U.S. DEPARTMENT OF THE INTERIOR
U.S. GEOLOGICAL SURVEY

A TEST TO EVALUATE THE EARTHQUAKE PREDICTION ALGORITHM, M8

by
John H. Healy[1], Vladimir G. Kossobokov[2],
and James W. Dewey[3]

OPEN-FILE REPORT 92-401

1992

[1]U.S. Geological Survey, Menlo Park, CA 94025
[2]Russian Academy of Sciences, Moscow, Russian Federation
[3]U.S. Geological Survey, Golden, CO 80401

THE DESIGN OF A TEST TO EVALUATE THE EARTHQUAKE PREDICTION ALGORITHM, M8

J. H. Healy, V. G. Kossobokov, and J. W. Dewey[1]

ABSTRACT

A test of the algorithm M8 is described. The test is constructed to meet four rules, which we propose to be applicable to the test of any method for earthquake prediction:

1. An earthquake prediction technique should be presented as a well documented, logical algorithm that can be used by investigators without restrictions.

2. The algorithm should be coded in a common programming language and implementable on widely available computer systems.

3. A test of the earthquake prediction technique should involve future predictions with a black box version of the algorithm in which potentially adjustable parameters are fixed in advance. The source of the input data must be defined and ambiguities in these data must be resolved automatically by the algorithm.

4. At least one reasonable null hypothesis should be stated in advance of testing the earthquake prediction method, and it should be stated how this null hypothesis will be used to estimate the statistical significance of the earthquake predictions.

The M8 algorithm has successfully predicted several destructive earthquakes, in the sense that the earthquakes occurred inside regions with linear dimensions from 384 to 854 km that the algorithm had identified as being in times of increased probability for strong earthquakes. In addition, M8 has successfully "post predicted" high percentages of strong earthquakes in regions to which it has been applied in retroactive studies. The statistical significance of previous predictions has not been established, however, and post-prediction studies in general are notoriously subject to success-enhancement through hindsight. Nor has it been

---

[1] J. H. Healy U. S. Geological Survey, 345 Middlefield Rd. Menlo Park CA 94025.
V. G. Kossobokov International Institute for Earthquake Prediction Theory and Mathematical Geophysics, Russian Academy of Sciences, Marshavskoye Shosse, 79/k. 2, Moscov 113556, Russian Federation. Phone: +7-095-110-7795. Fax: +7-095-310-7032.
J. W. Dewey U. S. Geological Survey, MS 967, Box 25046, Federal Center, Denver Colorado 80225-0046.

determined how much more precise an M8 prediction might be than forecasts and probability-of-occurrence estimates made by other techniques. We view our test of M8 both as a means to better determine the effectiveness of M8 and as an experimental structure within which to make observations that might lead to improvements in the algorithm or conceivably lead to a radically different approach to earthquake prediction.

Our implementation of the M8 algorithm will be directed to the prediction of earthquakes of magnitude 7.5 or greater. M8 will be applied to 147 circles of investigation located in the Circum Pacific seismic belt and Indonesia. Each circle has a radius of 427 km. The algorithm identifies Times of Increased Probability, TIP's, in which there is hypothesized to be an increased probability for the occurrence of $M \geq 7.5$ earthquakes. In the first forward prediction, covering the period 7/1/91 to 1/1/92, TIP's were identified for 38 of the 147 circles. The predictions will be updated every six months through 1997, by which time we anticipate that there will be a sufficient record to evaluate the algorithm. We will evaluate M8 against the null hypothesis that TIP's randomly distributed in the 147 circles of investigation are as effective as M8. We refer to the general test as "M8 Test". Two arms of M8 Test will be run in parallel: these are denoted M8 Test A and M8 Test B, and they differ only in whether or not a TIP is terminated by the occurrence of an $M \geq 7.5$ earthquake.

In M8 Test A, a TIP is terminated when an $M \geq 7.5$ earthquake occurs in the TIP region. Simulated forward predictions were made for the 13 six-month intervals between January 1, 1985 and July 1, 1991, using procedures and parameters as they will be used in M8 Test A. There were ten earthquakes of $M \geq 7.5$ in the circles of investigation. Six of the ten earthquakes occurred in circles with TIP's. On average, TIP's were declared in 22 percent of the circles in each six-month period. Evaluation of the success ratio, six earthquakes predicted out of ten, against the null hypothesis implies that successes of M8 between January 1, 1985 - July 1, 1991 are significant at only a 76 % level of confidence, which neither proves nor disproves the validity of the algorithm. The number of circle-years of TIP per predicted earthquake is 34.1.

The process of running simulated forward predictions for 1/1/85 - 7/1/91 suggested the modification of the M8 algorithm that we will use in M8 Test B. This modification may improve M8's effectiveness and is necessary to account for space-time volumes corresponding to TIP-regions in which $M \geq 7.5$ earthquakes have occurred. Two of the four earthquakes missed in Test A occurred in the same circle as a preceding and predicted $M \geq 7.5$ earthquake. The second of each pair of earthquakes in the two circles were not predicted by M8 because of the Test A convention that a TIP is terminated by a strong earthquake. In the B arm of M8 Test, a TIP is not turned off when an $M \geq 7.5$ earthquake occurs but is allowed to run its full term (commonly five years). With this modification, eight of

the ten strong earthquakes between 1/1/85 - 7/1/91 are predicted. Because Test B was formulated after the 1/1/85 - 7/1/91 data had been examined with Test A conventions, we cannot rigorously estimate the significance of the Test B conventions when applied to those data. Formal use of the same null hypothesis as used with Test A would suggest that the Test B results are significant at a 97 percent level of confidence. The numbers of circle-years of TIP per predicted earthquake is 27.7.

INTRODUCTION

After the Loma Prieta earthquake on October 17, 1989, a group of scientists from the International Institute for Earthquake Prediction Theory and Mathematical Geophysics in Moscow came to Menlo Park, California, to explain and demonstrate their earthquake prediction algorithms. One of the algorithms, M8, had successfully predicted the Loma Prieta earthquake, in that the earthquake occurred inside a circle of 280 kilometers radius that the M8 algorithm had identified as being in a five year Time of Increased Probability (TIP) for an $M \geq 7$ shock (Keilis-Borok and others, 1990). The successful TIP was presented at a meeting of the National Earthquake Prediction Evaluation Council (NEPEC) in 1988. (Updike, 1989, Appendix A, Page 10). An M8 TIP for an $M \geq 7.5$ shock in California was also presented at this meeting. Many seismologists in the U. S. Geological Survey have viewed the Loma Prieta prediction with skepticism: the M8 approach to earthquake prediction seems to these seismologists to be inconsistent with their understanding of earthquake genesis, and, in view of the large area and long time associated with the successful TIP, the successful prediction may have been fortuitous. In this paper we describe a test of the M8 algorithm. We have described what we believe to be the essential features of a test of an earthquake-prediction technique, and we have constructed a test that has these features. We refer to this test as M8 Test. The M8 Test will have two arms, Test A and Test B.

OVERVIEW OF M8 AND M8 TEST

The M8 algorithm was originally designed for diagnosis of Times of Increased Probability (TIP's) of the strongest, magnitude 8 and above, earthquakes (Keilis-Borok and Kossobokov, 1986). The formulation of the algorithm was normalized in such a way, that it can be applied without additional data-fitting to diagnose TIP's of strong earthquakes of magnitude less than 8 (Keilis-Borok and Kossobokov, 1990a).

TIP's are defined for regions within circles having pre-defined radii. The radii are five to ten times larger than the lengths of coseismic faulting typical of earthquakes of the magnitude being predicted: for the $M \geq 7.5$ earthquakes that are the objects of

prediction in M8 Test, the radii will be 427 km. The algorithm analyses seven functions of seismicity from within each circle. M8 announces a TIP when most of these functions are anomalously high with respect to the long-term record of seismicity in the circle. M8-TIP's are generally defined to last for 5 years beyond the date at which the criteria for TIP-declaration are fulfilled, provided that the criteria for TIP-declaration continue to be fulfilled at that date in the course of further semiannual updates of the seismicity data-base. A TIP will be terminated at the time of a semiannual update, before 5 years have elapsed, if the criteria for TIP-declaration are no longer satisfied with the updated data-base. A TIP may be extended beyond 5 years if the levels of seismicity (as parameterized by M8) continue to increase after the TIP-declaration criteria are first fulfilled.

In most studies with M8 run up to the present time, a TIP has been terminated on the occurrence of an earthquake of the size being predicted, and this convention has been continued in the A arm of M8 Test. The B arm of M8 Test will be run with a version of M8 in which a TIP is not terminated on the occurrence of an $M \geq 7.5$ earthquake within its circle. The decision to cancel or retain a TIP after a strong earthquake is rather the option of the user, since the rest of the analysis remains precisely the same. The choice depends on whether the user prefers to predict the second earthquake in a pair at the cost of increasing the total alarm time.

The algorithm might diagnose a TIP in response to a dramatic short-term increase of activity near the source region of the future strong earthquake, but most earthquakes are not preceded by such premonitory seismicity, and M8 is tuned more generally to detecting premonitory patterns over a broader space-time window than would be characteristic of a short-term, near-source, foreshock sequence.

A more detailed discussion of the M8 functions and a listing of the algorithm are given in Appendix IV.

Rationale for M8 Test

We propose M8 Test because M8 appears promising to us from several standpoints, and we think the time has come to subject the algorithm to a rigorous test.

A demonstration of the validity of the algorithm M8 would confirm the following general perceptions on which it is based.
- A strong earthquake is commonly preceded by specific intermediate-term change of seismic activity in the small and moderate magnitude ranges. Among these changes is an increase of activity, clustering of earthquakes and several other phenomena.
- These changes take place over large areas which may include many

active faults.
- These changes are similar in a wide variety of seismic regions.

M8 is based on a general geophysical hypothesis that has been widely proposed independently of M8. The general hypothesis, that a strong earthquake will commonly be preceded by an intermediate-term increase in small and moderate earthquake activity in a broad region that includes the future source of the strong earthquake, has been put forth in a number of different versions (e.g., Kanamori, 1981; Mogi, 1981; Reyners, 1981; Scholz, 1990): many versions postulate that the time-period and region of increased seismicity encompasses a smaller space-time window of seismic quiescence. Ideally, the parameterization of M8 should have achieved a partial optimization of the general hypothesis and have implicitly accounted for near-source seismic quiescence. The proceedings of the 1988 NEPEC meeting (Updike, 1989) contain several different perspectives on the philosophy of M8 and the process of data-fitting the M8 parameters.

As elaborated in Appendix IV, results of previous studies with the M8 algorithm are generally encouraging. Most previous studies have involved post-predictions. Analyses of the post-predictions suggest that the level of success is statistically significant. Although there have been several strong earthquakes that have occurred in regions of TIP's following the diagnosis of the TIP's, statistical estimates of the effectiveness of the algorithm have so far been based on post predictions.

From a practical standpoint, confirmation of the effectiveness of M8 may justify using TIP's as bases for some kinds of earthquake-mitigation efforts. The practical value of M8 TIP's would be particularly enhanced by development of techniques to more precisely define the source region of a future strong earthquake within the broad circle of investigation for which the TIP is issued (e.g., Keilis-Borok and Kossobokov, 1990b; Kossobokov and others, 1990).

The possibility that M8 might not be effective is implicit in our proposal to test the algorithm. The hypothesis may be false that regional seismicity tends to increase prior to strong earthquakes in the fashion parameterized by M8. The NEIC data base to which M8 is applied in Test A (Appendix I) may be too heterogeneous to permit recognition of precursory seismicity patterns. It is possible that successful predictions of M8 Test will be statistically significant but that the probability gain of M8 over well-established methods of calculating earthquake probabilities will be too small to justify additional earthquake mitigation efforts on the basis of M8 TIP's. Arguments that M8 might not be effective are presented in more detail in the proceedings of the 1988 NEPEC meeting (Updike, 1989).

The setting up and running of the M8 Test, besides accomplishing its primary purpose of testing the present formulation of M8, also provides a basis for collecting and organizing observations that may lead to improved formulations of M8 or possibly to substantially different approaches to earthquake prediction.

M8 TIP's - should they be called predictions?

Throughout this paper we refer to M8 TIP's as "predictions." We think our usage is consistent with the philosophy and characteristics of "earthquake prediction" suggested by the U.S. National Research Council, Panel on Earthquake Prediction of the Committee on Seismology (1976, p.7): "An earthquake prediction must specify the expected magnitude range, the geographical area within which it will occur, and the time interval within which it will happen with sufficient precision so that the ultimate success or failure of the prediction can readily be judged. Only by careful recording and analysis of failures as well as successes can the eventual success of the total effort be evaluated and future directions charted. Moreover, scientists should also assign a confidence level to each prediction."

Our usage of "prediction" is intended to be synonymous with "research prediction". The Seismological Society of America (1983) has accepted a definition of earthquake prediction which requires that the prediction have sufficient precision that actions to minimize loss of life and reduce damage to property are possible; Wallace and others (1984) propose that earthquake predictions should refer to a specific future earthquake. Under these more stringent definitions of "earthquake prediction," the TIP's issued by M8 Test might not be classified as predictions. A prediction that was to be acted on by society, or a prediction of a specific future earthquake, would involve integration of all available evidence, of which M8 TIP's would constitute only one part.

GENERAL RULES FOR TESTING EARTHQUAKE PREDICTION TECHNIQUES

Our goal is to conduct a test of M8 that will be acceptable to a broad spectrum of the scientific community. In particular, if the test convinces us that M8 is effective at identifying times of increased probability of earthquake occurrence, we would like the test also to convince colleagues that this is the case, even if these colleagues now find the assumptions of M8 to be counter-intuitive. Our goal is clearly similar to that of many who have developed earthquake prediction techniques and presented their predictions to the public.

We agreed that a convincing test would have to be based on the following rules, which can be applied to all earthquake prediction techniques:

1. An earthquake prediction technique should be presented as a well documented, logical algorithm that can be used by other investigators without restrictions.

2. The algorithm should be coded in a common programming language and implementable on widely available computer systems.

3. A test of the earthquake prediction technique should involve future predictions with a black box version of the algorithm in which potentially adjustable parameters are fixed in advance. The source of the input data must be defined and ambiguities in these data must be resolved automatically by the algorithm.

4. At least one reasonable null hypothesis should be stated in advance of testing the earthquake prediction method, and it should be stated how this null hypothesis will be used to estimate the statistical significance of the earthquake predictions.

The rules are intended to eliminate the possibility of cheating or self deception on the part of the predictor. Rules 1 and 2 seem to us necessary to facilitate communication with scientists who may be working from different earthquake-prediction paradigms.

SPECIFICS OF RUNNING AND EVALUATING M8 TEST

We will apply M8 systematically to the intermediate-term prediction of future M ≥ 7.5 earthquakes from 1 July 1991 through December 1997. These research predictions will be updated semiannually in February and August. The length of the test-period is equal to that of the simulated forward prediction study described below, in the section entitled "M8 TEST APPLIED TO THE INTERVAL 1/1/85 - 7/1/91", and should provide a sufficient record to evaluate the algorithm.

Values of seismicity functions are computed for circles of investigation at six-month intervals for a span of years that begins at 1975/01/01 and ends at "Te". At February semiannual updates, "Te" will be 1 January of the current year; at August semiannual updates "Te" will be 1 July of the current year. The Qth percentiles that are used to define anomalous values of seismicity functions are based on a span of years that begins at 1975/01/01 and ends at "T*." For most circles, "T*" will be six months earlier than "Te", in order that the definition of Qth percentiles be based entirely on Monthly PDE data. In some circles, past M ≥ 7.5 earthquakes have been retroactively identified as having been preceded by TIP's. For these circles, the values of T* are set to dates before the date of the

successfully post-predicted earthquake; it is assumed that the triggering threshold-values of seismicity functions should be those on which the successful retroactive TIP's were based.

We define a unit of TIP as six-months of a TIP in a single circle. TIP units are defined to span 1 January - 30 June or 1 July - 31 December respectively. 1 January - 30 June TIP units will therefore span between 98% and 99% of the time spanned by 1 July - 31 December TIP units. Our definitions of units of TIP are made for the purpose of evaluating the results of M8 with a null hypothesis; the 1 January - 30 or 1 July - 31 December Tip units are not intrinsic to the M8 computer program. TIP-lengths defined by the M8 program are measured in minutes, rather than calendar days or months, and TIP's issued by the program are of equal length (5 years). As a result, there may be a mismatch between the beginnings or endings of TIP's issued by the M8 program and the 1 January - 30 June, 1 July - 31 December TIP units defined for M8 Test. A program-issued TIP may span either 98 percent or more of a six-month TIP-unit or 2 percent or less of a TIP-unit. For purposes of M8 Test, if a program-issued TIP spans 98 percent or more of a six-month TIP-unit, the TIP is considered "on" for the entire TIP-unit. If a program-issued TIP spans 2 percent or less of a six-month span, we consider that there is not a TIP for the entire potential TIP-unit.

We consider M8 Test to have started on 1 July, 1991. Most potentially adjustable parameters of the test were fixed at that time. The TIP's for Table 1 were submitted to colleagues on 19 September 1991, with a draft of this paper. The decision to use the B arm of M8 Test was made several months later and was incorporated in a draft of this paper that we submitted to the National Earthquake Prediction Evaluation Council (NEPEC) for consideration at their meeting of 8 May, 1992. J. H. Dieterich submitted his proposed alternate null hypothesis (Appendix VI) to NEPEC for consideration at the same meeting.

We define future events to be those occurring on or after 1 July, 1991, and past events to be those occurring before that date. As of 8 May 1992, the NEIC data-base (Appendix I) contained no $M \geq 7.5$ shocks during the period following 1 July 1991.

For the purposes of the test, the M8 algorithm will be considered as a "black box", with potentially adjustable parameters in the algorithm fixed in advance of the date of the strong earthquakes that will be used to test the algorithm. We cannot rule out that a change in the structure of the NEIC data-base or our discovery of a logical oversight might force us to modify the algorithm in the course of running M8 Test. Any such change in the M8 algorithm would be announced in our reports of semiannual updates, prior to the beginnings of the TIP's that are diagnosed on the basis of the modified algorithm. Every half year we will run the same computer program in Moscow, Menlo Park, and Golden on the then-current NEIC

data-base. We claim that these procedures will enable us to satisfy the third of our General Rules for Testing Earthquake Prediction Techniques (previous section).

Data

Analyses will be based on the NEIC data-base existing at the times of each update. The data base, and the conventions used in M8 Test A to select earthquakes and earthquake-parameters from the data base, are described in Appendix I.

Programs

Hypocenters and magnitudes are extracted from the NEIC data-base using standard NEIC software (see Appendix I). Processing of the catalogs to obtain predictions is done by programs prepared at the International Institute of Earthquake Prediction Theory (Appendices II, III, and IV).

We judge the documentation and accessibility of the programs as only marginally satisfying the first and second of our general rules for testing of earthquake prediction techniques. The spirit of rules 1 and 2, for example, suggests that the programs be written in a single, well-known, portable programming language. The C language would fulfill this requirement. A good C program written for an IBM PC computer could be easily transported to many other systems. As it stands, however, we use programs written in C and Fortran. In addition, we use a program (the EDBS software used for extracting the most recent NEIC data) that is not maintained by any one of us and that could, in principle, be altered during the period of conducting M8 Test without our knowing it.

Regions covered by M8 Test

We will run the M8 algorithm for 147 circles with radii of 427 km that are located in the Circum-Pacific seismic belt and Indonesia and that have on average at least sixteen magnitude 4.0 shocks per year (Figure 1). Coordinates of the circles are given in Table 1.

Table 1.

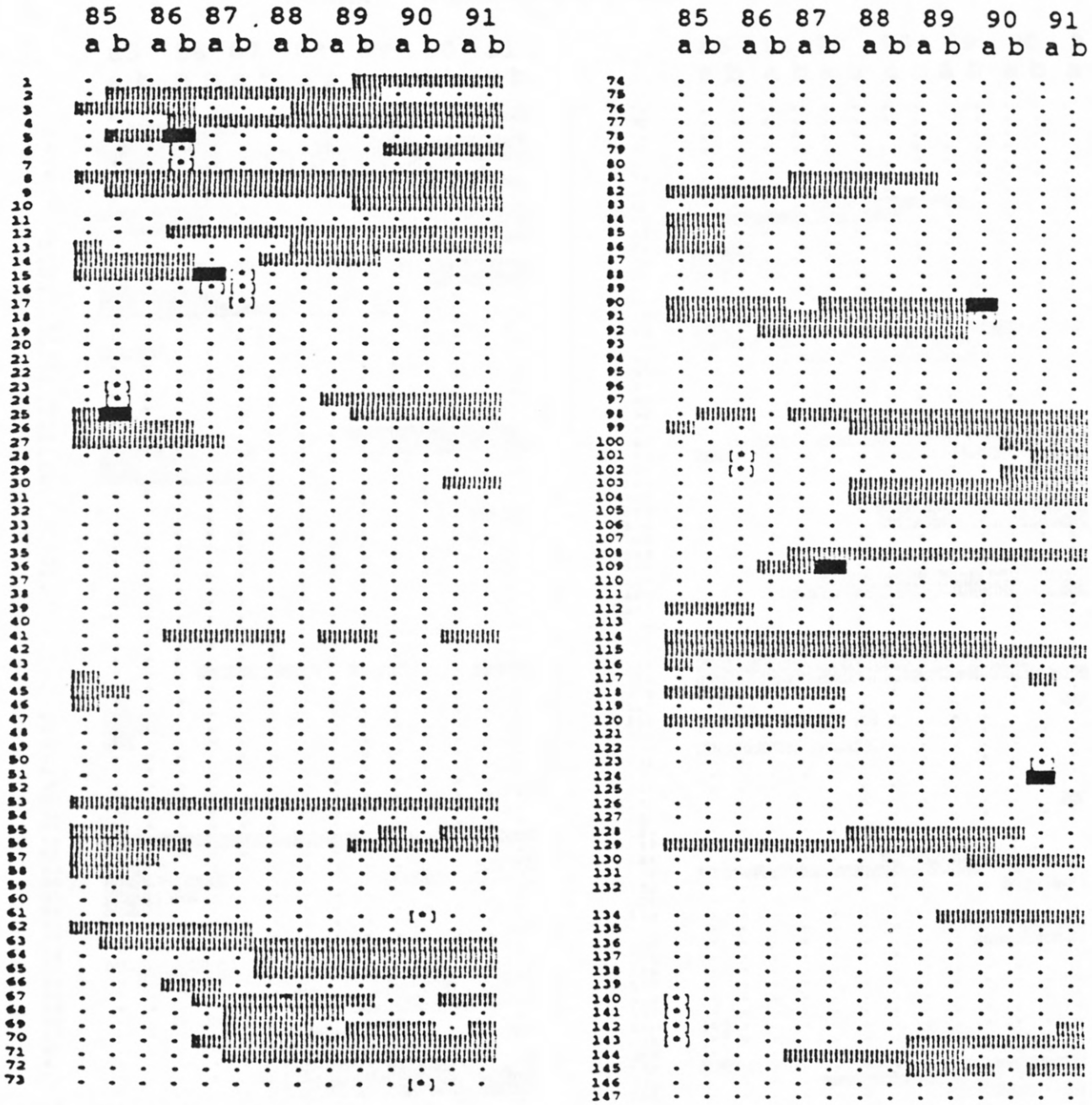
THE DISTRIBUTION OF TIMES OF INCREASED PROBABILITY (TIP) FOR EARTHQUAKES OF MAGNITUDE ≥ 7.5 IN THE CIRCUM PACIFIC SEISMIC ZONE.
The location of circles of investigation.

Reg No	Lat	Lon	Geographic Location
1	-15.00	-175.00	Tonga Trench
2	-17.50	-174.00	Tonga Trench
3	-20.00	-175.00	Tonga Trench
4	-22.50	-176.00	Tonga Trench
5	-25.00	-177.00	Tonga Trench
6	-27.50	-177.50	Tonga Trench
7	-30.00	-178.00	Kermadec Trench
8	-32.50	-179.00	Kermadec Trench
9	-35.00	180.00	Kermadec Trench
10	-37.00	178.00	Kermadec Trench
11	-2.00	136.00	New Guinea
12	-2.25	138.50	New Guinea
13	-2.50	141.00	New Guinea
14	-3.75	143.50	New Guinea
15	-5.00	146.00	New Guinea
16	-5.00	149.00	New Guinea
17	-5.00	152.00	Solomon Islands
18	-6.25	154.00	Solomon Islands
19	-7.50	156.00	Solomon Islands
20	-8.75	158.00	Solomon Islands
21	-10.00	160.00	Solomon Islands
22	-10.50	162.50	Solomon Islands
23	-11.00	165.00	Solomon Islands
24	-13.00	166.25	New Hebrides
25	-15.00	167.50	New Hebrides
26	-17.50	168.25	New Hebrides
27	-20.00	169.00	New Hebrides
28	-21.25	170.75	New Hebrides
29	9.50	93.75	Java Trench
30	7.00	94.50	Java Trench
31	5.00	95.75	Java Trench
32	3.00	97.00	Java Trench
33	2.00	98.50	Java Trench
34	-1.00	100.00	Java Trench
35	-3.00	101.50	Java Trench
36	-5.00	103.00	Java Trench
37	-6.50	105.00	Java Trench
38	-8.00	107.00	Java Trench
39	-8.50	109.50	Java Trench
40	-9.00	112.00	Java Trench
41	-9.25	114.50	Java Trench
42	-9.50	117.00	Java Trench
43	-9.50	119.50	Java Trench
44	-9.50	122.00	Java Trench
45	-8.25	124.50	Banda Sea
46	-7.00	127.00	Banda Sea
47	-6.00	129.00	Banda Sea
48	-5.00	131.00	Banda Sea
49	-3.50	129.25	Banda Sea
50	-2.00	127.50	Banda Sea
51	-1.00	125.25	Celebes Basin
52	.00	123.00	Celebes Basin
53	.00	120.50	Celebes Basin
54	1.50	125.00	Celebes Basin
55	3.00	127.00	Philippine Islands
56	5.25	126.50	Philippine Islands
57	7.50	126.00	Philippine Islands
58	9.75	125.50	Philippine Islands
59	12.00	125.00	Philippine Islands
60	13.50	123.00	Philippine Islands
61	15.00	121.00	Philippine Islands
62	35.00	136.00	Southern Japan
63	34.00	134.00	Southern Japan
64	33.00	132.00	Southern Japan
65	31.00	132.00	Southern Japan
66	29.00	131.00	Southern Japan
67	27.00	129.00	Ryukyu Islands
68	26.00	127.00	Ryukyu Islands
69	25.00	124.50	Ryukyu Islands
70	25.00	122.00	Ryukyu Islands
71	22.50	121.50	Taiwan
72	20.00	121.50	Taiwan
73	17.50	121.00	Taiwan
74	45.50	150.50	Kuril Islands
75	44.00	148.00	Kuril Islands
76	43.50	145.50	Kuril Islands
77	43.00	143.00	Japan
78	41.00	141.00	Japan
79	39.00	142.00	Japan
80	36.50	141.00	Japan
81	35.00	139.00	Japan
82	33.00	141.00	Izu Trench
83	31.00	142.00	Izu Trench
84	29.00	142.50	Izu Trench
85	27.00	143.00	Bonin Trench
86	25.00	143.00	Bonin Trench
87	23.00	143.00	Bonin Trench
88	21.00	144.50	Bonin Trench
89	19.00	146.00	Mariana Trench
90	16.50	147.00	Mariana Trench
91	14.00	146.00	Mariana Trench
92	12.00	144.00	Mariana Trench
93	12.00	141.00	Mariana Trench
94	46.00	152.00	Kuril Islands
95	48.50	155.50	Kuril Islands
96	51.00	158.00	Kamchatka
97	53.50	160.00	Kamchatka
98	56.50	161.50	Kamchatka
99	55.00	166.50	Aleutian Trench
100	51.50	176.00	Aleutian Trench
101	51.00	-178.50	Aleutian Trench
102	51.50	-173.00	Aleutian Trench
103	52.50	-167.50	Aleutian Trench
104	54.00	-162.50	Aleutian Trench
105	55.50	-157.50	Aleutian Trench
106	56.50	-152.00	Aleutian Trench
107	60.00	-153.00	Alaska
108	63.00	-151.00	Alaska
109	62.00	-145.00	Alaska
110	44.50	-130.00	Blanco Fault Zone
111	43.00	-126.00	Gorda Ridge
112	40.50	-128.00	Mendocino Fault Zone
113	40.50	-123.00	Mendocino Fault Zone
114	38.00	-119.00	Nevada
115	37.50	-122.00	San Andreas Fault Zone
116	35.00	-118.50	San Andreas Fault Zone
117	37.00	-118.00	Nevada
118	16.00	-88.00	Central America
119	16.00	-97.00	Central America
120	15.00	-94.00	Central America
121	14.00	-91.00	Central America
122	12.00	-88.00	Central America
123	10.00	-85.00	Central America
124	8.00	-82.50	Central America
125	5.00	-82.50	South America
126	6.00	-76.00	South America
127	8.00	-73.00	South America
128	.00	-80.00	South America
129	-5.00	-77.00	South America
130	-11.00	-74.00	South America
131	-12.00	-77.50	South America
132	-15.00	-75.00	South America
133	-17.50	-71.00	South America
134	-20.50	-69.00	South America
135	-22.00	-67.00	South America
136	-23.50	-70.00	South America
137	-25.00	-68.00	South America
138	-27.00	-71.00	South America
139	-28.00	-69.00	South America
140	-30.00	-71.50	South America
141	-31.00	-70.00	South America
142	-33.00	-72.50	South America
143	-34.00	-71.00	South America
144	-36.00	-73.00	South America
145	-56.00	-27.00	South Sandvich Islands
146	-57.00	-25.00	South Sandvich Islands
147	-58.50	-25.50	South Sandvich Islands

The symbols in the following tables indicate the status of each circle of investigation in six-month intervals from January 1, 1985 to January 1, 1992. ▥ indicates a TIP, • indicates no TIP, ■ indicates a successful prediction, [•] earthquake occurred at this location, and a blank indicates insufficient data to run the algorithm. Because the circles of investigation overlap an earthquake usually occurs in more than one circle.

