      GO TO 71
!  END OF SEARCH
   50 KE = 0
      IF (W(4)==0.D0 .OR. IW(2)<0) GO TO 100
      GO TO 52
!  ERROR CODE DEFINITION
   57 KE = KE + 1
   56 KE = KE + 1
   55 KE = KE + 1
   54 KE = KE + 1
   53 KE = KE + 2
   52 DO I = 1,N; W(I+4) = 0.D0; ENDDO
      W(3) = 0.D0
      IF (KE*(KE-3)/=0 .OR.(KE==3 .AND.(W(2)==1.D0 .OR. &
         (W(3)==0.D0 .AND.IW(3)<=N)))) GO TO 74
!  COMPUTATION OF THE ERRORS OF THE VARIABLES
!  RESTORE MATRIX G
      IF (W(2)==0.D0) GO TO 15
      J1 = JA
      I1 = N + 1
      DO 45 I = 2,I1
        IF (I>M) GO TO 45
        DO J = I,M; W(J1+J) = 0.D0
        ENDDO
        J1 = J1 + M
   45 ENDDO
      DO 49 I = 1,N
        DO I1 = I,N
          A = SNGL(W(4+N+I1))
          IF (IR==I) EXIT
        ENDDO
        IF (I1==I) GO TO 49
        J1 = JA + M* (I-1)
        J2 = JA + M* (I1-1)
        W(4+N+I1) = W(4+N+I)
        DO J = 1,N
          A = SNGL(W(J1+J))
```



```
          W(J1+J) = W(J2+J)
          W(J2+J) = A
        ENDDO
   49 ENDDO
      GO TO 66
!  INVERSE OF MATRIX D
   60 T = DSQRT(W(JA))
      J1 = JA
      DO I = 1,N
        S = W(J3+I)
        J2 = JS + I - LM
        DO L = 1,KV
          J1 = J1 + 1
          W(J1) = T*(W(J2+L*LM)-S)/W(JS+L*LM)
        ENDDO
      ENDDO
      CALL INVATA(KV,N,IR,W(JA+1),W(JD+1),X)
      IF (IR==0) GO TO 20
      GO TO 74
!  MATRIX G = A*INVERSE OF D
   62 DO L = 1,M
        J1 = L + JA - M
        DO I = 1,N
          I1 = JD + (I*I-I)/2
          I2 = 1
          S = 0.D0
          DO J = 1,N
            I1 = I1 + I2
            IF (J>=I) I2 = J
            S = S + W(I1)*W(J1+J*M)
          ENDDO
          X(I) = S
        ENDDO
        DO J = 1,N; W(J1+J*M) = X(J); ENDDO
      ENDDO
!  DIAGONAL ELEMENTS OF G(T)*G
   66 J1 = JA
      DO I = 1,N
        S = 0.D0
        DO L = 1,M
          J1 = J1 + 1
          S = S + W(J1)*W(J1)
        ENDDO
        W(4+N+I) = DSQRT(S)
      ENDDO
!  STANDARD ERRORS AND ERROR CORRELATIONS
      CALL INVATA(M,N,IR,W(JA+1),W(JD+1),X)
      IF (IR/=0) GO TO 74
      DO I = 1,N
        W(I+4) = DSQRT(W(JD+ (I*I+I)/2))
        W(4+N+I) = W(I+4)*W(4+N+I)
      ENDDO
      J1 = JD
      DO I = 1,N
        DO J = 1,I
          J1 = J1 + 1
          W(J1) = W(J1)/ (W(I+4)*W(J+4))
        ENDDO
      ENDDO
```

```
      !  ERROR RENORMALISATION FACTOR
         71 S = 0.D0
5785        DO I = 1,M; S = S + W(JM+N+I); ENDDO
            W(3) = DSQRT(DABS(W(JM+LM)-S*S/M)/MAX0(M-N-1,1))
            DO I = 1,N; W(I+4) = W(I+4)*W(3); ENDDO
      !  RESTORE OPTIMUM VALUES TO X AND F
         74 IW(4) = M - N - 1
5790        IF ((KE-5)* (KE-6)/=0) GO TO 75
            IW(3) = J - 2
            IW(4) = MF - 1
         75 DO I = 1,N; X(I) = W(JM+I); ENDDO
            DO I = 1,M; F(I) = W(JM+N+I); ENDDO
5795        W(4) = W(JM+LM)
        100 IF (KE==1) IW(3) = IW(3) + 1
            END SUBROUTINE

