

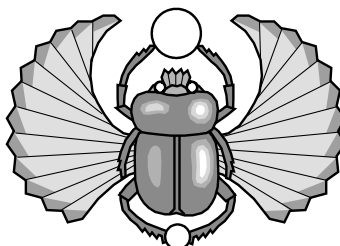
Multithread Version of P4 Including Parallelization of the VSOP87 Subroutine

Hans Jelitto
Hamburg, November 2015

© 2015 Hans Jelitto

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 multi-thread 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 planètes–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

The text of this brief program description (first three pages) is licensed under Creative Commons “CC” BY-NC-SA 4.0. Concerning the copyrights of H. Jelitto, the executable P4-4 program with all of its supplemental program files, text files, and data files listed in Table 1 of Ref. [1] can be used freely for private, scientific, and educational purposes, but may not be used for any commercial purpose. In case of use for any publication including any sort of presentation, appropriate quotation of the author(s) must be given. For the other program parts (see below), it has to be checked whether permission from the copyright owners is necessary. For any kind of commercial use, a written permission from the author is required. This program is distributed in the hope that it will be useful, but without any kind of warranty!

Further Copyrights

Further copyrights apply to: **1.** Subroutine VSOP87 and associated data files (see Table 1 in Ref. [1]), **2.** Program package FITEX (consisting of four subroutines at the end of the source code of P4 and P4-4), and **3.** Subroutine DELTA_T and numbered equations in the “Universal Time” section of Ref. [1]. Concerning these subroutines, see the copyright section in Ref. [1] on page 137!

P4-4 (Fortran 95)

PLANETENKORRELATION DER PYRAMIDEN VON GIZA

Parallelisierte Version fuer 4 Threads
 Programmlogik and Ergebnisse identisch zu p4

```

10      =
11      =
12      = P 4
13      = Programm
14      = zur Berechnung
15      = der Planetenposi-
16      = tionen und zur Bestim-
17      = mung des Zeitpunktes, der
18      = durch die Pyramidenanordnung
19      = bzw. Kammeranordnung vorgegeben
20      = ist. Grundlage sind Messungen namhaf-
21      = ter Aegyptologen sowie die planetarische
22      = Theorie V50P87 von Bretagnon und Francou
23      = (IMCCE, Paris). Das Programm ist eine viel-
24      = seitige Weiterentwicklung des Programms P3.
25      =
26      =
27      =
28      =
29      =

```

30 Hans Jelitto, Hamburg, 6. Juni 2015

Kurzbeschreibung

35 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:

40 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

45 2. Buch 2 (in Vorbereitung)

```

50      *
51      *
52      *
53      *
54      *
55      *
56      *
57      *
58      *
59      *
60      *
61      *
62      *
63      *
64      *
65      *
66      *
67      *
68      *
69      *
70      *
71      *
72      *
73      *
74      *
75      *
76      *
77      *
78      *
79      *
80      *
81      *
82      *
83      *
84      *
85      *
86      *
87      *
88      *
89      *
90      *
91      *
92      *
93      *
94      *
95      *
96      *
97      *
98      *
99      *
100     *
101     *
102     *
103     *
104     *
105     *
106     *
107     *
108     *
109     *
110     *
111     *
112     *
113     *
114     *
115     *
116     *
117     *
118     *
119     *
120     *
121     *
122     *
123     *
124     *
125     *
126     *
127     *
128     *
129     *
130     *
131     *
132     *
133     *
134     *
135     *
136     *
137     *
138     *
139     *
140     *
141     *
142     *
143     *
144     *
145     *
146     *
147     *
148     *
149     *
150     *
151     *
152     *
153     *
154     *
155     *
156     *
157     *
158     *
159     *
160     *
161     *
162     *
163     *
164     *
165     *
166     *
167     *
168     *
169     *
170     *
171     *
172     *
173     *
174     *
175     *
176     *
177     *
178     *
179     *
180     *
181     *
182     *
183     *
184     *
185     *
186     *
187     *
188     *
189     *
190     *
191     *
192     *
193     *
194     *
195     *
196     *
197     *
198     *
199     *
200     *
201     *
202     *
203     *
204     *
205     *
206     *
207     *
208     *
209     *
210     *
211     *
212     *
213     *
214     *
215     *
216     *
217     *
218     *
219     *
220     *
221     *
222     *
223     *
224     *
225     *
226     *
227     *
228     *
229     *
230     *
231     *
232     *
233     *
234     *
235     *
236     *
237     *
238     *
239     *
240     *
241     *
242     *
243     *
244     *
245     *
246     *
247     *
248     *
249     *
250     *
251     *
252     *
253     *
254     *
255     *
256     *
257     *
258     *
259     *
260     *
261     *
262     *
263     *
264     *
265     *
266     *
267     *
268     *
269     *
270     *
271     *
272     *
273     *
274     *
275     *
276     *
277     *
278     *
279     *
280     *
281     *
282     *
283     *
284     *
285     *
286     *
287     *
288     *
289     *
290     *
291     *
292     *
293     *
294     *
295     *
296     *
297     *
298     *
299     *
300     *
301     *
302     *
303     *
304     *
305     *
306     *
307     *
308     *
309     *
310     *
311     *
312     *
313     *
314     *
315     *
316     *
317     *
318     *
319     *
320     *
321     *
322     *
323     *
324     *
325     *
326     *
327     *
328     *
329     *
330     *
331     *
332     *
333     *
334     *
335     *
336     *
337     *
338     *
339     *
340     *
341     *
342     *
343     *
344     *
345     *
346     *
347     *
348     *
349     *
350     *
351     *
352     *
353     *
354     *
355     *
356     *
357     *
358     *
359     *
360     *
361     *
362     *
363     *
364     *
365     *
366     *
367     *
368     *
369     *
370     *
371     *
372     *
373     *
374     *
375     *
376     *
377     *
378     *
379     *
380     *
381     *
382     *
383     *
384     *
385     *
386     *
387     *
388     *
389     *
390     *
391     *
392     *
393     *
394     *
395     *
396     *
397     *
398     *
399     *
400     *
401     *
402     *
403     *
404     *
405     *
406     *
407     *
408     *
409     *
410     *
411     *
412     *
413     *
414     *
415     *
416     *
417     *
418     *
419     *
420     *
421     *
422     *
423     *
424     *
425     *
426     *
427     *
428     *
429     *
430     *
431     *
432     *
433     *
434     *
435     *
436     *
437     *
438     *
439     *
440     *
441     *
442     *
443     *
444     *
445     *
446     *
447     *
448     *
449     *
450     *
451     *
452     *
453     *
454     *
455     *
456     *
457     *
458     *
459     *
460     *
461     *
462     *
463     *
464     *
465     *
466     *
467     *
468     *
469     *
470     *
471     *
472     *
473     *
474     *
475     *
476     *
477     *
478     *
479     *
480     *
481     *
482     *
483     *
484     *
485     *
486     *
487     *
488     *
489     *
490     *
491     *
492     *
493     *
494     *
495     *
496     *
497     *
498     *
499     *
500     *
501     *
502     *
503     *
504     *
505     *
506     *
507     *
508     *
509     *
510     *
511     *
512     *
513     *
514     *
515     *
516     *
517     *
518     *
519     *
520     *
521     *
522     *
523     *
524     *
525     *
526     *
527     *
528     *
529     *
530     *
531     *
532     *
533     *
534     *
535     *
536     *
537     *
538     *
539     *
540     *
541     *
542     *
543     *
544     *
545     *
546     *
547     *
548     *
549     *
550     *
551     *
552     *
553     *
554     *
555     *
556     *
557     *
558     *
559     *
560     *
561     *
562     *
563     *
564     *
565     *
566     *
567     *
568     *
569     *
570     *
571     *
572     *
573     *
574     *
575     *
576     *
577     *
578     *
579     *
580     *
581     *
582     *
583     *
584     *
585     *
586     *
587     *
588     *
589     *
590     *
591     *
592     *
593     *
594     *
595     *
596     *
597     *
598     *
599     *
600     *
601     *
602     *
603     *
604     *
605     *
606     *
607     *
608     *
609     *
610     *
611     *
612     *
613     *
614     *
615     *
616     *
617     *
618     *
619     *
620     *
621     *
622     *
623     *
624     *
625     *
626     *
627     *
628     *
629     *
630     *
631     *
632     *
633     *
634     *
635     *
636     *
637     *
638     *
639     *
640     *
641     *
642     *
643     *
644     *
645     *
646     *
647     *
648     *
649     *
650     *
651     *
652     *
653     *
654     *
655     *
656     *
657     *
658     *
659     *
660     *
661     *
662     *
663     *
664     *
665     *
666     *
667     *
668     *
669     *
670     *
671     *
672     *
673     *
674     *
675     *
676     *
677     *
678     *
679     *
680     *
681     *
682     *
683     *
684     *
685     *
686     *
687     *
688     *
689     *
690     *
691     *
692     *
693     *
694     *
695     *
696     *
697     *
698     *
699     *
700     *
701     *
702     *
703     *
704     *
705     *
706     *
707     *
708     *
709     *
710     *
711     *
712     *
713     *
714     *
715     *
716     *
717     *
718     *
719     *
720     *
721     *
722     *
723     *
724     *
725     *
726     *
727     *
728     *
729     *
730     *
731     *
732     *
733     *
734     *
735     *
736     *
737     *
738     *
739     *
740     *
741     *
742     *
743     *
744     *
745     *
746     *
747     *
748     *
749     *
750     *
751     *
752     *
753     *
754     *
755     *
756     *
757     *
758     *
759     *
760     *
761     *
762     *
763     *
764     *
765     *
766     *
767     *
768     *
769     *
770     *
771     *
772     *
773     *
774     *
775     *
776     *
777     *
778     *
779     *
780     *
781     *
782     *
783     *
784     *
785     *
786     *
787     *
788     *
789     *
790     *
791     *
792     *
793     *
794     *
795     *
796     *
797     *
798     *
799     *
800     *
801     *
802     *
803     *
804     *
805     *
806     *
807     *
808     *
809     *
810     *
811     *
812     *
813     *
814     *
815     *
816     *
817     *
818     *
819     *
820     *
821     *
822     *
823     *
824     *
825     *
826     *
827     *
828     *
829     *
830     *
831     *
832     *
833     *
834     *
835     *
836     *
837     *
838     *
839     *
840     *
841     *
842     *
843     *
844     *
845     *
846     *
847     *
848     *
849     *
850     *
851     *
852     *
853     *
854     *
855     *
856     *
857     *
858     *
859     *
860     *
861     *
862     *
863     *
864     *
865     *
866     *
867     *
868     *
869     *
870     *
871     *
872     *
873     *
874     *
875     *
876     *
877     *
878     *
879     *
880     *
881     *
882     *
883     *
884     *
885     *
886     *
887     *
888     *
889     *
890     *
891     *
892     *
893     *
894     *
895     *
896     *
897     *
898     *
899     *
900     *
901     *
902     *
903     *
904     *
905     *
906     *
907     *
908     *
909     *
910     *
911     *
912     *
913     *
914     *
915     *
916     *
917     *
918     *
919     *
920     *
921     *
922     *
923     *
924     *
925     *
926     *
927     *
928     *
929     *
930     *
931     *
932     *
933     *
934     *
935     *
936     *
937     *
938     *
939     *
940     *
941     *
942     *
943     *
944     *
945     *
946     *
947     *
948     *
949     *
950     *
951     *
952     *
953     *
954     *
955     *
956     *
957     *
958     *
959     *
960     *
961     *
962     *
963     *
964     *
965     *
966     *
967     *
968     *
969     *
970     *
971     *
972     *
973     *
974     *
975     *
976     *
977     *
978     *
979     *
980     *
981     *
982     *
983     *
984     *
985     *
986     *
987     *
988     *
989     *
990     *
991     *
992     *
993     *
994     *
995     *
996     *
997     *
998     *
999     *
1000    *

```

Bezogen auf das Copyright von H. Jelitto stehen das

```

60      Programm P4-4 und die uebrigen Programmteile, mit Aus-
61      nahme der Datei "p4-manual-06-2015.pdf" und ihren vor-
62      hergehenden Versionen, fuer wissenschaftliche, private,
63      Ausbildungs- und paedagogische Zwecke zur freien Ver-
64      fuegung, solange der Name des Urhebers ordnungsgemaess
65      genannt wird, und duerten nicht fuer kommerzielle
66      Zwecke irgendeiner Art verwendet werden. Kommerzielle
67      Nutzung bedarf der schriftlichen Genehmigung. Fuer die
68      anderen Programmteile (A. bis C.), die im Folgenden
69      aufgezahlt sind, ist zu pruefen, ob eine Genehmigung
70      der Urheber bzw. Copyright-Inhaber erforderlich ist.

```

```

71      (Informationen zur Nutzung und zum Copyright der Datei
72      "p4-manual-06-2015.pdf" stehen zu Anfang jener Datei.)

```

```

73      Das Programm P4-4 wird in der Hoffnung zur Veruegung
74      gestellt, dass es fuer andere nuetzlich ist, jedoch
75      ohne irgendeine Art von Garantie oder Gewaehrleistung.

```

```

76      Die folgenden Angaben (A. bis D.) beziehen sich ent-
77      sprechend auf das Programm P4-4, die vorherige Version
78      P3 und alle zugehoerigen, unten aufgefuehrten Dateien.

```

```

79      A. Unterprogramm V50P87Y (basierend auf der Theorie
80      "Variations Seculaires des Orbits Planetaires") und
81      zugehoerige Dateien: P. Bretagnon und G. Francou,
82      Institut de mecanique celeste et de calcul des
83      ephemerides (IMCCE), 77 Avenue Denfert-Rochereau,
84      F-75014 Paris, France.

```

```

85      B. Programmpaket FITEX (bestehend aus 4 Unterprogrammen
86      im hinteren Programmteil): KIT, Karlsruhe Institute of
87      Technology (zuvor: FZK, Forschungszentrum Karlsruhe in
88      der Helmholtz-Gemeinschaft), Institut fuer Kernphysik,
89      Postfach 3640, D-76021 Karlsruhe. FITEX wurde von
90      G.W. Schweimer um 1972 entwickelt und erstmals ver-
91      oeffentlicht in: H.J. Gils: "The Karlsruhe Code MODINA
92      for Model Independent Analysis of Elastic Scattering
93      of Spinless Particles." KfK 3063, Nov. 1980, Kernfor-
94      schungszentrum Karlsruhe (KfK), Zyklotron Laboratorium,
95      and KfK 3063, 1. Supplement, Dec. 1983.

```

```

96      C. Umrechnung von "terrestrial time" (TT) in "universal
97      time" (UT) mittels delta-T = TT - UT: Fred Espenak,
98      und Jean Meeus, NASA Eclipse Web Site, Polynomial
99      expressions for DELTA-T.

```

```

100     D. Das Hauptprogramm P4-4 und die uebrigen Programmteile,
101     einschliesslich der Modifikation des Unterprogramms
102     V50P87 (-> "V50P87Y"): (c) 2014, 2015 Hans Jelitto,
103     Ewaldsweg 12, D-20537 Hamburg, Germany.

```

```

104     ----- Danksagung -----

```

```

105     Das Unterprogramm jdedate zur Umrechnung von JDE in
106     ein Kalenderdatum basiert auf einem Algorithmus aus dem
107     Buch von Jean Meeus: "Astronomical Algorithms", 1991,
108     Willmann-Bell, Inc., P.O.Box 35025, Richmond, Virginia
109     23235, USA, S.63. Dafuer und fuer die Aufllistung der
110     -----