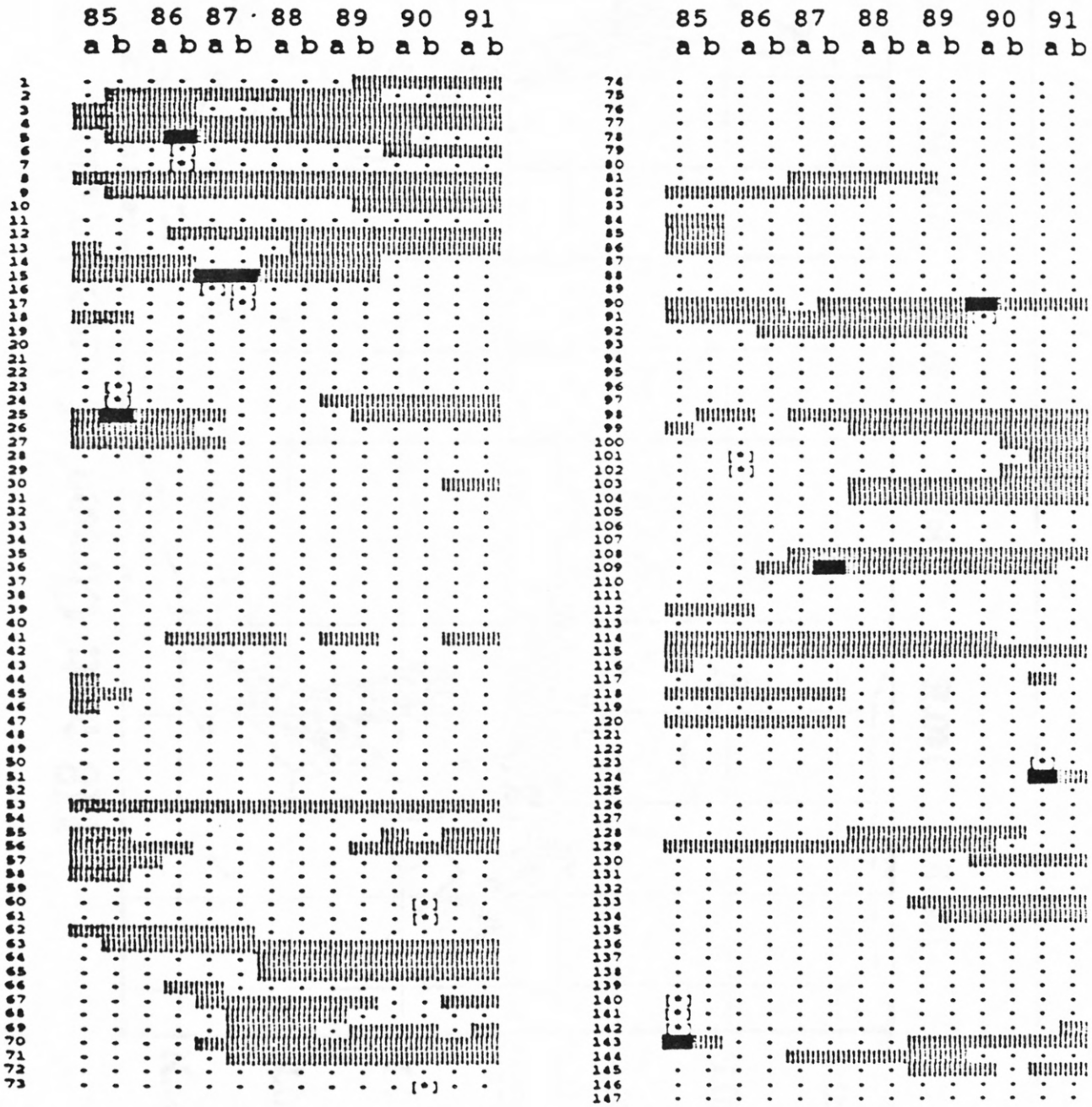
M8 test A results for years 1985-1991.

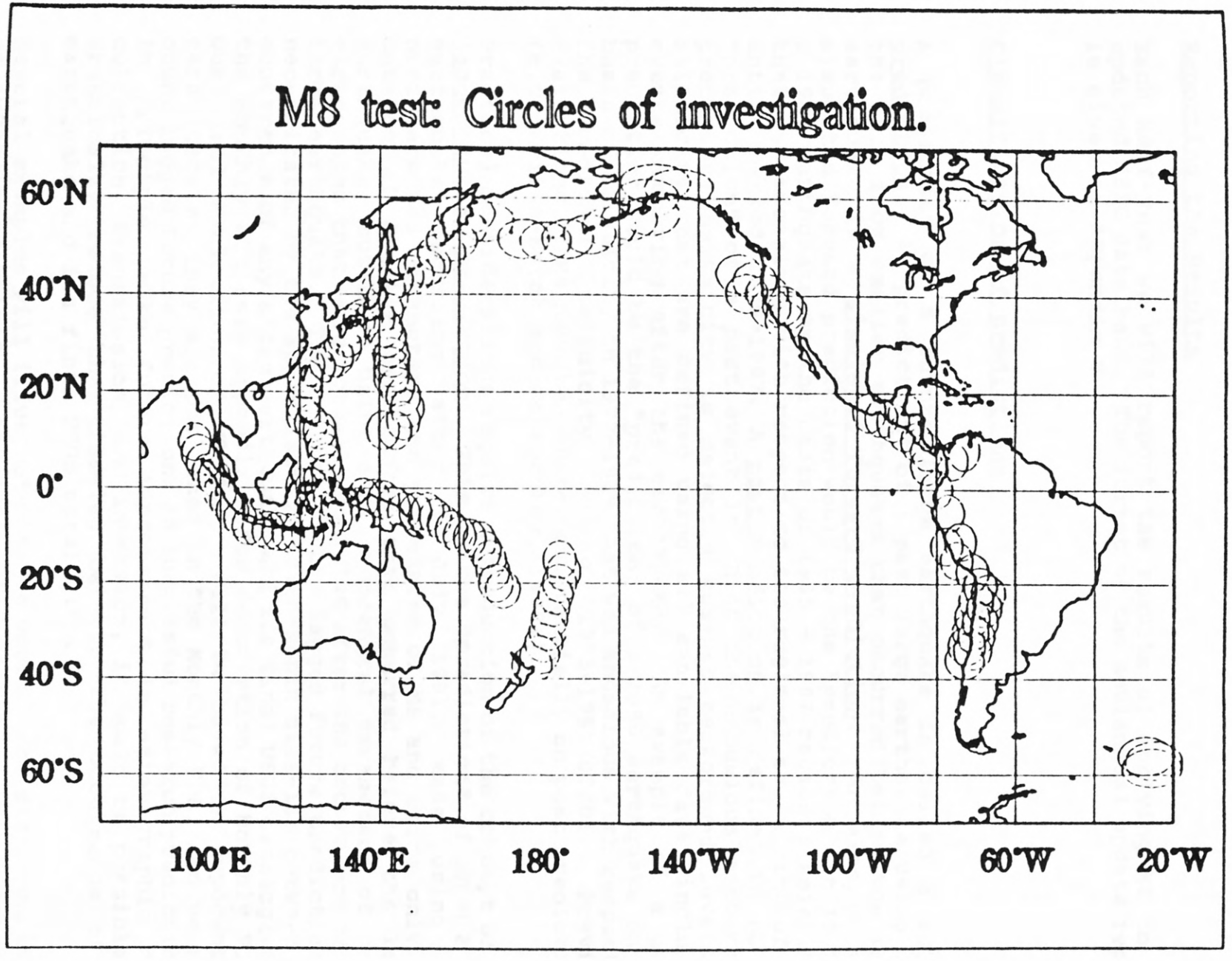
▥ - TIP
. - no TIP
▬ - successful prediction
[•] - earthquake located in this circle
  - blank - insufficient data



M8 test B results for years 1985-1991.

- TIP
- no TIP
- successful prediction
- earthquake located in this circle
- blank - insufficient data




M8 test: Circles of investigation.
60°N
40°N
20°N
0°
20°S
40°S
60°S
100°E
140°E
180°
140°W
100°W
60°W
20°W

Reporting the Results

Each half-year we will report the results of applying M8 to the updated NEIC data base. The format of the semiannual update report is given in Appendix V.

Classification of predictions

A prediction of a future large earthquake is called a future prediction. A "prediction" of a past large earthquake using only the data from smaller earthquakes that occurred before the large earthquake is a simulated forward prediction. An example of a simulated forward prediction would be the "prediction" in 1991 of a 1988 earthquake on the basis of 1985 - 1987 regional seismicity that was anomalous with respect to the regional seismicity of the entire period 1975-1987. A post prediction is defined to be the "prediction" of a past event in which the anomalous pattern of precursory seismicity is detected against background levels of seismicity that are defined using all available data, including events occurring after the earthquake. An example of a post-prediction would be the "prediction" of a 1980 earthquake on the basis of seismicity in 1977-1979 that was anomalous with respect to the regional seismicity during 1975-1991/07/01. Previous evaluations of M8 have been based substantially on post predictions (e.g. Keilis-Borok and Kossobokov, 1990a).

Practical considerations require introduction of the concept of the lagged future prediction. This is the "prediction" of an $M \geq 7.5$ earthquake occurring after 1 July 1991, made using the predetermined parameters and procedures of M8 and using only the catalog of smaller earthquakes that occurred before the large earthquake, but made using some hypocentral parameters of prior earthquakes that were actually computed after the occurrence of the large earthquake. The concept of the lagged future prediction is necessitated by the approximately seven-month time-lag between the occurrence of any sized earthquake and its final USGS cataloging in the Monthly PDE (see Appendix I for description of Monthly PDE). Our final evaluation of M8 Test will be based on hypocentral parameters as they are published in the Monthly PDE, and we will count lagged future predictions as successful research predictions. In principle, with future improvements in seismographic data collection, transmission, and inversion, it would be possible to drastically reduce the time-lag between the occurrence of an earthquake and its final USGS cataloging.

Several examples will illustrate how we would judge different types of future predictions. The terms NEIC data-base and PDE-weekly

are defined in Appendix I.

Scenario 1 - Successful future prediction with no complications. The M8 algorithm recognizes an episode of anomalous seismicity in a time period for which the NEIC data-base contains only Monthly PDE hypocentral parameters. A TIP is accordingly identified and a prediction issued. The prediction is issued prior to a strong earthquake that occurs within the TIP space-volume. The earthquake is assigned $M \geq 7.5$ at every stage of the NEIC cataloging process.

Scenario 2 - Successful lagged future prediction at every stage of the NEIC cataloging process. A strong earthquake of magnitude about 7.5 occurs in early January 1994, before we have had a chance to analyze regional seismicity data from July - December 1993. When we perform a semiannual update of TIP's in early February 1994, we see that the strong earthquake occurred in a TIP that was identifiable in a time period that includes July - December 1993, for which the NEIC data-base contains PDE-weekly hypocenters. And we see that the earthquake is assigned a magnitude of 7.5 in the PDE-weekly. The TIP is provisionally judged a successful lagged future prediction. When we next perform a semiannual update in early August 1994, the NEIC data-base contains entirely Monthly PDE hypocenters in the time-period in which the TIP was declared. Reanalysis of the data-base in August 1994 confirms the TIP. In addition, in August the Monthly PDE magnitude of the January earthquake is confirmed to be 7.5. The successful lagged future prediction is confirmed, and the episode is counted as a successful prediction in our final evaluation of M8.

Scenario 3 - Apparently successful lagged future prediction that is later disqualified on the basis of data in Monthly PDE. A strong earthquake occurs in early January 1994, before we have had a chance to analyze smaller earthquake data from July - December 1993. When we perform a semiannual update of TIP's in February 1994, the PDE-weekly data indicate that the earthquake had $M = 7.5$ and occurred in a TIP. But either the TIP does not emerge when M8 is applied to the data-base as it exists in August 1994, or the strong earthquake is seen to have $M < 7.5$ in the Monthly PDE. The provisionally successful lagged future prediction has not been confirmed, and the episode is not counted as a successful prediction in our final evaluation of M8.

Scenario 4 - A successful lagged future prediction that is identifiable only after issuance of Monthly PDE. A strong

earthquake occurs in early January 1994. In our semiannual update of February 1994, either the earthquake is assigned M < 7.5 or a TIP is not recognized prior to the earthquake. But in August 1994, application of M8 to the augmented data base reveals that the January earthquake did occur in a TIP. Moreover, the Monthly PDE for January shows that the strong earthquake had M ≥ 7.5. The episode is therefore a successful lagged future prediction and is counted as a successful prediction in our final evaluation of M8.

Evaluation of results

In evaluation of these results we will look at two statistics: the number of successful predictions and the number of units of TIP's. Our evaluation of test results will not consider failures-to-predict except indirectly, as these are reflected in the count of successful predictions. The circles of investigation overlap and earthquakes usually occur in more than one circle. When circles with TIP's intersect circles without TIP's, the region of intersection is counted as being in a TIP. If the same earthquake is predicted by more than one TIP, we give credit for only one successful prediction.

We define a null-hypothesis algorithm in which TIP's are randomly distributed in the 147 circles of investigation. The number of TIP's declared in the null-hypothesis algorithm shall be equal to the number of TIP's declared by M8 during the period of comparison. The level of confidence of M8 will be taken to be the percentage of runs with the null-hypothesis algorithm in which TIP units overlap fewer earthquakes than are successfully predicted by M8. We think this satisfies the fourth of our General Rules for Testing Earthquake Prediction Techniques.

The test can be described by analogy to a gambling game in which the player is charged one dollar for each TIP declared during each six-month interval and wins a fixed sum for each earthquake predicted. The goal is to determine the amount the house can afford to pay for each successful prediction. This test can apply to any algorithm that assigns TIP's to circles. The results from other algorithms can be compared with M8 and with our null-hypothesis algorithm. Appendix VI presents one such alternative algorithm, proposed by J. Dieterich after he had read an earlier draft of this paper.

M8 TEST A APPLIED TO THE INTERVAL JANUARY 1, 1985 - JULY 1,1991

In the A-arm of M8 Test, TIP's declared by M8 will be terminated on the occurrence of an M ≥ 7.5 earthquake within the circle(s) of investigation containing the TIP(s). We have run the algorithm as formulated for Test A and made simulated forward predictions for

six-month intervals between 1/1/85 - 7/1/91. These results are presented in Table 1, together with the results of the first future prediction (column 91b).

The total number of TIP units for 1/1/85 - 7/1/91 (columns 85a through 91a, Table I) is 409. There are 1883 possibilities for TIP units. Approximately 22 percent of the possible TIP-units actually had TIP's. Six earthquakes were ''predicted'' out of ten (Table 2) that occurred in the circles of investigation during the simulated forward prediction. There were therefore 68.2 TIP-units per predicted earthquake, or 34.1 circle-years per predicted earthquake.

TABLE 2
Strong (M ≥ 7.5) earthquakes affecting outcome of M8 Test A applied to 1/1/85 - 7/1/91

Date	Time(UTC)	Lat	Long	Predicted by M8?
1981/10/16	0325	33.13S	73.07W	*
1985/03/03	2247	33.13S	71.87W	no
1985/11/28	0349	13.98S	166.18E	yes(region 25)
1986/05/07	2247	51.52N	174.78W	no
1986/10/20	0646	28.12S	176.37W	yes(region 5)
1987/02/08	1833	6.09S	147.69E	yes(region 15)
1987/10/16	2048	6.26S	149.06E	no
1987/11/30	1923	58.67N	142.78W	yes(region 109)
1990/04/05	2112	15.12N	147.59E	yes(region 90)
1990/07/16	0726	15.68N	121.17E	no
1991/04/22	2156	9.68N	83.07W	yes(region 124)

* responsible for failure to predict 1985/03/03 (see text)

During the test period, six TIP's were terminated by earthquakes of magnitude ≥ 7.5, and 29 TIP's ended without an earthquake. Eighty-three percent of the TIP's therefore ended without the occurrence of a large earthquake. Table 3 summarizes the number of TIP-units and the number of analyzed circles in each time interval.

Table 3.
The number of circles under alarm from 1.

Year	1985		1986		1987		1988		1989		1990		1991	
	a	b	a	b	a	b	a	b	a	b	a	b	a	b
Alarms	31	31	24	27	27	28	31	31	35	39	35	32	38	38
Circles	139	141	143	144	144	146	146	146	146	147	147	147	147	147

To test the significance of the 1/1/85 - 7/1/91 results using our proposed null hypothesis, we randomly assigned TIP's, equal in number to the number of TIP's in Table 3, to circles equal in

number to the number of circles in Table 3. We used a random number generator to assign TIP's, and we made one million realizations of the null hypothesis algorithm. The results of these calculations are given in Figure 2 and Table 4. Column 1 gives N, the number of earthquakes predicted. Column 2 is the number of times N earthquakes were predicted in 1,000,000 tries. Column 3 is the cumulative distribution of column 2, the number of times that at least N earthquakes were predicted; the level of confidence is the percentage of cases in which the null hypothesis performs worse than M8.

In the set of 1,000,000 realizations, the null hypothesis algorithm performed worse than M8 (less than 6 earthquakes predicted) 76.4

percent of the time, implying that the M8 results are significant at a 76.4 percent level of confidence.

Table 4

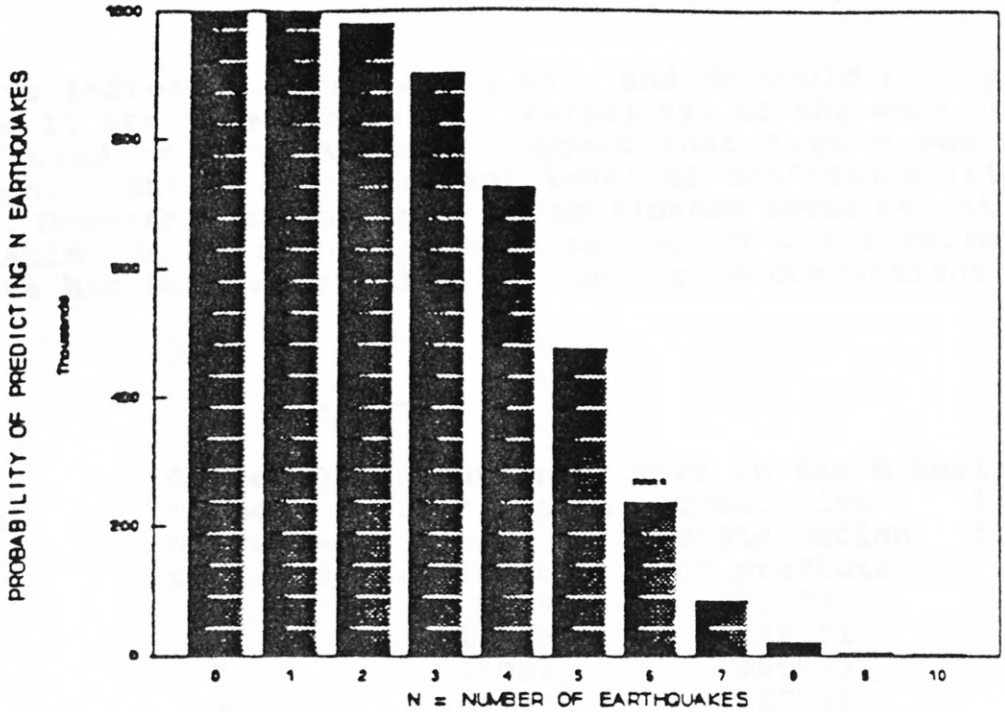
Random Distribution of TIPs in the A-test

Number of quakes predicted	Distribution of predictions	Cumulative distribution of predictions	Level of confidence %	
0	2078	1000000	0.00	
1	18893	997922	0.21	
2	76382	979029	2.10	
3	174846	902647	9.74	
4	251732	727801	27.22	
5	239570	476069	52.39	
6	152444	236499	76.35	**
7	64040	84055	91.59	
8	17237	20015	98.00	
9	2635	2778	99.72	
10	143	143	99.99	

M8 TEST B - A TEST OF A MODIFIED FORM OF M8

The B-arm of M8 Test, in which TIP's are not terminated at the times of $M \geq 7.5$ earthquakes, was designed as a consequence of the simulated forward prediction study with Test A conventions. Examination of M8 Test A applied to 1/1/85 - 7/1/91 revealed that two of the earthquakes that were not predicted occurred in regions for which TIP's had been declared within the previous five years, but in which previous strong earthquakes had occurred following declarations of the TIP's. The two shocks were those of 03/03/85 and 10/16/87. Region 143 (South America - 16), in which 03/03/85 occurred, had a TIP declared for it beginning 12/31/80. The TIP was

Figure 2. M8 Test A, Number of earthquakes predicted in 1000000 random trys.



followed by a magnitude 7.5 earthquake that occurred on 10/16/81 (Table 2), which turned off the TIP that would otherwise have extended at least to 12/31/85. A TIP for region 15 (New Guinea - 5), in which 10/16/87 occurred, was followed by the shock of 02/08/87 (Tables 1 and 2) which turned off the TIP which would otherwise have extended at least to 12/31/89.

The three authors of this report agreed that the Test B modification is desirable for the purposes of testing M8. Otherwise, some space-time volumes (such as that in which 1987/10/16 occurred) are formally inaccessible to M8 TIP's, and other space-time volumes (such as that in which 03/03/85 occurred) are strongly biased against M8 TIP's. With the unmodified Test A convention, the occurrence of a strong ($M \geq 7.5$) earthquake in effect resets the clock used to evaluate TIP's; a TIP cannot in principle be announced for a region until at least one year has elapsed following the strong earthquake, and any subsequent TIP's can be based only on anomalous seismicity occurring after the strong earthquake. With the proposed modification, the number of TIP-units in a five year run will inevitably increase, but we will not have situations of strong shocks occurring in unclassifiable space-time volumes that are neither ''TIP'' nor ''non-TIP''. Test B will be identical in every respect to the algorithm in Test A, except that a TIP is not terminated at the occurrence of an $M \geq 7.5$ earthquake.

Had Test B conventions been in force in our simulated forward prediction study of 1/1/85 - 7/1/91, the number of TIP-units would

have increased from 409 to 440, and M8 would have predicted 8 of the 10 strong earthquakes. Formal use of the same null hypothesis as used in Test A would suggest that Test B was statistically significant at a 97 percent level of confidence (Figure 3, Table 5). However, this estimate of confidence level cannot be considered statistically rigorous, because Test B was formulated after the data had been examined using the Test A conventions.

Table 5

Random Distribution of TIPs in the B-test

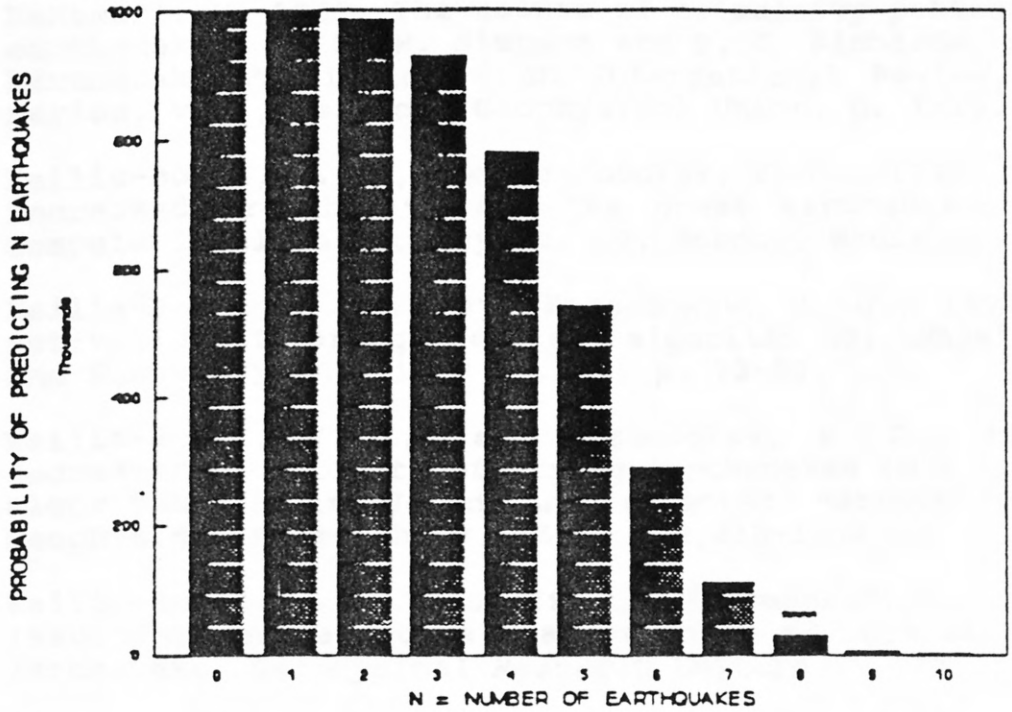
Number of quakes predicted	Distribution of predictions	Cumulative distribution of predictions	Level of confidence %	
0	1239	1000000	0.00	
1	12768	998761	0.12	
2	58081	985993	1.40	
3	148613	927912	7.21	
4	238153	779299	22.07	
5	252113	541146	45.89	
6	177789	289033	71.10	
7	82531	111244	88.88	
8	24350	28713	97.13	**
9	4105	4363	99.56	
10	258	258	99.97	

CONCLUSION

We have presented a description of a test of the earthquake prediction algorithm M8. Two arms of M8 Test are formulated, which we have denoted M8 Test A and M8 Test B. We have attempted to formulate M8 Tests A and B so that they are compatible with four general principles that we believe should be applied to any earthquake prediction test: we have documented the algorithms we will use and the conventions we will use, the algorithms are coded in common programming languages and run on commonly available computers, the tests will involve future predictions, and we have described the null hypothesis with which we will evaluate the performance of M8.

The future predictions issued in the course of our test will be research predictions. They are ''predictions'' in the sense that they are forecasts of time/space windows that have anomalously high probabilities of strong earthquakes and in the sense that their success or failure can be evaluated. They are ''research'' because they are generated for the purpose of testing the M8 algorithm and determining the algorithm's effectiveness.

Figure 3. M8 Test B, Number of earthquakes predicted in 1000000 random trys.



Most of the areas of investigation are known to include regions of high seismic hazard, irrespective of M8 results. It is important to periodically review seismic hazard in such regions. If a prediction resulting from this study is of concern, it would not be inappropriate to conduct a seismic hazard study tailored to the needs of the region.

ACKNOWLEDGEMENTS

This study was perfomed as a part of Project III of the US-USSR Cooperative Program in Earthquake Prediction, a program directed by the U.S. Geological Survey under the auspices of the Environmental Protective Agency. We gratefully acknowledge the helpful suggestions and criticisms of our colleagues, including B. Bakun, J. Dieterich, J. Eaton, B. Ellsworth, T. Hanks, V. Keilis-Borok, I. Kuznetsov, W. H. K. Lee, A. Lindh, A. McGarr, A. Michael, G. Molchan, P. Muffler, S. Nishenko, D. Oppenheimer, I. Primakov, P. Reasenberg, J. Savage, P. Shebalin, W. Spence, and J. Stock, P. Switzer, I. Johnstone, W. Thatcher. We thank P. Shebalin, in addition, for his programs on duplicate identification and aftershock removal and for his providing documentation of these programs for our Appendices II and III.

REFERENCES

Kanamori, H, 1981, The nature of seismicity patterns before large earthquakes, in D. W. Simpson and P. G. Richards, eds., Earthquake Prediction - an International Review, Maurice Ewing Series, v. 4, American Geophysical Union, p. 1-19.

Keilis-Borok, V. I., and Kossobokov, V. G., 1986, Time of Increased Probability for the great earthquakes of the world: Computational Seismology, v. 19, Moscow, Nauka, p. 48-58.

Keilis-Borok, V. I., and Kossobokov, V. G., 1990a, Premonitory activation of earthquake flow: algorithm M8: Physics of the Earth and Planetary Interiors, v. 61, p. 73-83.

Keilis-Borok, V. I., and Kossobokov, V. G., 1990b, Times of increased probability of strong earthquakes (M ≥ 7.5) diagnosed by algorithm M8 in Japan and adjacent territories: Journal of Geophysical Research, v. 95, p. 12,413-12,422.

Keilis-Borok, V. I., Knopoff, L., Kossobokov, V., and Rotvain, I., 1990, Intermediate-term prediction in advance of the Loma Prieta earthquake: Geophysical Research Letters, v. 17, p. 1461-1464.

Kossobokov, V. G., Keilis-Borok, V. I., and Smith, S. W., 1990, Localization of intermediate-term earthquake prediction: Journal of Geophysical Research, v. 95, p. 19,763-19772.

Mogi, K., 1981, Seismicity in western Japan and long-term earthquake forecasting: in D. W. Simpson and P. G. Richards, eds, Earthquake Prediction - An International Review, Maurice Ewing Series, v. 4, American Geophysical Union, p. 43-51.

Reyners, M., 1981, Long- and intermediate-term seismic precursors to earthquakes - state of the art: in D. W. Simpson and P. G. Richards, eds, Earthquake Prediction - An International Review, Maurice Ewing Series, v. 4, American Geophysical Union, p. 333 - 347.

Scholz, C. H., 1990, The mechanics of earthquakes and faulting: Cambridge University Press, 439pp.

Seismological Society of America, 1983, Guidelines for earthquake predictors: Bulletin Seismological Society America, v. 73, p. 1955-1956.

Updike, R. G., 1989, Proceedings of the National Earthquake Prediction Evaluation Council: U. S. Geological Survey Open-File Report 89-114, 25 pp. plus 9 appendices.

U. S. National Research Council, Panel on Earthquake Prediction of the Committee on Seismology, 1976, Predicting earthquakes, a scientific and technical evaluation - with implications for society: National Academy of Sciences, Washington, D. C., 62 pp.