      !-------------------------------------------------------------------------
5800  !     FIT1                                                M O D I N A 8 7
      !-------------------------------------------------------------------------
      !
      !  PROGRAMM BESCHREIBUNG NR. 309 VON G. W. SCHWEIMER (VERSION 1985)
      !
5805  !  MINIMISATION OF A FUNCTION F(X) OF ONE VARIABLE X
      !  CALLING SEQUENCE
      !     KE=0
      !     I(2)=MAXIMUM NUMBER OF FUNCTION EVALUATIONS
      !     W(1)=START VALUE OF X
5810  !     W(3)=FIRST STEP SIZE
      !     W(4)=ABSOLUTE SEARCH ACCURACY
      !     W(5)=RELATIVE SEARCH ACCURACY
      !    1 W(2)=FUNCTION VALUE F(X) AT X=W(1)
      !     OPTIONAL WRITE VI(1),X,F
5815  !     CALL FIT1(KE,VI,W)
      !     IF(KE==1) GO TO 1
      !     XMIN=W(1)
      !     FMIN=W(2)
      !     NF=VI(1)
5820  !  KE = ERROR CODE: KE=0 NO ERRORS, KE=
      !    2 MAXIMUM NUMBER OF FUNCTION EVALUATIONS
      !    3 ROUNDING ERRORS, PROB. BECAUSE BOTH W(4) AND W(5) ARE TOO SMALL
      !  THE WORKING FIELDS I AND W HAVE THE LENGTH 3 AND 11 RESPECTIVELY
      !  THEY CONTAIN ALL INFORMATION FOR THE CONTINUATION OF THE SEARCH
5825  !  THEREFORE A SEARCH WITHIN ANOTHER SEARCH CAN BE DONE JUST CHANGING
      !  THE WORKING FIELDS
      !  IF 2 FUNCTION VALUES F1 AND F2 ARE KNOWN FOR X = X1 AND X2 RESPEC-
      !  TIVELY WITH X1 NE X2 ENTER THE CALLING SEQUENCE AFTER DEFINING :
      !  KE = 1; I(1) = 3; W(6) = X1; W(7) = X2; W(9) = F1; W(10) = F2 AND
5830  !  W(1) = USERS CHOICE
      !  WORKING FIELD VARIABLES:
      !  I(1): CURRENT NUMBER OF FUNCTION EVALUATIONS
      !  I(2): MAXIMUM NUMBER OF FUNCTION EVALUATIONS
      !  I(3): MINIMUM POINTER, THE MINIMUM FUNCTION VALUE IS AT W(7+I(3))
5835  !  W(1): CURRENT VALUE OF X
      !  W(2): USER SUPPLIED FUNCTION VALUE
      !  W(3): FIRST STEP SIZE
      !  W(4 AND 5): SEARCH ACCURACIES
      !  W(6, 7 AND 8): X1, X2 AND X3 WITH X1 < X2 < X3
5840  !  W(9, 10 AND 11): FUNCTION VALUES AT X1, X2 AND X3 RESPECTIVELY
      !
```



```
      !-------------------------------------------------------------------------
            SUBROUTINE FIT1(KE,V,W)
            IMPLICIT NONE
5845        INTEGER(4) :: KE,IV,J,K
            REAL(8) :: V(3),W(11)
            IF (KE==1) GO TO 2
            KE = 1
            V(1) = 1
5850        V(3) = -1
            W(6) = W(1)
            W(9) = W(2)
          1 W(1) = W(1) + W(3)
            GO TO 12
5855      2 IF (V(1)>2.D0) GO TO 3
            V(3) = 0.D0
            W(7) = W(1)
            W(10) = W(2)
            IF (W(2)<=W(9)) GO TO 1
5860        V(3) = -1.D0
            W(1) = W(6) - W(3)
            GO TO 12
          3 IF (V(1)>3.D0) GO TO 5
            W(8) = W(1)
5865        W(11) = W(2)
            DO 4 J = 1,3
              K = 7 - MOD(J,2)
              IF (W(K)<=W(K+1)) GO TO 4
              W(1) = W(K)
5870          W(K) = W(K+1)
              W(K+1) = W(1)
              K = K + 3
              W(1) = W(K)
              W(K) = W(K+1)
5875          W(K+1) = W(1)
          4 ENDDO
            V(3) = 0.D0
            IF (W(9)<W(10).AND.W(9)<W(11)) V(3) = -1.D0
            IF (W(11)<W(10).AND.W(11)<W(9)) V(3) = 1.D0
5880        GO TO 9
      !  SORT IN THE NEW VALUES OF X AND F
          5 IF (V(3)==0.D0) GO TO 6
            J = IDINT(V(3))
            W(7-J) = W(7)
5885        W(10-J) = W(10)
            IF ((W(7+J)-W(1))*(W(1)-W(7))>0.D0) GO TO 7
            W(7) = W(7+J)
            W(10) = W(10+J)
            W(7+J) = W(1)
5890        W(10+J) = W(2)
            IF (W(2)>=W(10)) V(3) = 0.D0
            GO TO 9
          6 J = -1
            IF (W(1)<W(7)) J = 1
5895        IF (W(2)>W(10)) GO TO 8
            W(7+J) = W(7)
            W(10+J) = W(10)
          7 W(7) = W(1)
            W(10) = W(2)
5900        IV = IDINT(V(3))
```