```

gekürzten Reihen der VSOP87D-Parameter gilt mein herzlicher 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	Ausführbare 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	Ausführlichere Information zur Theorie "Planetary Solutions VSOP87"
out.txt	Ergebnis-Datei. Wenn diese nicht bereits existiert, wird sie bei entsprechender Option vom Programm erstellt.
inedit.t	Datei zum Editieren der Eingabeparameter --> Parametersatz fuer "inparm.t"
inparm.t	Input gemäss Schnellstart-Optionen
inpdata.t	Parameter f. FITEX, Kammer-Koordinaten in der Cheops-P. und Pyramiden-Koord.
inserie.t	Transitserien fuer Merkur und Venus
invsop3.t	VSOP87D, gekuerzt, Meeus; Astr. Alg. Polynomdarstellung der Bahnelemente, berechn. aus VSOP82, Meeus; Astr. Alg.
VSOP87A.mer	VSOP87A, kart. Koord. (Ekl. J2000.0) Merkur
VSOP87A.ven	(Diese und die folgenden Dateien enthalten die Parameter zur VSOP87-Theorie vollständig.)
VSOP87A.ear	Erde
VSOP87A.mar	Mars
VSOP87A.jup	Jupiter
VSOP87A.sat	Saturn
VSOP87A.ura	Uranus
VSOP87A.nep	Neptun
VSOP87A.emb	Erde-Mond-Schwerpunktsystem
VSOP87C.mer	VSOP87C, kart. Koord. (Ekl. d. Epoche) 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 Schnellstartoptionen - aber nicht im Eingabe-Menue angezeigt - 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 Abschnitte, 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 gewünscht 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 editiert werden koennen (Option 999).
- > e) Koordinaten der drei Kammern der Cheops-Pyramide zum Positionsvergleich mit den Planeten. 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. Automatische Erkennung und Markierung der Planetenkonstellationen 1 bis 14 bei Verwendung beliebigier Optionen.
- > j) Uebertragung der Positionen von Merkur bis Neptun ins Pyramidengebiet auf Basis der Pyramiden- bzw. Kammeranordnung (bei 3D-Berechnung mit FITEX, Einzelberechnung, Konst. 1 bis 14). Geographische Koord. (GPS) nur bei Konst. 12, alle Etappen, fuer Merkur bis Mars. Kombination VSOP87-Kurzversion und -Vollversion: Konstellationen, die mit der Kurzversion gefunden wurden, werden automatisch mit der Vollversion nachberechnet. Darueber hinaus: "Zeitintervall um Aphel bzw. um Perihel" auch fuer die Vollversion VSOP87 (sinvoll 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.
! > m) Zeitraeume werden nicht mehr mit der k-Nummer
! > des Aphel- bzw. Periheldurchgangs des Merkurs
! > angegeben, sondern mit der eher gebrauchlich-
! > en Jahreszahl.
! > n) Die Berechnungen mit VSOP87 wurde auf den Zeit-
! > 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
! > des Sonnensystems in einer Reihe: Sonne, Mer-
! > kur, Venus, Erde und optional auch Mars.
! > p) Zusätzlich werden Merkur- und Venustransite
! > vor der Sonnenscheibe registriert (VSOP87C).
! > q) Zum Testen der Transit-Berechnung kann man
! > 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-
! > 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-
! > 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,
! > waerzuehlich die Positionswinkel des Planeten
! > waehrend der Phasen in Bezug auf die scheinbar-
! > re Bewegungsrichtung der Sonne zu berechnen.
! > Hierbei ist eine Zeilenlaenge auf dem Monitor
! > von mindestens 148 Zeichen erforderlich.
! > t) Fuer die Transitphasen gibt es die zwei Zeit-
! > systeme "terrestrial (dynamical) time" (TT)
! > und "universal time" (UT). Die Umrechnung mit
! >  $\text{delta-T} = \text{TT} - \text{UT}$  wird ueber analytische Glei-
! > chungen erreicht (F. Espenak und J. Meeus,
! > 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. greg-
! > orianischer Kalender) oder es wird der gregori-
! > anische 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
! > 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-
! > dig von der Kalenderwahl).
! > 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-
! > 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
! > 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),
! > 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
! > ausfuehrlichere Kopfzeilen, jetzt in Englisch.
! > Am Ende des Programmlaufs wird die benoetigte
! > Rechenzeit angegeben (CPU-time).
! > -----
! > Optionen insgesamt:
! > -----
! > 1-15 --> Die wesentlichen astr. Berechnungen
! > 111 --> Information zu Autoren u. Copyrights
! > 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 "inset-2.t"
! > (0) --> Parameter einzeln eingeben
! > -----
! > Planetenpositionen: -----
! > 1. Anordnung der 3 Pyramiden
! > 2. Anordnung der 3 Kammern der Cheops-Pyramide
! > 3. Konjunktionen (Transit, Syzygium)
! > -----
! > VSOP87-Version: -----
! > 1. Kombination von Kurz- u. Vollversion VSOP87
! > 2. VSOP87 Kurzversion (Buch von J. Meeus)
! > 3. Keplersche Gleichung mit VSOP82 (Meeus)
! > 4. VSOP87 Vollversion (IMCCE, Internet)
! > -----
! > Koordinatensystem in VSOP87: -----
! > 1. Ekliptik der Epoche (VSOP87C)
! > 2. J2000.0 (VSOP87A, Vollv. und Kepl. Gl.)
! > -----
! > Umfang der Programm-Ausgabe: -----
! > 1. normal (eine Zeile pro Konstellation)
! > 2. detailliert (mehrere Zeilen pro Konstell.)
! > -----
! > 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
| 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
| 5. Zeitpunkt voellig frei und Minimierung der
|   Abweichung zwischen Pyramiden und Planeten-
|   anordnung durch Variation des Zeitpunkts
|
| -----
| "Sonnenposition": -----
| 1. genau suedlich Mykerinos-Pyramide (1D)
| 2. genau suedlich Chefren-Pyramide (1D)
| 3. unbestimmt (2D und 3D)
|
| -----
| Berechnung ("Sonnenposition" unbestimmt): -----
| 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
|
| -----
| Referenzsystem bei 2D-Berechnung: -----
| 1. Ekliptikales System
| 2. Merkurbahn-System, Transformtion A, B oder
|   C (Gerade "Sonne - Merkur-Aphel" = x-Achse,
|   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)
|
| -----
| "Polaritaet" bei Projektion (2D): -----
| 1. Blick vom ekliptikalen Nordpol
| 2. Blick vom ekliptikalen Suedpol
| 3. Beide Optionen 1. oder 2.
|
| -----
| Vorgegebene Hoehenlagen (3D): -----
| 1. Grundflaechen der Pyramiden
| 2. Schwerpunkte " "
| 3. Spitzen " "
|
| -----
| Kammerpos. in Cheops-P. (3D, z-Koord.): -----
| 1. Ostwaende der Kammern
| 2. Mitte " "
| 3. Westwaende " "
|
| -----
| 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)
|
| -----
| Planeten in Konjunktion: -----
| 1. Alle Merkur-Transite in einem Zeitintervall
| 2. Alle Venus-Transite " "

```

```

415 | 3. Merkur bis Erde in einer Reihe (Syzygium)
| 4. Merkur bis Mars " " ( " " )
| 5. Syzygium (Pkt. 3./4.) mit simultanem Transit
|
| -----
| Transit-Bestimmung: -----
| 1. Transite: gleiche eklipt. Laenge Planet/Erde
| 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
|
| -----
| Kalendersystem: -----
| 1. Automatische Wahl des Kalenders
|   (Greg. < 4712 BC < Julian. < 1582 AD < Greg.)
| 2. Gregorianischer Kalender fuer alle Zeiten
|
| -----
| Zeitsysteme: -----
| 1. "terrestrial dynamical time" (TT) bzw. JDE
| 2. "universal time" (UT), basierend auf delta-T
|   (NASA Eclipse Web Site).
|
| -----
| Ausgabegeraet: -----
| 1. Monitor
| 2. Monitor + Datei auf Festplatte ("out.txt")
| 3. Spezial-Programmausgabe (auf Mon. + Datei)
| 4. Programm-Abbruch
|
| -----
| Anmerkungen:
|
| 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
| 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
| werden und der Programmstart mit der Option 999 erfolgt.
|
| Anstelle des FORTRAN-77-Compilers (IBM Professional For-
| tran Compiler, Version 1.0, Ryan McFarland) wird jetzt un-
| ter Ubuntu Linux der GNU-Compiler "gfortran" verwendet,
| 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:
| 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.)
|
| Zum Unterprogramm VSOP87 bzw. VSOP87Y:
| Die VSOP87-Routine wurde dahingehend modifiziert, dass die
| umfangreichen Dateien der VSOP87-Theorie nur einmal gelesen

```

470

Zum Unterprogramm VSOP87 bzw. VSOP87Y:

Die VSOP87-Routine wurde dahingehend modifiziert, dass die umfangreichen Dateien der VSOP87-Theorie nur einmal gelesen

```

475 ! und im Rechenpeicher in ein Array geschrieben werden. Da-
! rüber hinaus wurde das Unterprogramm auf der Basis des API
! "OpenMP" parallelisiert, so dass 4 Threads gleichzeitig be-
! arbeitet werden koennen. (Fortran-95-Standard)
!
480 ! Bei den Konstellationen 13, 14, sowie den "quick start
! options" 371 und 372 wird automatisch auch die jeweilige
! 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.
!
485 ! Dieses Quellprogramm enthaelt auch Code-Abschnitte, die de-
! aktiviert wurden (durch "if", "ih" bzw. "it") und fuer
! spezielle Zwecke gedacht sind. Das Aktivieren einiger Zeilen
! durch Entfernen von z.B. "ih" 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 "if"):
! Fuer einige Programmblaende koennen mehr Dezimalstellen ange-
! zeigt werden. Dafuer sind entsprechende Format-Statements zu
! ersetzen. Schnellstart-Optionen 4, 9: s. Ende des Hauptpro-
! 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
! zusaetzlich vorab initialisiert und mit "pre-init." markiert.
!
500 !-----Module-----
! module base ! GRUNDLEGENDE VARIABLEN UND KONSTANTEN
! save
!
! integer(4) :: lmax(15), jp(12,6), il(3)
! real(8) :: xyr(37), re(78), pyr(40)
! real(8) :: ax, ay, az, bx, by, bz, cx, cy, cz, ao, ai, at
!
! real(8), parameter :: pi = 3.1415926535897932d0, &
! pldg = pi/180.d0, zjd0 = 2451545.d0, &
! gdp1 = 180.d0/pi, c = 299792458.d0, &
! tcen = 36525.d0, AE = 149597870610.d0, &
! tmi1 = 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")
! R0 = 695508000.d0, & ! R-Sonne (Brown/Christensen-Alsgaard)
! R3a = 6378136.600, R3p = 6356751.90d0, & ! 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.)
! tssid = (/ 87.9693d0, 224.7008d0 /), & ! T-siderisch ( " , ")
! 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, &
! 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)
! 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)
! 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/)
! real(8) :: ser(-180:170,2), ase(-180:170), zstart
! integer(4) :: ise(-180:170), isflag, ismax
!
! end module
!
565 ! program P4_4
! -----Hauptprogramm-----
!
!-----Deklarationen und Initialisierungen
! use base; use astro
! implicit double precision (a-h,o-z)
! 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)
! dimension :: xx(5), yy(5), test(10), ort(0:9,4), rcm(3), acm(3)
! dimension :: iw1(8), iw2(8) ! (threads)
! dimension :: ihis(100) !h
! character(1) :: tl(3), tra(2), tr, dp, ts, sl
! character(2) :: dd, dn, ds, dss, kon
! 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)
! 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', &
! 'Jup','Sat','Ura','Nep','E-M'/
!
590

```



```

650 data titab/'body      x[m]      z[m]      dr[m] /
651 data tt/      (pyramid positions) /
652 data text/      , , of the "planets" , , &
653 7*,      , , barycenter --> /
654 data plan/'Sun      , 'Mercury      , 'Earth      , &
655      , 'Mars      , 'Jupiter      , 'Saturn      , 'Uranus      , &
656      , 'Neptune      , 'Earth-Moon /
657 data str/      --- /,str2/ --- /,str3/ --- /
658 data emp/      --- /,dn/ /,ds/ * /,dss/ </,dp/ : /
659 data zide0/0.d0/,ifitrun/0/,zjdelim/0.d0/,izmin/0/ ! pre-init.
660 data i0/0/,ic/0/,ierr/0/ ! pre-init.

!-----Input-Daten und Programmstart
661 call inputdata(ipla,ilin,imod,imo4,ikomb,imo,lv,ivers, &
662 itrans,isep,iuniv,ical,ika,iaph,imax,step,ison,ih,i,rb,ijd, &
663 zmin,zmax,ak,zjdel,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop0,iout)
664 if (iout==4) then; write(6,*); go to 500; endif
665 call cpu_time(zia)
666 call date_and_time(zdate,ztime,zzone,iw1) ! (--> threads)
667 write(6,(/' <P4-> Computation started ...'))

! .. Die Input-Parameter werden in die Datei "inedit.t" geschrieben.
668 ! Man kann sie dann gegebenenfalls manuell an geeigneter Stelle in
669 ! "inparm.t" (Liste der Schnellstart-Optionen) einfügen, wobei
670 ! allerdings im Unterprogramm "inputdata" die Schnellstart-
671 ! Optionen angepasst werden muessen.
672 if (iop0/=999 .and. iout/=1) then
673   call inputfile(ipla,ilin,imod,imo4,ikomb,imo,lv,ivers, &
674   itrans,isep,iuniv,ical,ika,iaph,imax,step,ison,ih,i,rb,ijd, &
675   zmin,zmax,ak,zjdel,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop,2,iout)
676 endif

! .. Parameter fuer Spezial-Output (Konst. l2) --> isl2 = 1
677 isl2 = 0
678 if (((ipla==1 .and. iaph==1) .or. (ipla==2 .and. &
679   iaph==2 .and. ika==1)) .and. imod<=2 .and. &
680   ikomb==0 .and. iuniv==1 .and. io==2 .and. &
681   ison==5 .and. ijd==12 .and. iout==3) isl2 = 1

682 ! .. Erstellung weiterer Parameter
683 if (iout==1) then
684   ix = 6
685 else
686   ix = 1; open(unit=ix,file='out.txt')
687   write(6,('9x','Output file: "out.txt"'))
688   endif
689 10 write(6,*); kmin = 0; kmax = 0
690 if (ipla/=3) then
691   if (ijd>=1 .and. ijd<=14) then
692     ak = akon(ijd); if (ipla==2 .and. iek==1) ak = ak - 1.d0
693     call ephim(0,iaph,ipla,ical,ak,iak,zjdel,zjahr,delt)
694     endif
695   if (ijd==15 .and. imod==2 .and. iaph<=2) &
696     call ephim(0,iaph,ipla,ical,ak,iak,zjdel1,zjahr,delt)
697   endif
698   if (ipla==3 .or. (ipla/=3 .and. ijd==15 .and. &
699     (imod/=2 .or. (imod==2 .and. (iaph==3 .or. iaph==4)))) then
700     call ephim(2,iaph,ipla,ical,ak,kmin,zjdemin,zmin,delt)
701     call ephim(2,iaph,ipla,ical,ak,kmax,zjdemax,zmax,delt)

```

```

650   if (ipla==3) izmin = idint(zmin)
651   endif
652 ! .. Parameter fuer Transit-Pruefung
653 if (ilin==1) then
654   itransit=i; il(1)=1; il(2)=3; il(3)=2
655 elseif (ilin==2) then
656   itransit=2; il(1)=2; il(2)=3; il(3)=1
657 else
658   itransit=0; il(1)=1; il(2)=4; il(3)=1
659   endif
660 !-----Einlesen der Startwerte und Parameter fuer FITEX
661 ! sowie der Koordinaten der Pyramiden bzw. Kammern
662 j0 = 0
663 if (ipla==1) j0 = 18
664 if (ipla==3) e(1) = 1.d-6
665 if (ipla==1 .or. ipla==2) then
666   open(unit=10,file='inpdata.t')
667   do i=1,8+j0; read(10,*); enddo
668   read(10,*) dummy,(x0(i),i=1,7)
669   read(10,*) dummy,(e(i),i=1,7)
670   read(10,*)
671   read(10,*) dummy,(iw0(i),i=1,4)
672   read(10,*) dummy,(w0(i),i=1,3)
673   read(10,*)
674   read(10,*) dummy,iter
675   read(10,*) ; read(10,*)
676   ! Indizes von rp: 1. Pyr./Kammern, 2. Koordinaten und "Hoehe"
677   read(10,*) dummy,(rp(1,i),i=1,4)
678   read(10,*) dummy,(rp(2,i),i=1,4)
679   read(10,*) dummy,(rp(3,i),i=1,4)
680   read(10,*)
681   if (ison==2 .or. ipla==2) .and. isl2==0 then
682     read(10,*) dummy,diff(2),diff(3)
683   else
684     read(10,*)
685     do i=1,22-j0; read(10,*); enddo
686     do i=1,4; read(10,*) dummy,zjda(i); enddo
687     close(10)
688     if (ipla==2 .and. imod/=3) call chambers(ika,rp)
689     endif
690 !-----Einlesen der Transitserien zum Festlegen der Startnummer(n)
691 if (ilin<=2) then
692   do i=-180,170
693     ase(i) = z0; ise(i) = i0
694     if (.not.(iop0==-803 .and. ilin==2)) ser(i,1) = z0
695     ser(i,2) = z0
696     enddo
697     if (iop0/= -803) then
698       open(unit=10,file='inserie.t')
699       do i=1,5; read(10,*); enddo
700       do i=-150,150,5; read(10,*) idummy,(ser(i+j,1),j=0,4); enddo
701       do i=1,4; read(10,*); enddo
702       do i=-10,15,5; read(10,*) idummy,(ser(i+j,2),j=0,4); enddo
703       close(10)
704     endif

```

```

710      ismax = -10000; zstart = 99.99d0
      endif
!-----Weitere Initialisierungen
      do i=0,4; inum(i) = i0; enddo
      iflag = i0; iflag1 = i0; iflag2 = i0
      ifl = i0; ipos = i0; nfit = 7; mfit = 9
      ipar = i0; if (isep==4) ipar = 2
      indx = 1; iek = 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
720
725
730
735
740
745
750
755
760
765
! .. 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
! dwin = 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 = dmint(tsy*(1.d0-dwin/180.d0))
! if (tsprung<1.d0) tsprung = 1.d0
760
! .. Blickrichtung von der suedlichen ekliptikalischen Hemisphaere
! if (iekk==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

```

```

770
775
780
785
790
795
800
805
810
815
820
825
!-----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
      do i=1,8
      write(6, '(5f12.6)') (pyr(5*(i-1)+j), j=1,5)
      enddo

```

```

! . . . Zusatze zur 3-dim. Berechnung
if (ison>=4) then
  pyr(19) = pyr(18) - pyr(17)
  paz = pyr(17); pbz = pyr(18); pcz = pyr(19)
  write(6, (' x: ',3f12.3)) (pyr(i),i=11,13)
  write(6, (' y: ',3f12.3)) (pyr(i),i=14,16)
  write(6, (' z: ',3f12.3)) (pyr(i),i=17,19)
! . . . Erzeugung eines Vektors pd, der auf pa und pb senkrecht steht.
pdx = pby * paz - pay * pbz
pdy = paz * pbz - pbx * paz
pdz = pbx * pay - pax * pby
aba = dsqrt(pax*pax + pay*pay + paz*paz)
abb = dsqrt(pbx*pbx + pby*pby + pbz*pbz)
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
! . . . Modellwerte fuer FITEX
if (ison==5) then
  z(1) = z0; z(2) = z0; z(3) = z0
  z(4) = paz; z(5) = paz; z(6) = paz
  z(7) = pbx; z(8) = pby; z(9) = pbz
endif
endif
! . . . Laengen, Laengenverhaeltnisse, Winkel
if (ison<=2) then
  pyr(24) = pbx/pax
  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)
  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+pez*pbz)/(pyr(21)*pyr(22)))
  pyr(28) = dacos((pax*pcx+pay*pcy+pez*pcz)/(pyr(21)*pyr(23)))
  pyr(29) = dacos((pbx*pcx+pby*pcy+pbz*pcz)/(pyr(22)*pyr(23)))
endif
!-----Einlesen aller Parameter der V50P870-Kurzversion (Meeus)
20 if (imod==1) then
  open(unit=10,file='inv50p1.t')
  read(10,*)
  do n=1,12
    read(10,*) ; read(10,*) lmax(n)
    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)
      enddo
    enddo
  enddo
  close(10)
endif

```