Wallace, R. E., Davis, J. F., and McNally, K. C., 1984, Terms for expressing earthquake potential, prediction, and probability: Bulletin Seismological Society America, v. 74, p. 1819-1825.

APPENDIX I

The NEIC Earthquake Data Base System and M8

James W. Dewey
U. S. Geological Survey
Golden, Colorado

Stages of Cataloging Earthquakes at the NEIC

The National Earthquake Information Center (NEIC) routinely issues catalogs of instrumentally computed earthquakes within three different time-frames and at three levels of completeness. In order of increasing time-lapse from the occurrence of an earthquake and in order of increasing completeness, these are:

1. The "Quick Epicenter Determination" (QED) listing, issued within seven days of an earthquake. It is available to computers with dial-up capabilities..

2. The "Preliminary Determination of Epicenters (weekly listing), published within four weeks of the occurrence of an earthquake. In this report, this catalog is referred to as the PDE-weekly.

3. The "Preliminary Determination of Epicenters - Monthly Listing," published within about seven months of the occurrence of an earthquake. In this report, this catalog is referred to as the Monthly PDE.

The QED listings will not be used in M8 Test, for two reasons. First, the QED hypocenters and magnitudes are not stored, even temporarily, in the computer data-base system from which we will obtain the data that M8 will process. Second, the M8 functions depend on earthquakes of a magnitude for which the QED listings are significantly incomplete. The incompleteness is due to the relatively low percentage of world seismograph stations that send data in to the NEIC rapidly enough that the data can be used in the QED computation for a given earthquake. Neither the hypocenter-cataloging conventions of the NEIC nor the incompleteness of the QED listings are fundamentally unchangeable, however. Conceivably M8 could be applied to QED listings in the future.

The PDE-weekly listings typically contain about half the number of events that will be listed for the same time-period in the Monthly PDE. The incompleteness of the PDE-weekly listings varies geographically: many of the events not listed in the PDE-weekly are locally recorded events contributed by regional seismographic network operators, and for some regions of M8 Test the PDE-weekly listings are practically complete for earthquakes of the size that

are used in the M8 functions. The PDE-weekly hypocenters and magnitudes are stored only temporarily on the computer data-base system from which we will extract data for M8 Test; the PDE-weekly hypocenters and magnitudes are replaced in this data-base by those of the Monthly PDE when the latter become available. We will use PDE-weekly hypocenters and magnitudes in M8 Test for the time-periods during which they are stored on the computer data-base. Results obtained with M8 applied to time-periods containing PDE-weekly data will be considered provisional. Final scoring of M8 Test will be based on M8 applied entirely to Monthly PDE data, as these data are stored by the NEIC Earthquake Data Base System.

The Monthly PDE, in addition to listing more earthquakes than the PDE-weekly, also commonly lists slightly different hypocenters and magnitudes for earthquakes that were listed previously in the PDE-weekly. M8 functions based on the Monthly PDE may therefore differ from those based on the PDE-weekly even in regions for which the same events are listed in the PDE-weekly and Monthly PDE. The difference in magnitudes may cause an earthquake to change roles in M8, from being an object-of-prediction (M => 7.5) to not being an object-of-prediction, or vice-versa. Again, final scoring is based on the Monthly PDE, as these data are stored by the NEIC Earthquake Data Base System.

Both the PDE-weekly and Monthly PDE data, as they are stored by the NEIC Earthquake Data Base System, are somewhat abbreviated from the form in which they are distributed in print. The Earthquake Data Base System always stores the mb and MS magnitudes published by the NEIC, but it stores no more than two, additional, contributed magnitudes. For teleseismically recorded earthquakes, preference is given to storing the magnitudes computed by the seismographic stations at Berkeley (BRK) and Pasadena (PAS).

The data stored by the NEIC Earthquake Data Base System reside on a disk of a mini-mainframe computer at Golden, Colorado, from whence they are distributed as requested to members of the academic community, the private sector, and other governmental agencies. The data are updated weekly and monthly with hypocenters and magnitudes from respectively the PDE-weekly listings and the Monthly PDE listings. Besides the PDE data, the larger data-base contains data from numerous special catalogs. Some of these catalogs cover time-intervals for which PDE listings are nonexistent or incomplete; others present hypocenters and magnitudes that are intended to be improvements over those of the PDE. Hypocenters and magnitudes computed by the International Seismological Centre (ISC) are also stored in the larger data-base.

Some earthquakes whose epicenters and magnitudes were originally published in the PDE are not presently stored as PDE listings in the NEIC Earthquake Data Base System. Some of the "missing" earthquakes occurred as late as the early-1970's. Those PDE data-gaps of which we are aware correspond to geographic regions (such

as Central and Southern California) or individual exceptional earthquakes (such as the great 1964 Alaska earthquake) that are covered by non-PDE contributions in the data-base. We surmise that at early stages of the decades-long process of assembling the data-base, the compilers decided that the non-PDE contributions should be authoritative and that the PDE hypocenters and magnitudes did not need to be listed. The existence of the PDE-gaps means that we must include non-PDE contributions in our analysis: even though the M8 Test functions are only calculated for 1975 and later, some of the functions are influenced by seismicity occurring as early as 1963 (see Appendix IV).

The NEIC data-base for M8 Test

For analysis of seismicity occurring through 1988, we use a subset of the larger data-base that was published as a CD-ROM in 1989. This is the Global Hypocenter CD-ROM Data Base (Version 1.0), available from the NEIC. Hypocenters and magnitudes are retrieved from the CD-ROM by the program EPIC, which is supplied by the NEIC along with the CD-ROM. The CD-ROM contains the PDE data that are available on the larger data-base. The non-PDE contributions that cover the PDE gaps that we have identified are also on the CD-ROM. Unlike the larger data-base, the CD-ROM does not include the ISC hypocenters and magnitudes.

For analysis of seismicity occurring after 1988, we will use the hypocenters and the four or fewer magnitudes (NEIC mb and MS and two additional contributed magnitudes) that have been listed in the Monthly PDE and PDE-weekly and that are stored by the NEIC Earthquake Data Base System (EDBS). These data are accessed by EDBS software; this software is maintained by the NEIC, from whom documentation may be obtained.

In this paper, the term NEIC data-base refers to the CD-ROM data, for shocks occurring through 1988, plus the Earthquake Data Base System entries from the Monthly-PDE and PDE-weekly, for shocks in 1989 and later.

Conventions adopted in M8 Test for processing the NEIC data-base

We adopt the following conventions on data-selection for M8 Test:

Duplicate entries are identified by the procedure outlined in Appendix II. The hierarchy for selecting the preferred hypocenter/magnitude from a set of multiple entries is also given in that Appendix.

Magnitudes used in M8 Test will be the largest of as many as four magnitudes that may be listed with a particular earthquake entry in the NEIC data-base. If there are multiple entries for a given earthquake, the preferred entry is first selected, then the

magnitude taken as the maximum value from the magnitudes listed in the preferred entry.

Explosions and other human-induced events are not removed from the data base. Human-induced earthquakes will in general be a source of noise in M8 Test, though it is conceivable that some types of human-induced activity might reflect changes in regional stress in the same way that natural seismicity is postulated to reflect changes in regional stress prior to large earthquakes.

APPENDIX II

AUTOMATIC DUPLICATE IDENTIFICATION IN A SET OF EARTHQUAKE CATALOGUES MERGED TOGETHER (detailed description of the algorithm and program DUPLI_VX, version 2, 1991)

Peter N. Shebalin[1]

1. Introduction

The use of information from several earthquake catalogues is often more preferable for many seismological studies. For example, if we decide to work with hypocentral data from the International Seismological Centre (ISC), to obtain information about surface-wave magnitudes (MS), we need to refer to other data sources as ISC has published MS information on a routine basis starting in 1978 only. The other example is a study of a territory covered by two or more independent seismological networks.

When the data sets that are used are not large, the correspondence of different sources can be determined easily by hand, otherwise, it is natural to automatize this process. The simplest way to make this determination is to define the time-space "window" and to consider data from different sources within this window as a "duplicate". But our experience shows that discrepancies in epicentre location by different agencies, even recently, can exceed hundreds of kilometres. Discrepancies in time are often larger than the time difference of two near seismic events. So, the "window" approach will lead to inevitable errors of one or both types: false and missed duplicates. Our estimation of the number of such errors, depending on the sizes of the window, is at least 1/20 of the number of all duplicated events with magnitude $M \geq 4$ in worldwide catalogues from 1960 to 1980. Frequency of errors is higher for older data because of the rather significant number of errata in publications prepared without the help of a computer and the errors in their translation into computer-readable form.

In this paper a more complicated algorithm for automatic duplicates determination in several earthquake catalogues is presented. This algorithm tests a large set of possible errata and errors leading to large discrepancies of origin time and epicentre coordinates in different catalogues. It also allows the presence of duplicates in separate catalogues.

General ideas of the algorithm have been published in (Shebalin, 1987). The new version is significantly improved.

---

[1] International Institute for Earthquake Prediction Theory and Mathematical Geophysics, Russian Academy of Sciences. Address: Warshavskoye sh., 79-2, 113556 Moscow, RUSSIA. Phone: +7-095-1104848; Fax: +7-095-3107032.

The program DUPLI_VX for MS-DOS C compiler, version 5.1, uses the algorithm for data in NEIC/USGS VX-format. VX-format is used for NEIC/USGS CD-ROM Earthquake Hypocentres database and its updates.

2. Algorithm for identification of duplicates

The main idea of the algorithm is to model the manual analysis of data. This is based on the author's experience in finding duplicates with the aid of a computer in worldwide and regional earthquake catalogues that merged together.

How did we do it? First of all, the merged set of catalogues were ordered by time increase. Then, starting with the first event we found the "candidates" to be duplicates for each event not yet processed. All candidates were united into subsets. The possible duplicates of each element of the subset were also added to it, and so forth, until all "candidates" to "candidates" were exhausted. Then we analyzed the subset. The evident duplicates were combined into sub-subsets. Some sub-subsets could also be united, providing they did not contain obviously different events. Finally, the rest of the subset elements were associated with some sub-subsets or treated as separate events. To do that, we used some intuitive notion about the degree of "duplicateness" for each pair of subset elements.

The trial to formalize such degrees of "duplicateness" is made in paragraph 2.2 below. The formalization of the method used to split subsets of possible duplicates into several clusters of elements corresponding to the same events is described in paragraph 2.3. The manual work on detection of duplicates has two important results.

First, we discovered that the information contained in the catalogues is enough to identify duplicates in an overwhelming majority of cases, although we used station bulletins whenever possible. This has encouraged us to construct the algorithm.

Second, we found the most widely distributed types of errata and errors leading to significant time and epicentre location discrepancies, and we found some specific features of the data, which could help to identify duplicates. This gave us the basis for the formalized "degree of duplicateness".

2.1. Terminology and the task

We consider the catalogues in the form of sequential files with fixed record length sorted in time order. We shall use the terms "string" or "record" to designate the set of earthquake parameters: origin time, epicentre coordinates, focus depth, magnitude (sometimes several types), epicentral intensity and some additional information (e.g. characteristics of accuracy of the values of

parameters, their reliability, indications on non-seismic events, etc.).

Some catalogues list data about the same event from different sources. Such catalogues a priori contain duplicates. Using double abbreviation of catalogue/data source we can avoid this situation. Further, we shall use the term "catalogue" to indicate data sets with coinciding double abbreviations.

All records of all catalogues under consideration form the set $W = (w_i)$, $i = 1,\ldots,I$, I is their total number.

The task is to divide the set W in advance into unknown number J of non-overlapping subsets $D = (D_j)$, $j=1,\ldots,J$ of duplicates, so that the elements of one subset correspond to the same event, and the elements of different subsets correspond to different events. Of course, some quantity of errors of duplicates identification is inevitable.

2.2. Proximity function

To formalize the "degree of duplicateness" the real non-negative function $\rho(w,w')$ is defined for each pair (w,w') of records. Its larger values should correspond to more likely duplicates, so that the largest values correspond to evident duplicates, and the smallest values correspond to evident separate events.

The causes of duplicates in one separate catalog and in different catalogs are principally different. In the first case, they are not normally expected and occur occasionally. Accordingly, the definition of function ρ is different for those two cases. This is an important difference in the new version of the algorithm. In the earlier version (Shebalin, 1988) each separate catalog was assumed to be "cleaned" manually: now, catalogues which contain duplicates also can be processed.

The formal definition of proximity function $\rho(w,w')$ follows:

$$\rho(w,w')=C_I(w,w')*\max(R_{sp}(w,w'),\ C(w,w')*\max(R(w,w'),R_t(w,w'))) \qquad (1)$$

$C_I(w,w')=0.75$ if the intensity is known in both records w and w', and the difference is more than 2 units, and $C_I(w,w')=1$ otherwise.

$R_{sp}(w,w')=\max(r_j^{sp}(f_t,f_r,f_m,\delta(w,w')\cdot T_j^{sp}(w,w'))$, where $T_j^{sp}(w,w')$ is the logical value (0 or 1) of the result of the j-th test of possible error in epicentre coordinates, where 4 tests are applied; f_t, f_r, f_m are indices that correspond to the difference in time, epicentre coordinates, and magnitude in w and w' (see 2.2.1 - 2.2.3);

$\delta(w,w')$ is equal to 0 if the data source for the whole record and the source for time and epicentre are the same in w and w', and is equal to 1 otherwise;

$r_j^{ep}(f_t,f_r,f_n,\delta)$ represents the parameters of the algorithm.

$R_{ep}(w,w')$ is responsible for possible errors in the epicentre coordinates in w or w'.

$R(w,w')=r(f_t,f_r,f_n,\delta(w,w'))$, where $r(f_t,f_r,f_n,\delta)$ represents the parameters of the algorithm. $R(w,w')$ is responsible for "normal" discrepancies in w and w' (in significant or no errors in time and epicentre).

$R_t(w,w')=\max(r_j^t(f_t,f_r,f_n,\delta(w,w'))\cdot T_j^t(w,w'))$, where $r(f_r,f_n,\delta)$ are the parameters of the algorithm;

$T_j^t(w,w')$ is the result of i-th test of possible error in origin time of quake, where 8 tests are applied (see 2.2.4).

$C(w,w')$ is the coefficient obtained from additional tests of the pair (w,w') (see 2.2.6).

2.2.1. Definition of $f_t(w,w')$

$f_t(w,w')$ in (1) is the index corresponding to the value of the discrepancy in time determination in records w and w'. If time is given within an accuracy of seconds in both records, f_t is simply the interval number to which the direct time difference $dt(w,w')$ in w and w' belongs. If the origin time in one or both records is given with less accuracy, the determination of $f_t(w,w')$ is more complicated. The function $dt1(w,w')$ is used instead of the direct time difference $dt(w,w')$; its values may be less, equal, or larger than those of dt. (The formal definition of function dt1 is given below.) Finally, $f_t(w,w')$ is the interval number to which the value of $dt1(w,w')$ belongs. The boundaries of the intervals are fixed separately as the parameters of the algorithm for $\delta(w,w')$ equal to 0 and to 1 (same and different data sources). The value belongs to the interval if it is grater or equal to the left boundary value and smaller than the value of the right boundary. Values of thresholds correspond to the right boundaries of the intervals; the left of the first interval is zero.

The following cases of time representation in each record are considered:

1) Time is given with accuracy of seconds;
2) No seconds - seconds are blanked or given as 00 or 00.0;
3) No minutes (blanked);
4) No hour (blanked);
5) No month or no day (blanked).

For each combination of cases in w and w' dt1(w,w') is defined by the appropriate sub-function:

	1	2	3	4	5
1	dt	dt_m	dt_h	dt_d	dt_{mo}
2	dt_m	dt*	dt_h	dt_d	dt_{mo}
3	dt_h	dt_h	dt*	dt_d	dt_{mo}
4	dt_d	dt_h	dt_d	dt*	dt_{mo}
5	dt_{mo}	dt_{mo}	dt_{mo}	dt_{mo}	dt*

The following tables contain the definitions of those sub-functions. In the left columns of the tables the conditional expressions are listed in the in which order they are checked. If a higher condition is true, then the value from the right column of the table is chosen.

da and da', hr and hr', and mn and mn' are the values of dates, hours, and minutes in w and w'.

d, h, and m are integer numbers which minimize the left parts of the corresponding expressions.

Determination of $dt_m(w,w')$

Condition	Value, s
dt(w,w') = 0	0
dt(w,w') < 60	5
abs(dt-h*3600)<60,h>=1:	
a) mn = mn'	h*3600+8
b) else	h*3600-20
abs(dt-h*3600) ≥ 210 and mn=30 or mn'= 30	
a) abs(dt-h*3600) < 300	h*3600+90
b) abs(dt-h*3600) < 450	h*3600+120
c) abs(dt-h*3600) < 600	h*3600+180
d) abs(dt-h*3600) < 900	h*3600+240
e) abs(dt-h*3600) < 1200	h*3600+300
f) abs(dt-h*3600) ≥ 1200	dt(w,w')
abs(dt-h*3600)>=150 and mn or mn' equal to 15 or 45:	
a) abs(dt-h*3600) < 180	h*3600+90
b) abs(dt-h*3600) < 300	h*3600+120
c) abs(dt-h*3600) < 450	h*3600+240
d) abs(dt-h*3600) < 600	h*3600+300
e) abs(dt-h*3600) ≥ 600	dt(w,w')
abs(dt-h*3600) ≥ 150 and mn or mn' equal to 10, 20, 40 or 50:	
a) abs(dt-h*3600) < 180	h*3600+120
b) abs(dt-h*3600) < 300	h*3600+150
c) abs(dt-h*3600) ≥ 300	dt(w,w')
abs(abs(dt-h*3600)-m*60) < 30	h*3600+m*60-20
abs(abs(dt-h*3600)-m*60) ≥ 30	h*3600+m*60+20

Determination of $dt_h(w,w')$

Condition	Value, s
$dt(w,w') - h * 3600 = 0$	$dt + 150$
$dt - h * 900 = 0$:	
a) $abs(dt-d*86400) = 900$	$d * 86400 + 150$
b) $abs(dt-d*86400) = 1800$ or 2700	$d * 86400 + 240$
c) else	dt
$abs(dt - d * 86400) < 3600$, $d >= 1$:	
a) $abs(dt - d * 86400) < 60$	$d * 86400 + 10$
b) $abs(dt - d * 86400) < 180$	$d * 86400 + 40$
c) $abs(dt - d * 86400) < 960$	$d * 86400 + 90$
d) $hr = hr'$	$d * 86400 + 120$
e) $abs(dt - d * 86400) < 2110$	$d * 86400 + 150$
f) $abs(dt - d * 86400) \geq 2110$	$d * 86400 + 300$
$hr = hr'$:	
a) $dt \leq 180$	10
b) $dt \leq 300$	30
c) $dt \leq 600$	60
d) $dt \leq 900$	90
e) $dt \leq 1800$	150
f) $dt > 1800$	180
$abs(dt - h * 3600) < 180$	$h * 3600 + 100$
$abs(dt - h * 3600) < 300$	$h * 3600 + 120$
$abs(dt - h * 3600) < 600$	$h * 3600 + 150$
$abs(dt - h * 3600) < 900$	$h * 3600 + 180$
$abs(dt - h * 3600) \geq 900$	$h * 3600 + 240$

Determination of $dt_{mo}(w,w')$

Condition	Value, s
year and month are the same in w and w' or month is blanked in w or w'	600
other	1000000

Determination of dt*(w,w')

Condition	Value, s
dt(w,w') = 0 :	
a) month is blanked	150
b) day is blanked	120
c) hour is blanked	90
d) other	0
dt = h * 3600 + 0 and mn or mn1 = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 or 55:	
a) month is blanked	dt + 440
b) day is blanked	dt + 340
c) hour is blanked	dt + 240
d) minute is blanked	dt + 140
e) other	dt + 40
else	
a) month is blanked	dt + 500
b) day is blanked	dt + 380
c) hour is blanked	dt + 260
d) minute is blanked	dt + 140
e) other	dt + 20
dt < 7200	dt = 30
dt ≥ 7200	dt = 40

2.2.2. Definition of $f_r(w,w')$

$f_r(w,w')$ in (1) is the index corresponding to the value of the discrepancy in the epicentre coordinates in records w and w'. Distance dr(w,w') in degrees on the Earth's surface between the points given by the epicentre coordinates in records w and w' is calculated. If a fractional part of both the latitude and longitude in w or w' is equal to zero, then the value of dr is diminished. Formally, if φ, φ', λ, and λ' are latitudes and longitudes in w and w':

$$dr = (\delta\varphi^2 + \delta\lambda^2 * \cos^2(\varphi/2+\varphi'/2))^{1/2},$$

where

$\delta\varphi$ = min {abs(φ-φ'),abs(abs(φ-φ')-0.5)},

$\delta\lambda$ = min {abs(λ-λ'),abs(abs(λ-λ')-0.5)}, if a fractional part of φ and λ or φ' and λ' is equal to zero, and

$\delta\varphi$ = abs (φ-φ'), $\delta\lambda$ = abs (λ-λ') otherwise.

2.2.3. Definition of $f_m(w,w')$

Magnitude values from different sources for the same event may be very different depending on the type of magnitude and its data source. To calculate the magnitude index $f_m(w,w')$ we consider three cases:

1) values of the same type and the same source exist in w and w';

2) values of the same type but from different sources exist in w and w';

3) no values of magnitude of the same type exist in w and w'.
(The actual case number defines the value of dm_case(w,w').)

$f_x(w,w')$ is the interval number to which the difference of magnitudes belongs. In each case the lowest possible difference in magnitude value is considered. Intervals boudaries are given separately for the different values of d(w,w') and dm_case(w,w'). Three intervals are fixed. If no magnitude data exists in w or w', the value of $f_x(w,w')$ is assigned to 2 (the largest value of M in w and w' is larger than the parameter value M_big). Otherwise, $f_x(w,w') = 0$.

2.2.4. Tests of possible errors in time, $T_3^t(w,w')$

2.2.4.1. Local time, $T_1^t(w,w')$

This test of the possible use of local time is applied only if $dt1(w,w') \geq 6000$ and if $t1 < dt1(w,w') < t2$, where

$t1 = (\min (abs(\lambda)/15, abs(\lambda')/15) - 2) * 3600,$
$t2 = (\max (abs(\lambda)/15, abs(\lambda')/15) + 3) * 3600,$ and
λ and λ' are the epicentral longitudes in w and w'.

Otherwise, the value of $T_1^t(w,w')$ is 0.

The minimal value $\delta t = abs(dt1(w,w') - h * 3600)$ is determined, there h is an integer. Finally,

$T_1^t = 2.3$, if $\delta t \leq t\_err_1(\delta(w,w'))/15$,
$T_1^t = 1.7$, if $\delta t \leq t\_err_1(\delta(w,w'))/6$,
$T_1^t = 1.0$, if $\delta t \leq t\_err_1(\delta(w,w'))$,
$T_1^t = 0.7$, if $\delta t \leq t\_err_1(\delta(w,w'))*2$,
$T_1^t = 0$, if $\delta t > t\_err_1(\delta(w,w'))*2$,

where $t\_err_1(\delta(w,w')$ is the parameter of the algorithm.

2.2.4.2. Incorrect hour, $T_2^t(w,w')$

This test of possible inaccurate translation of local time to GMT is applied only if $dt1(w,w') \geq 3500$ and the minutes are not blanked in both records. $T_2^t = 1$, if a positive integer h exists, so that $dt1(w,w') = h * 3600$, $h < 16$.

2.2.4.3. Possible error in hour, $T_3^t(w,w')$

The most common reason for errors in the hour is inaccurate translation of local time to GMT.

The minimal value δt = abs(dt1(w,w') - h * 3600) is determined, where h is equal to 1 or 2.

T_3^t = 2.3, if $\delta t \leq t\_err_b(\delta(w,w'))/15$,
T_3^t = 1.7, if $\delta t \leq t\_err_b(\delta(w,w'))/6$,
T_3^t = 1.0, if $\delta t \leq t\_err_b(\delta(w,w'))$,
T_3^t = 0.7, if $\delta t \leq t\_err_b(\delta(w,w'))*2$,
T_3^t = 0, if $\delta t > t\_err_b(\delta(w,w'))*2$,

where $t\_err_b(\delta(w,w')$ is the parameter of the algorithm.

2.2.4.4. Possible error in day or month, $T_4^t(w,w')$

If the dates in w and w' are different then the minimal value δt = abs(dt1(w,w') - d * 86400) is determined, where h is equal to 1 or 2.

T_4^t = 2.3, if $\delta t \leq t\_err_d(\delta(w,w'))/15$,
T_4^t = 1.7, if $\delta t \leq t\_err_d(\delta(w,w'))/6$,
T_4^t = 1.0, if $\delta t \leq t\_err_d(\delta(w,w'))$,
T_4^t = 0.7, if $\delta t \leq t\_err_d(\delta(w,w'))*2$.

If the dates are the same and the hours are not blanked in both w and w', and the month values are different in w and w', then the month value in w is substituted by the value in w', and δt = dt1(modified w,w') is calculated.

Finally,

T_4^t = 1.7, if $\delta t \leq t\_err_{mo}(\delta(w,w'))/6$,
T_4^t = 1.0, if $\delta t \leq t\_err_{mo}(\delta(w,w'))$,
T_4^t = 0, if $\delta t > t\_err_{mo}(\delta(w,w'))$,

where $t\_err_{mo}(\delta(w,w'))$ is the parameter of the algorithm.

2.2.4.5. Possible error in minute, $T_5^t(w,w')$

This test is applied if the seconds are not blanked and are not equal to 0 in both w and w'. The value of T_5^t is equal to 1 if a valid modification of the first digit in w or w' exists, such that dt1(modified strings) $\leq t\_err_s(\delta(w,w'))$, and equal to 0 otherwise, where $t\_err_s(\delta(w,w'))$ is the parameter of the algorithm.

2.2.4.6. Possible reversal of two digits in time, $T_6^t(w,w')$

This test is applied if dt1(w,w') $\geq$ 300. The value of $T_6^t(w,w')$ is equal to 1 if a valid transposition of two neighbouring digits in the time field of w or w' can be found, such that dt1 is less

than or equal to the value of the parameter t_fit(j,δ(w,w')), and is equal to 0 otherwise. j is the number of the first transposed digit counting from the first digit of the month.

2.2.4.7. Possible reversal of two numbers in time, $T_7^t(w,w')$

This test is applied if dt1(w,w') $\geq$ 300. The value of $T_7^t(w,w')$ is equal to 1 if a valid transposition of two neighbouring two-digit numbers in the time field (month, day, hour, minute) of w or w' can be found, such that dt1 is less than or equal to the value of the parameter t_fit(j,δ(w,w')), and is equal to 0 otherwise. j is the number of the first digit of the transposed number counting from the first digit of the month.

2.2.4.8. Possible omission or insertion of one digit in time, $T_8^t(w,w')$

This test is applied if dt1(w,w') $\geq$ 300. The value of $T_8^t(w,w')$ is equal to 1 if a valid insertion of a digit into the time field of w or w' can be found, shifting the rest of the digits to the right, such that dt1 is less than or equal to the value of the parameter t_fit(j,δ(w,w')), and is equal to 0 otherwise. j is the number of the position where the digit was inserted, counting from the first digit of the month.

2.2.5. Tests of possible errors in epicentre, $T_3^{ep}(w,w')$

2.2.5.1. Incorrect sign of latitude or longitude, $T_1^{ep}(w,w')$

This test is applied if $f_t(w,w') < 5$ (not too large a difference in time). In w and w'the latitude and longitude of the epicentre are substituted by northern latitudes and eastern longitudes, respectively, and the value δr is obtained as a value of the function dr modified strings.

$T_1^{ep}(w,w') = 1$, if dr(w,w') > 1.0 and $\delta r < 0.5$,
or if dr(w,w') > 5.0 and $\delta r < 2.0$,
or if dr(w,w') > 10.0 and $\delta r < 3.0$,
$T_1^{ep}(w,w') = 0$, otherwise.

2.2.5.2. Possible format error in epicentre field, $T_2^{ep}(w,w')$

The value of $T_2^{ep}(w,w')$ is equal to 1 if a valid modification of w or w' can be founded by multiplying the values of the latitude or longitude by the factor of 10 or 100, such

that the value of the function dr after modification is less than 0.5; otherwise its value is 0.

2.2.5.3. Possible omission of one digit in epicentre field, $T_3^{ep}(w,w')$

The value of $T_3^{ep}(w,w')$ is equal to 1 if a valid modification of w or w' can be found by inserting one digit into the latitude or longitude field, such that the value of the function dr after modification is less than 0.5, if modification was made in the latitude field and is less than 0.3 in longitude field; otherwise, its value is 0.

2.2.5.4. Possible error in one digit in epicentre field, $T_4^{ep}(w,w')$

This test is applied if $dr(w,w') \geq 2.0$. The valid modification of the latitude and longitude (integer portion of the values only) in w and w' is made: the digit is replaced by a corresponding digit from the other string and also is increased or decreased by 1, and the minimal value of dr after modification is found. The value of $T_4^{ep}(w,w')$ is equal to 1 if the modified value is less than 2.0 and refers to the tens of a degree, and is equal to 0.7 when it refers to the integer units of a degree. Otherwise, the value of T_4^{ep} is 0.

2.2.6. Definition of C(w,w'). Additional tests

$$C(w,w') = C_0(w,w') * \max_{j=1,..,7} (C_j(w,w'),$$

$$C_j(w,w') = c_j(d(w,w')^{Ta_j(w,w')},$$

where $c_j(d(w,w'))$, $j=0,1,...,7$ are parameters of the algorithm, $Ta_j(w,w')$ - additional tests described below.

2.2.6.0. $Ta_0(w,w')$

This test is different for different values of $\delta(w,w')$.

For $\delta(w,w') = 1$ (different data sources in w and w') the character comparison of the fields of time, epicentre, focus depth and magnitudes is made. If the difference is in one or two of these characters, then $Ta_0(w,w') = 1$; otherwise, it equals 0.

For $\delta(w,w') = 0$ (same data sources in w and w') $Ta_0(w,w') = 1$ if

the abbreviation of the data source is in the list of data sources, which normally may contain duplicates.