```
            IF (W(2)<=W(10+IV)) V(3) = 0.D0
            GO TO 9
          8 W(7-J) = W(1)
            W(10-J) = W(2)
5905        9 IV = IDINT(V(3))
            J = 7 + IV
      !   ERROR TESTS
            IF (W(6)==W(7) .OR. W(7)==W(8) .OR. &
               (W(9)==W(10).AND.W(10)==W(11))) GO TO 15
5910        IF (V(1)>=V(2)) GO TO 16
            IF (V(3)==0.D0) GO TO 10
      !   STEP SIZE LIMITATION
            W(1) = W(J) + 2.D0*V(3)* (W(8)-W(6))
            GO TO 12
5915      10 W(1) = DMIN1(W(8)-W(7),W(7)-W(6))/(W(8)-W(6))
            IF (W(1)>0.1D0) GO TO 11
            W(1) = .5D0* (W(6)+W(8))
            GO TO 12
      !   PREDICTION OF THE POSITION OF THE MINIMUM
5920      11 W(1) = ((W(9)-W(10))/ (W(6)-W(7))- (W(10)-W(11))/ (W(7)-W(8)))/ &
             (W(6)-W(8))
            W(1) = .5D0* (W(6)+W(8)+ (W(11)-W(9))/ (W(1)* (W(6)-W(8))))
      !   TEST OF CONVERGENCE
            W(2) = DABS(W(1)-W(J))
5925        IF (W(2)<DABS(W(4)) .OR. W(2)<DABS(W(5)*W(J))) GO TO 13
          12 V(1) = V(1) + 1.D0
            RETURN
          13 KE = 0
          14 IV = IDINT(V(3))
5930        W(1) = W(7+IV)
            W(2) = W(10+IV)
            RETURN
          15 KE = KE + 1
          16 KE = KE + 1
5935        GO TO 14
            END SUBROUTINE

      !-----------------------------------------------------------------------
      !     INVATA                                            M O D I N A 8 7
5940  !-----------------------------------------------------------------------
      !
      !   PROGRAMM BESCHREIBUNG NR. 320 VON G. W. SCHWEIMER   (VERSION 1985)
      !
      !   INVERSION OF THE PRODUCT MATRIX A(TRANSPOSED)*A
5945  !   THE MATRIX A IS REDUCED TO AN UPPER TRIANGULAR MATRIX R BY
      !   HOUSEHOLDER TRANSFORMATIONS. THE REMAINING COMPUTATION IS STRAIGHT
      !   FORWARD.
      !   INPUT VARIABLES: N: NUMBER OF COLUMNS OF MATRIX A
      !                    M: NUMBER OF ROWS OF MATRIX A, M >= N > 0
5950  !                    A: INPUT MATRIX (DESTROYED)
      !   OUTPUT VARIABLES:   IR: ERROR CODE
      !                          IR=-2: M LT N OR N LT 1
      !                          IR=-1 RANK OF MATRIX A IS ZERO
      !                          IR=0 NO ERROR, RANK OF MATRIX A IS N
5955  !                          IR>0 RANK OF MATRIX A IS IR, THE INVERSE
      !                               OF A(T)*A IS COMPUTED CONSIDERING THE
      !                               IR COLUMNS OF A INDICATED BY THE FIRST
      !                               IR COMPONENTS OF IP
      !                  A: TRIANGULAR MATRIX R, R=A(I,J) I<=J=1,N
```