```

!-----Bahnparameter als Polynome 3. Grades aus V50P82 (Meeus)
if (io==2 .or. irb/=1 .or. imod==3 .or. ipla==3) then
  open(unit=10,file='inv50p3.t')
  do ll=1,2
    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
  close(10)
endif
!-----Titelzeilen
do iu=ix,6,5
  call titel1(iaph,ijd,iu,ison,ipla,ilin,isep,nurtr, &
    iuniv,isl2,iop0)
  call titel2(iu,imod,ivers,irb,ipla, &
    ison,ihl,iek,ijd,ika,iaph,ilin,ical,ak,zjdel,zjahr,delt, &
    dwi,dwikomb,dwi0,dwi2,dwi3,iamax,step,ikomb,zmin,zmax)
! . . . Tabellenkopf
  call tabe(iaph,imod,iek,iu,io,ison,ipla,ilin,itrans,isl2, &
    iop0,iout)
enddo
if (iaph==5) go to 200
if (ipla==3) go to 300
! Anmerkung: In jedem Programmlauf wird nur eine
! der drei folgenden Hauptschleifen verwendet.
!----- 1. Hauptschleife -----
!----- 1. Hauptschleife (Pyramiden- und Kammerpositionen-
! sowie Aphel- und Perihelzeitpunkte des Merkur)
k = kmin
zk = dfloat(k)
if (imod==2 .and. ijd==15 .and. iaph<=2) zk = ak
  isw = 1; if (iaph<=2 .and. iout==3) isw = 2
  jmax = i0; ncount = i0
!-----JDE-Zeitpunkt (Merkur im und ausserhalb des Aphels)
120 zjde = zjdel
  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
      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
    else
      account = dfloat(ncount)
      if (ijd==15) then
        ak = zk + step * (account - zamax * 0.5d0)/ymer
        call ephim(i0,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
      endif

```

```

945     else
      zjde = zjde1 + step * (account - zamax * 0.5d0)
      call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
    endif
  endif
endif
950
if (ijd==i0) call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
ik = idmint(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

970 !.....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

975

!.....Variante 3 (Keplersche Gleichung, Polynome 3. Grades nach VSOP82)
if (io==2 .or. irb/=1 .or. imod==3) then
  inmax = 3
  if (io==2) imax = 4
  if (ir/=i0) go to 500
  do i=1,imax; ii = 6*i
  call vsop3(lv,i,ix,ir,time,res)
  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

980

!.....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)

```

```

1005 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
endif

1010 !.....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==10)) then
  if (ikomb==1 .and. imod==1) then
    imod = 2; dw1 = 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
    enddo
  enddo
endif
endif

1040 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

1045

1050 else
  if ((iaph==3 .or. iaph==4) .and. isw==1 .and. ijd==15) then
    ifl = i0; if (xyr(36)<=dwi2) ifl = 1
  endif
endif

1055 !.....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; dw1 = dwikomb; go to 140
  endif
  inum(2) = inum(2) + 1

```

```

1065 ! Sonnenposition
      call sonpos(ison,iek,ix,ip(3,1),rp(3,2),rp(3,3),rcm,dmi, &
      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
      if (isw==1) then
      call konst(ik,kon)
      do iu=ix,6,5
      if (ison==5) then
      if (ipla==2) then
      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)
      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
      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, &
      (xyr(30+i),i=1,4),dd,xyr(36)
      endif
      endif
      enddo
      else
      if (((xyr(36)<=dwi2.or.iaph<=2).and.ijd==2).and.ijd==15).or. &
      ijd/=15.or.imod==2) then
      if (iout==3) then
      call konst(ik,kon)
      delh = delh * 24.d0
      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
      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*pi/dg)>0.d0) sonne = sonne + 180.d0
      call reduz(sonne,i0,i0)
      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, &
      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
      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
      write(iu,256)kon,ik,zjahr,x(5)*gdpi,xma, &
      sonne,(xyr(30+i),i=1,4),dd,xyr(36)
      endif
      enddo
      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 '
      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
      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), &
      (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)
      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
      write(iu,365)dk,ik,zjde,xyr(35),ncount-iamax/2, &
      (xyr(30+i),i=1,4),dd,xyr(36)
      endif
      endif
      enddo
      endif
      !h
      call histogramm(xyr(36),ihis) !h
      endif
      endif
      !.....Weiterer Output
      do iu=ix,6,5
      if (ic=1 .and. imod/=3 .and. io==2 .and. is12==0) then
      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)
      write(iu,'(ix,6f9.6,f22.8,'%')' xyr(11),xyr(12), &
      xyr(14),xyr(15),xyr(17),xyr(18),xyr(36)
      call linie(iu,2)
      endif
      endif
      if (is12/=0) call linie(iu,1)
      if (is12==0 .and. ic==1 .and. imod==3 .and. io==2) call linie(iu,2)

```

```

1185 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): ', f12.6, &
          & ' ', f12.6)') re(34), re(40), re(52)
      else
        write(iu, '( " ascending node (M/V/E/Ma): ', f12.6, &
          (re(28+6*i), i=1,4)
        )' )
      endif
      write(iu, '( " inclination i (M/V/E/Ma): ', f12.6, &
        (re(29+6*i), i=1,4)
        )' )
      write(iu, '( " perihelion pi (M/V/E/Ma): ', f12.6, &
        (re(30+6*i), i=1,4)
        )' )
      if (imod/=3 .and. irb/=1) &
        write(iu, '( " ang. par. (omega, i, tau): ', f12.6, &
          ao*gdpi, ai*gdpi, at*gdpi
          )' )
      if (ison==5) then
        write(iu, '( " transl. X1, X2, X3; del-t: ', f12.6, &
          & f9.3, ' days')' ) (x(i), i=1,3), delt
      do i=4,6; call reduz(x(i), 1, i0); enddo
      write(iu, '( " Euler angl. X4, X5, X6; M: ', f12.6, &
        & fl3.0)') (x(i)*gdpi, i=4,6), xyr(35)
      write(6, '( " X7: ', f12.6)') x(7)
    endif
  enddo
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)
  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<=i0) .or. ison==4) write(iu, &
  & '( " 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 V50P87A,
! 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; dw1 = dwi0
endif

```

```

1235 if (iaph==3 .or. iaph==4) then
  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
endif

```

```

1245 ! . . Standardruecksprung
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)

```

```

!-----Ende der 1. Hauptschleife (Pyramiden- und Kammerpositionen)-----
go to 400

```

```

!===== 2. Hauptschleife =====
!=====

```

```

!-----2. Hauptschleife (freier Zeitpunkt und Minimierung von Fpos-----
! fuer Pyramiden- und Kammeranordnung, Tabelle 51 in "Pyramiden
! 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
! beim Ruecksprung (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
  (optimiert fuer alle Kammerzuoordnungen)
  zfact = 0.1d0; zstep = 0.2d0
endif

```

```

1275 !
!.....Startparameter fuer "fitmin"
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
do i=1,5
  xx(i) = z0; yy(i) = z0
enddo
xx(i) = zjde; go to 250

```

```

1280 call ephim(1, iaph, ipla, ical, ak, iak, zjde, zjahr, delt)
240 tau = (zjde - zjd0)/tml
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

```

```

1295

```

```

1300 !.....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

1305 !.....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

1310 !.....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)

1320 ! . . . 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; irestart = i0
      endif; imod = 1

1325 ! . . . Bedingung zum Aufruf von fitmin pruefen
      if (xyr(36)>dwi.and.(ifitrun==i0) go to 290
      if (ikomb==1) imod = 2

1330 ! . . . 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

1340 ! . . . verhindert, dass fitmin endlos ins vorherige Minimum faellt
      if (dabs(zjde-zjdevor)<=0.1d0) then
        zjde = zjdelim; go to 290
      endif; zjdevor = zjde

1345 !.....Hauptbedingung pruefen (ison = 5) . . . . .
      if (xyr(36)>=dwikomb) go to 290
      inum(2) = inum(2) + 1

1350 ! . . . 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)

```

```

1360 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
        enddo
      ih call histogramm(xyr(36),ihis) !h

1380 ! . . . Standardruecksprung
      290 zjump = xyr(36)*zfact + zstep
      zjde = zjde + zjump; x36 = xyr(36)
      if (zjde<=zjdemax) go to 220

1385 !-----Ende der 2. Hauptschleife (freier Zeitpunkt)-----
      go to 400

1390 !-----3. Hauptschleife -----
!-----3. Hauptschleife (Suche von Linear konstellationen)-----
! Syzygium von Sonne, Merkur, Venus, Erde und Mars,
! sowie Bestimmung der Transite von Merkur und Venus.

1400 ! "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
      zjdestep = zjde
      310 if (iilin==2.and.inum(0)>1.and.iop0/=-803) dfd = 0.02d0
      call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
      ik = idhint(ak)
      inum(0) = inum(0) + 1
      if (iilin==3) itransit = i0
      do i=1,2; tra(i) = ' '; enddo
      if (ison==5) ifitrun = i0
      if (iilin<=2) ifitrun = 1

1410
1415

```

```
!.....Startparameter fuer "fitmin", "sekante" und "ringfit"
```

```
320 if (ison==5) then
  iflag = i0; ke = 1
  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
    yy(i) = z0
  enddo
  xx(1) = zjde
  endif
```

```
330 zjde = xx(indx)
  call ephim(1,iaph,ipla,ical,ak,iak,zjde,zjahr,delt)
340 time = (zjde - zjd0)/tcen
  tau = (zjde - zjd0)/tmi1
  inum(1) = inum(1) + 1
```

```
!.....Variante 1 (VSOP87D, Kurzversion aus "Meeus", mult. threads)
  if (imod==1) then
```

```
    !$omp parallel do default(shared) private(i, resu)
      do i=1,12; call vsop1(i,tau, resu); re(i) = resu; enddo
    !$omp end parallel do
  if (i1in<=2) then
    call kartko(ison)
    do i=1,9; rk(i) = xyr(i); enddo
  endif
endif
```

```
!.....Variante 2 (VSOP87A/C, Vollversion)
```

```
350 if (imod==2) then
  do i=1(1),i1(2),i1(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)
    if (i1in<=2) rk(ii+j) = r(j)
  enddo
enddo
endif
```

```
!.....Variante 3 (Keplersche Gleichung, Polynome 3. Grades nach VSOP82)
```

```
if (imod==3) then
  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)
  re(27+ii) = res(3); re(30+ii) = res(6)
  if (i<=4) re(3*i-2) = res(11)
  enddo
endif
```

```
!.....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)
do i=1,6; call reduz(df(i),i0,i0); enddo
```

```
if (i1in==3) difm = dmax1(dabs(df(1)),dabs(df(2)),dabs(df(4)))
if (i1in==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
```

```
  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,sepl)
    difm = dabs(sepl)
  endif
endif
```

```
if (ison==5) yy(indx) = difm
!... Test-Ausdruck (-> !t)
```

```
!t difr = re(7)-re(1)
!t call reduz(difr,i0,i0)
!t do iu=ix,6,5; write(iu,'(1imod,ifit,dt,le-lm,jde,difm = ',2i2,&
  &f5.1,f6.1,f18.7,f13.7)imod,ifitrun,step,difr,zjde,difm;enddo
!t
```

```
!.....Hauptbedingung pruefen. ....
  if (difm>dw1.and.ifitrun/=1) go to 370
```

```
!... Ruecksprung fuer ikomb = 1
  if (ikomb==1.and.imod==1 .and.i1in==3) then
    ifitrun = 1; imod = 2
    dw1 = dwikomb
    go to 350
  endif
```

```
!... Minimierung des Gesamtwinkels difm mit "fitmin" fuer ison = 5
! (Das heisst, "ison" hat hier eine andere Funktion und bedeutet
! Minimumsuche.)
```

```
if (ison==5) then
```

```
  ifitrun = 1; step = 1.d0
  if (i1in>=3 .and. itransit==i0) then
    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
    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
```

```
    eep = e(1)
    if (ikomb==1 .and.imod==1 .and.isep>=3) eep=1.d2*e(1)
    imodus = 1
    if (ddx1<dfc.or.ddx2<dfc) imodus = 2
    call fitmin(imod,imodus,iaph,ke,xx,yy,eep,dfd,nu, &
      iflag,ddx1,ddx2,test,itin,indx,ix)
    zjde = xx(indx)
  endif
endif
```

```
if (ke==1 .or. (isep==1 .and.ke==5)) go to 330
```

```
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-
```



```

1535 ! 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
1545 ! . . . 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.
1550 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
1555 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
1560     do iu=ix,6,5
!       write(iu,'(i12,'. syzygy''') inum(3)
!     enddo
!   endif
1565 ! . . . Pruefen des Transits (nur bei imod = 1, 2)
!   if (itransit==1 .and.ison==5) then
!     if (itransit==i0.or.ilin<=2) call memo(zjde,zjahr, &
!       deit,df(1),df(2),df(3),difm,zmem,iak,imem)
1570     if (itransit==1 .or.itransit==2) then
!       call transit(itransit,ikomb,imod,ipla,ilin,iaph,ivers, &
!         isep,ical,iuniv,tr,sepl,itt,sep,zjde,id5,da5,dmo5, &
!         zjahr,rk,md,ddx1,ddx2,dfd,dfd,test,itin,is,irs,ix,pan,sd,sl,&
!         iop0,inum)
1575     tra(itransit) = tr
!     endif
!     . . . Ereignis mit Transit und Output
!     if ((ilin>=3 .and.itransit==2).or. &
!       (ilin<=2 .and.tr/=1)) then
1580     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
1585     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
1590           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

```

```

1595     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
1600     enddo
!   enddo
! else
!   if (iop0==803 .and.(zjahr<=13000.d0 .or. &
!     zjahr>=17000.d0)) go to 390; ts = ' '
!   if (tra(ilin)/='W'.and.tra(ilin)/='V') ts=tra(ilin)
1605   if (iuniv==2) call delta_T(zjde)
!   call jdedate(zjde,ical,ida,da,dmo)
!   if (ida(3)>=izmin) then
!     do iu=ix,6,5
1610     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
1615     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
1620         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) &
1625         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) &
1630         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) &
1635         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) &
1640         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
1650   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

```

```

1655     else
1656     if (isep==3) then
1657     if (itt==3) &
1658     write(iu,559)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1659     ((id5(l,i),dp,i=4,5),id5(l,6),l=1,5),sep,sl,irs
1660     if (itt==2) &
1661     write(iu,561)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1662     ((id5(l,i),dp,i=4,5),id5(l,6),str2,l=1,3,2), &
1663     (id5(5,i),dp,i=4,5),id5(5,6),sep,sl,irs
1664     if (itt==1) &
1665     write(iu,571)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1666     str2,str2,(id5(3,i),dp,i=4,5),id5(3,6), &
1667     str2,str2,sep,sl,irs
1668     else
1669     if (itt==3) &
1670     write(iu,759)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1671     ((id5(l,i),dp,i=4,5),id5(l,6),l=1,5),sep,sl, &
1672     (pan(i),i=1,5),sd(1),sd(2),irs
1673     if (itt==2) &
1674     write(iu,761)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1675     ((id5(l,i),dp,i=4,5),id5(l,6),str2,l=1,3,2), &
1676     (id5(5,i),dp,i=4,5),id5(5,6),sep,sl,pan(1), &
1677     str3,pan(3),str3,pan(5),sd(1),sd(2),irs
1678     if (itt==1) &
1679     write(iu,771)kon,ts,da5(3,7),dmo5(3),id5(3,3), &
1680     str2,str2,(id5(3,i),dp,i=4,5),id5(3,6), &
1681     str2,str2,sep,sl,str3,str3,pan(3), &
1682     str3,str3,sd(1),sd(2),irs
1683     endif
1684     if (itt==i0.and.iu==6) inum(2) = inum(2) - 1
1685     endif
1686     if (isep<=2 .and. iu==6) then
1687     if (ts=='m'.or.ts=='v') inum(3) = inum(3) + 1
1688     if (ts=='c'.or.ts=='c') inum(4) = inum(4) + 1
1689     endif
1690     enddo
1691     else
1692     ic = i0; iwo = i0; inum(2) = inum(2) - 1
1693     endif
1694     endif
1695     if (itransit=i0.or.ilin<=2) zjde0 = zjde
1696     read(*,*) it
1697     ! . . . Ereignis ohne Transit-Pruefung (z.B. imod = 3), Output
1698     else
1699     do iu=ix,6,5
1700     if (dabs(df(2))<1.d-4) then
1701     write(iu,456)kon, ', tra(1), tra(2), ik, &
1702     zjde,zjahr,delt,(df(i),i=1,3),difm
1703     elseif (dabs(df(3))<1.d-4) then
1704     write(iu,457)kon, ', tra(1), tra(2), ik, &
1705     zjde,zjahr,delt,(df(i),i=1,3),difm
1706     else
1707     write(iu,455)kon, ', tra(1), tra(2), ik, &
1708     zjde,zjahr,delt,(df(i),i=1,3),difm
1709     endif
1710     enddo

```