The following tests are applied only if $\delta(w,w') = 0$.

2.2.6.1. Different number of stations, $Ta_1(w,w')$

This test is applied if the number of stations used for epicentre determination is given in both w and w'. The result is equal to 1 if the difference in the number of stations is more than 5 or if the minimum number of stations is less than 10, and the difference is more than 3. Otherwise, the result is equal to 0.

2.2.6.2. No number of stations in one string, $Ta_2(w,w')$

The result of this test is equal to 1 if in one string (w or w') the number of stations used for epicentre determination is not given; otherwise, the result is equal to 0.

2.2.6.3. Unreliable data, $Ta_3(w,w')$

The result of this test is equal to 1 if in w or w' the flag corresponding to an unreliable epicentre or origin time determination is turned on; otherwise, the result is equal to 0.

2.2.6.4. Poor accuracy of time/epicentre, $Ta_4(w,w')$

The result of this test is equal to 1 if in w or w' the quality indicator for time and epicentre determination corresponds to "poor" data or accuracy worse than 50 km for epicentre and 10 s for time, or if the number of stations is given, but is less than 7; otherwise the result is equal to 0.

2.2.6.5-7. Possible redetermination of depth

$Ta_5(w,w')$ is equal to 1 if in one string the value of depth is assigned a value of 0, 25, or 33, or a special flag is turned on, and in the other string another value of depth is given; otherwise, $Ta_5(w,w')$ equals 0.

The following tests are applied only if the focus depth values are given in both w an w', and if they are not assigned; otherwise; their results are equal to 0.

$Ta_6(w,w')$ is equal to 1 if in one string the value of depth is less than 50 km, and in the other it is greater than 50 km.

$Ta_7(w,w')$ is equal to 1 if the focus of depth values in w and w' are not equal.

2.2.7. Evident duplicates.

In some cases it is evident, that the two strings are duplicates. To avoid long tests in these cases special values of function $\rho(w,w')$ are assigned. Those values are also parameters of the algorithm:

Condition	Value of ρ
$f_t = 0$, $f_M < 3$, $f_r < 5$	ev1
$f_t = 0$, $f_M < 3$, $f_r = 5$	ev2
$\delta(w,w') = 1$, $f_t = 0$, $f_r < 4$	ev3
$\delta(w,w') = 1$, $f_t < 2$, $f_r < 3$, $f_M < 3$	ev4
$dt1(w,w') = 0$, $dr(w,w') < drmax$	ev5
$f_t = 0$, $dr(w,w') < drmax$, $f_M < 2$	ev6
$\delta(w,w') = 1$, $dt(w,w') = 0$, $f_r < 5$, $f_M < 4$	ev7

where drmax is the value of the largest threshold for f_r determination multiplied by 5/3.

2.2.8. Time limit for duplicates search

To diminish calculation time function $\rho(w,w')$ is assigned a negative value if the difference in origin time in w and w' is more than the value of the parameter dtmax.

2.2.9. Determination of the parameters of the algorithm.

All parameters can be different for different periods of time in the catalogs. All "threshold" parameters are chosen empirically.

Parameters r_j, r_j^{ep}, r_j^t, and c_j are determined on the basis of a learning set. A learning set is formed from actual catalogs merged together so that, if possible, each combination of values of indices ft, fr, and fm will occur several times, and also all tests T will be applicable with different values of corresponding

indices. Duplicates are marked in the learning set manually.

The techniques for the determination of parameters is beyond the scope of this paper. General ideas are found in (Shebalin, 1987).

The parameter values used for the program DUPLI_VX and determined for the set of catalogs from the database on CD-ROM, produced by NEIC/USGS, are listed in Appendix 1.

2.3. Cluster of duplicates

Studying the values of proximity function for pairs in the learning set, we can specify two rough pairs of thresholds:

$$\Delta_1(\delta(w,w') \text{ and } \Delta_2(\delta(w,w').$$

Values of $\rho < \Delta_1$ correspond to "most likely" duplicates and values of $\rho > \Delta_2$ correspond to obviously different events. Later on those values can be determined more precisely.

Set W of catalog records is divided into subsets of possible duplicates constructed pair by pair with proximity function values less than Δ_1, i.e., "most likely" duplicates. Thus, a record is considered as a possible duplicate or an event only if it has at least one twin with $\rho < \Delta_1$ belonging to this same event. The subsets can include pairs with $\rho > \Delta_2$ corresponding to different earthquakes. In that case, the subset must be split into two or more parts, each part corresponding to one separate event, and within each part, all pairs will have values of $\rho \leq \Delta_2$. It is natural to force any element within such a part to be "closer" in terms of proximity function to other elements of the part, rather than to any element outside that part. We call such parts "clusters of duplicates". We do not require that the maximum value of proximity function inside one cluster be less than the value for any pair formed by one element inside and another outside the part, as is common in clustering analysis. Otherwise, the erroneous records, which have larger values of proximity function with their duplicates in comparison with "normal" duplicates, could influence the occurrence of extra clusters corresponding to false events.

The following is the formal definition of cluster of duplicates.

Definition 1. Subset G belonging to the set W is called Δ_1-linked if for any two of its elements w' and w" it is possible to find elements $w_1, w_2, \ldots, w_s$ also belonging to G: $w'=w_1, w"=w_s$ and $\rho(w_i, w_{i-1}) \leq \Delta_1$ for all $i=1,2,\ldots,s-1$.

Definition 2. Subset G′ belonging to the set W is called Δ_2-limited if for any two of its elements w′ and w" $\rho(w',w") < \delta_2$.

Definition 3. Subset K belonging to the set W is called cluster if for any three elements of W: w, w′ belonging to K and w" not belonging to K

$$\rho(w,w')/\Delta_1(\delta(w,w')) < \rho(w,w")/\Delta_1(\delta(w,w")).$$

Main definition. Subset D belonging to the set W is called cluster of duplicates if it is 1) a cluster, 2) Δ_1-linked, 3) Δ_2-limited, and 4) if a larger subset of W that includes D with the same features does not exist.

It is easy to show that clusters of duplicates do not overlap and that the subdivision of the set W into clusters is unique.

Values of parameters Δ_1 and Δ_2 used in program DUPLI_VX are listed in Appendix 1.

3. Program DUPLI_VX

The program DUPLI_VX is the realization of the algorithm described for data in the format VX. Format VX is used for catalogs in NEIC/USGS CD-ROM Earthquake Hypocentres database and PDE and the final catalog produced by NEIC/USGS.

The program is written for MS-DOS C compiler, version 5.1. All values of parameters are fixed inside the code of program. Values of parameters are different for the two different time periods: data through 1964 and data since 1965. The program automatically choses the appropriate set of values. Parameter values for both time periods are listed in Appendix 1.

Input for the program is the set of catalogs in VX-format merged together and sorted by increase of time.

The output is in the same format. The 116-th byte of the format is used to mark duplicates. If only one string (data from one catalog) corresponds to the event, the value of 116-th byte is blank. If several strings correspond to one event, all those strings (duplicates) follow each other in the output, even if they initially have been separated by other strings (e.g., due to incorrect date in one string). Inside the cluster of duplicates the strings are sorted by increase of earthquake origin time. The string with the earliest time is marked in 116-th byte by the symbol "+"; all others are marked by number: 1,2,...,9,0,1,2, etc. All first strings of clusters of duplicates keep the order of time increase.

4. Application to NEIC/USGS CD-ROM Earthkuake Hypocentre database and its updates

The program DUPLI_VX has been applied to all data on CD-ROM from 1900 to 1988 and its updates for 1989 through June 1991. Data from CD-ROM have been retrieved by EPIC software with database duplicates removal option switched off.

The total number of data strings is 457340. They have been subdivided by the program into 389406 clusters of duplicates.

Examples of duplicates found by the program as well as erroneous results are given in Appendix 2.

Following tables show the distribution of the difference in time and space for duplicates and separate events for several time intervals. The tables show that any simple algorithm of duplicate identification which uses time/space thresholds only would give a very significant number of not-found duplicates and/or false duplicates.

1900 - 1927
18107 records and 13092 groups of duplicates

DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	1560	99	191	118	73	35	24	10	39	248
50 km	108	10	20	11	7	3	2	2	2	19
75 km	115	12	33	12	1	1	4	0	0	12
100 km	61	26	27	12	2	0	0	0	0	9
125 km	42	13	21	7	3	2	1	0	1	14
150 km	18	11	9	1	0	0	1	0	1	3
175 km	24	11	4	2	1	0	1	0	1	3
200 km	10	12	12	9	1	2	0	0	0	4
>200km	90	36	48	27	12	7	3	4	5	69

NON - DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	4	0	0	10	96	244	345	671	865	749
50 km	0	0	0	0	4	26	36	52	44	73
75 km	1	0	0	5	7	15	27	18	69	70
100 km	0	0	0	0	1	6	5	19	23	42
125 km	0	0	0	0	1	10	8	9	8	46
150 km	0	0	0	0	1	4	4	8	14	33
175 km	0	0	0	0	0	1	9	9	7	14
200 km	2	1	0	1	8	6	5	11	2	26
>200km	46	6	14	24	70	155	214	423	736	4007

Comment: EPIC software with duplicate removal option switched on (thresholds 99 s in time and 99 km in space) gives 14229 events (approximately 8.7% of the events are duplicates not removed).

1928 - 1948
29307 records and 22583 groups of duplicates

DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	1777	126	108	94	57	7	4	4	19	239
50 km	407	46	67	15	4	5	4	0	2	23
75 km	335	27	36	3	5	2	0	0	3	18
100 km	182	19	17	4	2	2	1	0	1	10
125 km	113	21	12	3	1	2	0	0	0	7
150 km	71	8	4	1	1	0	3	0	1	4
175 km	68	19	4	2	1	0	0	0	2	6
200 km	30	12	6	1	0	0	2	0	1	6
>200km	182	83	32	10	8	6	4	3	8	130

NON - DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	1	4	3	46	257	492	727	1115	1561	738
50 km	0	0	0	3	10	19	82	90	138	150
75 km	0	0	0	0	2	11	30	76	90	97
100 km	0	0	0	0	1	8	11	34	49	40
125 km	0	0	1	0	2	6	17	22	54	28
150 km	0	0	0	0	2	4	15	19	36	23
175 km	0	0	0	0	2	4	10	20	45	19
200 km	0	0	0	0	0	1	14	12	14	19
>200km	30	17	47	77	137	277	474	1112	2135	5635

Comment: EPIC software with duplicate removal option switched on (thresholds 99 s in time and 99 km in space) gives 23963 events (approximately 6.1% of the events are duplicates not removed).

1979 - 1981
29288 records and 24167 groups of duplicates

DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	4114	23	21	24	14	2	0	0	3	74
50 km	369	3	3	7	5	3	0	0	0	8
75 km	70	0	2	0	3	0	0	0	0	1
100 km	32	1	2	1	0	1	0	0	1	1
125 km	20	5	0	0	0	0	0	0	0	0
150 km	4	1	0	1	0	0	0	0	0	0
175 km	1	5	0	1	0	0	0	0	0	0
200 km	0	6	1	2	0	0	0	0	0	0
>200km	2	7	7	1	0	0	0	0	0	2

NON - DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	1	5	11	76	312	491	791	1428	2383	344
50 km	1	0	2	2	39	88	191	305	573	107
75 km	0	0	1	4	20	36	65	129	226	35
100 km	0	0	1	1	7	17	31	41	103	32
125 km	0	0	0	0	5	9	16	22	41	14
150 km	0	0	0	0	0	6	7	24	41	12
175 km	0	0	0	1	0	4	3	15	31	10
200 km	0	0	0	0	0	6	14	31	36	21
>200km	99	116	222	422	928	1983	3447	7740	15766	4548

Comment: EPIC software with duplicate removal option switched on (thresholds 60 s in time and 99 km in space) gives 24372 events (approximately 0.8% of the events are duplicates not removed).

1982 - 1988
83783 records and 78002 groups of duplicates

DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	4477	36	68	86	46	13	1	1	30	172
50 km	356	9	12	22	16	12	0	0	4	25
75 km	74	1	4	10	3	0	1	0	2	16
100 km	30	2	0	2	1	1	0	0	2	5
125 km	10	4	2	5	2	1	0	0	0	2
150 km	5	4	1	2	1	0	0	0	0	1
175 km	1	3	0	3	0	0	0	0	0	1
200 km	1	2	1	2	0	0	0	0	1	2
>200km	10	7	10	6	0	0	0	0	1	15

NON - DUPLICATES

	15 s	30 s	1 m	2 m	4 m	8 m	15 m	30 m	1 h	> 1 h
25 km	4	7	47	226	580	1227	1686	2697	4036	430
50 km	0	1	7	24	96	219	355	713	1206	108
75 km	0	1	1	3	47	98	143	275	564	54
100 km	0	0	0	4	22	46	73	170	300	20
125 km	0	0	0	7	12	31	81	116	204	11
150 km	0	0	2	2	14	24	57	117	222	18
175 km	0	0	0	4	5	18	35	81	143	19
200 km	0	0	0	1	8	23	25	50	132	12
>200km	414	445	893	1746	3490	7115	12969	27842	54063	5149

Comment: EPIC software with duplicate removal option switched on (thresholds 60 s in time and 99 km in space) gives 78524 events (approximately 0.7% of the events are duplicates not removed).

REFERENCE

P.N. Shebalin, 1987. Compilation of earthquake catalogs as the task of clustering with learning. - Doklady Ac. Sci. USSR, V.292, N 5, 1083-1086.

APPENDIX 1.

Values of parameters of the algorithm used in program DUPLI_VX

1. Values of Δ_1 and Δ_2.

	through 1964	since 1965
the same data source	$\Delta_1 = 150$ $\Delta_2 = 50$	$\Delta_1 = 200$ $\Delta_2 = 100$
different data sources	$\Delta_1 = 350$ $\Delta_2 = 210$	$\Delta_1 = 250$ $\Delta_2 = 150$

2. Values of dtmax (maximum time difference to search duplicates)

through 1600	1600-1799	1800-1899	1900-1949	since 1950
32000000 s	2650000 s	263000 s	180000 s	87000 s

3. Thresholds for determination of indices f_t, f_r and f_m

a) time (six thresholds, sec)

	the same data source	different data sources
through 1964	6 31 91 149 181 600	12 31 91 155 310 900
since 1965	6 20 50 90 150 450	10 20 30 70 120 550

b) space (six thresholds, km)

	the same data source	different data sources
through 1964	25 50 75 150 300 650	25 50 75 200 500 1500
since 1965	5 25 50 100 200 400	25 40 80 130 300 800

c) magnitude (three thresholds for three cases)

case 1: the same data source and type of magnitude

	the same data source			different data sources		
through 1964	0	1	6	0	2	8
since 1965	1	2	7	0	2	8

case 2: the same type of magnitude only

	the same data source			different data sources		
through 1964	3	5	18	3	5	18
since 1965	1	3	8	2	5	12

case 3: different sources and types of magnitude (minimal possible difference in values is considered)

	the same data source			different data sources		
through 1964	4	7	20	4	9	20
since 1965	3	7	15	4	9	25

M_big = 5.0

In the following tables the rows correspond to increasing values of f_t, the columns to increasing values of f_r.

4. Values of $r(f_t, f_r, f_M, \delta(w,w'))$

a) Through 1964, the same data source

$f_M = 0$						$f_M = 1$						$f_M = 2$						$f_M = 3$					
570	470	360	160	130	120	999	480	320	280	170	130	430	310	260	170	135	120	400	249	200	90	80	0
440	360	250	155	125	115	975	370	310	245	148	125	428	300	255	165	115	120	380	160	140	130	120	0
400	330	240	150	122	114	950	330	285	240	133	124	408	280	250	160	180	170	350	160	130	120	110	0
370	310	150	135	120	113	900	300	275	145	130	123	360	270	245	135	115	110	300	150	120	110	100	0
360	300	145	130	117	112	800	280	265	140	127	122	350	260	240	120	110	100	200	140	110	100	0	0
330	240	140	125	115	110	330	270	255	135	125	120	300	250	235	190	185	180	150	130	100	0	0	0

b) Through 1964, different data source

$f_M = 0$						$f_M = 1$						$f_M = 2$						$f_M = 3$					
896	815	611	470	304	148	999	930	900	530	232	200	888	843	647	540	234	146	774	707	430	200	130	120
874	767	498	210	203	200	893	792	667	340	200	184	869	760	661	530	215	191	681	646	372	120	120	120
864	757	494	203	200	200	875	767	640	530	177	202	859	750	630	520	160	141	572	510	281	120	120	120
850	734	484	203	201	200	864	760	622	518	200	157	856	740	612	508	155	135	499	427	220	120	120	120
848	704	271	204	203	200	853	749	299	221	190	180	842	730	286	211	158	133	328	315	198	120	120	120
230	220	190	180	149	147	340	230	210	190	185	160	220	213	200	180	150	172	200	196	159	125	120	120

c) Since 1965, the same data source

$f_M = 0$						$f_M = 1$						$f_M = 2$						$f_M = 3$					
717	630	360	205	200	125	853	735	370	270	207	130	810	663	365	203	199	127	634	471	256	150	100	100
652	616	489	260	250	150	770	680	385	300	270	205	743	631	345	270	260	175	616	523	331	200	190	170
597	575	500	270	255	160	711	646	466	335	308	206	678	622	464	280	224	177	612	571	377	170	150	140
545	447	350	260	250	150	651	554	330	345	260	150	613	518	290	250	208	145	549	489	208	190	147	113
523	352	260	250	240	140	588	360	287	203	200	150	521	390	200	155	145	130	404	289	185	140	130	112
500	262	240	150	125	100	500	270	205	182	130	105	454	250	195	90	80	90	150	90	80	50	74	92

d) Since 1965, different data source

$f_M = 0$						$f_M = 1$						$f_M = 2$						$f_M = 3$					
885	833	442	151	104	106	930	834	480	328	240	100	887	693	439	190	107	105	793	493	276	133	90	50
842	827	453	200	195	100	922	851	490	270	185	150	883	730	514	209	154	110	772	567	367	214	114	50
850	795	388	193	150	100	925	877	495	165	155	150	893	821	576	250	153	100	815	636	481	252	125	0
659	675	316	128	155	100	865	855	465	180	135	130	804	762	594	257	132	70	696	656	501	194	84	2
304	356	257	197	140	100	729	710	442	270	140	125	668	594	454	220	108	70	506	424	248	59	35	2
166	291	293	247	108	100	575	540	355	250	135	120	565	461	374	216	105	70	437	289	185	50	28	2);

5. Values of $r_j^{*p}(f_t, f_r, f_M, \delta(w,w'))$

For values corresponding to each test four strings are given: the first and second are for data trough 1964, the third and fourth are since 1965, the first and third are for data from the same data source, the second and fourth are from different sources.

	$f_M = 0$						$f_M = 1$						$f_M = 2$						$f_M = 3$					
Test 1.																								
	250	250	200	160	150	140	280	260	230	200	150	140	210	200	150	140	130	120	200	150	130	110	0	0
	418	374	200	100	0	0	429	405	250	0	0	0	409	374	0	0	0	0	200	200	0	0	0	0
	290	280	270	260	250	240	333	323	313	303	250	200	323	313	303	193	183	145	333	333	333	333	200	150
	457	357	300	300	100	50	557	457	320	160	100	50	467	377	300	100	50	0	0	0	0	0	0	0
Test 2.																								
	265	255	250	0	0	0	275	265	255	0	0	0	270	260	250	0	0	0	250	250	125	0	0	0
	400	350	250	200	198	180	420	370	270	220	185	183	400	350	260	190	166	166	300	250	200	0	0	0
	400	250	200	195	190	180	500	300	250	180	170	150	450	357	300	200	180	100	400	306	291	150	130	120
	369	314	301	224	75	0	388	326	311	234	170	0	356	312	301	199	172	0	242	233	189	171	107	0
Test 3.																								
	290	290	0	0	0	0	295	245	230	200	150	0	275	225	210	190	130	0	0	0	100	100	0	0
	400	350	250	205	200	180	420	370	270	220	205	183	400	350	260	190	166	166	300	250	200	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	150	120	100	90	70	50	160	130	110	100	80	60	155	125	105	95	0	0	100	70	50	0	0	0
Test 4.																								
	170	160	150	140	130	120	180	170	160	150	140	130	150	140	130	120	110	0	140	130	120	110	0	0
	170	160	150	140	130	120	180	170	160	150	140	130	150	140	130	120	110	0	140	130	120	110	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	150	120	100	0	0	0	160	130	110	0	0	0	155	125	105	0	0	0	100	70	0	0	0	0
Test 5.																								
	170	160	150	140	130	120	180	170	160	150	140	130	150	140	130	120	110	0	140	130	120	110	0	0
	170	160	150	140	130	120	180	170	160	150	140	130	150	140	130	120	110	0	140	130	120	110	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6. Values of $r_j^*(f_t, f_r, f_n, \delta(w,w'))$

For values corresponding to each test four strings are given: the first and second are for data trough 1964, the third and fourth are since 1965, the first and third are for data from the same data source, the second and fourth are from different sources. Rows correspond to increasing values of f_r.

$f_n = 0$						$f_n = 1$						$f_n = 2$						$f_n = 3$					
Test 1.																							
210	154	153	152	94	0	214	155	154	153	105	0	150	130	100	90	0	0	90	120	100	0	0	0
215	210	208	205	204	190	225	220	218	210	205	195	220	215	213	207	204	150	204	204	176	151	91	51
250	200	197	194	0	0	205	204	203	200	0	0	210	195	150	0	0	0	0	0	0	0	0	0
230	210	200	160	140	120	250	245	210	160	140	130	235	225	205	155	135	125	160	130	110	90	70	0
Test 2.																							
144	150	147	0	0	0	146	139	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
300	290	280	232	162	152	310	300	290	250	100	71	150	100	90	70	50	0	0	0	0	0	0	0
Test 3.																							
160	150	149	148	78	67	154	153	152	151	80	0	135	134	133	132	78	0	85	95	95	0	0	0
296	278	265	202	201	170	308	295	272	246	207	180	218	212	209	208	206	150	205	203	175	150	90	50
230	180	170	0	0	0	241	200	195	190	0	0	208	193	90	0	0	0	0	0	0	0	0	0
210	200	180	140	120	110	230	220	200	148	130	0	225	215	195	145	125	0	150	120	100	0	0	0
Test 4.																							
150	140	139	128	101	89	135	119	116	80	80	0	0	0	0	0	0	0	0	0	0	0	0	0
210	206	198	197	200	150	220	215	190	170	160	130	215	202	150	130	0	0	83	83	55	50	0	0
200	190	160	0	0	0	210	200	166	195	0	0	210	195	90	0	0	0	0	0	0	0	0	0
200	190	170	120	90	0	220	210	190	130	100	0	210	200	180	120	0	0	140	110	100	0	0	0
Test 5.																							
155	153	152	0	0	0	137	153	140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220	215	203	200	200	100	230	220	205	200	140	140	220	215	204	203	200	140	0	0	0	0	0	0
150	140	130	0	0	0	100	200	220	200	0	0	0	0	0	0	0	0	0	0	0	0	0	0
150	130	110	90	0	0	160	140	120	100	0	0	155	135	115	80	0	0	135	115	0	0	0	0
Test 6.																							
250	150	0	0	0	0	150	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Test 7.																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	200	166	0	0	0	0	220	200	0	0	0	0	500	250	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Test 8.																							
148	140	130	125	145	140	149	130	120	146	136	133	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
210	206	0	0	0	0	210	206	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7. Values of $c_j(\delta(w,w'))$ (additional tests)

	j	1	2	3	4	5	6	7	8	9
through	1964	999	693	800	900	612	466	761	623	300
since	1965	999	700	800	600	471	390	830	876	100