```
5960  !                  D: VECTOR OF LENGTH (N*(N+1))/2, IT CONTAINS THE
      !                       UPPER TRIANGULAR PART OF THE INVERSE OF A(T)*A
      !                  IP: PERMUTATION VECTOR OF LENGTH N, ITS FIRST IR
      !                       COMPONENTS CONTAIN THE LABELS OF THE USEFULL
      !                       COLUMNS OF A, THE LAST COMPONENTS CONTAIN
5965  !                       THE LABELS OF THE COLUMNS WHICH ARE LINEAR
      !                       COMBINATIONS OF THE FIRST.
      !     THE RANK OF THE MATRIX A IS DETECTED COMPARING THE RESULT
      !     OF A SUM WITH THE SUM OF ABSOLUTE VALUES.
      !     IF SUM OVER I OF T(I) <= EPS * (SUM OF ABS(T(I))) THEN
5970  !     SUM IS SET TO EXACTR ZERO.
      !-----------------------------------------------------------------------
            SUBROUTINE INVATA(M,N,IR,A,D,VP)
            IMPLICIT NONE
            INTEGER(2) :: IR
5975        INTEGER(4) :: M,N,I,I1,IJ,J,K,L
      !     Size of D changed (see above, FITEX)
            REAL(8) :: A(M,N),D(15*N),VP(N)
            REAL(8) :: EPS,P,Q,R,S,SIG,T,U,V,C
            DATA EPS/1.D-8/
5980        DATA I1/0/ ! pre-init.
            IR = INT(N,KIND=2)
            IF (M<N.OR.N<1) GO TO 19
            DO I = 1,IR; VP(I) = I; ENDDO
      !   HOUSEHOLDER LOOP
5985        K = 0
          2 K = K + 1
      !   PIVOT ELEMENT
            3 C = 0.D0
              DO 4 I = K,M
5990            IF (DABS(A(I,K))<=C) GO TO 4
                C = DABS(A(I,K))
                I1 = I
              4 ENDDO
              IF (C>0.D0) GO TO 8
5995          IR = IR - INT(1,KIND=2)
              IF (K>IR) GO TO 13
      !   SET UP THE PERMUTATION VECTOR IP AND PERMUTE THE COLUMNS OF MATRIX A
              L = IDINT(VP(K))
              DO J = K,IR; VP(J) = VP(J+1); ENDDO
6000          VP(IR+1) = L
              DO I = 1,M
                C = A(I,K)
                DO J = K,IR; A(I,J) = A(I,J+1); ENDDO
                A(I,IR+1) = C
6005          ENDDO
              GO TO 3
      !   ROTATION OF THE LOWER COLUMN FRAGMENTS OF A(K)
            8 DO J = K,IR
                C = A(K,J)
6010            A(K,J) = A(I1,J)
                A(I1,J) = C
              ENDDO
              S = A(K,K)
              V = 0.D0
6015          DO I = K,M
                U = A(I,K)/S
                V = V + U*U
              ENDDO
```

```
            V = 1.D0/DSQRT(V)
6020        SIG = S/V
            U = S + SIG
            A(K,K) = -SIG
            IF (K>=IR) GO TO 13
            L = K + 1
6025        DO J = L,IR
              S = V*A(K,J)
              P = DABS(S)
              DO I = L,M
                R = (A(I,K)/SIG)*A(I,J)
6030            S = S + R
                P = P + DABS(R)
              ENDDO
              IF (DABS(S)<=EPS*P) S = 0.D0
              T = (A(K,J)+S)/U
6035          IF (DABS(T)<=EPS*DABS(S/U)) T = 0.D0
              A(K,J) = -S
              DO I = L,M
                Q = A(I,J)
                P = T*A(I,K)
6040            R = Q - P
                IF (DABS(R)<=EPS*DABS(P)) R = 0.D0
                A(I,J) = R
              ENDDO
            ENDDO
6045        GO TO 2
      !  END OF HOUSEHOLDER LOOP
         13 IF (IR==0) GO TO 20
      !  INVERSE OF THE TRIANGULAR MATRIX R STORED IN D
            IJ = 0
6050        DO 16 K = 1,IR
              D(IJ+K) = 1.D0/A(K,K)
              IF (K==1) GO TO 16
              I = K
              DO L = 2,K
6055            I1 = I
                I = I - 1
                S = 0.D0
                DO J = I1,K; S = S + A(I,J)*D(IJ+J); ENDDO
                D(IJ+I) = -S/A(I,I)
6060          ENDDO
              IJ = IJ + K
         16 ENDDO
      !  INVERSE OF THE PRODUCT MATRIX
            IJ = 0
6065        DO J = 1,IR
              DO I = 1,J
                IJ = IJ + 1
                I1 = IJ
                L = J - I
6070            S = 0.D0
                DO K = J,IR
                  S = S + D(I1)*D(I1+L)
                  I1 = I1 + K
                ENDDO
6075            D(IJ) = S
              ENDDO
            ENDDO
```