```

1715     call memo(zjde,zjahr,delt,df(1),df(2),df(3),difm,zmem, &
1716     iak,imem)
1717     endif
1718     endif
1719     ! . . . Ruecksprung fuer Transit-Pruefung
1720     370 if (itransit=1 .and. ison==5 .and. ilin>=3) then
1721     if (itransit=i0) zjde = zjde0
1722     if (ison==5 .and. ic=1 .and. ilin>=3) itransit=itransit+1
1723     if (itransit=i0.or.itransit>2 .or. ilin<=2) go to 380
1724     go to 320
1725     endif
1726     ! . . . Bedingung fuer Zeitsprung zur Verkuerzung der Rechenzeit
1727     380 if (ilin>=3 .and. dwin<=21.d0) then
1728     iflag2 = iflag1; iflag1=i0; if (dabs(df(4))<=dwin) iflag1=1
1729     endif; ifitrun = i0
1730     ! . . . Weiterer Output
1731     do iu=ix,6,5
1732     if ((ilin<=2 .and. (tra(1)/=''.or.tra(2)/='').and. &
1733     ((isep<=2 .or. (isep>=3 .and. itt/=0)).and. &
1734     ida(3)>=izmin)).or. (ic==1 .and. ilin>=3)).and. &
1735     i0==2 .and. iwo==1) then
1736     if (imod/=3) then
1737     call linie(iu,2+ipar); write(iu,57) (zmem(i),i=11,19)
1738     write(iu,57) (zmem(i),i=20,22); call linie(iu,2)
1739     endif
1740     if (ic==1 .and. imod==3 .and. i0==2) call linie(iu,2)
1741     immin = 1; if (imod==3) immin = 5
1742     immax = 4; if (ilin>=3) immax = 8
1743     if (immin<=immax) then
1744     do i=immin,immax; ii = 6*i
1745     call vsop3(lv,i,ix,ir,time,res); if (ir/=i0) go to 500
1746     zmem(25+ii) = res(1); zmem(28+ii) = res(5)
1747     zmem(26+ii) = res(2); zmem(29+ii) = res(4)
1748     zmem(27+ii) = res(3); zmem(30+ii) = res(6)
1749     enddo
1750     endif
1751     if (ilin<=2) then
1752     if (ivers==3) then
1753     write(iu,('' ascending node (M/V/E/Ma): ''',2fl2.6,&
1754     & ''',fl2.6)')zmem(34),zmem(40),zmem(52)
1755     else
1756     write(iu,('' ascending node (M/V/E/Ma): ''',4fl2.6)') &
1757     (zmem(28+6*i),i=1,4)
1758     endif
1759     write(iu,('' inclination i (M/V/E/Ma): ''',4fl2.6)') &
1760     (zmem(29+6*i),i=1,4)
1761     write(iu,('' perihelion pi (M/V/E/Ma): ''',4fl2.6)') &
1762     (zmem(30+6*i),i=1,4)
1763     else
1764     do i=31,78; re(i) = zmem(i); enddo
1765     call elements(iu,ivers,pla)
1766     endif
1767     call linie(iu,1+ipar)
1768     endif
1769     enddo
1770     390 if (ikomb==1 .and. imod==2) then; imod = 1; dwi = dwi0; endif

```

```

1775 ! . . Bedingter groesserer Zeitsprung
      if (i1in<=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
      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
      else
      zjde = zjde + stepl
      endif
      endif
1780
1785 ! . . Ergaenzung (Tabellenkopf fuer Transit-Test mit inum(2)=0)
      if (i1in<=2 .and. inum(2)==0) then
      do iu=ix,6,5
      call zweizeile(iu,io,zmem(1),ilin,imod,isep,ical,izp)
      enddo
      endif
1790
1795 !-----Ende der 3. Hauptschleife (Linearkonstellation, Transit)-----
!=====
!----- Ende der Hauptschleifen -----
!=====
1800 400 do iu=ix,6,5; if (io/=2) call linie(iu,1+ipar); enddo
! . . Ruecksprung bei Option -803 und Speichern von "inser-2.t"
      if (iop0==803) then
      if (ilin==1) then
      ilin = 2; zmin = -30000; zmax = 30000
      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 contime(2,zia,zib,iw1,iw2,ihour2,imin2,sec2) ! ( " )
      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)
      if (iplac<2 .and. imod<=2 .and. ison>=3) then
      write(iu, '(7x,a24,a33)') 'Frequency of deviations ', &
      & ' Fpos(0 to 5%) in steps of 0.05%:'
      call linie(iu,1)
      do i=0,4
      write(iu, '(2(3x,10i3)') (ihis(j+i*20),j=1,20)
      enddo; call linie(iu,1); write(iu,*)
      endif
      close(iu)
      enddo
      500 continue

```

```

1830 !-----Ende des Hauptprogramms-----
      stop
1835 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)
      57 format(1x,3(f9.4,f8.4,f9.6))
      58 format(1x,a2,i7,f10.3,3f8.3,3f7.1,f5.1,a2,f7.3)
      59 format(1x,a2,i7,f10.3,3f8.3,2f7.1,a7,f5.1,a2,f7.3)
      60 format(1x,a2,i7,f10.3,3f7.2,f5.2,a2,f7.3)
      61 format(1x,a2,i7,f10.3,2f8.3,i3,i4,3f7.1,f6.1,a2,f7.3)
      62 format(1x,a2,i7,f10.3,2f8.3,i3,i4,3f7.1,f6.2,a2,f7.3)
      63 format(1x,f14.5,f7.1,f7.2,f7.3,f7.2,3f7.1,f6.1,a2,f7.3)
      64 format(1x,a2,i7,f10.3,f8.2,f7.3,f7.2,4f7.1,a2,f7.3)
      65 format(1x,a2,i7,f7.1,f7.2,f7.3,f7.2,f6.2,a2,f7.3)
      66 format(1x,a2,i7,f10.3,f8.2,f7.3,f8.2,3f7.2,f6.2,a2,f7.3)
      67 format(1x,a3,i8,f13.3,f12.0,i6,1x,3f7.1,f5.1,a2,f7.3)
      68 format(1x,a3,i8,f13.3,f12.0,i2,i4,3f7.1,f6.1,a2,f7.3)
      69 format(1x,a3,i8,f13.3,f12.0,i6,1x,2f7.1,a7,f5.1,a2,f7.3)
      70 format(1x,a3,i8,f13.3,f12.0,i6,1x,3f7.2,f5.2,a2,f7.3)
      71 format(1x,a3,i8,f13.3,f12.0,i2,i4,3f7.2,f6.2,a2,f7.3)
      72 format(1x,a2,i7,f11.3,f8.3,f9.3,f9.4,f8.1,2f7.1,1x,a2,f7.3)
      73 format(1x,a2,i7,f15.5,f11.3,i3,i4,f8.2,f7.2,a2,f6.3)
      74 format(1x,a2,i7,f15.5,f11.3,i3,i4,f8.2,f7.2,f6.2,a2,f6.3)
      75 format(1x,a2,3a1,i7,f15.5,f11.3,5f8.3)
      76 format(1x,a2,3a1,i7,f15.5,f11.3,2f8.3,f6.1,f10.3,f8.3)
      77 format(1x,a2,3a1,i7,f15.5,f11.3,3f8.3,f6.1,f10.3)
      78 format(1x,a2,i7,f5.0,a5,i6,i3,2(a1,i2),4f8.3,f7.1,i5)
      79 format(1x,a2,a1,f4.0,a5,i6,i3,2(a1,i2),4(i4,2(a1,i2)),f7.1,a1,i4)
      80 format(1x,a2,a1,f4.0,a5,i6,i3,2(a1,i2),2(a10,i4,2(a1,i2)), &
      f7.1,a1,i4)
      81 format(1x,a2,a1,f4.0,a5,i6,1x,a8,2x,a8,i4,2(a1,i2),2(2x,a8), &
      f7.1,a1,i4)
      82 format(1x,a2,a1,i7,f5.0,a5,i5,i4,2(a1,i2),4f8.3,f7.1,i4)
      83 format(1x,a2,a1,f4.0,a5,i5,i4,2(a1,i2),f7.1,a1,i3)
      84 format(1x,a2,a1,f4.0,a5,i5,i4,2(a1,i2),2(a10,i4,2(a1,i2)), &
      f7.1,a1,i3)
      85 format(1x,a2,a1,f4.0,a5,i5,2a10,i4,2(a1,i2),2a10,f7.1,a1,i3)
      86 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)
      87 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)
      88 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)
      89 format(1x,a2,a1,f4.0,a5,i5,1x,5(i4,a1,i2,a1,i2),f7.1,1x,a1, &
      3x,5f8.2,4x,2f8.2,i6)
      90 format(1x,1x,a1,3x,f8.2,a8,f8.2,a8,f8.2,4x,2f8.2,i6)
      91 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)
      92 . . Ausgabe einer grosseren Stellenanzahl zur Feinabstimmung bzw.
      93 Minimierung von F[%] fuer die Schnellstart-Optionen 4 und 9.
      94 Dies wurde verwendet fuer Buch 1.
      95 Suche in der Umgebung des Merkur-Aphels bzw. Merkur-Perihels
      96 format(1x,f14.5,f8.2,f7.2,f8.4,f6.1,3f7.1,f5.1,a2/65x,f14.8)
      97 format(1x,f14.5,f8.2,f7.2,f7.3,f7.2,f5.1,a2/65x,f14.8)
      end program P4_4

```

```

1890 subroutine inputdata(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
      itrans,isep,iuniv,ical,ika,iaph,iamax,step,ison,ih,i,irb,ij,d, &
      zmin,zmax,ak,zjdel,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop0,iout)
!-----Inputdaten und Programmstart-----
implicit double precision (a-h,o-z)
character(36) :: com
data ita/0/ ! pre-init.
iy = 6; ipla = 1; itrans = 1
io = 0; ire = 0; z0 = 0.d0
write(iy, '(//27x,27(.....))')
write(iy, '(30x, "PLANETARY CORRELATION" )')
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/), &
& 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)', &
'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
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
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

! . . . Verborgene Optionen fuer Tabellen aus beiden oben genannten
! Buechern, s.a. im Programmkopf unter "Neue Optionen, b")
if (iop==0 .and.iop==15).or. &
(iop==390 .and.iop==392).or.(iop==400 .and.iop==402).or. &
(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. &
(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. &
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
ire = 1; call emes(ire,com,dm)
enddo

```

```

! . . Auswertung der eingegebenen Option
if (iop<0 .or.iop>15) then
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
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
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
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
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
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
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
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

! . . Einlesen der Parameter aus "inparm.t"
call inputfile(ipla,ilin,imod,imo4,ikomb,io,lv,ivers, &
itrans,isep,iuniv,ical,ika,iaph,iamax,step,ison,ih,i,irb,ij,d, &
zmin,zmax,ak,zjdel,dwi,dwikomb,dwi2,dwi3,nurtr,iek,iop,1,iout)
return

! . . . Menues fuer Einzeleingabe der Parameter.....
! . . Planetenpositionen
do
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)
enddo

```

```

! . . Linearkonstellation, Transite
ilin = 4
if (ipla==3) then
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
call emes(ire, com, dm)
enddo
endif

2020 ! . . VSOP, Theorie-Variante
! 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
if (ipla/=3) then
write(iy, '( " VSOP87 combi. (1), short version (2), '' / &
& ', advance='no')
read(*, *, iostat=iox) imod
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')
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')
read(*, *, iostat=iox) imod
if (imod>=1 .and. imod<=2 .and. iox==0) exit
endif
endif
call emes(ire, com, dm)
enddo

2045 ! Aendern des Parameters "imod"
! (imod4 wird eingefuehrt, da imod wechselt, falls ikomb = 1 ist.)
! imod4 = 0
if (imod==1) ikomb = 1
if (imod==2) imod = 1
if (imod==4) then
imod = 2; imod4 = 1
endif

2050 ! . . 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
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
call emes(ire, com, dm)

2060

```

```

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.)
! if (ipla==3 .and. ikomb==1 .and. ilin>=3) then
! do
! write(iy, '( " Check planetary transit yes (1), no (2) : '' &
! & ', advance='no')
! read(*, *, iostat=iox) itran
! if ((itran==1 .or. itran==2) .and. iox==0) exit
! call emes(ire, com, dm)
! enddo
! if (itran==2) io = 1
! endif

2080 ! . . 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
! write(iy, '( " Date equ.L.(1), nearest (2), phases (3)'' / &
! & ', advance='no')
! read(*, *, iostat=iox) isep
! if (isep>=1 .and. isep<=4 .and. iox==0) exit
! call emes(ire, com, dm)
! enddo
! endif

2090 ! . . Julian/Gregorian calendar: Automatic choice of calendar or
! only Gregorian calendar
! ical = 0
! do
! write(iy, '( " Calendar only Greg. (1), Jul./Greg. (2) : '' &
! & ', advance='no')
! read(*, *, iostat=iox) ical
! if ((ical==1 .or. ical==2) .and. iox==0) exit
! call emes(ire, com, dm)
! enddo

2100 ! . . Terrestrial Time bzw. Universal Time
! iuniv = 1
! if (itran==1 .and. ilin<=2 .and. isep>=3) then
! do
! write(iy, '( " Time system
! & ', advance='no')
! read(*, *, iostat=iox) iuniv
! if ((iuniv==1 .or. iuniv==2) .and. iox==0) exit
! call emes(ire, com, dm)
! enddo
! endif

2110 ! . . Zuordnung der Planeten Erde (E), Venus (V) und Merkur (M) zu
! Koenigs-, Koeniginnen- und Felsenkammer in dieser Reihenfolge
! ika = 0

```

```

2125 if (ipla=2 .and. imod/=3) then
do
write(iy, '( " Planets E-V-M (1), E-M-V (2), V-E-M (3), "/ &
& ', advance='no')
read(*, *, iostat=i ox) ika
if (ika=1 .and. ika<=6 .and. iox==0) exit
call emes(ire, com, dm)
enddo
endif

2130

2135 ! .. Zeitpunkte im/um Aphel bzw. Perihel oder freier Zeitpunkt
iaph = 1
iamax = 0
step = 24.00
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=i ox) 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=i ox) 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=i ox) iaph
if (iaph>=1 .and. iaph<=4 .and. iox==0) exit
else
write(iy, '( " Passage aphelion (1), perihelion (2) : ' ' &
& ', advance='no')
read(*, *, iostat=i ox) iaph
if ((iaph==1 .or. iaph==2) .and. iox==0) exit
endif
call emes(ire, com, dm)
endif

2160 write(iy, '( " Steps per Mercury passage : ' ' ', advance='no')
read(*, *, iostat=i ox) iamax
if (iimax>0 .and. iimax<=200000 .and. iox==0) exit
call emes(ire, com, dm)
enddo

2170 do
write(iy, '( " Steps per Mercury passage : ' ' ', advance='no')
read(*, *, iostat=i ox) iamax
if (iimax>0 .and. iimax<=200000 .and. iox==0) exit
call emes(ire, com, dm)
enddo

2175 write(iy, '( " Step width (hours, real) : ' ' ', advance='no')
read(*, *, iostat=i ox) step
if (step>z0 .and. iox==0) exit
call emes(ire, com, dm)
enddo

2180 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=i ox) 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

2200

2205 ! .. 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=i ox) ison2
if ((ipla==1 .and. ison2>=1 .and. ison2<=3) .or. &
(ipla==2 .and. (ison2==2 .or. ison2==3))) .and. iox==0) exit
enddo
call emes(ire, com, dm)
endif

2220 if (ison2==2) ison = 4
if (ison2==3) ison = 5
endif

2225 ! .. 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=i ox) ihi
if (ihi>=1 .and. ihi<=3 .and. iox==0) exit
call emes(ire, com, dm)
enddo
endif

```

```

! . . Grundebene Ekliptik, Merkur- oder Venusbahn
2245  irb = 1
      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=i ox) 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
2265  write(iy, '( " Constell. (1..14), years (15), JDE (0) : ' &
          & ), advance='no')
        endif
        read(*,*, iostat=i ox) ijd
        if (ijd>=0 .and.ijd<=15 .and.iox==0) exit
        call emes(ire, com, dm)
      enddo
    endif
2270  ak = z0
      zmin = z0
      zmax = z0
      if (ijd==15) then
2275  do
        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
        enddo
        else
2280  do
          write(iy, '( " from year (real): ' ' ', advance='no')
          call pcheck(1, zmin, 1, dm, imod, ire)
          if (ire==0) exit
        enddo
      do
2285  write(iy, '( " until year (real): ' ' ', advance='no')
          call pcheck(1, zmax, 1, dm, imod, ire)
          if (zmin>=zmax .and.ire==0) then
            call emes(ire, com, dm)
            ire = 1
          endif
          if (ire==0) exit
        enddo
2295  endif
      endif
      if (ipla==3) then
        step = z0
        if (ilin>=3 .and.ikomb==0) then
2300  do

```

```

        write(iy, '( " Step width [hrs] (min.-search 0.) (real) : ' &
          & ), advance='no')
        read(*,*, iostat=i ox) 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
      zjdel = z0
      if (ijd==0) then
      do
2315  write(iy, '( " JDE (real) : ' ' ', advance='no')
          call pcheck(1, zjdel, 3, dm, imod, ire)
          if (ire==0) exit
        enddo
      endif
2320  ! . . Winkelintervall bzw. relativer Fehler
      dwi = z0
      dwi2 = z0
      dwi3 = z0
      dwikomb = z0; dm = 99.99d0
2325  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
      enddo
      else
2340  do
        if (ison<=2) then
          if (imod/=3) dm = 10.d0
2345  write(iy, '( " Tolerance ecl. long. VSOP short (real) ', &
            & ' ' ' : ' ' ', advance='no')
        else
2350  if (iaph/=5 .or.(iaph==5 .and.ikomb==1)) then
          write(iy, '( " Max. F-pos VSOP short ver. (real) [%] ', &
            & ' ' ' : ' ' ', advance='no')
        else
          write(iy, '( " Max. F-pos, VSOP short, start fitmin [%] ', &
            & ' ' ' : ' ' ', advance='no')
        endif
        call pcheck(2, dwi, 1, dm, imod, ire)
        if (ire==0) exit
      enddo
2355  do
        if (ison<=2) then
2360  if (ison<=2) then

```