APPENDIX 2

Examples of possible duplicates in NEIC/USGS CD-ROM found by program DUPLI_VX

```
JTAV  1900 10091228        58.000-152.000                    7.70MsABE  8.30UKPAS   13   8
BDA   1900 10091228        60.000-142.000 60.                8.20UKBDA            2  1.
ABE   1900 1009122800      58.000-158.000                    7.70MsAN2           11   .
dt max = 0 sec, dr max = 847 km

SSR   1901 04052230     I  45.000 148.000 30.               7.50MsSSR          221GGF.
ABE   1901 0405233000       45.000 148.000                  7.40MsAN2          221   .
BDA   1901 0405233045       45.000 148.000 60.              7.90UKBDA          221 19.
dt max = 3645 sec, dr max = 0 km

EUR   1901 05220638     ER 47.583   7.600  4.               3.00UKEUR          B44BBA5
EUR   1901 05220738     FR 47.600   7.467                                     B44   4
dt max = 3600 sec, dr max = 9 km

SSR   1902 01010520     I  55.000 165.000 20.               7.10MsSSR            4GEF.
ABE   1902 0101052000       55.000-165.000                  7.00MsAN2            9   .
JTAV  1902 0101052030       55.000-165.000 25.              7.00MsABE  7.80UKPAS 9   .
BDA   1902 0101052030       55.000-165.000 60.              7.80UKBDA            9  1.
dt max = 30 sec, dr max = 1719 km

ABE   1902 0830214800       40.000  77.000                  6.80MsAN2          320   .
SSR   1902 08302150         37.000  71.000200.        6.9M 07.70MsSSR          717GGE7
dt max = 120 sec, dr max = 557 km

EUR   1903 04200845     ER 42.300  -3.100 10.               4.30UKEUV          377 A 6
IGNI  1903 0420084500.00   42.300   3.283                                     378   6
dt max = 0 sec, dr max = 471 km

EUR   1903 06211325     ER 42.500   1.200 33.               4.70UKEUV          378 B 7
EUR   1903 06211329     TI 42.600  12.100 25.                                  381   7
dt max = 240 sec, dr max = 802 km

LEI   1904 0830           31.200 100.900                    6.00UKLEI          307   8
ABE   1904 0830114200      30.000 101.000                  7.40mBABE1 6.80MsAN2 307   .
dt max = 42120 sec, dr max = 120 km

BDA   1905 0430170700       -1.000-168.000 60.              7.60UKBDA          624 39.
ABE   1905 0430170700     -20.000-175.000                   7.10MsAN2          173   .
dt max = 0 sec, dr max = 2021 km

ABE   1906 0825134700        4.000  33.000                  6.80MsAN2          557   .
GUTE  1906 0825134736.00     9.000  39.000                  6.75UKPAS          558   .
dt max = 36 sec, dr max = 778 km

ABE   1906 1002015000       -4.000 149.000                  7.20MsAN2          203   .
BDA   1906 10020252         -4.000 149.000 60.              7.70UKBDA          203 16.
dt max = 3720 sec, dr max = 0 km

SSR   1906 10162215     I  46.600  27.300100.       5.5 C   5.50MsSSR          358GFE6
EUR   1906 10162315     ER 46.600  27.300130.               5.50UKEUV          3582226
dt max = 3600 sec, dr max = 0 km

EUR   1908 05191020     ER 37.200   4.100 33.               4.20UKEUV          387 B 6
IGNI  1908 0519102000.00   37.199  -4.100                                     377   4
dt max = 0 sec, dr max = 653 km

EUR   1909 04240247     ER 38.900  -8.800 33.               4.90UKEUV          376 A 7
EUR   1909 04240347     ER 39.000  -9.000 33.               4.90UKEUV          376 C 7
IGNI  1909 0424034700.00   41.699  -8.599                                     376   7
dt max = 3600 sec, dr max = 279 km

BDA   1914 10031722l2       16.000 -61.000100.              7.40UKBDA            92  7.
ABE   1914 10031722l2      -16.000  61.000100.              7.40mBABE1          425   .
GUTE  1914 10031722l2.00    16.000 -61.000100.              7.40UKPAS            92   .
dt max = 0 sec, dr max = 12613 km

EUR   1914 11171626     ER 37.800  21.800 17.               4.60UKEUV          3682226
EUR   1914 11181626     ER 37.700  21.700 33.               4.60UKEUV          368 C .
dt max = 86400 sec, dr max = 12 km

EUR   1919 0626185e     ER 36.700  -1.100 33.               4.20UKEUV          387 C .
ISS   1919 0626185900.00   34.500 -10.000                                     393   .
IGNI  1919 0626190019.00   36.666  -1.083                                     387  4.
dt max = 139 sec, dr max = 756 km

ABE   1919 0831072046     -16.000 169.000180.              7.30mBABE1          186   .
ISS   1919 0831172034.00  -15.000 165.000                                     186   .
BDA   1919 0831172046     -16.000 169.000180.              7.25UKBDA          186 14.
GUTE  1919 0831172046.00  -16.000 169.000180.              7.25UKPAS          186   .
dt max = 36000 sec, dr max = 398 km

SSR   1920 0225172348       36.500  71.000180.       5.4M 06.20MsSSR          717DGZ6
GUTE  1928 0225172358.00    37.500  67.000                5.60UKPAS          717   .
dt max = 10 sec, dr max = 334 km

DGS   1929 0717083406.00    51.000-180.000                                     7   .
JTAV  1929 0717083757       51.000-179.500                                     7   .
dt max = 231 sec, dr max = 31 km
```

```
SSR   1929 1101064721       45.900  26.800190.        6.6 05.9E 06.60MxSSR              3505E07
EUR   1929 11010657     KR 45.900  26.800160.                  6.60UKXXX               3505337
EUR   1929 1101065714   KR 46.000  26.100 33.                                          350 B .
EUR   1929 1101065721   KR 45.900  26.600198.                  6.60UKXXX               350 B 7
GUTE  1929 1101065721.00   45.900  26.800160.                  5.75UKPAS               350   .
MED   1929 1101065725.00   45.900  26.800160.                  6.20MxMED               350   .
dt max = 604 sec, dr max = 35 km

ABE   1930 1221145124       30.000 122.250170.                  7.00mBABE1              348   .
GUTE  1930 1221145124.00    30.000 122.250170.                  6.90UKPAS               348   .
LEI   1930 122112352        33.300 120.400                      6.80UKLEI               344   .
dt max = 32436 sec, dr max = 372 km

BCIS  1935 0419160000.00    32.500  16.000                                              400  6.
EUR   1935 0419175747   KR 30.800  15.500 33.                  5.30UKXXX               401 B .
ISS   1935 0419175747.00   30.800  15.800                                              401   .
dt max = 7067 sec, dr max = 175 km

EUR   1936 08122224     KR 37.000   2.800 15.                  4.90UKXXV               387470.
EUR   1936 0812323420   KR 37.000  28.700 33.                  4.90UKXXX               366 C .
dt max = 20 sec, dr max = 2068 km

CGS   1937 10111858948.00  13.300 -92.000                                              68   .
ISS   1937 10121559946.00  14.000 -92.000                                              69   .
dt max = 86398 sec, dr max = 70 km

SSR   1937 1114105810       35.000  73.000200.          6.9E 07.20UKSSR  7.30MxSSR  720DGE9
BDA   1937 1114105812       36.500  70.800240.                  7.20UKBDA               718 53.
ABE   1937 1114105812       36.500  70.800240.                  7.10mBABE1              718   .
GUTE  1937 1114105812.00    36.500  70.800240.                  7.20UKPAS               718   .
dt max = 2 sec, dr max = 252 km

EUR   1938 0529000456   KR 39.800 -33.700 33.                  4.80UKXXV               404 C .
ISS   1938 0529000456.00   39.800  33.700                                              366   .
dt max = 0 sec, dr max = 5199 km

CGS   1939 0125141148.00   13.000 -90.000                                              71   .
ISS   1939 0127141148.00   13.100 -89.500                                              73   .
dt max = 172800 sec, dr max = 49 km

IGNI  1939 0508162719.00   37.000 -23.900                                              404   7
ISS   1939 0509162719.00   37.000 -23.900                                              404   .
dt max = 86400 sec, dr max = 0 km

EUR   1939 07311332     KR 39.800  29.600 15.                  4.80UKXXV               366620.
EUR   1939 0731133245   KR 39.800  29.600 33.                  4.80UKXXX               366 B .
ISS   1939 0731133245.00   39.800 -29.600                                              405   .
dt max = 45 sec, dr max = 4547 km

EUR   1939 12262357     KR 39.500  38.200 25.                                          366 B T
SSR   1939 1226235716   I  39.700  39.700 18.                  8.00MxSSR               366BIFE
EUR   1939 1226235716   KR 39.700  39.700 33.                  8.00UKXXV               366 B E
BDA   1939 1226235721      39.500  38.500 60.                  7.90UKBDA               366 30.
EUR   1939 1226235721   KR 39.500  38.500 33.                  8.00UKXXV               366 C .
ABE   1939 1226235721      39.500  38.500                      7.70mBABE1 7.80MxABE1   366   .
GUTE  1939 1226235721.00   39.500  38.500                      8.00UKPAS               366   .
MED   1939 1226235721.00   39.800  39.500                      8.00MxMED               366   .
dt max = 21 sec, dr max = 117 km

EUR   1941 0605210220   KR 73.300   2.400 33.                  5.00UKXXV               640 C .
EUR   1941 0605210224   KR 72.000   -.500 33.                  5.00UKXXV               639 C .
GUTE  1941 0606210224.00   72.000   -.500                      5.60UKPAS               639   .
dt max = 86404 sec, dr max = 156 km

EUR   1941 11211212     KR 39.700  23.800 15.                  4.90UKXXV               369510.
EUR   1941 1121121224   KR 39.700  23.800 33.                  4.90UKXXX               365 C .
ISS   1941 1121121224.00  -39.700  23.800                                              587   .
dt max = 24 sec, dr max = 7940 km

BDA   1943 0503015912       12.500-125.500 60.                  7.40UKBDA               611 39.
ABE   1943 0503015912       12.500 125.500                      7.50mBABE1 7.30MxABE1   251   .
GUTE  1943 0503015912.00    12.500 125.500                      7.40UKPAS               251   .
dt max = 0 sec, dr max = 10641 km

CDMG  1944 0902213645.00P  34.233-116.433                      3.00MLPAS               430   .
CGS   1944 0903013642.00   34.200-116.400                                              43   .
dt max = 14397 sec, dr max = 4 km

ISS   1948 0211032927.00   36.100 118.800                                              664   .
CDMG  1948 0211032928.30P  36.090-118.870 11.                  4.60MLPAS               39A  6
dt max = 1 sec, dr max = 9883 km

GS    1976 0102063348.40* 66.024 -16.770 33. N      4.4 0                              637 13.
EMSC  1976 0102063359.70   65.510 -14.340 10. A                  4.60mbEMSC              638 16.
dt max = 11 sec, dr max = 111 km

GS    1976 0109235010.60   32.985  76.024 22.        4.7 0                              303 23.
SSR   1976 0110035027      39.400  72.500                      4.50UKSSR  3.89UKSSR    716   .
dt max = 14417 sec, dr max = 702 km

GS    1976 0122192846.00* 44.518 149.310 33. N      4.3 0                              221 15.
SSR   1976 0122192850      44.500 149.300 30.                  3.06MxSSR               221   .
SSR   1976 0123192857      44.500 149.300 30.                  3.78MxSSR               221   .
dt max = 86411 sec, dr max = 1 km

SSR   1976 0424191332      41.300  51.600  8.                  4.60UKSSR  4.75MxSSR    338   .
EMSC  1976 0424191334.70   41.560  51.070 10. A                3.40MxEMSC 4.70mbEMSC   338 45.
GS    1976 0424191335.40   41.459  50.939 33. N      4.8 0                              338 24.
SSR   1976 0424191337      42.000  50.300                      4.90UKSSR  4.17MxSSR    338   .
dt max = 5 sec, dr max = 120 km
```

```
GS    1976 0505002306.60*  64.985-135.509 33. N                                    679 10.
JTAV  1976 0505022240.20   66.900-135.570  0.                                     677  .
dt max = 7172 sec, dr max = 8 km

GS    1976 0612034143.50   37.388  20.863 38.      4.3 0   4.30MLATH               399 19.
EMSC  1976 0612034142.50   37.360  20.180 10. A            3.60MsEMSC 3.80mbEMSC 399 14.
dt max = 3599 sec, dr max = 30 km

ISC   1976 0805113903      36.800  71.300180.                                     717  .
GS    1976 0805143903.00   36.842  71.330304.      4.5 0                          717 17.
dt max = 10800 sec, dr max = 26 km

ISC   1976 0828233952      46.000 142.700350.                                     662  .
GS    1976 0829233951.80   46.185 143.194360. G    4.6 0                          662 18.
dt max = 86399 sec, dr max = 38 km

EMSC  1976 0906141136.90   39.160  20.370 10.              3.80MsEMSC 3.80mbEMSC 392 12.
GS    1976 0906141138.30*  39.104  28.971 26.                                     366 10.
dt max = 2 sec, dr max = 666 km

ISC   1976 0915204242      42.200  77.250                  2.78MsISC              330  .
ISC   1976 0916204242      42.200  77.200                  2.78MSISC              330  .
dt max = 86400 sec, dr max = 3 km

GS    1976 1219235655.30*    .896 126.027 33. N    5.6 0                          266 31.
GS    1976 1220235654.60     .828 126.074 33. N    5.5 0                          266 61.
dt max = 86399 sec, dr max = 8 km

GS    1977 0511234616.70   33.235  48.001 33. N    4.8 0                          347 32.
EMSC  1977 0512234620.30   33.230  48.000 51.              4.70mbEMSC             347 52.
dt max = 86404 sec, dr max = 1 km

ISC   1977 0701094914      42.900  45.300  5.              4.00MSISC  3.89MsISC   337  .
EMSC  1977 0701214917.40   43.120  45.580 10. A                                   337 29.
GS    1977 0701214919.90   43.000  45.464 53.      4.8 0                          337 22.
dt max = 43205 sec, dr max = 30 km

GS    1978 0517123441.40* -31.570 -68.190 10. G                                    137  6.
GS    1978 0518123441.30* -31.538 -67.909 10. G                                    137  7.
dt max = 86400 sec, dr max = 24 km

IGNT  1978 0717141558.30   37.813  -7.561 13.              3.80MW                 376 183
EMSC  1978 0717141558.80   37.790   7.960 10. A                                   387  9.
dt max = 0 sec, dr max = 1226 km

ISC   1978 0928085538      51.300 160.000 33.              5.00MSISC  3.61MsISC   219  .
GS    1978 0928095538.20*  51.361 159.744 33. N    4.7 0                          219 37.
dt max = 3600 sec, dr max = 17 km

NEDC  1978 1129145200.00   16.010 -96.590 18.              7.70MsGANSE            60   .
GS    1978 1129195247.40   16.010 -96.591 18. D    6.4 07.7E 07.90MLTRV           60343.
ABE   1978 1129195249      16.100 -96.600 23.              7.30mbABE2 7.60MsABE1  60   .
dt max = 18049 sec, dr max = 9 km

GS    1982 0601233720.27*  17.664-101.814 43.    1.004.6 6                        58 15.
GS    1982 0603013719.427  17.694-101.818 33. N  1.31                             58 13.
dt max = 7199 sec, dr max = 3 km

GS    1982 0805132932.448  44.895   8.218 10. G  1.01                           545 24.
GS    1982 0806132932.75   45.004   8.227 10. G  1.384.6 1      4.10MLTRI         545 36.
EMSC  1982 0806132934.70   44.910   8.230 10. A            3.80MsEMSC 4.20mbEMSC 545 69.
dt max = 86402 sec, dr max = 11 km

GS    1982 0810050036.948  39.397  22.985 10. G   .38                           364  8.
GS    1982 0811050035.62*  39.352  23.010 10. G   .95      3.20MLATH            365  7.
EMSC  1982 0811050037.10   39.350  23.070 10. A            3.00MsEMSC            365 20.
dt max = 86401 sec, dr max = 7 km

GS    1983 0321064815.258 -33.151 -70.410109.     .94                           127  9.
GS    1983 0321104917.68  -33.003 -70.347 94.    1.024.5 4                      127 252
dt max = 14462 sec, dr max = 16 km

GS    1983 0503012319.038  36.273-120.068  7. G             3.20MLBRV             39  4.
GS    1983 0503093416.578  36.191-120.313  9. G             3.20MDGS              39  3.
dt max = 28857 sec, dr max = 22 km

GS    1983 0507054357.078  36.223-120.288  9. G             3.60MLBRV  3.60MDGS   39 12.
GS    1983 0507124331.328  36.310-120.384  6. G             3.60MLBRX             39 12.
dt max = 25114 sec, dr max = 12 km

EMSC  1983 0806163643.50   40.230  24.790 10. A             3.80MsEMSC            365 10.
GS    1983 0806193750.748  39.934  24.617 10. G  1.80       3.30MLATH            365  6.
dt max = 10867 sec, dr max = 33 km

GS    1984 0512000839.93*  41.450  14.058 10. G  1.393.7 3      4.10MLTRI         390 32.
EMSC  1984 0512000845.50   41.670  13.910 10. A             4.00MsEMSC 3.76mbEMSC 390 64.
GS    1984 0512010826.457  42.098  13.436 10. G   .26       3.70MDFRA            381  4.
dt max = 3587 sec, dr max = 79 km

GS    1984 0528135711.248  40.428  28.124 10. G   .28                           366  5.
GS    1984 0528135714.327  41.509  20.216 10. G  1.20                           548  5.
dt max = 3 sec, dr max = 1237 km

GS    1985 0103235954.91   45.355   6.553 12.      .81      3.90MLLDG            538 23.
GS    1985 0104235955.91   45.434   6.484 10. G  1.05                           538 22.
EMSC  1985 0104235956.60   45.250   6.620 10. A             3.80MsEMSC            538 69.
dt max = 86402 sec, dr max = 20 km

GS    1985 0627050716.79*  39.177  57.233 33. N   .704.410                      340 19.
EMSC  1985 0627050717.80   37.100  55.310 10. A             3.90MsEMSC 4.33mbEMSC 341 29.
dt max = 1 sec, dr max = 256 km
```

```
GS    1986 0316000400.26*  41.876  19.152 10. G   .97          3.20[illegible]          391  8.
GS    1986 0316010401.00   42.014  19.388 10. G   .87          3.30[illegible]          383  9.
dt max = 3601 sec, dr max = 22 km

GS    1986 0507230745.0054 51.800-176.800 33. M     5.5 8                               7 21.
GS    1986 0508010415.61   51.609-174.697 33. M   .925.353                            7195.
dt max = 7110 sec, dr max = 12 km

GS    1986 0525233818.53   39.460  28.353 10. G   .68                                 366 13.
GS    1986 0526233818.51t  39.461  28.350 10. G   .71                                 366 13.
dt max = 86400 sec, dr max = 0 km

GS    1986 0527024405.08t  38.856  27.826 10. G   .62                                 366  5.
GS    1986 0527044430.14   38.935  27.829 10. G   .89                                 366 18.
dt max = 7245 sec, dr max = 8 km

GS    1988 0525101007.49t  40.142  27.225 10. G  1.16                                 366  5.
GS    1988 0525101111.38   37.266  22.139 89. *  1.233.9 63.3I 1                       368 24.
dt max = 64 sec, dr max = 490 km

GS    1988 0810103432.69  -10.143 160.484 35. *  1.055.6155.3I 85.[illegible]         193102.
GS    1988 0810183433.06  -10.549 160.964 33. M  1.364.9125.4I 1                       193 80.
dt max = 18001 sec, dr max = 62 km

GS    1988 0811155439.02t  46.848   -.303 10. G  1.30          2.90[illegible]          838 14.
GS    1988 0812155439.02t  46.848   -.303 10. G  1.30          2.90[illegible]          838 14.
dt max = 86400 sec, dr max = 0 km
```

APPENDIX 3

Examples of errors of duplicates identification by program DUPLI_VX

```
**********

ISS  1917 0426093945.00    43.900   9.500
EUR  1917 04260936      T1 43.480  13.120 25.                    8.50M&TRI              380   . +
                                                                                         381   X 1
EUR  1917 0426093559    KR 43.500  13.100 33.                    8.50UYYYY              381 A X +
MED  1917 0426093559.00    43.500  13.100                        8.50M&MED              381   . 1
ISS  1917 0426093609.00    43.600  13.000                                               381   . 2

**********

ISS  1923 10092302S1.00    39.300  21.000                                               364   . +
EUR  1923 10092303      KR 41.300  19.500  5.                    4.20UXXXX              3915217 2

EUR  1923 10092304      KR 41.500  19.500 33.                                           391 C 7

ISS  1923 10092305S1.00    39.300  21.000                                               364   .

**********

EUR  1927 05180258      KR 44.100  20.500 10.                    4.60UYYYY              3835326 +
EUR  1927 05180259      KR 44.100  20.500                        3.80UXXXX              383   . 1

EUR  1927 051803        KR 44.100  20.500 33.                    3.80UYYYY              383 B 6 +
EUR  1927 05180303      KR 44.000  20.600                        3.80UYYYY              383   . 1

**********

GS   1978 0307024839.40   31.962 137.610442.        6.0 0       6.40UYBRY              211406. +
ABS  1978 0307024848      32.000 137.600441.                    7.10M&BABSS            211   . 1
GS   1978 0307024847.60   32.005 137.609439.        6.9 0       7.00UYPAS              2111216

**********

BSR  1978 0323140643      44.100 149.300 40.                    4.60UYSSR  3.30M&SSR   221   .
GS   1978 0323140725.50*  44.090 149.277 33. W      4.8 0                              221 20.

**********

GS   1978 0829023658.20   49.839  78.008  0. G      5.0 0                              329116.
GS   1978 0829023706.50   50.008  78.996  0. G      5.9 04.02 0                        329199.

**********

GS   1978 0903051845.80*  48.306   9.044 10. G                                        543  5.
EUR  1978 0903052010    FR 48.300   9.000                        2.80UYFRG              543   .
GS   1982 0118200002.73   39.843  24.071 10. G  1.174.115       4.90M&ATH              365 43. +
EMSC 1982 0118200004.50   39.770  24.240 10. A                   5.10M&EMSC 4.30M&EMSC 365 46. 1
EMSC 1982 0118200004.60   39.830  24.150 10. A                   5.00M&EMSC 4.70M&EMSC 365  7.

**********

GS   1982 1220174502.20* -24.574-175.811 33. W  1.234.9 6                             175 20.
GS   1982 1220174536.02  -23.864-175.743 33. W  1.345.215                             174 34. +
GS   1982 1220174608.82  -23.545-175.939 33. W  1.185.6166.3210                       174 85. 1

**********

GS   1984 1230213623.03* -36.749 177.411 40.    1.115.2 2                             160 25.
GS   1984 1230213656.41  -36.662 177.512 39.    1.386.2256.82156.80M&BRY  6.60M&PAS  160133.

**********

GS   1985 0403202122.52   28.115 139.538465.      .914.816                           212 80.
GS   1985 0403202136.26   28.230 139.525468. D911.185.999       6.20M&PAS             2134273

**********

GS   1985 0820203221.77   28.110 140.735 33. W  1.224.711                            212 34.
GS   1985 0820203243.00   28.096 140.603 33. W  1.024.9 74.72 2                       212 36.

**********

GS   1985 0831063033.98*  28.082 140.831 15. *  1.384.0 2                            212 223
GS   1985 0831063043.06   28.068 140.774 33. W  1.085.216                            212 86.

**********

GS   1985 1112033411.52* -36.514 -97.918 10. G  1.404.6 5                            686 23.
GS   1985 1112033418.35  -36.499 -98.158 10. G  1.045.6315.72155.90M&BRX             692 98.

**********
GS   1986 1114213005.07   23.911 121.654 33. W  1.385.4 3                            244394. +
GS   1986 1114213010.88   23.901 121.574 33. G  1.196.3237.82177.50M&BRX             244 798 1

**********

GS   1988 0217075255.107  21.707 174.232 33. W   .824.4 1                             5  4.
GS   1988 0217075314.60   21.659 174.753 33. W  1.165.219                             5 54.

**********

GS   1988 0224084435.48   13.321 124.714 33. W  1.304.4 3                            249 56.
GS   1988 0224084454.61   13.318 124.811 33. W  1.135.2 95.62 2                       249 33.
```

APPENDIX 4

Description of the program DPL_ONE

Program DPL_ONE chooses for each cluster of duplicates determined and marked by program DUPLI_VX one record (string) from one data source.

The input is filed with extension .DP@ in VX-format. 116-th byte in .DP@ files is used to mark duplicates.

The output is filed with the same name and extension .DPL in VX-format. Output is sorted by increase of time.

The rule for choosing one record from the set of duplicates follows.

1. Records of the highest priority data sources are taken. Following is the list of descending priorities:
CGS, NOS, ERL, GS, PDE, GUTE, G-R, CDMG, NGDC, EMSC, SSR, EUR, ABE, BDA, LEE.

2. If several records have the same data source, the data source of time/epicentre (pos.25-26 in VX-format) is checked. The following list of descending priorities is chosen:
"KR", " ", anything exept ("% ","? ","* "), "% ", "? ", "* ".

3. If several records still have the same priority, those which have at least one non-zero value of magnitude of any type are chosen.

4. The record with the largest number of stations used for epicentre determination is chosen, if this number exists and is greater than 7.

5. Records that have a non-zero value for mb or MS magnitudes, or have a contributed magnitude 1 or 2 with the same data source as that for epicentre and time, are chosen from the rest.