```
            GO TO 20
         19 IR = -2
6080     20 IF (IR==0) IR = -1
            IF (IR==N) IR = 0
            END SUBROUTINE

      !-----------------------------------------------------------------------
6085  !     LILESQ                                              M O D I N A 8 7
      !-----------------------------------------------------------------------
      !
      !  PROGRAMM BESCHREIBUNG NR. 320 VON G. W. SCHWEIMER   (VERSION 1985)
      !
6090  !  LINEAR LEAST SQUARES PROBLEM !!B-A*X!!=MIN(X)
      !  SOLVED BY HOUSEHOLDER TRANSFORMATIONS
      !  REDUNDANT VARIABLES ARE DETECTED BY THE METHOD OF G.GOLUB,
      !  NUMERISCHE MATHEMATIK, VOL. 7, PAGE 206-216, (1965)
      !  INPUT VARIABLES:M: NUMBER OF ROWS OF A AND B
6095  !                  N: NUMBER OF COLUMNS OF A AND ROWS OF X
      !                  A: M*N MATRIX (DESTROYED)
      !                  B: VECTOR OF M COMPONENTS (DESTROYED)
      !  OUTPUT VARIABLES: X: VECTOR OF VARIABLES, THE REDUNDANT VARIABLES
      !                       ARE SET TO ZERO. THE !!X!!=MIN IS NOT USED
6100  !                       BECAUSE THE COMPONENTS OF X ARE ASSUMED TO BE
      !                       NOT COMMENSURABLE
      !                  IP: PERMUTATION VECTOR OF N COMPONENTS, IT CONTAINS
      !                       THE COLUMN LABLES OF MATRIX A ORDERED ACCORDING
      !                       THEIR IMPORTANCE IN REDUCING THE EUCLIDEAN NORM
6105  !                  A: THE UPPER PART CONTAINS THE TRANSFORMED INPUT A
      !                       A(2,1) CONTAINS THE SQUARE OF THE EUCLIDEAN
      !                       NORM
      !                  B: TRANSFORMED INPUT B
      !                  IER: ERROR CODE
6110  !                       IER=0 NO ERROR
      !                       IER=-1 ALL COMPONENTS OF X ARE ZERO AND MAY BE
      !                       REDUNDANT
      !                       IER=-2 NO ACTION BECAUSE M < N OR N < 1
      !                       IER>0 THE FIRST IER COMPONENTS OF IP CONTAIN
6115  !                       THE LABELS OF THE NONZERO COMPONENTS OF X, THE
      !                       REMAINING COMPONENTS OF X ARE ZERO AND MAY BE
      !                       REDUNDANT
      !  NOTE: ALL ARITHMETIC OPERATIONS ARE PERFORMED IN DOUBLE PRECISION,
      !  AN ITERATIVE IMPROVEMENT IS IMPOSSIBLE WITHOUT SAVING A AND B.
6120  !  THE ROUND OFF ERROR OF !!B-A*X!!**2 IS APPROXIMATLY GIVEN BY
      !  !!B(INITIAL)!!**2 - !!B(TRANFORMED)!!**2
      !-----------------------------------------------------------------------
            SUBROUTINE LILESQ(M,N,IER,A,B,X,VP)
            IMPLICIT NONE
6125        INTEGER(2) :: IER
            INTEGER(4) :: M,N,I,IP,J,K,L,L1,L2
            REAL(8) :: C,DELTA,EPS,P,Q,R,S,SIG,T,U,V,W
            REAL(8) :: A(M,N),B(M),VP(N),X(N)
            DATA EPS/1.D-8/
6130        DATA W/0.d0/,SIG/0.d0/,L2/0/,L1/0/,L/0/ ! pre-init.
            IER = 0
            IF (M<N.OR.N<1) GO TO 19
            DO J = 1,N; VP(J) = J
            ENDDO
6135  !  ROTATION LOOP
            DO 10 K = 1,N
```