```

2365 if (imod/=3) dm = 10.d0
      write(iy, '( : ', advance='no')
      & ' : ', advance='no')
      else
      if (iaph/=5 .or. (iaph=5 .and. ikomb==1)) then
        write(iy, '( : ', advance='no')
        & ' : ', advance='no')
      else
        write(iy, '( : ', advance='no')
        & ' : ', advance='no')
      endif
      call pcheck(2, dwikomb, 1, dm, imod, ire)
      if (ire==0) exit
    enddo
    endif
    if (iaph==3 .or. iaph==4) then
      do
        write(iy, '( : ', advance='no')
        & ' : ', advance='no')
        call pcheck(2, dwi2, 1, dm, imod, ire)
        if (ire==0) exit
      enddo
    do
      write(iy, '( : ', advance='no')
      & ' : ', advance='no')
      print beyond aphelion/per. [%]', &
      call pcheck(2, dwi3, 1, dm, imod, ire)
      if (ire==0) exit
    enddo
    endif
  endif
  if (ipla==3 .and. ilin>=3) then
    if (ikomb==0) then
      do
        write(iy, '( : ', advance='no')
        & ' : ', advance='no')
        call pcheck(2, dwi, 1, dm, imod, ire)
        if (ire==0) exit
      enddo
    else
      do
        write(iy, '( : ', advance='no')
        & ' : ', advance='no')
        call pcheck(2, dwi, 1, dm, imod, ire)
        if (ire==0) exit
      enddo
    do
      write(iy, '( : ', advance='no')
      & ' : ', advance='no')
      call pcheck(2, dwi, 1, dm, imod, ire)
      if (ire==0) exit
    enddo
  endif
  ! . . 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, '( : ', advance='no')
      & ' : ', advance='no')
      read(*, *, iostat=iox) nurtr
      if ((nurtr==1 .or. nurtr==2) .and. iox==0) exit
      call emes(ire, com, dm)
    enddo
  endif
  ! . . Blickrichtung auf die Planetenbahnen
  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
          write(iy, '( : ', advance='no')
          & ' : ', advance='no')
          read(*, *, iostat=iox) iek
          if (iek>=1 .and. iek<=2 .and. iox==0) exit
        else
          write(iy, '( : ', advance='no')
          & ' : ', advance='no')
          read(*, *, iostat=iox) iek
          if (iek>=1 .and. iek<=3 .and. iox==0) exit
        endif
        call emes(ire, com, dm)
      else
        iek = 1
        if ((ijd>=6 .and. ijd<=11) .or. ijd==13 .or. ijd==14) iek<=2; exit
      endif
    enddo
  endif
  ! . . Ausgabe
  if (io==0) then
    io = 2; if (iaph==5) io = 1
    if (imo4==0 .and. iaph/=5) then
      do
        write(iy, '( : ', advance='no')
        & ' : ', advance='no')
        read(*, *, iostat=iox) io
        if ((io/=2 .or. io==2) .and. iox==0) exit
        call emes(ire, com, dm)
      enddo
    endif
  endif
  ! . . Ausgabegeraet
  do
    if (imod<=2 .and. ipla<=2 .and. ison==5) then
      write(iy, '( : ', advance='no')
      & ' : ', advance='no')
      read(*, *, iostat=iox) iout
      if (iout>=1 .and. iout<=4 .and. iox==0) exit
    else
      write(iy, '( : ', advance='no')
      & ' : ', advance='no')
      read(*, *, iostat=iox) iout
      if ((iout==1 .or. iout==2 .or. iout==4) .and. iox==0) exit
    endif
  enddo

```



```

2480      endif
      call emes(ire,com,dm)
      enddo
      end subroutine

2485      subroutine inputfile(ipla,ilin,imod,imo4,ikomb,iol,lv,ivers, &
      itrans,isep,iuniv,ical,ika,iaph,iamax,step,ison,ih,i,rb,ijd, &
      zmin,zmax,ak,zjdel,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,itrans,isep,iuniv
      read(10,*) ical,ika,iaph,iamax,step
      read(10,*) ison,ih,i,rb,ijd
      read(10,*) zmin,zmax,ak,zjdel
      read(10,*) dwi,dwikomb,dwi2,dwi3
      read(10,*) nurtr,iek,iol,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,itrans,isep,iuniv
        write(10,'(3i3,i6,f10.5)') ical,ika,iaph,iamax,step
        write(10,'(3i3,i4)') ison,ih,i,rb,ijd
        write(10,'(3f13.5,f15.5)') zmin,zmax,ak,zjdel
        write(10,'(4f8.3)') dwi,dwikomb,dwi2,dwi3
        write(10,*) ('-',i=1,59)
        write(10,*) ('*',i=1,27), ' END ', ('*',i=1,27)
      endif
      close(10)
      end subroutine

2520      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

2530      subroutine pchange(imodus,iz,jz,rx,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)

```

```

2540      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
      end subroutine

2545      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

2585      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,'(///)') incorrect input. ', &
        & '(max. ',f6.2, ').'(//)' dm
      end subroutine

2590      if (ire==3) write(iy,'(///)') incorrect input.'(//)'
      if (ire==2) write(iy,'(///)') incorrect input. ', &
        & a36/)'com
      if (ire==3) write(iy,'(///)') incorrect input. ', &
        & '(max. ',f6.2, ').'(//)' dm
      end subroutine

2595      if (ire==3) write(iy,'(///)') incorrect input. ', &
      & '(max. ',f6.2, ').'(//)' dm
      end subroutine

```

```

2600 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' /
2605 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 = i->
  if (a1<ep.or.a2<ep) kon = tkon(i)
enddo
end subroutine
2615 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 = 365.248d0; B = 0.d0; C = -4711.9986d0 ! (Programm P3)
! C
2640 ! . . 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
2650 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)) &
  iak = ak + 0.5d0
endif
! . . Apheldurchgang der Erde
! c day = 2451547.507d0 + 365.2596358d0 * (ak + 0.5d0) &
! + 1.58d-8 * (ak + 0.5d0)**2
end subroutine
2675 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,ak,iak,zjd,y,delt)
if (y>-500.d0 .and.y<=500.d0) then
  u = y/100.d0
  delt = 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
  delt = 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
  delt = 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
  delt = 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
  delt = 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 &
  + 0.00000000875d0 * t**7
elseif (y>1860.d0 .and.y<=1900.d0) then
  t = y - 1860.d0
  delt = 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

```



```

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

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

2850 subroutine weekday(ZJD,wd)
      !-----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', /
      wd = wday(idnint(dmod(dint(ZJD + 700000001.5d0),7.d0)))
      end subroutine

2860 subroutine vsop1(l,tau, resu)
      !-----Berechnung der ekliptikalen Koordinaten (VSOP87D-Kurzversion)-----
      use base, only : gdpi,z0,lmax,ip; use astro, only : par1
      ! implicit double precision (a-h,o-z)
      resu = z0
      do j=1,lmax(l)
         sum0 = z0
         do i=1,ip(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)
      enddo

2870 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)
      dimension :: r(6),rku(3),md(0:9)
      character(11) :: afile(9),cfile(8)
      data afile/ 'VSOP87A.mar', 'VSOP87A.mer', 'VSOP87A.ven', 'VSOP87A.ear', &
                'VSOP87A.mar', 'VSOP87A.jup', 'VSOP87A.sat', 'VSOP87A.ura', &
                'VSOP87A.nep', 'VSOP87A.emb' /
      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))
         if (ivers==3) open(unit=10,file=cfile(ibody))
      endif

```

```

2895 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)
      write(6,/' ' x, y, z = ',3f14.10') (r(i),i=1,3)
      write(6,/' ' vx,vy,vz = ',3f14.10') (r(i),i=4,6)
      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
      enddo
      end subroutine

2900
      !-----subroutine vsop3(l,k,ix,ke,time,res)-----
      ! Bahn-Elemente, abgeleitet aus VSOP82 (nach Meeus)---
      ! 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,gdpl
      ! use astro, only : par3
      ! implicit double precision (a-h,o-z)
      ! dimension :: res(12)
      ! u360 = 360.d0; ke = 0
      ! eps = 1.d-13
      ! 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)
      !       res(j) = resu
      !    enddo
      ! enddo
      res(7) = res(1) - res(6)
      if (res(7)<0.d0) res(7) = res(7) + u360
      res(8) = res(6) - res(5)
      if (res(8)<0.d0) res(8) = res(8) + u360

2915 ! .. Loesung der Keplerschen Gleichung (Resultat: zen)
      ii = 0
      E = res(3)
      zm = res(7)*pidg
      ze = zm
      itmax = 100 ! Maximalzahl der Iterationen

2930 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)
               zen = ze + (zm + E*d sin(ze) - ze)/(1.d0 - E*dcos(ze))
            else
               ! 2. Fixpunktverfahren (Keplersche Gleichung)
               zen = zm + E*d sin(ze)
            endif
            if (dabs(zen-ze)<eps) exit
            if (ii>itmax) then; ke = 2; go to 20; endif
            ii = ii+1
            ze = zen
         enddo
      else
2950

```

```

! 3. Sekantenverfahren (verwendet Sekantensteigung)
ke = 1
ze2 = zm
fze2 = zm + E*dsein(ze2) - ze2
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
! gegenueber "sekante", da die Keplersche
! Gleichung deutlich weniger Rechenzeit
! benoetigt als "Ringfit" selbst.)
endif
go to 30

! .. zu viele Iterationen
20 do iu=ix,6,5
write(iu,'(711) ----> error in "vsop3"', &
& '(Keplers equation), ke = ', IZ/) ) ke
enddo
return
res(9) = zen*gdpi
30 if (res(9)<0.d0) res(9) = res(9) + u360

! .. 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
res(11) = res(10) + res(6)
if (res(11)>u360) res(11) = res(11) - u360
end subroutine

subroutine transit(ip, ikomb, imod, ip1a, ilin, iap, ivers, isep, &
ical, iuniv, tr, sepmin, itt, sep, zjde, id5, da5, dmo5, zjahr, &
rk, md, ddx1, ddx2, dfd, test, itin, is, ives, ix, pan, sd, sl, iop0, inum)
!-----Ueberpruefung der Transite von Merkur bzw. Venus-----
! Die berechneten Zeitpunkte sind optional dieselbe ekleptikale
! Laenge bei Erde und Merkur bzw. Venus, die minimale Separation
! 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
! 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
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(0:19), inum(0:4)
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','i','i','c','c'/'
data idr/'0','blim','0','d0','ba','0','d0','ang','0','d0','shift','0','d0'/' pre-init.
! .. Einige Konstanten
T = (zjde-zjd0)/tcen
Axl D. Wittmann: we = Schiefe der Ekliptik der Epoche
we = (23.4458042d0 - 0.856033d0 * &
dsin(0.015306d0 * (T + 0.507474d0))) * pdig

```

```

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 Abschaeetzung.)
do j=1,2; tcorr(j) = tsyn(j)/tsid(j); enddo
ee = dsqrt(R3a*R3a-R3p*R3p)/R3a
R3 = R3p/(AE*dsqrt(1.d0-(ee*dsein(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))))
bmin1 = a-bp; bmin2 = a-bp-b3
bmax1 = a+bp; bmax2 = a+bp+b3

!.....OPTIONEN 1/ 2: gleiche eklept. Laenge u. minimale Separation
if (isep==1) then
din = dcos(zi(ip)*pdg*tcorr(ip))
dre = (re(3*ip-1)-re(8))*pidg
ba = din*datan(re(3*ip)*dsein(dre)/(re(9)-re(3*ip)*dcos(dre)))
bap = dabs(ba)
else
bap = sepmin
endif
if (ikomb==1 .and. imod==1) bmax2 = bmax2*1.800
hout = bmax2*1.01d0; tr = tp(6)
if (bap<=bmin2) tr = tp(2*ip-1)
if (bap>bmin2 .and. bap<=bmax2) tr = tp(2*ip)
if (bap>bmax2 .and. bap<=hout .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)
endif
do iu=ix,6,5; write(iu,'(a15,a18,i3,5f8.5)') ip,bmin2,bmin1,' &
' bmax1,bmax2,bap = ', ip,bmin2,bmin1,bmax1,bmax2,bap; enddo

! .. Min. Separation (sep) zw. Sonne und Planet in
! Bogensekunden. Bei "plus" passiert der Planet das
! das Sonnenzentrum noerdlich, bei "minus" suedlich.
if (isep==1) then
sep = ba*wfact
else
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
endif

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

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

```

```

3070 do j=1,5
      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
        step = 5.d-2; iflag = 0
        ddx1 = dfd + 1.d0; nu = 0
        if (ilinc=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 ! V50P87-Berechnung mit Beruecksichtigung der Lichtlaufzeit
        if (imod=1) then
          call vsop1tr(ip,rk,(xj2-zj0-del)/tml,del,r3i,ept,inum,resu)
        else
          call vsop2tr(xj2-del,ivers,ip,md,ix,prec,lu,r,rk, &
            ierr,del,r3i,ept,inum,rku)
        endif
        if (iex=1) go to 20
3100 ! Bestimmung: auf- bzw. absteigender Knoten
        if (nu=1.or.nu=2) then
          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, &
          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
        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)
        endif
        sep = sep0i*wfact
        if (re(3*ip-1)<0.d0) sep = -sep

```

```

3130 xjdt = xj2; zjde = xj2
      if (iuniv=2) call delta_T(xjdt)
      call jdate(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.ilinc=2) then
        sd(1) = dsin(R0/(AE*re(9))) * wfact
        sd(2) = dsin(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)
        endif
        inum(4) = inum(4) + 1
3145 ! Mit der zeitlichen Verschiebung "shift" (in julian. Tagen)
        wird der spaeter folgende Startpunkt fuer "ringfit" bzw.
        "sekante" moeglichst nahe an die Nullstelle verlegt.
        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
      endif

3160 ! .. Vorbereitung zur naechsten Berechnung im selben Transit
        30 iis = 0; ke = 1
        itr = itr + 1
        Kontaktpunkt I
        if (itr=2) then
          idr = 1; blim = bmax1
          xj2 = zjde - shift
        endif
        Kontaktpunkt II
        if (itr=3) then
          idr = 2; blim = bmin1
          xj2 = zjde - shift
        endif
        Kontaktpunkt III
        if (itr=4) then
          idr = 4; blim = bmin1
          xj2 = zjde + shift
        endif
        Kontaktpunkt IV
        if (itr=5) then
          idr = 5; blim = bmax1
          xj2 = zjde + shift
        endif
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)/tml1

3190 !
!   VSOP87D Kurzversion (imod=1), VSOP87C Vollversion (imod=2)
  if (imod==1) then
    call vsop1tr(ip,rk,tau,del,r3i,ept,inum, resu)
  else
    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.
y12 = sep01-bljm
3200 call ringfit(xj1,xj2,xj3,yy1,yy2,yy3,eps,1.d-3,iis,ix,ke)
  if (ke==1 .or. ke==5) go to 40
  if (ke==2) go to 60
  xjdt = xj2 + del
  if (iuniv==2) call delta_T(xjdt)
  call jdedate(xjdt,ical,ida,da,dmo)

3205 ! . . 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)) &
  call tserie(ip,zjde,is,iop0,ires)
end subroutine

subroutine sepa(ip,iv,rk,sep0i)
!-----Berechnung der Separation Sonne-Merkur bzw. Sonne-Venus-----
!
! 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 - raumliche 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)/ &
(re(9)-re(3*ip)*cos0i))
3235 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.
!
! 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
! 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,gdpl,zjd0,tcen
  implicit double precision (a-h,o-z)
  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,
!
! 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 = (23.445804260 - 0.856033d0 * &
dsin(0.015306d0 * (T + 0.50747d0))) * pidg
  call rotmat(1,-eeps,0.d0,0.d0,rgeo)

3275 ! 2. Translation des heliozentrischen Koordinatenursprungs von der
! 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
enddo
  call transtat(xx(1),xx(2),xx(3),rgeo)

3280 ! 3. Umrechnung in sphaerische Koordinaten
! (Positionen von Sonne, Merkur und Venus)
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

3285 ! 4. Berechnung des Positionswinkels nach Andre Danjon: "Astronomie
! Generale", S.36, Gl."3 bis". Siehe auch Jean Meus: "Transits",
! S.15 ("kartesische" Koordinaten x und y in Bogensekunden).
  sdec = rgeo(2) * pidg
  dra = (rgeo(3*ip+1)-rgeo(1)) * pidg
  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)
  ang = datan(-x/y)*gdpl
3300 if (y*dcos(ang*pidg)<0.d0) ang = ang + 180.d0
  call reduz(ang,0,1)
end subroutine

```

```

3305 subroutine tserie(ip,zjde,is,iop0,ires)
!-----Bestimmung der Transit-Serie-----
! Die Seriennummern entsprechen denen der "NASA Eclipse Web Site".
! (Die Liste der Seriennummern "iserie.t" wird nur einmal verwendet, um die Startnummern, d.h. die Nummern zu bestimmen, die den ersten gefundenen Transiten zugeordnet werden. Danach werden alle 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,iflag,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
3320
! . . . 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
3330
! . . 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))
ic write(6,('a,x,b,ise(j),j,is,ismax = ',f9.3,f10.3,f16.6, &
ic & 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
ase(ires) = zjde
  ise(ires) = 1
end subroutine
3360

```

```

3365 subroutine VSOP87Y(tdj,ivers,ibody,prec,lu,r,ierr,md)
!-----
! >> UPGRADE (by H. Jelitto): As proposed by Bretagnon and Franco
! >> 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.
3380
! >> 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.
3395
3400
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

```



```

3425 ! heliocentric positions and velocities
! dynamical equinox and ecliptic J2000.
3: V50P87C.
! rectangular coordinates
! heliocentric positions and velocities
! dynamical equinox and ecliptic of the date.
3430 4: V50P87D.
! spherical coordinates
! heliocentric positions and velocities
! dynamical equinox and ecliptic of the date.
3435 5: V50P87E.
! rectangular coordinates
! barycentric positions and velocities
! dynamical equinox and ecliptic J2000.
!
! ibody
!
3440 0: Sun (not used here in V50P87Y)
! 1: Mercury
! 2: Venus
! 3: Earth
! 4: Mars
! 5: Jupiter
! 6: Saturn
! 7: Uranus
! 8: Neptune
! 9: Earth-Moon barycenter
!
! prec
! relative precision (real double precision).
3450 if prec is = 0 then the precision is the precision
! p0 of the complete solution V50P87.
! Mercury p0 = 0.6 10**-8
! 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
! 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 :
! 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.
! - 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
! 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
!
! lu
! logical unit index of the file (integer).
3480

```