6. Records with a larger number of non-blanked bytes #11-23,46-48, 53-63, 75-80 and 89-92 of VX-format are chosen.

7. The first record from the rest is chosen finally.

APPENDIX 5

Sorce codes of the programs to combine catalogs (DUPLI_VX, DPL_ONE)

```
/*****************************************************************/
/*                      dupli_vx - main program                  */
/*****************************************************************/

/* This program identifies and marks duplicates in input file [.vx]
and writes the result to the file [.dpt]. Last digit (11-th) in the
file [.dpt] is ' ' if the event has no duplicates,
'+' if it has, and digit for duplicates to record marked by '+' */

#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <malloc.h>
#include <ctype.h>
#include "getkey.h"
#include "param0.h"

static int ti=0;        /* index of time period for other parameters */

struct data
 {
 char record[115];
 long num;
 int st;
 struct data *p;
 struct data *pb;
 };
struct g_data
 {
 struct data *rc;
 int dlt;
 int jcst;
 double delta2;
 struct g_data *p;
 };

main(argc,argv)
int argc;
char **argv;
{
/* int allocated=1,freed=0; */

char ipfile[30];
char opfile[30];
FILE *ipf,*opf;
char inrec[130];

int dlt=0,dlt1=0;
double delta4=0.0;
int jcst=0,jcst1=0;
register int j=0,j1=0;
long nvpct=0;
int ern_max=0;          /* number of records in linked list */
int jr=0,jw=0;          /* markers of end of file
                           (0 - file exhousted, 1 - not yet) */
long ir=0L;             /* number of records read */
long iw=0L;             /* number of records written */
long ig=0L;             /* number of duplicates clusters */
long num_rec=0L;        /* number of records in file */
double step=0.0;        /* num_rec/79 */
double num_print=0.0;   /* next number to print symbol */

int flag=0,flag1=0,flag2=0,n_flag=0,w_flag=0;
int nnn=0;
int ng=0,n_ng=0,n_ng1=0;
struct data     *first,*next,*work,*last;
                /* pointers to elements of main linked list */
struct g_data   *g_first,*g_next,*g_work,*g_last;
                /* pointers to elements of linked list for cluster */
struct g_data   *ng_first,*ng_next,*ng_work,*ng_last;
struct g_data   *ng_first1,*ng_last1;
struct g_data   *ng_first2,*ng_last2;
                /* pointers to elements of linked list for elements
                   of group but not belonging to cluster */

struct find_t *file_info;

int t=0;        /* index of time period for dtmax */
double adlt=0.0,adlt2=0.0;
double adelta(int,int);
```

```
ipfile[0]=' ';
opfile[0]=' ';
inrec[0]=' ';

printf("\n\n\n\n\n**********************************************************************************");
  printf("**                                                                                **");
  printf("**      IDENTIFICATOR OF DUPLICATES IN EARTHQUAKE CATALOGUES MERGED TOGETHER       **");
  printf("**                                                                                **");
  printf("**                                VERSION 2.0-VX    1991                          **");
  printf("**                                                                                **");
  printf("**                                   Petar Shebalin                               **");
  printf("**            International Institute for Earthquake Prediction Theory            **");
  printf("**                              and Mathematical Geophysics                       **");
  printf("**      Varshavskoye sh. 79-2, 113556 Moscow USSR  Phone:3107795 Fax:3107033      **");
  printf("**                                                                                **");
printf("**********************************************************************************\n\n\n");

if (argc < 2)
 {
 printf("FORMAT: dupli_vx <input file name [.vx]> [<output file name [.dpt]>]\n");
 exit(1);
 }
strcpy(ipfile,argv[1]);
for (j=1;j<strlen(ipfile);j++) if (*(ipfile+j) == '.') break;
if (*(ipfile+j) != '.') strcpy(ipfile+j,".vx");
if (argc < 3) { strcpy(opfile,ipfile); strcpy(opfile+j,".dpt"); }
else strcpy(opfile,argv[2]);
for (j=1;j<strlen(opfile);j++) if (*(opfile+j) == '.') break;
if (*(opfile+j) != '.') strcpy(opfile+j,".dpt");

/* determine number and lenght of records */
file_info=(struct find_t *) malloc(sizeof(struct find_t));
_dos_findfirst(ipfile,_A_NORMAL,file_info);
num_rec=file_info->size;
free(file_info);

if ((ipf=fopen(ipfile,"rb")) == NULL)
 { printf("%s not found, exiting ...\n",ipfile); exit(1); }
if ((opf=fopen(opfile,"wb")) == NULL)
 { printf("%s cannot be open, exiting ...\n",opfile); exit(1); }

fread(inrec,130,1,ipf);
for (j=0;j<130;j++) if (*(inrec+j) == '\n') break; j++;
rvect=j-115;

num_rec/=(long)j;
step=(double)num_rec/79.0;
num_print=step;
fseek(ipf,0L,SEEK_SET);

/* allocate memory for linked lists of records */
first=(struct data *) malloc(sizeof(struct data));
first->pb=(struct data *) NULL;
q_first=(struct q_data *) malloc(sizeof(struct q_data));
nq_first=(struct q_data *) malloc(sizeof(struct q_data));
nq_work=(struct q_data *) malloc(sizeof(struct q_data));
next=first;
q_next=q_first;
qrs_max=1;
while(next != (struct data *) NULL)
 {
 nq_work=(struct q_data *) malloc (sizeof(struct q_data));
 if (nq_work == (struct q_data *) NULL) break;
 q_next->p=nq_work;
 q_next=q_next->p;
 q_next->p=(struct q_data *) NULL;
 work=(struct data *) malloc (sizeof(struct data));
 if (work == (struct data *) NULL) break;
 next->p=work;
 qrs_max++;
 (next->p)->pb=next;
 next=next->p;
 next->p=(struct data *) NULL;
 last=next;
 }
/* free memory for q_data chain (all but first) */
q_next=q_first->p;
while(q_next != (struct q_data *) NULL)
 {
 q_work=q_next->p;
 if(q_next != (struct q_data *) NULL) free(q_next);
 q_next=q_work;
 }
if (nq_first != (struct q_data *) NULL) free(nq_first);
if (nq_work  != (struct q_data *) NULL) free(nq_work);
printf("\n\nTld records to be processed\n",num_rec);

printf("\n.....\r");
/* Fill linked list */
next=first;j=0;
while(next != (struct data *) NULL)
 {
 jr=fread(next->record,115,1,ipf); fseek(ipf);
 *(next->record+115)='\0';
 if (!jr) {next->pb->p=(struct data *) NULL; break;}
 ir++;
 next->num=ir;
 next->st=0;
 next=next->p;
 }
/* printf("\nPress <Esc> to terminate program ...\n\n\n"); */

/* main loop */
nq_first=(struct q_data *) NULL;
```

```
g_first->p=(struct g_data *) NULL;
g_first->dlt=0;
g_first->jcat=0;
g_first->delta3=0.0;
n_mg=0;
ti=0;

while (chtoi(first->record+6,4) >= yr[t]) t++;
if (t > 4) ti=1;

while (iw < num_rec)
 {
 if (kbhit() && (getkey() == CTRL_END)) break;
 if (!ti && chtoi(first->record+6,4) >= yr[t]) { t++; if (t > 4) ti=1; }
 g_first->rc=first;
 g_first->delta3=0.0;
 g_first->dlt=0;
 g_last=g_first;
 mg_last=(struct g_data *) NULL;
 mg_first1=(struct g_data *) NULL;
 mg_first2=(struct g_data *) NULL;
 mg=1;
 adlt3=min(adelta(delta2[ti][0],0),adelta(delta2[ti][1],1));

/* check if first is listed in mg_data */

 if (n_mg)
   {
   if (mg_first->rc->num == first->num)
     {
     g_work=g_first;
     g_first=mg_first;
     free(g_work);
     g_work=(struct g_data *) NULL;
     n_mg--;
     }
   else
     {
     mg_next=mg_first;
     while(mg_next->p != (struct g_data *) NULL)
       {
       if ((mg_next->p)->rc->num == first->num) break;
       mg_next=mg_next->p;
       }
     if (mg_next->p)  /* found */
       {
       g_first->delta3=mg_next->p->delta3;
       g_first->p=mg_first;
       mg_work=mg_next->p;
       mg_next->p=(mg_next->p)->p;
       free (mg_work);
       mg_work=(struct data *) NULL;
       n_mg--;
       }
     else g_first->p=mg_first;  /* no first in mg_data list */
     }  /* g_first is not mg_first */

   mg_first1=g_first->p;
   g_first->p=(struct g_data *) NULL;
   mg=1; n_mg=0; n_mg1=0;
   mg_first=(struct g_data *) NULL;
   mg_last=(struct g_data *) NULL;
   g_last=g_first;
   mg_next=mg_first1;
/* pairs to first */
   while(mg_next)
     {
     dlt=delta(g_first->rc->record,mg_next->rc->record,&jcat,t);
     adlt=adelta(dlt,jcat);
     mg_next->dlt=dlt;
     mg_next->jcat=jcat;
     mg_work=mg_next->p;
     if ((dlt >= delta1[ti][jcat]) && (adlt >= g_first->delta3))
       {
       g_last->p=mg_next;
       g_last=g_last->p;
       mg++;
       }
     else  if ((dlt < delta2[ti][jcat]) || (adlt < g_first->delta3))
       {
       if (n_mg) { mg_last->p=mg_next; mg_last=mg_last->p; }
       else { mg_first=mg_next; mg_last=mg_next; }
       if (adlt > g_first->delta3) g_first->delta3=adlt;
       n_mg++;
       }
     else
       {
       if (n_mg1) { mg_last2->p=mg_next; mg_last2=mg_last2->p; }
       else { mg_first1=mg_next; mg_last2=mg_next; }
       n_mg1++;
       }
     mg_next=mg_work;
     }
   g_last->p=(struct g_data *) NULL;
   if (mg_last) mg_last->p=(struct g_data *) NULL;
   if (mg_last2) mg_last2->p=(struct g_data *) NULL;

   mg_first1=(struct g_data *) NULL;
   mg_last1=(struct g_data *) NULL;

/* pairs to second and further on (sort mg_first2) */
   mg_next=mg_first2;
   if (n_mg1) while (mg_next)
     {
```

```
      nq_work=nq_first;
      flag=0;
      if (n_nq) while (nq_work)
        {
        dlt=delta(nq_work->rc->record,nq_next->rc->record,&jcat,t);
        adlt=adelta(dlt,jcat);
        if (adlt > adelta(nq_next->dlt,nq_next->jcat))
          {
          if (n_nq) {nq_last->p = nq_next; nq_last=nq_last->p; }
          else { nq_first=nq_next; nq_last=nq_next; }
          n_nq++;
          if ((adlt-adelta(nq_next->dlt,nq_next->jcat)) > g_first->delta) g_first->delta-=adlt;
          flag=1; break;
          }
        nq_work=nq_work->p;
        }
      if (flag) { nq_next=nq_next->p; continue; }
      g_last->p=nq_next;
      g_last=g_last->p;
      nq_next=nq_next->p;
      nq++;
      }
    if (nq_last)  nq_last->p=(struct g_data *) NULL;
    g_last->p=(struct g_data *) NULL;
    n_nq1=0; nq_first1=(struct g_data *) NULL;
    }   /* n_nq != 0 */
  n_nq1=0;

/* update delta1-linked list */

/* now 1) g_first - dl-linked, d2-limited,
       2) nq_first - non-duplicates to first
       3) nq_first1 - the remaining nq */

/* find pairs to second and further on */
 g_next=g_first;
 while (g_next)
   {
   next=first->p;
   while (next != (struct data *) NULL)
     {
     if (next->et) { next=next->p; continue; }
     dlt=delta(g_next->rc->record,next->record,&jcat,t);
     if ((dlt < 0) && (next->num > g_next->rc->num)) break;
     adlt=adelta(dlt,jcat);
     if (adlt < adlt2) {next=next->p; continue; }
     next->et=1;
     nq_work=(struct g_data *) malloc(sizeof(struct g_data));
     nq_work->p=(struct g_data *) NULL;
     nq_work->rc=next;
     nq_work->delta=0.0;
     nq_work->dlt=delta(g_first->rc->record,next->record,&jcat1,t);
     nq_work->jcat=jcat1;
     adlt=adelta(nq_work->dlt,nq_work->jcat);
     if ((nq_work->dlt < delta2[i][nq_work->jcat]) || (adlt < g_first->delta))
       {
       if (n_nq) { nq_last->p=nq_work; nq_last=nq_last->p; }
       else { nq_first=nq_work; nq_last=nq_first; }
       n_nq++;
       }
     else
       {
       if (n_nq1) { nq_last1->p=nq_work; nq_last1=nq_last1->p; }
       else { nq_first1=nq_work; nq_last1=nq_first1; }
       n_nq1++;
       }
     next=next->p;
     }

/* sort nq_first1 to g_first or nq_first */

   nq_next=nq_first1;
   if (n_nq1) while (nq_next)
     {
     nq_work=nq_first;
     flag=0;
     if (n_nq) while (nq_work)
       {
       dlt=delta(nq_work->rc->record,nq_next->rc->record,&jcat,t);
       adlt=adelta(dlt,jcat);
       if (adlt > adelta(nq_next->dlt,nq_next->jcat))
         {
         if (n_nq) {nq_last->p = nq_next; nq_last=nq_last->p; }
         else { nq_first=nq_next; nq_last=nq_next; }
         n_nq++;
         if ((adlt-adelta(nq_next->dlt,nq_next->jcat)) > g_first->delta) g_first->delta-=adlt;
         flag=1; break;
         }
       nq_work=nq_work->p;
       }
     if (flag) { nq_next=nq_next->p; continue; }
     g_last->p=nq_next;
     g_last=g_last->p;
     nq_next=nq_next->p;
     nq++;
     }
   if (nq_last)  nq_last->p=(struct g_data *) NULL;
   g_last->p=(struct g_data *) NULL;
   n_nq1=0; nq_first1=(struct g_data *) NULL;

   g_next=g_next->p;
   }
```

```
  /* remove non-duplicates according delta3 condition */
    g_next=g_first;
    if (ng > 2) while (g_next->p)
      {
    /* remove far from g_first */
      if (adelta(g_next->p->dlt,g_next->p->jcat) < g_first->delta3)
        {
        ng_last->p=g_next->p;
        g_next->p=g_next->p->p;
        ng_last=ng_last->p;
        n_ng++; ng--;
        ng_last->p=(struct g_data *) NULL;
        continue;
        }
      if (!g_next->p->p) break;
      g_work=g_next->p;
      while (g_work->p)
        {
        flag1=0;
        dlt=delta(g_next->p->rc->record,g_work->p->rc->record,&jcat,t);
        adlt=adelta(dlt,jcat);
        if ((dlt < delta3[ti][jcat]) || (adlt < g_next->p->delta3) || (adlt < g_work->p->delta3))
          {
          ng--; flag1=rev(g_next->p,g_work->p);
          if (flag1) break;
          else
            {
            if (n_ng) { ng_last->p=g_work->p; ng_last=ng_last->p; }
            else { ng_first=g_work->p; ng_last=ng_first; }
            g_work->p=(g_work->p)->p;
            if (!g_work->p) g_last=g_work;
            ng_last->p=(struct g_data *) NULL;
            n_ng++;
            }
          }
        else g_work=g_work->p;
        }
      if (!flag1) g_next=g_next->p;
      else
        {
        if (n_ng) {ng_last->p=g_next->p; ng_last=ng_last->p; }
        else {ng_first=g_next->p; ng_last=ng_first; }
        g_next->p=(g_next->p)->p;
        if (!g_next->p) g_last=g_next;
        ng_last->p=(struct g_data *) NULL;
        n_ng++;
        }
      }
  /* while (g_next->p->p) */

  /* check if delta < delta to some ng_data element or older cluster */

    flag=1;
    ng_first2=ng_first;
    ng_last2=ng_last;
    n_ng=1;

    if (ng > 1) while(n_ng)
      {
      ng_first1=ng_first2;
      ng_last1=ng_last2;
      ng_first2=(struct g_data *) NULL;
      ng_last2=(struct g_data *) NULL;
      g_next=g_first;
      n_ng=0;

  /* update g_first->delta3 */
      ng_next=ng_first1;
      while(ng_next)
        {
        dlt=delta(first->record,ng_next->rc->record,&jcat,t);
        adlt=adelta(dlt,jcat);
        if (adlt > g_first->delta3) g_first->delta3=adlt;
        ng_next=ng_next->p;
        }
  /* check second and further on */
      while (g_next && g_next->p)
        {
  /*      if (((double)g_next->p->dlt/(double)delta1[ti][g_next->p->jcat]) <
  max((double)delta2[ti][g_next->p->jcat]/(double)delta1[ti][g_next->p->jcat],g_first->delta3))  */
        adlt=adelta(g_next->p->dlt,g_next->p->jcat);
        if ((adlt < g_first->delta3) || (adlt < g_next->p->delta3))
          {
          if (n_ng) { ng_last2->p=g_next->p; ng_last2=ng_last2->p; }
          else { ng_first2=g_next->p; ng_last2=ng_first2; }
          g_next->p=g_next->p->p;
          ng--; n_ng++;
          ng_last2->p=(struct data *) NULL;
          continue;
          }
        delta4=0.0;

  /* find maximum deltas to ng_data or older cluster */
        ng_next=ng_first1;
        flag1=0;
        while (ng_next != (struct g_data *) NULL)
          {
          dlt=delta(g_next->p->rc->record,ng_next->rc->record,&jcat,t);
          adlt=adelta(dlt,jcat);
          delta4=max(delta4,adlt);
          ng_next=ng_next->p;
          }
  /* check */
      n_flag=0;
```

```
    /* check delta with first */
    if (sdelta(g_next->p->dlt,g_next->p->jcat) < delta4)
      {
      if (n_mg) { mg_last3->p=g_next->p; mg_last3=mg_last3->p; }
      else { mg_first3=g_next->p; mg_last3=mg_first3; }
      g_next->p=(g_next->p)->p;
      mg_last3->p=(struct g_data *) NULL;
      mg--; n_mg++;
      continue;
      }
    /* check with other from g_data list */
     g_work=g_first;
     flag2=1;
     while (g_work->p)
       {
       flag1=0;
       if (g_work->p->rc->num == g_next->p->rc->num) {g_work=g_work->p; flag2=0; continue; }
       dlt=delta(g_work->p->rc->record,g_next->p->rc->record,&jcat,t);
       sdlt=sdelta(dlt,jcat);
       if ((dlt < delta3[t1][jcat]) || (sdlt < delta4))
         {
         mg--; flag1=rev(g_next->p,g_work->p);
         if (flag1) break;
         else if (flag2) { g_next=g_work; flag1=1; break; }
         else
          {
          if (n_mg) {mg_last3->p=g_work->p; mg_last3=mg_last3->p; }
          else {mg_first3=g_work->p; mg_last3=mg_first3; }
          g_work->p=(g_work->p)->p;
          mg_last3->p=(struct g_data *) NULL;
          n_mg++;
          }
         }
       else g_work=g_work->p;
       }
     if (!flag1) g_next=g_next->p;
     else
       {
       if (n_mg) {mg_last3->p=g_next->p; mg_last3=mg_last3->p; }
       else {mg_first3=g_next->p; mg_last3=mg_first3; }
       g_next->p=(g_next->p)->p;
       mg_last3->p=(struct g_data *) NULL;
       n_mg++;
       }
     }   /* while (g_next->p->p) */

  /* check if all are delta1-linked with first */
    mg_next=g_first->p;
    g_first->p=(struct g_data *) NULL;
    g_next=g_first;
    g_last=g_first;
    while (g_next)
     {
     while(mg_next)
       {
       dlt=delta(g_next->rc->record,mg_next->rc->record,&jcat,t);
       if (dlt < delta1[t1][jcat]) break;
       g_last->p=mg_next;
       mg_next=mg_next->p;
       g_last=g_last->p;
       g_last->p=(struct g_data *) NULL;
       }
     mg_work=mg_next;
     if (mg_work) while (mg_work->p)
       {
       dlt=delta(g_next->rc->record,mg_work->p->rc->record,&jcat,t);
       if (dlt < delta1[t1][jcat]) { mg_work=mg_work->p; continue; }
       g_last->p=mg_work->p;
       mg_work->p=mg_work->p->p;
       g_last=g_last->p;
       g_last->p=(struct g_data *) NULL;
       }
     g_next=g_next->p;
     }
    if (mg_next != (struct g_data *) NULL)
      {
      if (n_mg) { mg_last3->p=mg_next; mg_last3=mg_last3->p; }
      else { mg_first3=mg_next; mg_last3=mg_first3; }
      n_mg++;
      while(mg_last3->p) { mg_last3=mg_last3->p; n_mg++; }
      }
    if (flag) flag=0;
    else
      {
      if (mg_first != (struct g_data *) NULL) mg_last->p=mg_first1;
      else mg_first=mg_first1;
      if (mg_last1 != (struct g_data *) NULL) mg_last=mg_last1;
      }
    }   /* while (n_mg) */


  /* determine delta3 in mg_data list */
   n_mg=0;
   mg_next=mg_first;
   while (mg_next)
    {
    n_mg++;
    g_next=g_first;
    while (g_next)
     {
     dlt=delta((mg_next->rc)->record,(g_next->rc)->record,&jcat,t);
     sdlt=sdelta(dlt,jcat);
     if (sdlt > mg_next->delta3)  mg_next->delta3=sdlt;
     g_next=g_next->p;
```

```
    }
  mg_next=mg_next->p;
  }

/* Sort the cluster of duplicates and write to output file,
free g_data linked list */

if (!fwrite(first->record,115,1,opf))
  {
  printf("\n\nUnable to write to output file (disk full) ... exiting ...\n");
  exit(1);
  }
iw++;
ng=1;
ig++;
last->p=first;
first->pb=last;
first=first->p;
first->pb=(struct data *) NULL;
last=last->p;
last->p=(struct data *) NULL;
if (g_first->p == (struct g_data *) NULL) fwrite(" \r\n",3,1,opf);
else
 {
 fwrite("+\r\n",3,1,opf);

/* sorting */
 g_next=g_first;
 while((g_next->p)->p != (struct g_data *) NULL)
   {
   flag=0;
   g_work=g_next->p;
   while (g_work->p != (struct g_data *) NULL)
     {
     if (((g_work->p)->rc)->num < ((g_next->p)->rc)->num)
       {
       flag=1;
       g_last=g_next->p;
       g_next->p=g_work->p;
       g_work->p=(g_work->p)->p;
       (g_next->p)->p=g_last;
       break;
       }
     g_work=g_work->p;
     }
   if (!flag) g_next=g_next->p;
   }

/* writing and freeing g_data */
 flag1=0;
 g_next=g_first->p;
 next=last;
 while(g_next != (struct g_data *) NULL)
  {
  g_work=g_next->p;
  work=g_next->rc;
  jw=fwrite(work->record,115,1,opf);
  *(work->record)=(ng%10)+'0';
  memcpy((work->record)+1,"\r\n",2);
  if (jw) jw=fwrite(work->record,3,1,opf);
  if (!jw)
   {
   printf("\nUnable to write to output file (disk full) ... exiting ...\n");
   exit(1);
   flag1=1;
   break;
   }
  iw++;
  if (g_next != (struct g_data *) NULL) free(g_next);
  g_next=g_work;

  next->p=work;
  if (work->pb != (struct data *) NULL) { work=work->pb; work->p=(work->p)->p; (work->p)->pb=work; }
  else { first=first->p; first->pb=(struct data *) NULL; }
  (next->p)->pb=next;
  next=next->p;
  next->p=(struct data *) NULL;
  ng++;
  }
 if (flag1) break;
 } /* else */
g_first->p=(struct g_data *) NULL;

/* update main linked list */
next=last;
while (next != (struct g_data *) NULL)
 {
 if (jr) jr=fread(next->record,115,1,ipf);
 if (jr)
    {
    ir++;
    ffseek(ipf);
    next->num=ir;
    next->st=0;
    }
 else
    {
    next=next->pb;
    next->p=(struct data *) NULL;
    break;
    }
 last=next;
 next=next->p;
```

```
}
if (iv > (long) num_print) { printf("."); num_print+=step; }

} /* while (iv < ir) */

printf(".");
printf("\n\n%ld records read, %ld written, %ld groups of duplicates\n\n",ir,iv,ig);

if (iv < ir)  {printf("%c\nNon-recognised error during execution, please restart\n",(char)7); exit(1); }
exit(0);
}

/*****************************************************************/
int rsv(str1,str2)
struct g_data *str1,*str2;
{
double adelta();
if (adelta(str1->dlt,str1->jcat) < adelta(str2->dlt,str2->jcat)) return(1);
if (adelta(str1->dlt,str1->jcat) > adelta(str2->dlt,str2->jcat)) return(0);
if (str1->rc->num > str2->rc->num) return(1);
return(0);
}

/*****************************************************************/

double adelta(val,jct)
int val,jct;
{
return((double)val/(double)deltal[ti][jct]);
}

ffseek(ffff)
FILE *ffff;
 {
 static char a=' ';
 for (;;) { fread(&a,1,1,ffff); if (a == '\n') break; }
 return(1);
 }
/*****************************************************************/
/*                              dlt                              */
/*****************************************************************/


#include <string.h>
#include <stdlib.h>
#include <malloc.h>
#include <ctype.h>
#include <math.h>
#include "param.h"

#define NUMPAR 9

/* function returns the value of proximity function for recors first and next */
/* it is also returns pointer to value of jcat - 0 if data source is the same, */
/* 1 otherwise */
/* argument t is 0 (parameters for period prior 1961) or 1 (starting 1961)   */
/* value of parameters are stored in parax.h   */

delta(first,next,jcat,t)
char *first,*next;
int *jcat;
int t;
{
long ddt;          /* ddtl */
int ddr;
int fi1,fi2,la1,la2,dfi,dla,h1,h2;
int j,nt1,nt2;
int return_value;
int j_t,j_f,j_r;
int flag,fs;
double val,factor,factor1;
int ti;
int epi_error;
int time_error;
int r_add[NUMPAR];

long dtl();
ti=t/5;

ddt=dtl(first,next);
if (ddt > dtmax[t]) return(-1);
return_value=0;

time_error=1;
epi_error=1;
for (j=0;j<NUMPAR; j++) r_add[j]=0;

/* jcat = 0 if catalog and data source are the same */

*jcat=memcmp(first,next,8);
if (*jcat) *jcat=1;
else if ((*(first+24) == '*') || (*(first+24) == '?')) *jcat=0;
else if ((*(next+24) == '*') || (*(next+24) == '?')) *jcat=0;
else if (memcmp(first+24,next+24,2)) *jcat=1;
else *jcat=0;


for (j_t=0;j_t < 6; j_t++) if (ddt < t_w[ti][*jcat][j_t]) break;

fi1=chtoi(first+24,6);
la1=chtoi(first+33,7);
fi2=chtoi(next+24,6);
la2=chtoi(next+33,7);
```

```
  ddr=dr(fil,fil2,lal,la2);
  for (j_r=0;j_r < 6; j_r++) if (ddr < e_v[tl][*jcat][j_r]) break;

  j_v=dm(first,next,*jcat,tl);

  /* printf(" %d %d %d",j_t,j_r,j_m); */

  /* return 10000 if it is evident duplicate */
  if (!j_t  && (j_m < 3)) { if (j_r < 5) return(8000); if (j_r < 6) return(800); }
  if (*jcat && !j_t && (j_r < 4)) return(6000);
  if (*jcat && (j_t<2) && (j_r < 3) && (j_m<3)) return(5000);
  /* if ((j_t<3) && (j_r < 3) && (j_m<3)) return(10000); */
  if (!ddt && (ddr < (e_v[tl][1][5]*5/3)) && (j_m < 4)) return(1000);
  if (!j_t && (ddr < (e_v[tl][1][5]*5/3)) && (j_m < 2)) return(1000);
  if (*jcat && !dt(first,next) && (j_r < 5) && (j_m < 4)) return(4000);

  if (j_m < 0) return(0);

  /* big difference in intensity */

  nst1=(int)*(first+92)-(int)'0'; if (nst1 < 3) nst1=0;
  else if (nst1 > 9)
   {
   if (*(first+92) == 'X') nst1=10;
  /* else if (*(first+92) == 'E') nst1=11; */
  /* else if (*(first+92) == 'T') nst1=12; */
   else nst1=0;
   }

  nst2=(int)*(next+92)-(int)'0'; if (nst2 < 3) nst2=0;
  else if (nst2 > 9)
   {
   if (*(next+92) == 'X') nst2=10;
  /* else if (*(next+92) == 'E') nst2=11; */
  /* else if (*(next+92) == 'T') nst2=12; */
   else nst2=0;
   }
  if (nst1 && nst2 && (ddt > 5L))
    {
    if (abs(nst1-nst2) > 2) factor1=0.75;
    else if (abs(nst1-nst2) > 3) factor1=0.65;
    else if (abs(nst1-nst2) > 4) factor1=0.5;
    else factor1=1.000001;
    }
  else factor1=1.0000000001;

  /* tests of errors in coordinates */
  if ((j_t < 5) && (ddr > 100))
    {
     if (epi_err[tl][*jcat][0][j_m][j_t] && (dr(abs(fil),abs(fil2),abs(lal),abs(la2)) < 50))
      epi_error=0;     /* sign of latitude, longitude */
     else if ((ddr > 500) && epi_err[tl][*jcat][0][j_m][j_t] && (dr(abs(fil),abs(fil2),abs(lal),abs(la2)) < 200))
      epi_error=0;     /* sign of latitude, longitude */
     else if ((ddr > 1000) && epi_err[tl][*jcat][0][j_m][j_t] && (dr(abs(fil),abs(fil2),abs(lal),abs(la2)) < 300))
      epi_error=0;     /* sign of latitude, longitude */
     else if (epi_err[tl][*jcat][2][j_m][j_t] && epi_fat(fil,fil2,lal,la2))
      epi_error=2;     /* format error */
     else if (epi_err[tl][*jcat][1][j_m][j_t] && epi_dig(first,next,fil,fil2,lal,la2))
      epi_error=1;     /* one digit error in coord */
     else if (epi_err[tl][*jcat][3][j_m][j_t] && dig_miss(fil,lal,fil2,la2))
      epi_error=3;     /* one digit missed */
     else if (epi_err[tl][*jcat][4][j_m][j_t] && lat_lon(fil,fil2,lal,la2))
      epi_error=4;     /* lat and lon reversed or lat = lon  */
    }
  if (j_r > 5)
   {
   if ((j_t > 5) || (epi_error < 0)) return(0);
   return_value=epi_err[tl][*jcat][epi_error][j_m][j_t];
   return(return_value);
   }

  /* possible error in time */
  factor=1.0; rs=1;
  /* if ((j_r < 5) && (ddt > 60L)) */
  if (ddt > 60L)
    {
    if (time_err[tl][*jcat][2][j_m][j_r] && (rs=err_hour(ddt,first,next,tl,*jcat)))
      time_error=2;  /*  error in hour  */
    else if (time_err[tl][*jcat][3][j_m][j_r] && (rs=err_day(ddt,first,next,tl,*jcat)))
      time_error=3;    /*  error in day or month  */
    else if (time_err[tl][*jcat][0][j_m][j_r] && (rs=loc_time1(ddt,first,next,lal,la2,tl,*jcat)))
      time_error=0;  /*  local time 1  */
    else if (time_err[tl][*jcat][1][j_m][j_r] && (rs=loc_time2(ddt,first,next,lal,la2)))
      time_error=1;  /*  local time 1  */
    else if (time_err[tl][*jcat][4][j_m][j_r] && (rs=dig_rev(ddt,first,next,tl,*jcat)))
      time_error=4;    /*  digits reversed  */
    else if (time_err[tl][*jcat][5][j_m][j_r] && (rs=num_rev(ddt,first,next,tl,*jcat)))
      time_error=5;    /*  numbers reversed  */
    else if (time_err[tl][*jcat][6][j_m][j_r] && (rs=dig_time(ddt,first,next,tl,*jcat)))