```
      !  PIVOT ELEMENT
               U = 0.D0
               DO 4 J = K,N
6140             C = 0.D0
                 DO 2 I = K,M
                   IF (DABS(A(I,J))<=DABS(C)) GO TO 2
                   L2 = I
                   C = A(I,J)
6145        2    ENDDO
                 IF (C==0.D0) GO TO 4
                 S = 0.D0
                 T = 0.D0
                 DO I = K,M
6150               V = A(I,J)/C
                   S = S + V*V
                   T = T + V*B(I)
                 ENDDO
                 IF (U>=T* (T/S)) GO TO 4
6155             U = T* (T/S)
                 SIG = C*DSQRT(S)
                 W = T
                 L = J
                 L1 = L2
6160        4    ENDDO
                 IF (U==0.D0) GO TO 11
      !  PERMUTE A(K) AND B(K)
               I = IDINT(VP(L))
               VP(L) = VP(K)
6165           VP(K) = I
               DO I = 1,M
                 C = A(I,L)
                 A(I,L) = A(I,K)
                 A(I,K) = C
6170           ENDDO
               C = B(K)
               B(K) = B(L1)
               B(L1) = C
               DO J = K,N
6175             C = A(K,J)
                 A(K,J) = A(L1,J)
                 A(L1,J) = C
               ENDDO
      !  ROTATION OF THE LOWER COLUMN FRAGMENT OF A(K) AND B(K)
6180           U = SIG + A(K,K)
               V = A(K,K)/SIG
               DELTA = (B(K)+V*W)/U
               A(K,K) = -SIG
               B(K) = -V*W
6185           L = K + 1
               IF (L>M) GO TO 10
               IF (K>=N) GO TO 8
               DO J = L,N
                 S = V*A(K,J)
6190             P = DABS(S)
                 DO I = L,M
                   R = A(I,K)/SIG*A(I,J)
                   S = S + R
                   P = P + DABS(R)
6195             ENDDO
```



```
                 IF (DABS(S)<=EPS*P) S = 0.D0
                 T = (A(K,J)+S)/U
                 IF (DABS(T)<=EPS*DABS(S/U)) T = 0.D0
                 A(K,J) = -S
6200             DO I = L,M
                   Q = A(I,J)
                   P = T*A(I,K)
                   R = Q - P
                   IF (DABS(R)<=EPS*DABS(P)) R = 0.D0
6205               A(I,J) = R
                 ENDDO
               ENDDO
         8     DO I = L,M; B(I) = B(I) - DELTA*A(I,K); ENDDO
        10   ENDDO
6210   !  END OF ROTATION LOOP
             K = N
             GO TO 12
        11   K = K - 1
             IER = int(K,KIND=2)
6215   !  SQUARE OF THE EUCLIDEAN NORM
        12   S = 0.D0
             L = K + 1
             IF (K==M) GO TO 14
             DO I = L,M; S = S + B(I)*B(I); ENDDO
6220    14   A(2,1) = S
             IF (K==N) GO TO 16
       !  COMPONENTS OF X WHICH DO NOT REDUCE THE EUCLIDEAN NORM
             DO I = L,N
               DO J = L,N
6225             IP = IDINT(VP(J))
                 X(IP) = 0.D0
               ENDDO
             ENDDO
             IF (K==0) GO TO 20
6230   !  COMPUTATION OF X
        16   IP = IDINT(VP(K))
             X(IP) = B(K)/A(K,K)
             IF (K==1) GO TO 21
             DO J = 2,K
6235           L = K + 2 - J
               S = B(L-1)
               DO I = L,K
                 IP = IDINT(VP(I))
                 S = S - A(L-1,I)*X(IP)
6240           ENDDO
               IP = IDINT(VP(L-1))
               X(IP) = S/A(L-1,L-1)
             ENDDO
             GO TO 21
6245   !  ERROR CODE
        19   IER = IER - INT(1,KIND=2)
        20   IER = IER - INT(1,KIND=2)
        21   RETURN
             END SUBROUTINE
6250   !  Number of lines: 6250
```