```

!
! The file corresponds to a version of V50P87 theory and
! a body, and it must be defined and opened before the
! first call to subroutine V50P87.
!
3485 Output :
!
! r(6) array of the results (real double precision).
!
! for elliptic coordinates :
! 1: semi-major axis (au)
! 2: mean longitude (rd)
! 3: k = e*cos(pi) (rd)
! 4: h = e*sin(pi) (rd)
! 5: q = sin(i/2)*cos(omega) (rd)
! 6: p = sin(i/2)*sin(omega) (rd)
! e: eccentricity
! pi: perihelion longitude
! i: inclination
! omega: ascending node longitude
!
! for rectangular coordinates :
! 1: position x (au)
! 2: position y (au)
! 3: position z (au)
! 4: velocity x (au/day)
! 5: velocity y (au/day)
! 6: velocity z (au/day)
!
! for spherical coordinates :
! 1: longitude (rd)
! 2: latitude (rd)
! 3: radius (au)
! 4: longitude velocity (rd/day)
! 5: latitude velocity (rd/day)
! 6: radius velocity (au/day)
!
! ierr error index (integer).
! 0: no error.
! 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.
!
-----
!
! 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', &
! '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/6.2831853071795864769d0/
! data t/0.d0,1.d0,5*0.d0/
! data t2000/2451545.d0/
3540

```

```

3545 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
3550 t(1)=(tdj-t2000)/a1000
      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
3560   read (lu,1001,end=20) iv,bo,ic,it,inn
      iv2(ibody) = iv
      it2(it,ic,ibody) = 1
      in2(it,ic,ibody) = inn
      if (ideb==0) then
3565   ideb=1; ierr=1
      if (iv/=ivers) return
      ierr=2
      if (bo/=body(ibody)) return
      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
3575   go to 10
      md(ibody) = 2
      endif

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

```

```

3600   !$omp section
      itn = 0
      inn = in2(itn,ic,ibody); if (inn==0) go to 51
3605   do n=2,inn,4
      a = par2(1,n,itn,ic,ibody)
      b = par2(2,n,itn,ic,ibody)
      c = par2(3,n,itn,ic,ibody)
      r1 = r1 + a*dacos(b + c*t(1))*t(itn)
      enddo
3610   if (itn<=4) itn = it2(itn+1,ic,ibody)
      if (itn<=4 .and. itn/=1) then; itn = itn+1; go to 31
      endif
      !$omp section
      itn = 0
3615   inn = in2(itn,ic,ibody); if (inn==0) go to 52
      do n=3,inn,4
      a = par2(1,n,itn,ic,ibody)
      b = par2(2,n,itn,ic,ibody)
      c = par2(3,n,itn,ic,ibody)
      r2 = r2 + a*dacos(b + c*t(1))*t(itn)
      enddo
3620   if (itn<=4) itn = it2(itn+1,ic,ibody)
      if (itn<=4 .and. itn/=1) then; itn = itn+1; go to 32
      endif
      !$omp section
      itn = 0
3625   inn = in2(itn,ic,ibody); if (inn==0) go to 53
      do n=4,inn,4
      a = par2(1,n,itn,ic,ibody)
      b = par2(2,n,itn,ic,ibody)
      c = par2(3,n,itn,ic,ibody)
      r3 = r3 + a*dacos(b + c*t(1))*t(itn)
      enddo
3630   if (itn<=4) itn = it2(itn+1,ic,ibody)
      if (itn<=4 .and. itn/=1) then; itn = itn+1; go to 33
      endif
3635   !$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
      return
3650   ! -----
      ! Formats
      ! -----
3655   1001 format (17x,i1,4x,a7,12x,i1,17x,i1,i7)
      1002 format (79x,f18.11,f14.11,f20.11)
      end subroutine

```

```

3660 !-----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

!-----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)
! gemeins Vorzeichen der z-Komponente des Vektorproduktes a x c.
! if (i-15.or.i-20) then
!   if (iek/=3) iek = 1
!   if (iekk/=3) iekk = 1
!   ez = ax*cy-ay*cx

```

```

3720 ! 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 (iekk/=3) iekk = 2
! endif
! endif

! .. Berechnung der rel. Abweichung [%] --> xyr(36)
! Sonnenposition auf Nordsuuedachse
! 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- Chefred-Pyramide u. Mykerinos-
! Cheops-Pyramide auch ein anderes Streckenpaar verwenden, wie z.B.
! Mykerinos- Chefred-Pyramide und Chefred- 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(21)/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

!-----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

```

```

3780 if (iek==2) em = -1.d0
      if (ison<=3) then
3781   sax = (xvr(4)+xvr(1)) * .5d0
3782   say = (xvr(5)+xvr(2)) * .5d0
3783   sbx = (xvr(7)+xvr(1)) * .5d0
3784   sby = (xvr(8)+xvr(2)) * .5d0
3785   scx = (xvr(7)+xvr(4)) * .5d0
3786   scy = (xvr(8)+xvr(5)) * .5d0
3787   al1 = - em * pyr(31) - datan(ay/ax) + datan(say/sax)
3788   al2 = - em * pyr(32) - datan(by/bx) + datan(sby/sbx)
3789   al3 = - em * pyr(33) - datan(cy/cx) + datan(scy/scx)
3790   r1 = dsqrt(sax*sax + say*say)
3791   r2 = dsqrt(sbx*sbx + sby*sby)
3792   r3 = dsqrt(scx*scx + scy*scy)
3793   !
3794   !
3795   !
3796   !
3797   !
3798   !
3799   !
3800   !
3801   !
3802   !
3803   !
3804   !
3805   !
3806   !
3807   !
3808   !
3809   !
3810   !
3811   !
3812   !
3813   !
3814   !
3815   !
3816   !
3817   !
3818   !
3819   !
3820   !
3821   !
3822   !
3823   !
3824   !
3825   !
3826   !
3827   !
3828   !
3829   !
3830   !
3831   !
3832   !
3833   !
3834   !
3835   !
3836   !
3837   !
3838   !
3839   !
3840   !
3841   !
3842   !
3843   !
3844   !
3845   !
3846   !
3847   !
3848   !
3849   !
3850   !
3851   !
3852   !
3853   !
3854   !
3855   !
3856   !
3857   !
3858   !
3859   !
3860   !
3861   !
3862   !
3863   !
3864   !
3865   !
3866   !
3867   !
3868   !
3869   !
3870   !
3871   !
3872   !
3873   !
3874   !
3875   !
3876   !
3877   !
3878   !
3879   !
3880   !
3881   !
3882   !
3883   !
3884   !
3885   !
3886   !
3887   !
3888   !
3889   !
3890   !
3891   !
3892   !
3893   !
3894   !
3895   !
3896   !
3897   !
3898   !
3899   !
3900   !
3901   !
3902   !
3903   !
3904   !
3905   !
3906   !
3907   !
3908   !
3909   !
3910   !
3911   !
3912   !
3913   !
3914   !
3915   !
3916   !
3917   !
3918   !
3919   !
3920   !
3921   !
3922   !
3923   !
3924   !
3925   !
3926   !
3927   !
3928   !
3929   !
3930   !
3931   !
3932   !
3933   !
3934   !
3935   !
3936   !
3937   !
3938   !
3939   !
3940   !
3941   !
3942   !
3943   !
3944   !
3945   !
3946   !
3947   !
3948   !
3949   !
3950   !
3951   !
3952   !
3953   !
3954   !
3955   !
3956   !
3957   !
3958   !
3959   !
3960   !
3961   !
3962   !
3963   !
3964   !
3965   !
3966   !
3967   !
3968   !
3969   !
3970   !
3971   !
3972   !
3973   !
3974   !
3975   !
3976   !
3977   !
3978   !
3979   !
3980   !
3981   !
3982   !
3983   !
3984   !
3985   !
3986   !
3987   !
3988   !
3989   !
3990   !
3991   !
3992   !
3993   !
3994   !
3995   !
3996   !
3997   !
3998   !
3999   !
4000   !

```

```

3840   !
3841   !
3842   !
3843   !
3844   !
3845   !
3846   !
3847   !
3848   !
3849   !
3850   !
3851   !
3852   !
3853   !
3854   !
3855   !
3856   !
3857   !
3858   !
3859   !
3860   !
3861   !
3862   !
3863   !
3864   !
3865   !
3866   !
3867   !
3868   !
3869   !
3870   !
3871   !
3872   !
3873   !
3874   !
3875   !
3876   !
3877   !
3878   !
3879   !
3880   !
3881   !
3882   !
3883   !
3884   !
3885   !
3886   !
3887   !
3888   !
3889   !
3890   !
3891   !
3892   !
3893   !
3894   !
3895   !
3896   !
3897   !
3898   !
3899   !
3900   !
3901   !
3902   !
3903   !
3904   !
3905   !
3906   !
3907   !
3908   !
3909   !
3910   !
3911   !
3912   !
3913   !
3914   !
3915   !
3916   !
3917   !
3918   !
3919   !
3920   !
3921   !
3922   !
3923   !
3924   !
3925   !
3926   !
3927   !
3928   !
3929   !
3930   !
3931   !
3932   !
3933   !
3934   !
3935   !
3936   !
3937   !
3938   !
3939   !
3940   !
3941   !
3942   !
3943   !
3944   !
3945   !
3946   !
3947   !
3948   !
3949   !
3950   !
3951   !
3952   !
3953   !
3954   !
3955   !
3956   !
3957   !
3958   !
3959   !
3960   !
3961   !
3962   !
3963   !
3964   !
3965   !
3966   !
3967   !
3968   !
3969   !
3970   !
3971   !
3972   !
3973   !
3974   !
3975   !
3976   !
3977   !
3978   !
3979   !
3980   !
3981   !
3982   !
3983   !
3984   !
3985   !
3986   !
3987   !
3988   !
3989   !
3990   !
3991   !
3992   !
3993   !
3994   !
3995   !
3996   !
3997   !
3998   !
3999   !
4000   !

```

```

3895   if (w(5)==z0) go to 10
      j2=4+j2
      do i=1,n
      j1=j2+i; j2=j1+i-1
      write(iu,154) (w(j),j=j1,j2)
      enddo
      !c 10
      write(iu,*)
      write(iu,(' ',start x(1,1),il,','),7f13.3)) &
n,(xsta(i),i=1,3),(xsta(i)*gdpl,i=4,6),xsta(7)
      write(iu,(' ',y(1,1),il,','),9f13.3)) &
m,(ysta(i),i=1,m)
      write(iu,(' ',results x(1,1),il,','),7f13.3)) &
n,(x(i),i=1,3),(x(i)*gdpl,i=4,6),x(7)
      write(iu,(' ',y(1,1),il,','),9f13.3/)) &
m,(y(i),i=1,m)
      enddo
      endif
! . . . Berechnung der Sonnenposition im Pyramidengelaende mit Hilfe
! der gerade bestimmten Parameter x(1)..x(7) durch Transforma-
! tion des Koordinatenursprungs (Sonne)
3915   do i=1,m; y(i) = z0; enddo
      call transl(x(1),x(2),x(3),y)
      call rotmat(5,x(4),x(5),x(6),y)
      call mastab(x(7),y)
      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. Positionskordinaten der Felsenkammer)
3930   xyr(31) = xyr(31) + xp3
      xyr(32) = xyr(32) + yp3
      xyr(33) = xyr(33) + zp3
! . . . Fehlerabschaetzung fuer die Sonnenposition
3935   !c
      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
      xyr(34) = dsqrt(w(4))
      endif
      endif
3945   return
152 format(5x,2i5,1p,9e13.5)
153 format(3i5,1p,8e23.15)
154 format(' ',1p,6e13.5)
end subroutine
!-----Invertierung der 3x3-Matrix a, d.h. a -> inv(a)-----
3950   implicit double precision (a-h,o-z)

```

```

3955   dimension :: a(3,3),b(3,3)
! . . 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)
      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)
3960   ! . . Kehrwert der Determinante und Transponieren
      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)
!
! z.B. Dz(w1) = ( cos w1 sin w1 0 )
!              ( -sin w1 cos w1 0 )
!              ( 0 0 1 )
!
! iachse = 4: Drehung um Knotenlinie (Winkel w1, w2)
! 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)
      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 ! axis 1
      D(1,1) = one
      D(2,2) = c1
      D(2,3) = s1
      D(3,2) = - s1
      D(3,3) = c1
      else
      D(1,1) = c1
      if (iachse==2) then ! axis 2
      D(1,3) = s1
      D(2,2) = one
      D(3,1) = - s1
      D(3,3) = c1
      else
      D(1,2) = s1
      D(2,1) = - s1
      D(2,2) = c1
      D(3,3) = one
      endif
      endif
      enddo
      s2 = dsin(w2); c2 = dcos(w2)
      if (iachse==4) then ! axis 4
      D(1,1) = - s1 * s1 * (one - c2) + one

```

```

4015 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
4020 s3 = dsin(w3); c3 = dcos(w3)
      D(1,1) = c1 * c3 - s1 * c2 * s3
      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
4025 D(3,1) = s1 * s2
      D(3,2) = - c1 * s2
      D(3,3) = c2
      endif
4030 !. . . Ausfuehrung der Transformation (Merkur-, Venus- und Erdposition)
      !c do i=1,3; write(6,'(3f13.8)')(D(i,j),j=1,3); enddo
4035 do i=1,9; b(i) = z0; enddo
      do k=0,6,3
4040 do i=1,3
          do j=1,3
              b(k+i) = b(k+i) + D(i,j)*a(j+k)
          enddo
          enddo
          enddo
4045 do i=1,9; a(i) = b(i); enddo
      write(6,'(a12,3f13.8)') Mercury : ',(a(j),j=1,3)
      write(6,'(a12,3f13.8)') Venus : ',(a(j),j=4,6)
      write(6,'(a12,3f13.8)') Earth : ',(a(j),j=7,9)
      end subroutine
4050 !-----
      !-----Translation transl(a1,a2,a3,a)
      ! 3 Vektoren a(1..9) --> a(1..9)
      implicit double precision (a-h,o-z)
      dimension :: a(9)
4055 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
4060 !-----
      !-----Massstabsaenderung:-----
      ! 3 Vektoren a(1..9) --> a(1..9)
      implicit double precision (a-h,o-z)
      dimension :: a(9)
4065 do i=1,9
          a(i) = zmas * a(i)
          enddo
      end subroutine
4070 !-----
      !-----Subroutine transfo(irb,rku)
      ! 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.
4075 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
4080 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
4085 write(6,'(a10,f23.8)') re(4) 're(4) ',re(4)
      write(6,'(a10,f23.8)') re(40) 're(40) ',re(40)
      if (irb>=2 .and. irb<=4) then
          ai = dabs(datan(xyr(3)/(xyr(1)*dsin(ao))))
      else
4090 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 = dsin(dsin(ao)/dsqrt(1.d0-(dsin(ai)*dcos(ao)**2))+ao-pi
4095 a1 = ao; a2 = ai
          a3 = at
      write(6,'(a12,3f13.8)') Mercury : ',(xyr(j),j=1,3)
      write(6,'(a12,3f13.8)') Venus : ',(xyr(j+3),j=1,3)
      write(6,'(a12,3f13.8)') Earth : ',(xyr(j+6),j=1,3)
4100 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
4105 !. . . Matrix K(ao,ai)
          call rotmat(4,a1,a2,z0,xyt)
          !. . . Matrix Dz(at)
          at = datan(xyt(2)/xyt(1))
          a3 = at
          endif
          call rotmat(3,a3,z0,z0,xyt)
          endif
4110 !-----Transformation B --> Dz(at-ao) * Dx(ai) * Dz(ao)
      if (irb==3) then
4115 !. . . 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
4125 !-----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
4130

```

```

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

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 : gdp1
implicit double precision (a-h,o-z)
dimension :: rku(3)
ra = dsqrt(r1*r1 + r2*r2)
rku(1) = atan(r2/r1)*gdp1
rku(2) = atan(r3/ra)*gdp1
rku(3) = dsqrt(ra*ra + r3*r3)
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, &
  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)
! durchgeführt werden. Die Ortsabweichungen im Pyramidengelaende
! 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
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
! 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
! 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)
  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, &
  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
  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
!-----Umrechnung in kartesische Koordinaten
! call kartko(ison)
! Koordiantentransformation: Weltraum --> Pyramidengelaende
do i=4,6; y(i) = xyr(i); enddo
call transl(x(1),x(2),x(3),y)
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 + &
  + (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
do iu=ix,6,5
  write(iu,(' Mercuru aphelion coordinates [m]:', &
    & f13.2,f10.2,f9.2)) y(4),y(5),y(6),dr
  call linie(iu,1)
enddo
endif
end subroutine

subroutine plako(diff,ipla,ijd,ik,ison,ipos, &
  rcm,x,y,ort,fp,dd,dn,dss,pla,plan,emp,text,tt,titab, &
  isl2,dmi,zjda,zjde,ivers,md,ix,prec,lu,r,ierr,rku)
!-----Koordinaten fuer Merkur bis Neptun-----
! und Berechnung der "Planetenspositionen" im Giza-Gelaende fuer
! Konst. 1-14 mit ison = 5 (FITEK) und imod = 2 (VSOP87-Vollv.).
! Zusatzlich:
! 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)
use base
implicit double precision (a-h,o-z)
dimension :: diff(9),r(6),rku(3),md(9),x(7),y(9),rcm(3)
dimension :: ort(9,4),rp(3,4),zjda(4)
character(2) :: dd,dn,dss
character(3) :: pla(9),line
character(7) :: emp
character(10) :: plan(9)
character(18) :: date(4)

```