      time_error=6;    /*  digit missed or added */
    else if (time_err[tl][*jcat][7][j_m][j_r] && (rs=err_min(ddt,first,next,tl,*jcat)))
      time_error=7;    /*  error in minute  */
    if (rs == 3) factor=2.3;
    else if (rs == 2) factor=1.7;
    else if (rs == 6) factor=0.7;
    }

  if (j_t > 5) {
    if ((j_r > 5) || (time_error < 0)) return(0);
    return_value=time_err[tl][*jcat][time_error][j_m][j_r]; }
  else if (j_r < 6) {
    return_value = r_val[tl][*jcat][j_r][j_m][j_t];
    if (time_error >= 0)  {
```

```
    if (time_err[ti][*jcat][time_error][j_m][j_r] > return_value) {
      return_value=time_err[ti][*jcat][time_error][j_m][j_r]; } }
  if (epi_error >= 0) {
    if (epi_err[ti][*jcat][epi_error][j_m][j_t] > return_value)  {
      return_value=epi_err[ti][*jcat][epi_error][j_m][j_t];  } } }

if (!return_value) return(0);

/* do some additional arguments for duplicates exist ? */

if (*jcat)  /* almost all bytes are the same */
  {
  if (addition[ti][0])
    {
    flag=0;
    for(j=5;j<25;j++) if (*(first+j) != *(next+j)) flag++;
    for(j=26;j<86;j++) if (*(first+j) != *(next+j)) flag++;
    if (flag < 2)
      {
      r_add[0]=return_value;
      val=(1000.0+(double)addition[ti][0])/1000.0;
      return_value=(return_value*val);
      }
    }
  else
    {
    for (j=0;j<ni_src[ti];j++)
      {
      if ((!memcmp(first,src1[j],3) && (!memcmp(src1[j]+3,first+24,2) || !memcmp(src1[j]+3,"***",2)))
      || (!memcmp(first,src1[j],3) && (!memcmp(src1[j]+3,first+24,3) || !memcmp(src1[j]+3,"***",2))))
         {  return_value=(return_value*factor*factor1*1.5); return(return_value); }
      }
    }
  return_value=(return_value*factor*factor1);
  return(return_value);
  }
/* source which may contain duplicates */
for (j=0;j < n_src[ti]; j++)
                        if (!memcmp(first,src[j],3))
  {
  r_add[1]=return_value;
  val=addition[ti][1];
  val/=1000.0; val+=1.0;
  if (val > 0.9) return_value=(return_value*val);
  break;
  }

if (((*(first+91)-'0') >= 0) && ((*(first+91)-'0') <= 9)) mat1=chtoi(first+89,3);
    else mat1=0;
if (((*(next+91)-'0') >= 0) && ((*(next+91)-'0') <= 9)) mat2=chtoi(next+89,3);
    else mat2=0;

if ((!mat1 && mat2) || (mat1 && !mat2))   r_add[3]=return_value;

if (mat1 > 99) mat1=0;
if (mat2 > 99) mat2=0;

                                                  /* diff mat */
if (addition[ti][2] && mat1 && mat2 &&
  (((min(mat1,mat2) < 10) && (abs(mat1-mat2) > 3)) ||
  ((min(mat1,mat2) >= 10) && (abs(mat1-mat2) > 5))))
  r_add[2]=return_value;

if (addition[ti][4] && unrlbl(first,next))
                                          /* unreliable data */
  r_add[4]=return_value;
else if (addition[ti][5] && (j_t<6) && bad_acc(first,next,mat1,mat2))
                                          /* bad acc */
  r_add[5]=return_value;
h1=chtoi(first+41,3);
h2=chtoi(next+41,3);
if ((h1 == 25) || (h1 == 33) ||
(*(first+47) == 'A') || (*(first+47) == 'M') ||
(*(first+47) == 'D') || (*(first+47) == 'G') ||
(*(first+47) == '*') || (*(first+47) == '?')) h1=0;
if ((h2 == 25) || (h2 == 33) ||
(*(next+47) == 'A') || (*(next+47) == 'M') ||
(*(next+47) == 'D') || (*(next+47) == 'G') ||
(*(next+47) == '*') || (*(next+47) == '?')) h2=0;
if (h1 == h2) j=0;
else if (!h1 || !h2) j=1;
else if ((h1 > 50) && (h2 < 50)) j=2;
else if ((h2 > 50) && (h1 < 50)) j=2;
else j=3;
if (j)  r_add[5+j]=return_value;


val=0.0;
for (j=2;j<NUMPAR;j++) if (r_add[j] && ((double) addition[ti][j] > val))
                    val=addition[ti][j];
val/=1000.0; val+=1.0;
if (val > 0.9) return_value=(return_value*val);
return_value=(return_value*factor*factor1);
return(return_value);
}
/*****************************************************************************/
/*                              testpar                                      */
/*****************************************************************************/

/* The following are functions used by delta()     */
```

```
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include "param1.h"

static char rec[27]=" ";
static char c2[2]=" ";

/*********************************** dig_miss in epi ***************************/

dig_miss(lat1,lon1,lat2,lon2)
int lat1,lon1,lat2,lon2;
{
int nev,sign,base;
if (lat1<0) { sign=-1; nev=lat1*(-1); } else { sign=1; nev=lat1; }
base=nev/1000; if (base)
 { nev=nev%1000; if (dr(nev*sign,lat2,lon1,lon2) < 50) return(1);
 nev=base*100+(nev%100); if (dr(nev*sign,lat2,lon1,lon2) < 30) return(1); }

if (lat2<0) { sign=-1; nev=lat2*(-1); } else { sign=1; nev=lat2; }
base=nev/1000; if (base)
 { nev=nev%1000; if (dr(lat1,nev*sign,lon1,lon2) < 50) return(1);
 nev=base*100+(nev%100); if (dr(lat1,nev*sign,lon1,lon2) < 30) return(1); }

if (lon1<0) { sign=-1; nev=lon1*(-1); } else { sign=1; nev=lon1; }
base=nev/10000; if (base)
 { nev=nev%10000; if (dr(lat1,lat2,nev*sign,lon2) < 50) return(1);
 nev=base*1000+(nev%1000); if (dr(lat1,lat2,nev*sign,lon2) < 50) return(1); }
base=nev/1000; if (base)
 {
 if (base < 10) {nev=nev%1000; if (dr(lat1,lat2,nev*sign,lon2) < 50) return(1);}
 nev=base*100+(nev%100); if (dr(lat1,lat2,nev*sign,lon2) < 30) return(1);
 }

if (lon2<0) { sign=-1; nev=lon2*(-1); } else { sign=1; nev=lon2; }
base=nev/10000; if (base)
 { nev=nev%10000; if (dr(lat1,lat2,lon1,nev*sign) < 50) return(1);
 nev=base*1000+(nev%1000); if (dr(lat1,lat2,lon1,nev*sign) < 50) return(1); }
base=nev/1000; if (base)
 {
 if (base < 10) {nev=nev%1000; if (dr(lat1,lat2,lon1,nev*sign) < 50) return(1);}
 nev=base*100+(nev%100); if (dr(lat1,lat2,lon1,nev*sign) < 30) return(1);
 }
return(0);
}


epi_dig(str1,str2,lat1,lat2,lon1,lon2)
int lat1,lat2,lon1,lon2;
char *str1,*str2;
{
char buff[8];
int val;

memcpy(buff,str1+26,6);
if ((*(str2+28) != ' ') && (*(str2+28) != '-'))
  {
*(buff+1)=*(str2+27);
if (*(buff+1) == ' ') *(buff+1)='0';
else if (*(buff+1) == '-') *(buff+1)='0';
val=chtoi(buff,6);
if (dr(val,lat2,lon1,lon2) < 200) return(1);
if (dr(val+1000,lat2,lon1,lon2) < 200) return(1);
if (dr(val-1000,lat2,lon1,lon2) < 200) return(1);
*(buff+1)=*(str1+27);
  }
*(buff+2)=*(str2+28);
if (*(buff+2) == ' ') *(buff+2)='0';
else if (*(buff+2) == '-') *(buff+2)='0';
val=chtoi(buff,6);
if (dr(val,lat2,lon1,lon2) < 70) return(1);
if (dr(val+100,lat2,lon1,lon2) < 70) return(1);
if (dr(val-100,lat2,lon1,lon2) < 70) return(1);

memcpy(buff,str1+33,7);
if ((*(str2+36) != ' ') && (*(str2+36) != '-'))
 {
*(buff+2)=*(str2+35);
if (*(buff+2) == ' ') *(buff+2)='0';
else if (*(buff+2) == '-') *(buff+2)='0';
val=chtoi(buff,7);
if (dr(lat1,lat2,val,lon2) < 200) return(1);
if (dr(lat1,lat2,val+1000,lon2) < 200) return(1);
if (dr(lat1,lat2,val-1000,lon2) < 200) return(1);
*(buff+2)=*(str1+35);
 }
*(buff+3)=*(str2+36);
if (*(buff+3) == ' ') *(buff+3)='0';
else if (*(buff+3) == '-') *(buff+3)='0';
val=chtoi(buff,7);
if (dr(lat1,lat2,val,lon2) < 70) return(1);
if (dr(lat1,lat2,val+100,lon2) < 70) return(1);
if (dr(lat1,lat2,val-100,lon2) < 70) return(1);
*(buff+3)=*(str1+36);
if (*(str2+35) == ' ') return(0);
if (*(str2+35) == '-') return(0);
if (*(str2+36) == ' ') return(0);
if (*(str2+36) == '-') return(0);
*(buff+1)=*(str2+34);
val=chtoi(buff,7);
if (dr(lat1,lat2,val,lon2) < 130) return(1);
return(0);
}
```

```
/*********************************** epi_fmt ***********************************/

epi_fmt(f1,f2,l1,l2)
int f1,f2,l1,l2;
{
if (dr((int)(f1/10),f2,l1,l2) < 50)   return(1);
if (dr((int)(f1/100),f2,l1,l2) < 50) return(1);
if (dr(f1,f2,(int)(l1/10),l2) < 50)  return(1);
if (dr(f1,f2,(int)(l1/100),l2) < 50) return(1);
if (dr(f1,(int)(f2/10),l1,l2) < 50)  return(1);
if (dr(f1,(int)(f2/100),l1,l2) < 50) return(1);
if (dr(f1,f2,l1,(int)(l2/10)) < 50)  return(1);
if (dr(f1,f2,l1,(int)(l2/100)) < 50) return(1);
return(0);
}

lat_lem(f1,f2,l1,l2)
int f1,f2,l1,l2;
{
if (abs(l1) > 9000) return(0);
if (abs(l2) > 9000) return(0);
if (dr(f1,l2,l1,f2) < 50) return(1);
if (dr(l1,f2,f1,l2) < 50) return(1);
if ((abs(f1-f2) < 50) && ((abs(f1-l1) < 50) || (abs(f2-l2) < 50))) return(1);
if (((abs(l1-l2) < 50) || (abs(l1-l2) > 35950)) && ((abs(f1-l1) < 50)
  || (abs(f2-l2) < 50))) return(1);
return(0);
}

/*************************** local time 1 ******************************/

loc_time1(ddt,str1,str2,lon1,lon2,t,jcat)
int lon1,lon2;
char *str1,*str2;
long ddt;
int t,jcat;
{
int gmt,gmt1,gmt2,j,flag;
/* if (ddt < 3500) return(0); */
if (ddt < 6000) return(0);
/* if (*(str1+18) == ' ') return(0);
if (*(str2+18) == ' ') return(0); */
gmt=abs(lon1)/1500;
gmt1=abs(lon2)/1500;
gmt2=max(gmt,gmt1)+3;
gmt1=min(gmt,gmt1)-2;
gmt=(ddt/3600L);
j=(ddt%3600L);    if (j > 1800) { j=3600-j; gmt--; }
if (j > (int)(loc_t[t][jcat]*2)) return(0);
if (gmt < 2) return(0);    /* test of error in hour */
if (gmt > 23) return(0);
if (j <= (int)(loc_t[t][jcat]/15)) j=4;
else if (j <= (int)(loc_t[t][jcat]/6)) j=3;
else if (j <= (int)loc_t[t][jcat]) j=2;
else j=1;

if ((gmt < gmt2) && (gmt > gmt1)) flag=1;
else if ((abs(24-gmt) < gmt2) && (abs(24-gmt) > gmt1)
          && !memcmp(str1+13,str2+13,2)) flag=2;
else if (!(gmt%2) && ((gmt/2) < gmt2) && ((gmt/2) > gmt1)) flag=3;
else if (gmt < 13) flag=3;
else return(0);
j=max(0,j-flag+1);
if (!j) return(0);
j--; if (!j) return(4);
return(j);
}

/*************************** local time 2 ******************************/

loc_time2(ddt,str1,str2,lon1,lon2)
int lon1,lon2;
char *str1,*str2;
long ddt;
{
int gmt,gmt1,j;
if (ddt < 3500) return(0);
if ((*(str1+16) == ' ') || (*(str2+16) == ' ')) return(0);
if (!memcmp(str1,str2,5) && !memcmp(str1+24,str2+24,2) &&
  memcmp(str1+18,str2+18,2) &&
  ((*(str1+18) == ' ') || (*(str2+18) == ' '))) return(0);
gmt=abs(lon1)/1500;
gmt1=abs(lon2)/1500;
j=min(gmt,gmt1)-1;
gmt1=max(gmt,gmt1)+2;
gmt=max(0,j);
if (ddt < gmt*3600L) return(0);
if (ddt > gmt1*3600L) return(0);
gmt=ddt%3600L;
if (!gmt) return(1);
return(0);
}

/*************************** error in hour *****************************/

err_hour(ddt,str1,str2,t,jcat)
long ddt;
char *str1,*str2;
int t,jcat;
{
/* if (*(str1+18) == ' ') return(0);
/* if (*(str2+18) == ' ') return(0); */
if (labs(ddt-3600L) <= (long)(err_h[t][jcat]/15L)) return(3);
if (labs(ddt-3600L) <= (long)(err_h[t][jcat]/6L)) return(2);
```

```
if (labs(ddt-3600L) <= (long)err_h[t][jcat]) return(1);
if (labs(ddt-3600L) <= (long)(err_h[t][jcat]*2L)) return(4);
if (labs(ddt-7200L) <= (long)(err_h[t][jcat]/15L)) return(3);
if (labs(ddt-7200L) <= (long)(err_h[t][jcat]/6L)) return(2);
if (labs(ddt-7200L) <= (long)err_h[t][jcat]) return(1);
if (labs(ddt-7200L) <= (long)(err_h[t][jcat]*2L)) return(4);
if (memcmp(str1+13,str2+13,2)) return(0);
if (labs(ddt-36000L) <= (long)(err_h[t][jcat]/8L)) return(2);
if (labs(ddt-36000L) <= (long)(err_h[t][jcat]/4L)) return(1);
return(0);
}

/************************** error in day or month ***************************/

err_day(ddt,str1,str2,t,jcat)
long ddt;
char *str1,*str2;
int t,jcat;
{
long dt1();
long ddt1;
/* if (memcmp(str1+13,str2+13,2) && (*(str1+16) != ' ') && (*(str2+16) != ' '))  */
if (memcmp(str1+13,str2+13,2))
 {
 if  (labs(ddt-86400L) < (long)(err_d[t][jcat]/15L)) return(3);
 if  (labs(ddt-86400L) < (long)(err_d[t][jcat]/6L)) return(2);
 if  (labs(ddt-86400L) < (long)err_d[t][jcat]) return(1);
 if  (labs(ddt-86400L) < (long)(err_d[t][jcat]*2L)) return(4);
 if  (labs(ddt-172800L) < (long)(err_d[t][jcat]/15L)) return(3);
 if  (labs(ddt-172800L) < (long)(err_d[t][jcat]/6L)) return(2);
 if  (labs(ddt-172800L) < (long)err_d[t][jcat]) return(1);
 if  (labs(ddt-172800L) < (long)(err_d[t][jcat]*2L)) return(4);
 }
if (!memcmp(str1+11,str2+11,2)) return(0);
if (*(str1+14) == ' ') return(0);
if (*(str2+14) == ' ') return(0);
memcpy(rec,str2,25);   memcpy(rec+11,str1+11,2);
if ((ddt1=dt1(str1,rec)) < (long)(err_d[t][jcat]/6L)) return(2);
if (ddt1 < err_d[t][jcat]) return(1);
return(0);
}

/************************** error in minute *****************************/

err_min(ddt,str1,str2,t,jcat)
long ddt;
char *str1,*str2;
int t,jcat;
{
long dt();
int j;

if (ddt > 3000L) return(0);
if (*(str1+20) == ' ') return(0);
if (*(str2+20) == ' ') return(0);
if (!memcmp(str1+19,"00.00",5)) return(0);
if (!memcmp(str2+19,"00.00",5)) return(0);
if (!memcmp(str1+19,"00   ",5)) return(0);
if (!memcmp(str2+19,"00   ",5)) return(0);
memcpy(rec,str2,27);
for (j=0; j< 6; j++)  { *(rec+17)=j+'0'; if (abs((int)dt(str1,rec)) < err_m[t][jcat]) return(1); }
memcpy(rec,str1,27);
for (j=0; j< 6; j++)  { *(rec+17)=j+'0'; if (abs((int)dt(str2,rec)) < err_m[t][jcat]) return(1); }
return(0);
}

/************************** digits reversed *****************************/

dig_rev(ddt,str1,str2,t,jcat)
long ddt;
char *str1,*str2;
int t,jcat;
{
long dt2();
int j;
char c;

if (ddt < 300L) return(0);
memcpy(rec,str2,27);
for(j=11;j<19;j++)
 {
 if (*(str1+j) == ' ') break;
 if (*(str1+j+1) == ' ') break;
 if (*(rec+j) == ' ') break;
 if (*(rec+j+1) == ' ') break;
 if ((j < 17) && ((*(str1+j+2) == ' ') || (*(str2+j+2) == ' '))) break;
 if (*(rec+j) == *(rec+j+1)) continue;
 c=*(rec+j);
 *(rec+j)=*(rec+j+1);
 *(rec+j+1)=c;
 if (dt2(str1,rec) < t_fit[t][jcat][j-11]) return(1);
 c=*(rec+j+1);
 *(rec+j+1)=*(rec+j);
 *(rec+j)=c;
 }
memcpy(rec,str1,27);
for(j=11;j<19;j++)
 {
 if (*(str2+j) == ' ') break;
 if (*(str2+j+1) == ' ') break;
 if (*(rec+j) == ' ') break;
 if (*(rec+j+1) == ' ') break;
 if ((j < 17) && ((*(str2+j+2) == ' ') || (*(str1+j+2) == ' '))) break;
 if (*(rec+j) == *(rec+j+1)) continue;
```

```
    c=*(rec+j);
    *(rec+j)=*(rec+j+1);
    *(rec+j+1)=c;
    if (dt2(str2,rec) < t_fit[t][jcat][j-11]) return(1);
    c=*(rec+j+1);
    *(rec+j+1)=*(rec+j);
    *(rec+j)=c;
    }
   return(0);
   }

   /************************** numbers reversed ****************************/

   num_rev(ddt,str1,str2,t,jcat)
   long ddt;
   char *str1,*str2;
   int t,jcat;
   {
   long dt2();
   int j;

   if (ddt < 300L) return(0);
   memcpy(rec,str2,25);
   for(j=11;j<17;j+=2)
    {
    if (*(str1+j+3) == ' ') break; if (*(rec+j+3) == ' ') break;
    if (memcmp(rec+j,rec+j+2,2) == 0) continue;
    memcpy(c2,rec+j,2);  memcpy(rec+j,rec+j+2,2);  memcpy(rec+j+2,c2,2);
    if (dt2(str1,rec) < t_fit[t][jcat][j-11]) return(1);
    memcpy(c2,rec+j,2);  memcpy(rec+j,rec+j+2,2);  memcpy(rec+j+2,c2,2);
    }
   memcpy(rec,str1,25);
   for(j=11;j<17;j+=2)
    {
    if (*(str2+j+3) == ' ') break; if (*(rec+j+3) == ' ') break;
    if (memcmp(rec+j,rec+j+2,2) == 0) continue;
    memcpy(c2,rec+j,2);  memcpy(rec+j,rec+j+2,2);  memcpy(rec+j+2,c2,2);
    if (dt2(str2,rec) < t_fit[t][jcat][j-11]) return(1);
    memcpy(c2,rec+j,2);  memcpy(rec+j,rec+j+2,2);  memcpy(rec+j+2,c2,2);
    }
   return(0);
   }

   /********************************* dig_time ****************************/

   dig_time(ddt,str1,str2,t,jcat)
   long ddt;
   char *str1,*str2;
   int t,jcat;
   {
   int j;
   long dt2();

   if (ddt < 300L) return(0);

   /* - one digit missed */
   if ((*(str1+20) == ' ') || (*(str1+20) == '0'))
   {
   memcpy(rec,str1,25);  memmove(rec+14,rec+13,11);
   for (j=13; j< 17; j++)
    {
    if (*(str2+j+2) == ' ') break;
    if (*(str1+j+2) == ' ') break;
    *(rec+j)=*(str2+j);
    if (dt2(str2,rec) < t_fit[t][jcat][j-11]) return(1);
    *(rec+j)=*(rec+j+1);
    }
   }
   if ((*(str2+20) == ' ') || (*(str2+20) == '0'))
   {
   memcpy(rec,str2,25);  memmove(rec+14,rec+13,11);
   for (j=13; j< 17; j++)
    {
    if (*(str1+j+2) == ' ') break;  if (*(str2+j+2) == ' ') break;
    *(rec+j)=*(str1+j);
    if (dt2(str1,rec) < t_fit[t][jcat][j-11]) return(1);
    *(rec+j)=*(rec+j+1);
    }
   }
   /*  one digit more */
   memcpy(rec,str2,25);
   for (j=16;j<21;j++) if (*(rec+j) == ' ') {*(rec+j)='0'; break; }
   for(j=11;j<17;j+=2)
    {
    if (*(str1+j+3) == ' ') break; if (*(rec+j+3) == ' ') break;
    memcpy(c2,rec+j,2); memmove(rec+j,rec+j+1,22-j);
    if (dt2(str1,rec) < t_fit[t][jcat][j-11]) return(1);
    memmove(rec+j+1,rec+j,22-j);  memcpy(rec+j,c2,2);
    memmove(rec+j+1,rec+j+2,22-j);
    if (dt2(str1,rec) < t_fit[t][jcat][j-11]) return(1);
    memmove(rec+j+2,rec+j+1,22-j);   memcpy(rec+j,c2,2);
    }
   memcpy(rec,str1,25);
   for (j=16;j<21;j++) if (*(rec+j) == ' ') {*(rec+j)='0'; break; }
   for(j=11;j<17;j+=2)
    {
    if (*(str2+j+3) == ' ') break;   if (*(rec+j+3) == ' ') break;
    memcpy(c2,rec+j,2);   memmove(rec+j,rec+j+1,22-j);
    if (dt2(str2,rec) < t_fit[t][jcat][j-11]) return(1);
    memmove(rec+j+1,rec+j,22-j);  memcpy(rec+j,c2,2);
    memmove(rec+j+1,rec+j+2,22-j);
    if (dt2(str2,rec) < t_fit[t][jcat][j-11]) return(1);
    memmove(rec+j+2,rec+j+1,22-j);   memcpy(rec+j,c2,2);
    }
```

```
return(0);
}

/********************************** bad_accuracy ****************************/

bad_acc(str1,str2,mst1,mst2)
char *str1,*str2;
int mst1,mst2;
{
char c;
if (((mst1 > 1) && (mst1 < 7)) || ((mst2 > 1) && (mst2 < 7))) return(1);
c=*(str1+89);
if ((c == 'D') || (c == '*') || (c == 'V') || (c == 'E') || (c == 'G') ||
    (c == 'H') || (c == 'I')) return(1);
c=*(str2+89);
if ((c == 'D') || (c == '*') || (c == 'V') || (c == 'E') || (c == 'G') ||
    (c == 'H') || (c == 'I')) return(1);
return(0);
}

/********************************** smrlbl ****************************/

smrlbl(str1,str2)
char *str1,*str2;
{
if (!memcmp(str1+24,"I ",2)) return(1);
if (!memcmp(str1+24,"LT",2)) return(1);
if (*(str1+24) == '?')       return(1);
if (*(str1+24) == '*')       return(1);
if (*(str1+24) == '&')       return(1);
if (*(str1+24) == (char) 63) return(1);
if (*(str1+24) == (char) 37) return(1);
if (!memcmp(str2+24,"I ",2)) return(1);
if (!memcmp(str2+24,"LT",2)) return(1);
if (*(str2+24) == '?')       return(1);
if (*(str2+24) == '*')       return(1);
if (*(str2+24) == '&')       return(1);
if (*(str2+24) == (char) 63) return(1);
if (*(str2+24) == (char) 37) return(1);
return(0);
}
/*****************************************************************************/
/*                                dt                                         */
/*****************************************************************************/


/* Three functions are to determine discrepancies in time : */
/* dt() returns direct time difference                      */
/* dt1() is main function used in delta()                   */
/* dt2() is used to check some of possible errors in time   */

#include <stdlib.h>
#include <string.h>

static int cal[2][13] = {0,31,59,90,120,151,181,212,243,273,304,334,365,
0,31,60,91,121,152,182,213,244,274,305,335,366};

long dt(str1,str2)
char *str1, *str2;
{
int yr1,yr2,mo1,mo2,da1,da2,hr,mn,sc,vis1,vis2;
long val;

yr1=chtoi(str1+6,4); vis1=yr1%4; if (vis1) vis1=0; else vis1=1;
yr2=chtoi(str2+6,4); vis2=yr2%4; if (vis2) vis2=0; else vis2=1;
mo1=chtoi(str1+11,2); if (mo1 < 1) mo1=1; else if (mo1 > 12) mo1=12;
mo2=chtoi(str2+11,2); if (mo2 < 1) mo2=1; else if (mo2 > 12) mo2=12;
da1=chtoi(str1+13,2); if (da1<1) da1=1;
da2=chtoi(str2+13,2); if (da2<1) da2=1;
hr=chtoi(str2+15,2)-chtoi(str1+15,2);
mn=chtoi(str2+17,2)-chtoi(str1+17,2);
sc=chtoi(str2+19,2)-chtoi(str1+19,2);

val=((((long)((yr2-yr1)*(365+vis1)+cal[vis2][mo2-1]-cal[vis1][mo1-1]+da2-da1)*24L+hr)*60+mn)*60+sc);

return(val);
}
/************************** dt1() ****************************/

long dt1(str1,str2)
char *str1, *str2;
{
long val,val1;
int j,j1,ndx,mn;
long dt();
long sc_mn(),sc_hr(),sc_da(),sc_yr();

for (j=11;j<21;j++) if (*(str1+j) == ' ') break;
/* if (!memcmp(str1+17,"00  ",4)) j=17; */
if (!memcmp(str1+19,"00   ",5) || !memcmp(str1+19,"00.00",5))  j=19;
for (j1=11;j1<21;j1++) if (*(str2+j1) == ' ') break;
/* if (!memcmp(str2+17,"00  ",4)) j1=17; */
if (!memcmp(str2+19,"00   ",5) || !memcmp(str2+19,"00.00",5))  j1=19;

ndx=(22-min(j,j1))/2;
if (!ndx) return(labs(dt(str1,str2)));

if (j1 == j)
 {
 val=labs(dt(str1,str2));
 if (!val) { if (ndx < 2) return(0L); return((long)ndx*30L); }
 if (!(val%3600L))
   {
   mn=chtoi(str1,17,2);
```

```
    if (!(mnt5)) return(val+(long)ndx*100L-60L);
    return (val+(long)ndx*120L-100L);
    }
  if (val < 1800L) return(val-30L); /* no seconds */
  val1=(val/3600L); j1=(val%3600L); if (j1 > 1800) { val1++; j1-=3600; }
  if (val1 < 3L) return(val - 30 * j1 / abs(j1));
  return (val - 40 * j1 / abs(j1));
  }

 switch(j)
  {
  case 21: /* seconds not blanked in first */
   switch(j1)
    {
    case 19:  case 20: return(sc_mn(str2,str1));
    case 17:  case 18: return(sc_hr(str2,str1));
    case 15:  case 16: return(sc_da(str1,str2));
    case 13:  case 14: return(mo_yr(str1,str2,1));
    case 11:  case 12: return(mo_yr(str1,str2,0));
    default: return(val);
    }
  case 19: case 20: /* seconds are blanked in first */
   switch(j1)
    {
    case 21: return(sc_mn(str1,str2));
    case 17:  case 18: return(sc_hr(str2,str1));
    case 15:  case 16: return(sc_da(str1,str2));
    case 13:  case 14: return(mo_yr(str1,str2,1));
    case 11:  case 12: return(mo_yr(str1,str2,0));
    default: return(val);
    }
  case 17: case 18: /* minutes are blanked in first */
   switch(j1)
    {
    case 21: return(sc_hr(str1,str2));
    case 19:  case 20: return(sc_hr(str1,str2));
    case 15:  case 16: return(sc_da(str1,str2));
    case 13:  case 14: return(mo_yr(str1,str2,1));
    case 11:  case 12: return(mo_yr(str1,str2,0));
    default: return(val);
    }
  case 15: case 16: /* hours are blanked in first */
   switch(j1)
    {
    case 21: return(sc_da(str1,str2));
    case 19:  case 20: return(sc_da(str1,str2));
    case 17:  case 18: return(sc_da(str1,str2));
    case 13:  case 14: return(mo_yr(str1,str2,1));
    case 11:  case 12: return(mo_yr(str1,str2,0));
    default: return(val);
    }
  case 13: case 14: /* days are blanked in first */
   switch(j1)
    {
    case 15: case 16: case 17: case 18: case 19: case 20: case 21:
                                    return(mo_yr(str1,str2,1));
    case 11:  case 12: return(mo_yr(str1,str2,0));
    default: return(val);
    }
  case 11: case 12: /* month is blanked in first */
    return(mo_yr(str1,str2,0));
  default: return(val);
  }
}

long sc_mn(blanked,nblanked)
char *blanked, *nblanked;
 {
 long t;
 int t3;
 long t1,t2;
 int mn;
 t=dt(blanked,nblanked); if (!t) return(0L); t2=labs(t);
 if (t2 < 60L) return(5L);
 mn=chtoi(blanked+17,2);
 t1=t2/3600L; t3=(t2%3600); if (t3 > 1800) { t1++; t3= 3600-t3; }
 if (t3 < 60)
  {
  if (memcmp(blanked+17,nblanked+17,2)) return(t1*3600L+20L);
  return(t1*3600L+8L);
  }
 if (!(mnt30))
  {
  if (t3 < 210) mn=1;
  else if (t3 < 300) return(t1*3600L+90L);
  else if (t3 < 450) return(t1*3600L+120L);
  else if (t3 < 600) return(t1*3600L+180L);
  else if (t3 < 900) return(t1*3600L+240L);
  else if (t3 < 1200) return(t1*3600L+300L);
  else return(t2);
  }
 else if (!(mnt15))
  {
  if (t3 < 150) mn=1;
  else if (t3 < 180) return(t1*3600L+90L);
  else if (t3 < 300) return(t1*3600L+120L);
  else if (t3 < 450) return(t1*3600L+240L);
  else if (t3 < 600) return(t1*3600L+300L);
  else return(t2);
  }
 else if (!(mnt10))
  {
  if (t3 < 150) mn=1;
  else if (t3 < 180) return(t1*3600L+120L);
```

```
    else if (t3 < 300) return(t1*3600L+150L);
    else return(t2);
    }
  t3=(t2/3600L); t1=t2%3600L; if (t1 > 1800L) t3++;
  if (t > 0L) t=(t3*3600L);  else t=(t3*3600L); t3=labs(t);
  t1=(t2%60L); t2=(t3/60L);
  if (!t2)
    {
    if (t1 < 30L) return(t3*3600L+20L);
    return(t3*3600L+30L);
    }
  if (t < 0L)
    {
    if (t1 < 30L) return(t3*3600L+t2*60L + 30L);
    return(t3*3600L+t2*60L + 60L);
    }
  if (t1 < 30L) return(t3*3600L+t2*60L - 30L);
  return(t3*3600L+t2*60L + 30L);

  }

long sc_hr(blanked,nblanked)
char *blanked, *nblanked;
 {
 long t; int t1; long t2;
 t=dt(blanked,nblanked);
 if (!(t%3600L)) return(labs(t)+150L);
 if (!(t%900L))
   {
   t1=labs(t)/86400L; t2=labs(t)%86400L;
   if (t2 == 84600L) return((long) t1 * 86400L + 86640L);
   if (t2 == 85500L) return((long) t1 * 86400L + 86550L);
   if (t2 == 83700L) return((long) t1 * 86400L + 86640L);
   if (t1 && (t2 == 1800L)) return((long) t1 * 86400L + 240L);
   if (t1 && (t2 == 900L)) return((long) t1 * 86400L + 150L);
   if (t1 && (t2 == 2700L)) return((long) t1 * 86400L + 240L);
   t=labs(t);
   if (t1 || (t2 > 50000L)) return(t);
   t2=t/3600L; t1=(t%3600L);
   if (t1 > 1800) { t2++; t1=3600-t1; }
   if (t1 == 900) return(t2*3600L+150L);
   return(t2*3600L+240L);
   }
 if (labs(t) > 50000)
   {
   t1=labs(t)/86400L; t2=labs(t)%86400L;
   if (t2 > 3600L) { t1++; t2=86400-t2; }
   if (t2 > 3600L) return(labs(t));
   if (t2 < 60L) return((long)t1*86400+10L);
   if (t2 < 180L) return((long)t1*86400+40L);
   if (t2 < 960L) return((long)t1*86400+90L);
   if (!memcmp(blanked+15,nblanked-15,2)) return((long)t1*86400+120L);
   if (t2 < 2110L) return((long)t1*86400L+150L);
   return((long)t1*86400L+300L);
   }
 if (!memcmp(blanked+15,nblanked+15,2))
   {
   if (t < 181L) return(10L);
   if (t < 301L) return(30L);
   if (t < 601L) return(60L);
   if (t < 900L) return(90L);
   if (t < 1800L) return(150L);
   return(180L);
   }
 t=labs(t);
 t2=t/3600L; t1=(t%3600L); if (t1 > 1800) { t2++; t1=3600-t1; }
 if (t1 < 180L) return(t2*3600L+100L);
 if (t1 < 300L) return(t2*3600L+120L);
 if (t1 < 600L) return(t2*3600L+150L);
 if (t1 < 900L) return(t2*3600L+180L);
 return(t2*3600L+240L);
 }

long sc_da(first,next)
char *first, *next;
 {
 long t,t1; int t2,t3;
 t=labs(dt(first,next));
 if (memcmp(first+13,next+13,2)) t3=1; else t3=0;
 t2=t/86400L; t1=(t%86400L); if (t3 && (t1 > 43200L)) {t2++; t1=86400-t1; }
 if ((t2==1) && (t1 < 36000L)) t2--;
 t=(long)t2*86400;
 if (t1 < 3600L) return(t+60L+30L*(long)t3);
 if (t1 < 7200L) return(t+90L+60L*(long)t3);
 return(t+150L+120L*(long)t3);
 }

long sc_yr(first,next,flag)
char *first, *next; int flag;
 {
 if (!memcmp(first+6,next+6,4+(flag*3))) return(600L);
 return(1000000L);
 }

/************************** dt2() ***************************/


long dt2(str1,str2)
char *str1, *str2;
{
int yr1,yr2,mo1,mo2,da2,da1,hr,hr1,mn,mn1,sc,sc1,vis1,vis2,damax2,damax1;
long val;
int stta;
```

```
stts=0;
yr2=chtoi(str2+6,4); vis2=yr2 % 4; if (vis2) vis2=0; else vis2=1;
yr1=chtoi(str1+6,4); vis1=yr1 % 4; if (vis1) vis1=0; else vis1=1;
if ((*(str1+11) == ' ') || (*(str2+11) == ' '))
                { mo1=1;  mo2=1; vis1=0; vis2=0; }
else
 {
 mo2=chtoi(str2+11,2);
 if ((mo2 < 1) || (mo2 > 12)) return(999999L);
 mo1=chtoi(str1+11,2);
 if ((mo1 < 1) || (mo1 > 12)) return(999999L);
 }
damax2=cal[vis2][mo2]-cal[vis2][mo2-1];
damax1=cal[vis1][mo1]-cal[vis1][mo1-1];
if ((*(str1+13) == ' ') || (*(str2+13) == ' ')) { da1=1; da2=1; }
else
 {
 da2=chtoi(str2+13,2); if ((da2 < 1) || (da2 > damax2)) return(999999L);
 da1=chtoi(str1+13,2); if ((da1 < 1) || (da1 > damax1)) return(999999L);
 }
sc=chtoi(str2+19,2); if ((sc < 0) || (sc > 59)) return(999999L);
sc1=chtoi(str1+19,2); if ((sc1 < 0) || (sc1 > 59)) return(999999L);
if ((*(str1+19) == ' ') || (*(str2+19) == ' ')) { sc=0; stts=5; }
else sc-=sc1;
mn=chtoi(str2+17,2); if ((mn < 0) || (mn > 59)) return(999999L);