```

4250 character(23) :: text(0:9),tt(2)
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 ',&
' B Lm-L dev.' r[AU]
call linie(iu,2)
else
write(iu,'(/27x, ''Celestial positions in Giza'')')
call linie(iu,1)
write(iu,*)' body x[m] y[m] z[m] ', &
' dr[m] latitude N longitude E'
endif
enddo

4260 !.....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
icmax = 1; if (is12/=0) icmax = 4
10 if (is12/=0) then
zjde = zjda(icm)
do iu=ix,6,5
call linie(iu,2)
write(iu, '(4x,a18, 'JDE = ',f14.5)') date(icm),zjda(icm)
call linie(iu,2)
enddo
endif
if (is12/=0 .and. icm==1) then
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
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
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
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
enddo
endif
4305 endif

```

```

!....."Planetenpositionen" im Giza-Gelaende (kartesische Koord.)
if ((ij>=1 .and. ij<=14).or.(ik==4519 .and. ipla==1) &
.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
do j=1,3; re(j+3) = rku(j); enddo
call kartko(ison)
do j=4,6; y(j) = xyr(j); enddo
call transl(x(1),x(2),x(3),y)
call rotmat(5,x(4),x(5),x(6),y)
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 &
+ (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 &
+ (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)
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
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
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
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)
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 ', &
& f10.2,2f10.2,f9.2)') (ort(0,j),j=1,4)
enddo
4365

```



```

4370 do i=1,imax
      dd = dn
      if ((i>=1 .and. i<=3) .or. i==9) dd = dss
      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)
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)
end subroutine

4385
      subroutine geoko(x,y,ipla,iB1,zB2,iL1,zL2)
!-----Berechnung der geographischen Koordinaten-----
! (iB1,zB2 und iL1,zL2, jeweils in Grad und Minuten)
! use base, only : pi, pidg, R3a, R3p
! implicit double precision (a-h,o-z)

4390
! . . . Erdumfang ueber Pole. Anstelle von Ue = 40008 km folgt
! Ellipsenumfang nach Srinivasa Ramanujan.
4395 zL = 3.40*((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
4405 zL0 = 31.134276d0 ! der Koeniginnenkammer
      endif

! . . . 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
4415 iB1 = idint(zB)
      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

```

!-----Abstand eines Punktes der geographischen Breite B-
! zur Erdachse (u) und zur Aequatorebene (v) (kartesische Koord.)-----
4430 use base, only : pidg, R3a, R3p
! implicit double precision (a-h,o-z)
      u = R3a/dsqrt(1.d0 + (dtn(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
      if (i/=0) a = a*gdpi
      ab = dabs(a)
      if (ab>u360) ab = dmod(ab,u360)
      if ((j==0 .and. ab>180.d0) .or. &
4445 (j==1 .and. a<0.d0)) ab = ab - u360
      a = z1 * ab
      if (i/=0) a = a * pidg
end subroutine

4450
      subroutine memo(zz1,zz2,zz3,zz4,zz5,zz6,zz7,zmem,ik,imem)
!-----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
      zmem(7) = zz7
4465 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,*) itext(i); 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 titell(iaph,ijd,ia,ison,ipla, &
      iLin,isep,nurtr,iuniv,isl2,iop0)
!-----Haupttitel und Untertitel-----
! implicit double precision (a-h,o-z)

```

```

4485 write(ia,*)
4486 if (iop0==350) then
4487   write(ia,'(20x,A20,A22)') 4 PLANETS IN A LINE ', &
4488   '(SYZYG), 17. MAY 3088'
4489 go to 20
4490 elseif (iop0==351) then
4491   write(ia,'(17x,A16,A31)') 'MERCURY TRANSIT ', &
4492   '(MIN. SEPARATION), 18. MAY 3088'
4493 go to 20
4494 elseif (iop0==360) then
4495   write(ia,'(18x,A14,A32)') 'VENUS TRANSIT ', &
4496   '(MIN. SEPARATION), 18. DEC. 3089'
4497 go to 20
4498 elseif (iop0==361) then
4499   write(ia,'(19x,A20,A23)') 3 PLANETS IN A LINE ', &
4500   '(SYZYG), 23. DEC. 3089'
4501 go to 20
4502 elseif (iop0==370) then
4503   write(ia,'(24x,A34)') 'SEARCH FOR "SHADOW-CONSTELLATIONS"'
4504 go to 10
4505 elseif (iop0==371 .or. iop0==372) then
4506   write(ia,'(16x,A20,A29)') 'PRECEDING "SHADOW-CO', &
4507   'NSTELLATION" 12, 22. MAY 3088'
4508 go to 20
4509 endif
4510 if (ipla==1) write(ia,*)
4511   'ALIGNMENT WITH THE PYRAMIDS OF GIZA' PLANETS IN ', &
4512   if (ipla==2) write(ia,*)
4513     'PLANETS IN ALIGNME', &
4514     'NT WITH THE CHAMBERS OF THE CHEOPS PYRAMID'
4515   if (ipla==3) then
4516     if (ilim>3) write(ia,'(28x,a11,a15)') 'PLANETS IN ', &
4517     'A LINE (SYZYG)'
4518     if (ilim==1) write(ia,'(31x,a19)') 'TRANSITS OF MERCURY'
4519     if (ilim==2) write(ia,'(32x,a17)') 'TRANSITS OF VENUS'
4520   endif
4521   10 if (ipla/=3 .and. is12/=0) then
4522     if (iaph==1 .and. ijd/=13 .and. ijd/=14) &
4523       write(ia,'(30x,a21)') '(Mercury at aphelion)'
4524     if (iaph==2 .and. ijd/=13 .and. ijd/=14) &
4525       write(ia,'(29x,a23)') '(Mercury at perihelion)'
4526     if (iaph==3 .or. (iaph==1 .and. (ijd==13 .or. ijd==14))) &
4527       write(ia,'(29x,a23)') '(Mercury near aphelion)'
4528     if (iaph==4 .or. (iaph==2 .and. (ijd==13 .or. ijd==14))) &
4529       write(ia,'(28x,a25)') '(Mercury near perihelion)'
4530     if (iaph==5) write(ia,'(24x,a34)') &
4531     '(time not restricted, F minimized)'
4532   elseif (ipla/=3 .and. is12/=0) then
4533     if (ipla==1) write(ia,'(17x,a48)') &
4534     '(more positions - coordinate system of pyramids)'
4535     if (ipla==2) write(ia,'(17x,a48)') &
4536     '(more positions - coordinate system of chambers)'
4537   else
4538     if (isep==1) then
4539       if (ison/=5) then
4540         write(ia,'(14x,a21,a33)') '(eclipt. longitudes, ', &
4541         'all within an angular range, JDE)'
4542       else
4543         if (ilim>=3) then
4544           if (nurtr==1) then

```

```

4545   write(ia,'(13x,a18,a37)') '(angular range of ', &
4546   'eclipt. longitudes dL minimized, JDE)'
4547   else
4548     write(ia,'(5x,a18,a52)') '(angular range of ', &
4549     'eclipt. longitudes dL minimized, only transits, JDE)'
4550   endif
4551   else
4552     write(ia,'(11x,a18,a41)') '(equal eclipt. lon', &
4553     'gitudes for Earth und transit planet, TT)'
4554   endif
4555   elseif (isep==2) then
4556     write(ia,'(14x,a54)') &
4557     '(minimum separation, without travel time of light, TT)'
4558   else
4559     if (iuniv==1) then
4560       write(ia,'(17x,a48)') &
4561       '(geocentric transit phases, terrestrial time TT)'
4562     else
4563       write(ia,'(18x,a46)') &
4564       '(geocentric transit phases, universal time UT)'
4565     endif
4566   endif
4567   20 if (isep/=4) then
4568     write(ia,'(34x,a8,i4,a2)') '< option', iop0, ' >'
4569   else
4570     write(ia,'(11x,a8,i4,a47)') '< option', iop0, &
4571     ' > (monitor line width minimal 148 characters)'
4572   endif
4573   end subroutine
4574
4575   subroutine titel2(ia,imod,ivers,irb,ipla, &
4576   ison,ih,i,iek,ijd,ika,iaph,ilin,ical,ak,zjdel,zjahr,delt, &
4577   dwt,dwikomb,dwi0,dwt2,dwi3,lamax,step,ikomb,zmin,zmax)
4578   !-----Zwei weitere Titelzeilen-----
4579   implicit double precision (a-h,o-z)
4580   dimension :: ida(7),da(7)
4581   character(5) :: ca(2),dmo
4582   character(7) :: cal(2)
4583   character(10) :: wd
4584   character(15) :: text0
4585   character(27) :: text1
4586   character(19) :: text2
4587   character(8) :: text3(0:6)
4588   character(25) :: text4
4589   character(22) :: text5(2)
4590   data ca/'(c1)', '(c2)'/,cal/'Gregor','Julian',/
4591   data text3/'', 'E-V-M', 'E-M-V', 'E-M-V', ' &
4592   ' V-E-M', 'V-M-E', 'M-E-V', 'M-V-E', /
4593   data text5/'', only Greg. calendar', Jul./Greg. calendar'/
4594   if (imod==1) text1 = ' V50P87D short ver.(Meeus)'
4595   if (imod==2 .and. ivers==1) text1 = ' V50P87A (2005) full ver.,'
4596   if (imod==2 .and. ivers==3) text1 = ' V50P87C (2005) full ver.,'
4597   if (imod==3) text1 = ' "Keplers equation"'
4598   if (ikomb==1 .and. ivers==1) text1 = ' V50P87A, comb. search,'
4599   if (ikomb==1 .and. ivers==3) text1 = ' V50P87C, comb. search,'
4600   if (ivers==1) text2 = ' standard J2000.0,'
4601   if (ivers==3) text2 = ' ecliptic of date,'

```

```

4605 if (ipla/=3) then
      if (irb=1) then
        if (imod=3) text4 = 'Sun' south of Myker. P.'
        if (imod=2) text4 = "Sun" south of sub. cham.'
        if (ison=2) text4 = "Sun" south of Chefred. P.'
        if (ison=3) text4 = "Sun position" free, 2D'
        if (ipla=1) then
          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'
          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'
          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
        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'
      else
        if (ilin=1) text4 = ' all Mercury transits'
        if (ilin=2) text4 = ' all Mercury transits'
        if (ilin=3) text4 = ' linear c., Merc. to Earth'
        if (ilin=4) text4 = ' linear c. Mercury to Mars'
      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/'
      if (ison>3 .or. iek=3) text0 = ' Ecl. N and S,'
      text0 = ' Period (yea'
    else;
      endif
      if (ijd=15 .and. (imod/=2 .or. (imod=2 .and. &
        (iaph=3 .or. iaph=4))) then
        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
            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
              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, &
                & '', tolerance F <='', f6.2, ''/'', f5.2, '' %'')) &
                text0, zmin, zmax, ca(ical), dwi, dwikomb
            endif
          endif
        endif
      endif
    endif
  endif

```

```

4665      endif
      endif
      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'', &
          & 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)
          return
        endif
      endif
      else
        call ephim(1, iaph, ipla, ical, ak, iak, zjdel, zjahr, delt)
        if (ijd>=1 .and. ijd<=14) then
          write(ia, '(a15, '' constellation'', i3, '', JDE ='', &
            & f15.5, '', year ='', f9.2, a5)') text0, ijd, zjdel, zjahr, ca(ical)
        else
          write(ia, '(a15, 20x, '' JDE ='', f15.5, '', year ='', f9.2, a5)') &
            text0, zjdel, zjahr, ca(ical)
        endif
        if (iaph<=2) then
          call jdedate(zjdel, ical, ida, da, dmo)
          call weekday(zjdel, wd)
          k = 1
          if (zjdel>=0.d0 .and. zjdel<2299161.d0 .and. ical=2) k = 2
          if (zjdel>=1356183.d0 .and. zjdel<=5373484.d0) then
            write(ia, '(25x, ''date ('', a7, '', TT) ='', &
              & f4.0, a5, i5, '', i3, 2('', i2, '', A10)') &
              cal(k), da(7), dmo, (ida(1), i=3, 6), wd)
            return
          else
            write(ia, '(24x, ''date ('', a7, '', TT) ='', &
              & f4.0, a5, i6, '', i3, 2('', i2, '', A10)') &
              cal(k), da(7), dmo, (ida(1), i=3, 6), wd)
            return
          endif
        endif
      endif
      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
        write(ia, ('' Consider without printing by tolerance ='', &
          & f8.4)') dwi2
        write(ia, ('' Print beyond aphelion (per.) by toler. ='', &
          & f8.4)') dwi3
      endif
      end subroutine
    endif
  endif
  tabe(iaph, imod, iek, ia, io, &
    ison, ipla, ilin, itran, is12, iop0, iout)
  !-----Tabellenkopf-----
  ! Bei Datumsberechnungen uebernimmt das Unterprogramm

```



```

4960 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
      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
      else
        if (ipla/=3) then
          write(ia,503) ' Detected constellations:', inum(2), te2, &
            ihour, te3, imin, te3, sec
        else
          if (ison==5) then
            inumber = inum(2)
          else
            write(ia,501) ' Detected constellations:', inum(2)
            inumber = inum(3)
          endif
          write(ia,503) ' Number of syzygies      ', inumber, te2, &
            ihour, te3, imin, te3, sec
          endif
        endif
      endif
    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) ) &
          'The file "inser-2.t" has been created.'
        write(ia,505) te2, ihour, te3, imin, te3, sec
        endif
      endif
    endif
  write(ia,506) te22, ihour2, te3, imin2, te3, sec2, te5
5000 format(1x,a25,i10,f6x,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)
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 histogram(zz,ihis) !h
!-----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)
!h i = idint(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"-----
! 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 gewünscht kann
! diese - durch Umbenennung in "inserie.t" - die vorherige bzw.
! fehlende Datei "inserie.t" ersetzen. Die Verwendung dieser
5010
5015

```

```

! 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, V50P87C full version'
write(10, '(34x,a9)') '(Mercury)      JDE', ('JDE',i=1,4)
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)
write(10, '(35x,a7)') '(Venus)      JDE', ('JDE',i=1,4)
write(10, '(a14,4(12x,a3)') 'S-No.
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
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)
end subroutine

subroutine vsop1tr(ip,rk,tau,del,r3i,eps,inum, resu)
!-----Berechnung der ekliptikalen Koordinaten (Kurzversion V50P87)-----
! Berücksichtigung der Laufzeit des Lichtes, die bei Berechnung
! 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)
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
enddo
call kartko(0)
do j=ist,ist+2; rk(j) = xyr(j); enddo
do
  tau1 = tau + del; inum(1) = inum(1) + 1
  do j=9; call vsop1(j,tau1, resu); re(j) = resu; enddo
  call kartko(0)
  do j=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)
  del = r3i*AE/(c*86400.d0*tmil); tau2 = tau + del
  if (dabs(tau2-tau1)<eps) exit
enddo
del = del*tmil
end subroutine

subroutine vsop2tr(xj2,ivers,ip,md, &
ix,prec,lu,r,rk,ierr,del,r3i,eps,inum,rku)
!-----Aufruf der V50P87-Subroutine (Vollversion)-----
! Berücksichtigung der Laufzeit des Lichtes
5030
5035
5040
5045
5050
5055
5060
5065
5070

```

```

5075 ! 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 "dl" vom Planet "ip" zur Erde
use base, only : re,c,AE
implicit double precision (a-h,o-z)
dimension :: rx(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

5105 subroutine fitmin(imod,imodus,iap,ke,x,y,ee1, &
  step,nu,iflag,ddx1,ddx2,test,iin,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

```

```

5135 10 iconv = 0
!c do lu=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
nulum = 1
!.....Bestimmung der ersten drei x- und y-Werte
if (iap==5 .and. imod==2) then
  nulum = 2
  if (nu==0) then
    indx = 1; go to 99
  endif
endif
if (nu<=nulum) 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)

5155 ! .. Pruefen auf numerisches Rauschen (im Minimum) und Konvergenz-
! problem. Letzteres Problem entsteht eventuell beim Umschalten
! von der VSOP87-Kurzversion zur -Vollversion.
5160 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
  if (i1==0 .and. i2==0) write(6,*) ' --> num. noise, nu = ',nu
  if (i2==1) write(6,'(a23,i3)') ' --> switch-pr.(dy), ',nu
  if (i1==1) write(6,'(a23,i3)') ' --> switch-pr.(dx), ',nu
  if (i1==1 .or. i2==1) then
    iconv = 1; go to 20
  endif
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)
      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)
      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
  endif
endif

```

```

5195     elseif ((dy1<ze.and.dy2>=ze).or.iflag==1) then
5196     select case (iflag)
5197     case(0)
5198     do i=1,2
5199     x(3+i) = x(2*i-1); y(3+i) = y(2*i-1)
5200     enddo
5201     x(3) = (x(3)+(zpa-1.d0)*x(2))/zpa
5202     indx = 3; iflag = 1
5203     case(1)
5204     x(1) = (x(1)+(zpa-1.d0)*x(2))/zpa
5205     indx = 1; iflag = 0
5206     end select
5207     else
5208     !.....Suche mit hyperbolischem Fit (imodus = 2)
5209     a1 = x(1)-x(2); a3 = x(3)-x(2)
5210     b1 = (y(2)**2-y(1)**2)*a3
5211     b2 = (y(3)**2-y(2)**2)*a1
5212     if (dabs(b1+b2)-ee2) then; ke = 0; return; endif
5213     d(1) = dabs(x(1)-b)
5214     d(2) = dabs(x(2)-b)
5215     d(3) = dabs(x(3)-b);      indx = 1
5216     if (d(2)>d(1).and.d(2)>d(3)) indx = 2
5217     if (d(3)>d(1).and.d(3)>d(2)) indx = 3
5218     x(indx) = b
5219     if (x(1)>x(2)) call pchange(2,1,2,rx,x,y,indx)
5220     if (x(2)>x(3)) call pchange(2,3,rx,x,y,indx)
5221     if (x(1)>x(2)) call pchange(2,1,2,rx,x,y,indx)
5222     endif
5223     ddX1 = dabs(x(2)-x(1))
5224     ddX2 = dabs(x(3)-x(2))
5225     ddX3 = dabs(x(3)-x(1))
5226     if (imodus==2) then
5227     do i=1,10
5228     if (dabs(ddX3-test(i))<1.d-7) ie = 1
5229     enddo
5230     endif
5231     !.....Hauptbedingung pruefen und Check auf Endlosschleife (ie=1)
5232     if (ddX1<=ee1.or.ddX2<=ee1.or.ie==1) then
5233     do iu=ix,6.5; write(iu,(' ',nu,imod,imods,indx,dx1,dx2,ie',' ,&
5234     & ' ',',',14,3i3,2f13.8,i3)) nu,imod,imods,indx,ddX1,ddX2,ie
5235     write(iu,('a12,3f18.8/')) x(1..3) = ',(x(i),i=1,3); enddo
5236     ke = 0; return
5237     endif
5238     if (imodus==2) then
5239     itin = itin + 1; if (itin>10) itin = 1
5240     test(itin) = ddX3
5241     endif
5242     99 nu = nu + 1
5243     write(6,('a11,2i2,3f18.7/')) m,n,x1-3 = ',imodus,nu,(x(i),i=1,3)
5244     if (nu<=100) return
5245     ke = 2
5246     do iu=ix,6.5
5247     write(iu,('/' ' -----> error in "fitmin", ke = ',I2/')) ke
5248     enddo
5249     end subroutine

```

```

5255     subroutine ringfit(x1,x2,x3,y1,y2,y3,ep,step,nu,itmax,ix,ke)
5256     !-----Nullstellenbestimmung-----
5257     ! Die Routine liefert fuer die Kreisfunktion, die durch (x1,y1),
5258     ! (x2,y2) und (x3,y3) verlaeuft, die naechstgelegene Nullstelle
5259     ! (neuer x2-Wert). Wie bei "sekante" ergibt wiederholtes Aufrufen
5260     ! von "ringfit" die Nullstelle einer stetig differenzierbaren Funk-
5261     ! tion. Abhaengig von den Optionen (ilin=<2, isep=3, 4 bzw. 1) ver-
5262     ! kuert sich die Rechenzeit um bis zu 3%, was wenig ist. Da die
5263     ! Grundidee und die Gleichungen jedoch auch eine gewisse Aesthetik
5264     ! besitzen, wurde diese Routine beibehalten. (Der Einsatz von
5265     ! "ringfit" ist nur sinnvoll, wenn die Berechnung der Ausgangs-
5266     ! funktion deutlich mehr Zeit erfordert als "ringfit" selbst.)
5267     implicit double precision (a-h,o-z)
5268     if (ke/=5) ke = 1; ep0 = 1.d-20
5269     if (nu<=1.or.ke==5) then
5270     call sekante(x1,x2,y1,y2,ep,step,nu,itmax,ix,ke); return
5271     endif
5272     if (nu==2) then ! Erzeugung des 3. Startpunktes
5273     x31 = x1; y31 = y1; x32 = x2; y32 = y2
5274     call sekante(x1,x2,y1,y2,ep,step,nu,itmax,ix,ke)
5275     if (x1==x31) then; x3 = x32; y3 = y32
5276     else; x3 = x31; y3 = y31
5277     endif; return
5278     endif
5279     sh = x2 ! Verschiebung (x2) zum Ursprung
5280     x1 = x1-sh; x2 = 0.d0; x3 = x3-sh
5281     do iu=ix,6.5; write(iu,('a16,i3,6f10.6/')) &
5282     'nu, x123, y123 = ',nu,x1,x2,x3,y1,y2,y3; enddo
5283     z1 = x1*x1 + y1*y1; ya = y2-y1; xa = -x1
5284     z2 = y2*y2; yb = y3-y2; xb = x3
5285     z3 = x3*x3 + y3*y3; yc = y1-y3; xc = x1-x3
5286     xy = 0.5d0/(x1*yb + x3*ya)
5287     if (dabs(xy)<ep0) then
5288     x1 = x1+sh; x2 = sh; ke = 5; return
5289     endif
5290     x0 = (z1*yb + z2*yc + z3*ya)*xy
5291     y0 = -(z1*xb + z2*xc + z3*xa)*xy
5292     wu = x0*x0 + (y2-y0)**2 - y0*y0
5293     if (wu<0.d0) then; ke = 4; go to 10; endif
5294     wu = dsqrt(wu)
5295     xx = x0 + wu; xx2 = x0 - wu ! (2 Loesungen)
5296     if (dabs(xx)>dabs(xx2)) xx = xx2
5297     d1 = dabs(x1-xx); d2 = dabs(xx); d3 = dabs(x3-xx)
5298     if (d3>d1.and.d3>d2) then; x3 = 0.d0; y3 = y2
5299     elseif (d1>d2.and.d1>d3) then; x1 = 0.d0; y1 = y2
5300     endif
5301     x1 = x1+sh; x2 = xx+sh; x3 = x3+sh; nu = nu+1
5302     if (dabs(x2-x1)<ep.or.dabs(x3-x2)<ep) then
5303     do iu=ix,6.5; write(iu,('a8,7x,a1,i3,3f14.10/')) &
5304     'nu, x123', ', ',nu,x1-sh,x2-sh,x3-sh; enddo
5305     ke = 0; return
5306     endif
5307     if (nu==itmax) return
5308     ke = 2
5309     10 do iu=ix,6.5
5310     write(iu,('/' ' -----> error in "ringfit", ke = ',I2/')) ke
5311     enddo
5312     end subroutine

```



```

5315 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) verläuft. 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)
!c
5320 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
x2 = x1 + step
return
endif
if (y1==y2) then
ke = 3; go to 10
endif
x0 = x2-y2*(x2-x1)/(y2-y1)
if (dabs(y2)<dabs(y1)) then
x1 = x2
y1 = y2
endif
x2 = x0
if (dabs(x2-x1)<ep.and.nu>2) then
do iu=ix,6,5; write(iu,'(a16,i3,2f16.6)') &
'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. <<
!-----
5355 FITEX
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(TRANSPPOSED)*A)
FIT1(ONE DIMENSIONAL MINIMUM SEARCH)
CALLING SEQUENCE
KE=0
M=NUMBER OF FUNCTIONS, M GE N
N=NUMBER OF VARIABLES, N GE 1

```

```

5370 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(I4,(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: STEP SIZE LIMITATION
=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.
5415 W(4+N+I)=ERROR ENHANCEMENTS
W(4+N+I+(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(I4,(N*(N+5))/2)+(M+N+1)*(K+1)+M*N
ADDRESSES IN IW
4+L: LABELS OF THE QUADRATIC SUMS
ADDRESSES IN W
4+I: STANDARD ERROR OF X(I)
4+N+I: ERROR ENHANCEMENT FOR X(I)
FROM 4+N+I: MATRIX D AND ERROR CORRELATIONS
FROM JS+1 MATRIX S; JS = 4+MAX0(I4,(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
! THE SEARCH. THIS ALLOWS A SEARCH WITHIN ANOTHER SEARCH JUST CHANGING
! THE WORKING FIELDS
!
!-----

```

```

5435 SUBROUTINE FITEX(KE,M,N,F,X,E,W,IW)
      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.)
      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
! >> 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.
      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
      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)
      DO L = 1,K
         IW(L+4) = 1 + K - L
         W(JS+LM*L) = 7.D75
      ENDDO
      KE = 1
      KV = K
      JA = JS + LM* (K+1)
      JM = JS + LM*IW(5) - LM
      J3 = JA - LM
      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
      DO I = 1,K
         J1 = JS + LM*IW(I+4)
         IF (W(4)<W(J1)) GO TO 4
      ENDDO
      GO TO 37
      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
      IF (I2==0) GO TO 6
      DO J = 1,I2
         I1 = I1 - 1

```

```

      IW(I1+1) = IW(I1)
      ENDDO
      IW(I1) = L
      JM = JS + LM*IW(5) - LM
! NEW ROW
      6 J1 = JS + LM* (L-1)
      DO I = 1,N
         JI = J1 + 1
         W(JI) = X(I)
      ENDDO
      DO I = 1,M
         JI = J1 + 1
         W(JI) = F(I)
      ENDDO
      W(JI+1) = W(4)
! TEST MAXIMUM NUMBER OF FUNCTION EVALUATIONS
      IF (IW(3)>=IW(2)) GO TO 53
      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)
      IF (MF==1) GO TO 12
      X(MF-1) = W(3)
      J2 = JS + N
      S = 0.D0
      DO I = 1,M
         T = F(I) - W(J2+I)
         S = S + T*T
      ENDDO
      J = 2
      IF (S<EPS*EPS*W(JS+LM)) GO TO 55
      W(3) = S
      J1 = 2 + N + MF
      W(J1) = DSQRT(W(3))
      IF (MF<=2) GO TO 12
      I1 = N + 1
      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))
         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)
      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
      DO L = 1,KV
         T = W(JS+LM*L)
         S = S + 1.D0/ (T*T)
      ENDDO
      W(JA) = 1.DG/S
! CENTRE OF THE VARIABLES AND FUNCTIONS
      I1 = M + N

```

```

5550 DO I = 1, I1
      J1 = JS
      S = 0.D0
      DO L = 1, KV
        T = W(J1+LM)
        S = S + W(J1+I) / (T*T)
        J1 = J1 + LM
      ENDDO
      W(J3+I) = S*W(JA)
    ENDDO
5555 IF (KE/=1) GO TO 60
      IF (W(2)=0.D0) GO TO 20
      J1 = JA - M - 1
      DO I = 1, M; W(J1+I) = F(I); ENDDO
      GO TO 23
    ! MATRIX A
      20 J1 = JA
      DO I = 1, N
        U = W(J3+I)
        DO J = 1, M
          J1 = J1 + 1
          J2 = JS
          S = 0.D0
          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
          ENDDO
          W(J1) = S*W(JA)
        ENDDO
      ENDDO
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
      24 J1 = JD
      DO I = 1, N
        T = W(J3+I)
        DO J = 1, I
          J1 = J1 + 1
          J2 = JS
          S = 0.D0
          U = W(J3+J)
          DO L = 1, KV
            V = W(J2+LM)
            S = S + (W(J2+I) - T) * (W(J2+J) - U) / (V*V)
            J2 = J2 + LM
          ENDDO
          W(J1) = S*W(JA)
        ENDDO
      ENDDO
5600 ! 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

```

```

5610 I2 = 1
      J1 = JD + (I*I-I)/2
      S = 0.D0
      DO J = 1, N
        J1 = J1 + I2
        IF (J>=I) I2 = J
        S = S + W(J1)*W(J+4)
      ENDDO
      X(I) = W(J3+I) - S
    ENDDO
5615 ! TEST OF CONVERGENCE
      31 A = 0.E0
      DO I = 1, N
        W(I+4) = X(I) - W(JM+I)
        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
      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
      GO TO 100
    ! RANDOM PREDICTION
      35 DO I = 1, N
        A = SNGL(W(J3+I))
        X(I) = W(JM+I) + W(1)*E(I) * &
          (MOD(IABS(INT(IR, KIND=4)), 200) - 100) / 100.D0
      ENDDO
      IW(4) = 3
      GO TO 100
    ! ONE DIMENSIONAL SEARCH
      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
      IR = 3
      W(5) = IR
      IR = 20
      W(6) = IR
      W(8) = 0.5D0
      W(11) = 0.D0
      W(12) = 0.D0
      W(13) = 0.D0
      W(14) = 1.D0
      W(16) = W(JM+LM)
      W(17) = W(4)
      GO TO 40
      39 W(9) = W(4)
      CALL FIT1(KE, W(5), W(8))
      DO I = 1, N; X(I) = W(JM+I) + W(8)*W(J3+I); ENDDO
      IF (KE=3) KE = 2
      IF (KE=2) GO TO 53
      KE = 1
      W(3) = W(8)
      GO TO 100

```

```

5665 ! 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) = DSORT(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 = SINGL(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 = SINGL(W(J1+J))

```

```

5725 W(J1+J) = W(J2+J)
   W(J2+J) = A
   ENDDO
49 ENDDO
GO TO 66
! INVERSE OF MATRIX D
60 T = DSORT(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-1)/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) = DSORT(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) = DSORT(W(JD+ (I*I+1)/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) = DSORT(DABS(W(JM+LM)-S*S/M)/MAXG(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
      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
      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
      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 I1 RESPECTIVELY
      THEY CONTAIN ALL INFORMATION FOR THE CONTINUATION OF THE SEARCH
      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
      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(I(3))
      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
      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
      V(3) = -1
      W(6) = W(1)
      W(9) = W(2)
      1 W(1) = W(1) + W(3)
      GO TO 12
      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
      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)
      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)
        W(K) = W(K+1)
        W(K+1) = W(1)
        K = K + 3
        W(1) = W(K)
        W(K) = W(K+1)
        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
      GO TO 9
      5 IF (V(3)=0.D0) GO TO 6
        J = IDINT(V(3))
        W(7-J) = W(7)
        W(10-J) = W(10)
        W(7) = W(7+J)
        W(10) = W(10+J)
        W(7+J) = W(1)
        W(10+J) = W(2)
        IF (W(2)>=W(10)) V(3) = 0.D0
      6 GO TO 9
      J = -1
      IF (W(1)<W(7)) J = 1
      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)
      IV = IDINT(V(3))

```



```

6020 V = 1.D0/DSQRT(V)
      SIG = S/V
      U = S + SIG
      A(K,K) = -SIG
      IF (K>=IR) GO TO 13
      L = K + 1
      DO J = L, IR
        S = V*A(K,J)
        P = DABS(S)
        DO I = L,M
          R = (A(I,K)/SIG)*A(I,J)
          S = S + R
          P = P + DABS(R)
        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
      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
        A(I,J) = R
      ENDDO
    ENDDO
  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
  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
      II = I
      I = I - 1
      S = 0.D0
      DO J = II,K; S = S + A(I,J)*D(IJ+J); ENDDO
      D(IJ+I) = -S/A(I,I)
    ENDDO
    IJ = IJ + K
  16 ENDDO
! INVERSE OF THE PRODUCT MATRIX
  IJ = 0
  DO J = 1, IR
    DO I = 1,J
      IJ = IJ + I
      II = IJ
      L = J - I
      S = 0.D0
      DO K = J, IR
        S = S + D(II)*D(II+L)
        II = II + K
      ENDDO
      D(IJ) = S
    ENDDO
  ENDDO

```

```

6080 GO TO 20
      IR = -2
      20 IF (IR==0) IR = -1
        IF (IR==N) IR = 0
      END SUBROUTINE
-----
6085 LILESO
-----
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
      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 !IX!!=MIN IS NOT USED
      BECAUSE THE COMPONENTS OF X ARE ASSUMED TO BE
      NOT COMMENSURABLE
      IP: PERMUTATION VECTOR OF N COMPONENTS, IT CONTAINS
      THE COLUMN LABELS OF MATRIX A ORDERED ACCORDING
      THEIR IMPORTANCE IN REDUCING THE EUCLIDEAN NORM
      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
      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
      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.
      THE ROUND OFF ERROR OF !B-A*X!!*2 IS APPROXIMATELY GIVEN BY
      !!B(INITIAL!!**2 - !!B(TRANSFORMED!!**2
-----
6110 SUBROUTINE LILESO(M,N,IER,A,B,X,VP)
      IMPLICIT NONE
      INTEGER(2) :: IER
      INTEGER(4) :: M,N,I,IP,J,K,L,L1,L2
      REAL(8) :: C,DELTA,EPS,P,O,R,S,SIG,T,U,V,W
      REAL(8) :: A(M,N),B(M),VP(N),X(N)
      DATA EPS/1.D-8/
      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
6115 ! ROTATION LOOP
      DO 10 K = 1,N

```

```

! PIVOT ELEMENT
U = 0.D0
DO 4 J = K,N
  C = 0.D0
DO 2 I = K,M
  IF (DABS(A(I,J))<=DABS(C)) GO TO 2
  L2 = I
  C = A(I,J)
ENDDO
2 IF (C==0.D0) GO TO 4
  S = 0.D0
  T = 0.D0
DO I = K,M
  V = A(I,J)/C
  S = S + V*V
  T = T + V*B(I)
ENDDO
IF (U>=T* (T/S)) GO TO 4
U = T* (T/S)
SIG = C*DSQRT(S)
W = T
L = J
L1 = L2
4 ENDDO
IF (U==0.D0) GO TO 11
! PERMUTE A(K) AND B(K)
I = IDINT(VP(L))
VP(L) = VP(K)
VP(K) = I
DO I = 1,M
  C = A(I,L)
  A(I,L) = A(I,K)
  A(I,K) = C
ENDDO
C = B(K)
B(K) = B(L1)
B(L1) = C
DO J = K,N
  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)
U = SIG + A(K,K)
V = A(K,K)/SIG
DELTA = (B(K)+V*W)/U
A(K,K) = -SIG
B(K) = -V*W
L = K + 1
IF (L>M) GO TO 10
IF (K>N) GO TO 8
DO J = L,N
  S = V*A(K,J)
  P = DABS(S)
DO I = L,M
    R = A(I,K)/SIG*A(I,J)
    S = S + R
    P = P + DABS(R)
  ENDDO

```

```

6200 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
      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
        A(I,J) = R
      ENDDO
      ENDDO
      DO I = L,M; B(I) = B(I) - DELTA*A(I,K); ENDDO
10 ENDDO
! 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
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
    IP = IDINT(VP(J))
    X(IP) = 0.D0
  ENDDO
  ENDDO
  IF (K==0) GO TO 20
! 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
  L = K + 2 - J
  S = B(L-1)
  DO I = L,K
    IP = IDINT(VP(I))
    S = S - A(L-1,I)*X(IP)
  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

```