mn1=chtoi(str1+17,2); if ((mn1 < 0) || (mn1 > 59)) return(999999L);
if ((*(str1+17) == ' ') || (*(str2+17) == ' ')) { mn=0; stts=4; }
else mn-=mn1;
hr=chtoi(str2+15,2); if ((hr < 0) || (hr > 23)) return(999999L);
hr1=chtoi(str1+15,2); if ((hr1 < 0) || (hr1 > 23)) return(999999L);
if ((*(str1+15) == ' ') || (*(str2+15) == ' ')) { hr=0; stts=3; }
else hr-=hr1;

val=((((long)(yr2-yr1)*365+
 cal[vis2][mo2-1]-cal[vis1][mo1-1]+da2-da1)*24L+hr)*60+mn)*60+sc);

return(labs(val));
}
/*****************************************************************************/
/*                                    dr                                     */
/*****************************************************************************/

#include <math.h>
#include <stdlib.h>

/* function returns integer value of distance between points specified as */
/* epicentres in first and next records */
/* The correction is used for cases of zero values of fractional parts of */
/* both latitude and longitude at least in one record */


dr(fi1,fi2,la1,la2)
int fi1,fi2,la1,la2;
{
float co;
long dl,df;
df=labs((long)fi1-(long)fi2);
dl=labs((long)la1-(long)la2); if (dl > 180000L) dl=360000L-dl;

/* if fractional parts in both latitude and longitude are zeroes */
if (((fi1%100) && !(la1%100)) || (!(fi2%100) && !(la2%100)))
 {
 if (df > 25L) df=labs(df-50L);
 if (dl > 25L) dl=labs(dl-50L);
 }
co=cos (0.000873 * (double)(fi1+fi2)); dl*=co;
dl*=dl;dl+=(df * df); df = (sqrt((double) dl)+0.5);
return((int) df);
}
/*****************************************************************************/
/*                                    dr1                                    */
/*****************************************************************************/


#include <string.h>
#include <stdlib.h>

/* function returns index of magnitudes difference in first and next records */

/* there are 3 types of magnitude difference:
   0 - type and source are same,
   1 - only type is same
   2 - minimal difference among all values

function returns:

0 - if no magnitude in one and mmin < mbig in other
1 - ddx <= ddx0[0][type][jcat]
2 - no magnitude in one and mmin >= mbig in other
    or ddx <= ddx0[1][type][jcat]
3 - ddx <= ddx0[2][type][jcat]
-1 - otherwise */



static int mbig=50;

static int ddx0[2][3][3][2] = {
0,0, 3,3, 4,4,  1,2, 5,5, 7,9,  6, 8, 18,18, 20,20,
1,0, 1,2, 3,4, 2,2, 3,5, 7,9, 7, 8, 8,12, 15,25};

dm(first,next,jcat,t)
char *first,*next;
```

```
int jcat,t;
{
int j,j1;
int m1[6],m2[6],mlmin,m2min,ddm,ddm1; /* magnitudes */

m1[0]=chtoi(first+53,3);
m1[1]=chtoi(first+58,3);
m1[2]=(chtoi(first+64,4)+5)/10;
m1[3]=(chtoi(first+75,4)+6)/10;
m2[0]=chtoi(next+53,3);
m2[1]=chtoi(next+58,3);
m2[2]=(chtoi(next+64,4)+5)/10;
m2[3]=(chtoi(next+75,4)+6)/10;

mlmin=100; for (j=0;j<4;j++) if (m1[j] && (m1[j] < mlmin)) mlmin=m1[j];
m2min=100; for (j=0;j<4;j++) if (m2[j] && (m2[j] < m2min)) m2min=m2[j];
if (mlmin == 100) mlmin=0;
if (m2min == 100) m2min=0;
if (mlmin < 10) mlmin=0;
if (m2min < 10) m2min=0;

if (!mlmin && (m2min < mbig)) return(0);
if (!m2min && (mlmin < mbig)) return(0);
if (!mlmin && (m2min >= mbig)) return(2);
if (!m2min && (mlmin >= mbig)) return(2);

if ((mlmin > 25) && (m2min > 25))
{

ddm=100;
for (j=0;j<2;j++)
for (j1=0;j1<2;j1++)
 {
 if (!m1[j+2]) continue;
 if (!memcmp(first+68+j*11,"UK",2)) continue;
 if (!memcmp(first+68+j*11,"MsSSR",5)) continue;
 if (!m2[j1+2]) continue;
 if (!memcmp(next+68+j1*11,"UK",2)) continue;
 if (!memcmp(next+68+j1*11,"MsSSR",5)) continue;
 ddm1=abs(m1[j+2]-m2[j1+2]);
 if (!memcmp(first+68+j*11,next+68+j1*11,5) && (ddm1 < ddm)) ddm=ddm1;
 }
if (!jcat)
 {
 if (m1[0] && m2[0]) ddm1=abs(m1[0]-m2[0]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 if (m1[1] && m2[1]) ddm1=abs(m1[1]-m2[1]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 }
if (ddm < 100)
 {
 for (j=0;j<3;j++) if (ddm <= ddm0[t][j][0][jcat]) break;
 if (j == 3) return(-1);
 return (j+1);
 }

/* same type */

ddm=100;
for (j=0;j<2;j++)
for (j1=0;j1<2;j1++)
 {
 if (!m1[j+2]) continue;
 if (!m2[j1+2]) continue;
 if (!memcmp(first+68+j*11,"MsSSR",5)) continue;
 if (!memcmp(next+68+j1*11,"MsSSR",5)) continue;
 ddm1=abs(m1[j+2]-m2[j1+2]);
 if (!memcmp(first+68+j*11,next+68+j1*11,5) && (ddm1 < ddm)) { ddm=ddm1;break; }
 if (!memcmp(first+68+j*11,"UK",2)) continue;
 if (!memcmp(next+68+j1*11,"UK",2)) continue;
 if (!memcmp(first+68+j*11,next+68+j1*11,2) && (ddm1 < ddm)) ddm=ddm1;
 }
if (m1[0] && jcat)
 {
 if (m2[2] && !memcmp(next+68,"ab",2)) ddm1=abs(m1[0] - m2[2]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 if (m2[3] && !memcmp(next+79,"ab",2)) ddm1=abs(m1[0] - m2[3]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 }
if (m2[0] && jcat)
 {
 if (m1[2] && !memcmp(first+68,"ab",2)) ddm1=abs(m2[0] - m1[2]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 if (m1[3] && !memcmp(first+79,"ab",2)) ddm1=abs(m2[0] - m1[3]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 }
if (m1[1] && jcat && memcmp(first,"SSR",3))
 {
 if (m2[2] && !memcmp(next+68,"Ms",2) && memcmp(next+70,"SSR",3))
    ddm1=abs(m1[1] - m2[2]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 if (m2[3] && !memcmp(next+79,"Ms",2) && memcmp(next+81,"SSR",3))
 ddm1=abs(m1[1] - m2[3]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 }
if (m2[1] && jcat && memcmp(next,"SSR",3))
 {
 if (m1[2] && !memcmp(first+68,"Ms",2) && memcmp(first+70,"SSR",3))
      ddm1=abs(m2[1] - m1[2]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 if (m1[3] && !memcmp(first+79,"Ms",2) && memcmp(first+81,"SSR",3))
      ddm1=abs(m2[1] - m1[3]); else ddm1=100;
 if (ddm1 < ddm) ddm=ddm1;
 }
```

```
  if (jcat)
   {
   if (m1[0] && m2[0]) ddm1=abs(m1[0]-m2[0]); else ddm1=100;
   if (ddm1 < ddx) ddx=ddx1;
   if (m1[1] && m2[1]) ddm1=abs(m1[1]-m2[1]); else ddm1=100;
   if (ddm1 < ddx) ddx=ddm1;
   }
  if (ddx < 100)
   {
   if (max(m1min,m2min) > 60) ddx--1;
   else if (max(m1min,m2min) > 70) ddx--2;
   for (j=0;j<3;j++) if (ddx <= ddx0[t][j][1][jcat]) break;
   if (j == 3) return(-1);
   return (j+1);
   }
  } /* if mmin > 29 */

  /* different types */

  ddx=100;
  for (j=0;j<4;j++)
  for (j1=0;j1<4;j1++)
   {
   if (!m1[j]) continue;
   if (!m2[j1]) continue;
   ddm1=abs(m1[j]-m2[j1]);
   if (ddm1 < ddx) ddx=ddm1;
   }
  if (ddx < 100)
   {
   if (max(m1min,m2min) < 35) ddx--2;
   else if (max(m1min,m2min) > 59) ddx--2;
   else if (max(m1min,m2min) > 62) ddx--3;
   else if (max(m1min,m2min) > 67) ddx--4;
   else if (max(m1min,m2min) > 72) ddx--6;
   for (j=0;j<3;j++) if (ddx <= ddx0[t][j][2][jcat]) break;
   if (j == 3) return(-1);
   return (j+1);
   }
  return(2);
  }
  /*****************************************************************************/
  /*                                chtoi                                      */
  /*****************************************************************************/

  /* This is an analog to standard atoi function, but it skips periods (.), */
  /* replaces blanks by zeroes, and stops at byte #(len-1) from the start  */

  chtoi(ptr,len)
  char *ptr;
  int len;
  {
  int j;
  int val;
  int factor;
  char c;
  factor=1;
  val=0;
  for (j=len-1;j>=0;j--)
   {
   c=*(ptr+j);
   if (c == ' ') c='0';
   else if (c == '.') continue;
   else if (c == '-') { val*=(-1); return(val); }
   else if (c == '+') return(val);
   val+=(factor*(c-'0'));
   factor*=10;
   }
  return(val);
  }
  /*********************/
  /*                   */
  /*      getkey()     */
  /*                   */
  /*********************/

  #include <bios.h>

  int getkey(void)
  /* Does the BIOS to read the next keyboard character */
  /* See MACROS in getkey.h                            */
  {
   int key, lo, hi;

   key = _bios_keybrd(_KEYBRD_READ);
   lo = key & 0x00ff;
   hi = (key & 0xff00) >> 8;
   return((lo == 0) ? hi + 256 : lo);
  }
  static int r_val[2][2][6][4][6] = {
  970,670,360,160,130,120,   999,680,320,250,170,130,   630,310,260,170,135,120,   400,249,200, 90, 80,  0,
  440,350,250,155,125,115,   975,370,310,245,148,125,   428,300,255,165,115,120,   380,160,140,130,120,  0,
  400,330,240,150,122,114,   950,330,285,240,133,124,   408,280,250,160,180,170,   350,160,130,120,110,  0,
  370,310,150,135,120,113,   900,300,275,145,130,123,   360,270,245,135,115,110,   300,150,120,110,100,  0,
  340,300,145,130,117,112,   800,280,265,140,127,122,   350,260,240,120,110,100,   200,140,110,100,  0,  0,
  330,240,140,125,115,110,   330,370,255,135,125,120,   300,250,235,190,185,180,   150,130,100,  0,  0,  0,

  886,815,611,470,204,148,   999,930,900,550,212,200,   888,843,647,540,234,146,   774,707,430,200,120,120,
  874,767,498,210,203,200,   893,792,667,540,200,154,   869,760,661,530,215,191,   681,646,372,120,120,120,
  864,757,494,203,200,200,   875,767,640,530,177,202,   859,750,630,520,160,141,   572,510,281,120,120,120,
  850,734,484,203,201,200,   864,760,622,518,200,157,   856,740,612,508,155,135,   499,427,220,120,120,120,
  848,704,271,204,203,200,   853,749,299,221,190,180,   842,730,286,211,158,133,   328,315,198,120,120,120,
  230,220,190,180,149,147,   240,230,210,190,185,160,   220,213,200,180,150,172,   200,196,159,125,120,120,
```

```
717,630,360,205,200,125,  853,735,370,270,207,130,  810,663,345,203,199,127,  634,471,256,150,100,100,
682,616,489,260,250,150,  770,680,385,300,270,205,  743,631,345,270,260,175,  616,523,331,200,190,170,
647,575,500,270,255,160,  711,646,466,335,308,206,  678,622,466,280,224,177,  612,571,377,170,150,140,
545,447,350,260,250,150,  651,554,330,345,260,150,  613,518,290,250,208,145,  549,489,208,190,147,113,
533,352,260,250,240,140,  588,360,207,203,200,180,  521,390,200,185,145,130,  404,289,185,140,130,112,
500,262,240,150,125,100,  500,270,205,152,130,105,  454,250,195, 90, 80, 90,  150, 90, 80, 50, 74, 92,

885,833,442,151,104,106,  930,834,480,328,240,100,  887,693,439,190,107,105,  793,493,276,133, 90, 50,
843,827,453,200,155,100,  922,851,490,270,185,150,  883,730,514,209,154,110,  772,567,367,214,114, 50,
850,795,388,193,150,100,  925,877,495,165,185,150,  893,821,576,250,153,100,  815,636,481,252,125,  0,
699,675,316,128,155,100,  865,855,465,180,135,130,  804,762,594,257,132, 70,  696,656,501,194, 84,  2,
304,356,257,197,160,100,  729,710,442,270,140,125,  668,894,454,220,108, 70,  506,424,248, 89, 35,  2,
166,291,293,247,108,100,  875,840,355,250,135,120,  945,461,374,216,105, 70,  637,289,185, 50, 26,  2};

static int epi_err[2][2][5][6][6] = {
250,250,200,160,150,140,  280,260,230,200,150,140,  210,200,150,140,130,120,  200,150,130,110,  0,  0,
265,255,250,  0,  0,  0,  275,265,255,  0,  0,  0,  270,260,250,  0,  0,  0,  250,250,125,  0,  0,  0,
290,290,  0,  0,  0,  0,  285,265,230,200,150,  0,  275,225,210,190,130,  0,    0,  0,100,100,  0,  0,
170,160,150,140,130,120,  180,170,160,150,140,130,  150,140,130,130,110,  0,  140,130,120,110,  0,  0,
170,160,150,140,130,120,  180,170,160,150,140,130,  150,140,130,120,110,  0,  140,130,120,110,  0,  0,

418,374,200,100,  0,  0,  429,408,250,  0,  0,  0,  409,274,  0,  0,  0,  0,  300,200,  0,  0,  0,  0,
400,350,250,200,198,180,  420,370,370,220,185,183,  400,350,260,190,166,166,  300,250,200,  0,  0,  0,
400,350,250,205,200,180,  420,370,370,220,205,183,  400,350,260,190,166,166,  300,250,200,  0,  0,  0,
170,160,150,140,130,120,  180,170,160,150,140,130,  150,140,130,120,110,  0,  140,130,120,110,  0,  0,
170,160,150,140,130,120,  180,170,160,150,140,130,  150,140,130,120,110,  0,  140,130,120,110,  0,  0,

290,280,270,260,250,240,  333,323,313,303,250,200,  323,313,303,193,183,165,  333,333,333,333,200,150,
400,250,200,195,190,180,  500,300,250,180,170,150,  450,357,300,200,180,100,  400,306,291,150,130,120,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
457,357,200,200,100, 50,  557,457,320,160,100, 50,  467,377,300,100, 50,  0,    0,  0,  0,  0,  0,  0,
369,314,301,224, 75,  0,  388,326,311,234,170,  0,  356,312,301,199,172,  0,  243,233,189,171,107,  0,
150,120,100, 90, 70, 50,  160,130,110,100, 80, 60,  155,125,105, 95,  0,  0,  100, 70, 50,  0,  0,  0,
150,120,100,  0,  0,  0,  160,130,110,  0,  0,  0,  155,125,105,  0,  0,  0,  100, 70,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0};

static int time_err[2][2][8][6][6] = {
210,154,153,152, 94,  0,  214,155,154,153,105,  0,  150,130,100, 90,  0,  0,   90,120,100,  0,  0,  0,
144,150,147,  0,  0,  0,  146,139,100,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
160,150,149,148, 78, 67,  154,153,152,151, 80,  0,  135,134,133,132, 78,  0,   85, 95, 95,  0,  0,  0,
150,140,139,128,101, 89,  135,119,116, 80, 80,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
155,153,152,  0,  0,  0,  137,153,140,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
250,150,  0,  0,  0,  0,  150,150,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
148,140,130,125,145,140,  149,130,120,144,136,133,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,

215,210,208,205,204,190,  225,220,218,210,205,195,  220,215,213,207,204,150,  206,204,176,151, 91, 51,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
296,278,265,202,201,170,  308,295,272,246,207,180,  218,212,209,208,206,150,  205,203,175,150, 90, 50,
210,206,198,197,200,150,  220,215,190,170,160,130,  215,202,150,130,  0,  0,   83, 83, 55, 50,  0,  0,
220,215,203,200,200,100,  230,220,205,200,140,140,  220,215,204,203,200,140,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  200,166,  0,  0,  0,  0,  230,200,  0,  0,  0,  0,  500,250,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,

250,200,197,194,  0,  0,  205,204,203,200,  0,  0,  210,195,150,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
230,180,170,  0,  0,  0,  241,200,195,190,  0,  0,  208,193, 90,  0,  0,  0,    0,  0,  0,  0,  0,  0,
200,190,160,  0,  0,  0,  210,200,166,195,  0,  0,  210,195, 90,  0,  0,  0,    0,  0,  0,  0,  0,  0,
150,140,130,  0,  0,  0,  100,200,220,200,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
210,206,  0,  0,  0,  0,  210,206,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,

220,210,200,160,140,120,  250,245,210,160,140,130,  235,225,205,155,135,125,  160,130,110, 90, 70,  0,
300,290,280,232,162,152,  310,300,290,250,100, 71,  150,100, 90, 70, 50,  0,    0,  0,  0,  0,  0,  0,
210,200,180,140,120,110,  230,220,200,148,130,  0,  225,215,195,145,125,  0,  150,120,100,  0,  0,  0,
200,190,170,120, 90,  0,  220,210,190,130,100,  0,  210,200,180,120,  0,  0,  140,110,100,  0,  0,  0,
150,130,110, 90,  0,  0,  160,145,120,100,  0,  0,  155,135,115, 80,  0,  0,  135,125,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0,    0,  0,  0,  0,  0,  0 };

static int addition[2][9]={
   999, 693, 800, 900, 612, 466, 761, 623, 300,
   999, 700, 500, 600, 471, 390, 830, 876, 100 };

static long dtmax[6] = { 320000000L, 2650000L, 2630000L, 1800000L, 870000L, 870000L };
static int t_w[2][2][6] = {
6, 31, 91, 149, 181, 600,
12, 31, 91, 155, 310, 900,
6, 30, 50, 90, 150, 450,
10, 20, 30, 70, 120, 550};

static int e_w[2][2][6] = {
25, 50, 75, 150, 300, 650,
25, 50, 75, 200, 500, 1500,
5, 25, 50, 100, 200, 400,
25, 40, 80, 130, 300, 800};

static int n_src[2] = { 3, 5 };
static char src[5][4]={"JTA","EUR","ESR","ERA","NOS"};
static int n1_src[2] = {8 , 9};
static char src1[9][6]={"IES***","ESR***","NOS***","IGN***","BDA***","GUT***","EURTI*","ERA***","BCI***"};


static int delta1[2][2]={250,200,350,150};
static int delta2[2][2]={150,100,210,50};
int yr[6]={1600,1800,1900,1950,1961,3000};
/*
 350 210 800 150 50 800
 250 150 800 200 100 800
```

```
*/

static long loc_t[2][2] = { 121L, 151L, 30L, 120L };
static int err_b[2][2] = { 91, 151, 45, 120 };
static int err_d[2][2] = { 121, 151, 45, 120 };
static int err_m[2][2] = { 6, 15, 3, 10 };
static long t_fit[2][2][8] = {
121L, 61L, 61L, 31L, 31L, 8L, 8L, 1L,
180L, 120L, 120L, 60L, 60L, 10L, 10L, 1L,
0L, 0L, 0L, 15L, 15L, 8L, 8L, 1L,
0L, 0L, 0L, 30L, 30L, 10L, 10L, 3L};
#define BACKSPACE      8          /* back space key */
#define FORWARD       12
#define ENTER         13          /* carriage return (enter) */
#define ESC           27
#define PLUS          43
#define MINUS         45
#define HOME         327
#define END          335
#define UP           328          /* up arrow */
#define DOWN         336          /* down arrow */
#define PGUP         329
#define PGDN         337
#define LEFT         331          /* left arrow */
#define RIGHT        333          /* right arrow */
#define INSERT       338
#define DEL          339
#define CTRL_LEFT    371
#define CTRL_RIGH    372
#define CTRL_HOME    375
#define CTRL_END     373
#define TAB            9
#define SHIFT_TAB    271
#define F1           315
#define F2           316
#define F3           317
#define F4           318
#define F5           319
#define F6           320
#define F7           321
#define F8           322
#define F9           323
#define F10          324
#define ALT_HOME       7
#define ALT_END        1


/*************************************************************************/
/*                          dpl_one - main program                       */
/*************************************************************************/


#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <malloc.h>
#include <ctype.h>
#include <math.h>
#include <doc.h>
#define PERFUM 35

/* This program chooses one record from each cluster of duplicates in .dpl file */

struct data
  {
  char rec[116];
  int prior;
  struct data *p;
  };

main(argc,argv)
int argc;
char **argv;
{
char ipfile[30];
char opfile[30];
FILE *ipf,*opf;

struct data *first,*next,*last,*work;
struct data *b_first,*b_work,*b_next;

struct info
  {
  char memrec[6];
  char ta[3];
  long num;
  long num0;
  struct info *p;
  struct info *pb;
  } *info_first,*info_last,*info_next,*info_work;

long ir=0L,iw=0L;
int jcat=0;
int reclen=118;
char inrec[130];
int prior_max=31999;
int flag=0;
struct find_t *file_info;
long num_rec=0L;
double num_print=0.0;
double step=0.0;        /* num_rec/79 */

int j=0,jr=0,ji=0,grs=0,grs1=0,info_w=0;
```

```
ipfile[0]=' ';
opfile[0]=' ';
inrec[0]=' ';

if (argc < 2)
 {
 printf("This program reads input .DPf file, chooses one record from\n");
 printf("each cluster of duplicates according default syster of priorities\n");
 printf("for data sources and information completeness, and writes chosen\n");
 printf("records to output .DPL file\n");
 printf("FORMAT: dpl_one <input file name [.dpf]> [<output file name [.dpl]> \n");
 exit(1);
 }
strcpy(ipfile,argv[1]);
if (argc < 3)  strcpy(opfile,ipfile);
else if (argv[2][0] != '+') strcpy(opfile,argv[2]);
else { info_v=1; strcpy(opfile,ipfile); }
if (argc > 3) info_v=1;

for (j=1;j<strlen(ipfile);j++)
 if (*(ipfile+j) == '.') break;
if (*(ipfile+j) != '.') strcpy(ipfile+j,".dpf");
for (j=1;j<strlen(opfile);j++)
 if (*(opfile+j) == '.') break;
if (*(opfile+j) != '.') strcpy(opfile+j,".dpl");

/* determine number of records */
file_info=(struct find_t *) malloc(sizeof(struct find_t));
_dos_findfirst(ipfile,_A_NORMAL,file_info);
num_rec=file_info->size;
free(file_info);

if ((ipf=fopen(ipfile,"rb")) == NULL)
 { printf("%s not found, exiting ...\n",ipfile); exit(1); }
if ((opf=fopen(opfile,"wb")) == NULL)
 { printf("%s cannot be open, exiting ...\n,opfile"); exit(1); }

/* determine length of record */
fread(inrec,130,1,ipf);
for (reclen=0;reclen<130;reclen++) if (*(inrec+reclen) == '\n') break; reclen++;
fseek(ipf,0L,SEEK_SET);
num_rec/=(long)reclen;

printf("\n\n\n\n\n\n**********************************************************************************");
  printf("**                                                                              **");
  printf("**       REMOVAL OF DUPLICATES FROM THE FILE PRODUCED BY DUPLI_VX PROGRAM       **");
  printf("**                                                                              **");
  printf("**                              according to default rule                       **");
  printf("**                                                                              **");
  printf("**                                   VERSION 1.0                                **");
  printf("**                                                                              **");
  printf("**********************************************************************************\n\n\n");

printf("\n%ld  records to be processed\n",num_rec);
printf("\n0.....................................................................\r");
step=(double)num_rec/79.0;
num_print=step;
info_first=(struct info *) NULL;
first=(struct data *) malloc (sizeof(struct data));
jr=fread(first->rec,116,1,ipf); ffseek(ipf);
ir=1L;
b_first=(struct data *) NULL;
while (first || b_first)
 {
/* write the elements from buffer if it is a time */
 while(b_first && ((cmp(b_first->rec,first->rec) < 0) || (first == (struct data *) NULL)))
    {
    if (!fwrite(b_first->rec,116,1,opf)) {printf("\nDisk full"); exit(1); }
    fwrite("\r\n",2,1,opf);
    iw++;
    b_work=b_first;
    b_first=b_first->p;
    free(b_work);
    }
 if (!first) break;
 last=first;
 qrs=1;
 for (;;)
  {
  last->p=(struct data *) malloc (sizeof(struct data));
  if (jr) jr=fread(last->p->rec,116,1,ipf);
  if (!jr) { last->p=(struct data *) NULL; break; }
  ffseek(ipf);
  last=last->p; ir++;
  if (ir > (long) num_print) { printf("."); num_print+=step; }
  if ((last->rec[115] == ' ') || (last->rec[115] == '+')) break;
  qrs++;
  }
qrs1=qrs;
for (j1=0;j1<4;j1++)
 {
 if (qrs < 2) break;
 next=first;
 prior_max=31999;
 for (j=0;j<qrs;j++)
   {
   next->prior=priority(next->rec,j1);
   if (next->prior < prior_max) prior_max=next->prior;
   next=next->p;
   }
 while (first->prior > prior_max) { work=first; first=first->p; qrs--; free(work); }
 next=first;
 for (j=1;j<qrs;j++)
   if (next->p->prior > prior_max) { work=next->p; next->p=next->p->p; qrs--; /*j--;*/ free(work); }
```

```
      else next=next->p;
    }
  /* free all but first elements */
  for(j=1;j<qrs;j++) { work=first->p; first->p=first->p->p; free (work); }

  /* last byte in record */
  if (qrs1 < 10) first->rec[115]=qrs1+'0';
  else first->rec[115]='*';

  /* statistics */
  if (info_w)
    {
    flag=0;
    if (!info_first)
       {
       info_last = (struct info *) malloc(sizeof (struct info));
       info_last->pb=(struct info *) NULL;
       memcpy(info_last->source,first->rec,5);
       memcpy(info_last->ts,first->rec+24,2);
       info_last->num=0L;
       info_last->num0=0L;
       info_last->p=info_first;
       info_first=info_last;
       }
    else
       {
       info_next=info_first;
       while (memcmp(first->rec,info_next->source,5) || memcmp(first->rec+24,info_next->ts,2))
        if (info_next->p)  info_next=info_next->p;
       else {flag=1; break;}
       if (flag)
          {
          info_next->p=(struct info *) malloc(sizeof(struct info));
          info_last=info_next->p;
          info_last->pb=info_next;
          info_last->p=(struct info *) NULL;
          info_last->num=0L;
          info_last->num0=0L;
          memcpy(info_last->source,first->rec,5);
          memcpy(info_last->ts,first->rec+24,2);
          }
       else info_last=info_next;
       }
    info_last->num++;
    if (!memcmp(first->rec+53,"   ",3) && !memcmp(first->rec+58,"   ",3)
        && !memcmp(first->rec+64,"   ",3) && !memcmp(first->rec+75,"   ",3)) info_last->num0++;
    }

 /* write if ordered correctly */
 /*  if (first->p && (cmp(first->rec,first->p->rec) < 0)) out(first,opf); */
  if (first->p && (cmp(first->rec,first->p->rec) < 0))
     {
     if (!fwrite(first->rec,116,1,opf)) {printf("\nDisk full"); exit(1); }
     fwrite("\r\n",2,1,opf);
     iw++;
     work=first;
     first=first->p;
     free(work);
     }
  else    /* add first to buffer */
     {
     if (!b_first) { b_first=first; first=first->p; b_first->p=(struct data *) NULL; }
     else if ( cmp(first->rec,b_first->rec) < 0 )
              { work=first->p; first->p=b_first; b_first=first; first=work; }
     else /* sort */
        {
        b_work=b_first;
        while(b_work->p && (cmp(first->rec,b_work->p->rec) >= 0)) b_work=b_work->p;
        last=first->p; first->p=b_work->p; b_work->p=first; first=last;
        }
     }
  last=first;
  }
fclose(ipf);
fclose(opf);
printf("\n\n%ld records read from %s, %ld written to %s\n",
ir,ipfile,iw,opfile);

/* print statistics if required */

if (info_w)
 {
 strcpy(inrec,opfile);
 for (j=1;j<strlen(opfile);j++) if (*(opfile+j) == '.') break;
 strcpy(opfile+j,".pri");
 opf=fopen(opfile,"w"); if (!opf) exit(0);

fprintf(opf,"%ld record read from %s, and %ld written to %s\n\n",
ir,ipfile,iw,inrec);

/* sort */
info_next=info_first->p;
info_first->p=(struct info *) NULL;
info_last=info_first;

while(info_next)
 {
 info_work=info_first;
 flag=0;
 while (memcmp(info_next->source,info_work->source,5) > 0)
   {
   if (info_work->p) info_work=info_work->p;
   else { flag=1; break; }
   }
```

```
    if (!flag) while (memcmp(info_next->ta,info_work->ta,2) > 0)
      {
      if (info_work->p) info_work=info_work->p;
      else { flag=1; break; }
      }
    if (flag)
      {
      info_last->p=info_next;
      info_next->pb=info_last;
      info_last=info_last->p;
      info_next=info_next->p;
      info_last->p=(struct info *) NULL;
      }
    else  if (info_work->pb)
      {
      info_work->pb->p=info_next;
      info_next->pb=info_work->pb;
      info_work->pb=info_next;
      info_next=info_next->p;
      info_work->pb->p=info_work;
      }
    else
      {
      info_work=info_next;
      info_next=info_next->p;
      info_work->p=info_first;
      info_first->pb=info_work;
      info_first=info_work;
      info_first->pb=(struct info *) NULL;
      }
    }
    while(info_first)
      {
      info_first->source[5]='\0';
      info_first->ta[2]='\0';
      fprintf(opf,"%5s\"%2x\" - %6ld (%ld no magnitude)\n", info_first->source,info_first->ta,info_first->num,info_first->num0);
      info_first=info_first->p;
      }
    printf("see statistics in %s",opfile);
    }
  exit(0);
  }

  /*****************************************************/

  ffseek(ffff)
  FILE *ffff;
   {
   char a=' ';
   for (;;)
    {
    fread(&a,1,1,ffff); if (a == '\n') break;
    }
   return(1);
   }

  /*****************************************************************/

  priority(str,num_test)
  char *str;
  int num_test;
   {
  static char src[PRNUM][5] = {
  "CGS  ","NOS  ","ERL  ","GS   ",
  "CGS**","NOS**","ERL**","GS **",
  "CGS% ","NOS% ","ERL% ","GS % ",
  "CGS? ","NOS? ","ERL? ","GS ? ",
  "CGS* ","NOS* ","ERL* ","GS * ",
  "PDE**","PDE  ","PDE% ","PDE? ",
  "PDE* ","GTTEE","G-REE","CDMEE",
  "MGDEE","EMSEE","SSREE","EURAR",
  "ABEEE","BDAEE","LEEEE"};
  int return_value;
  int nat,j;

  return_value=0;
  switch (num_test)
   {
  case 0:   /* data source and time/location authority */
   for (return_value=0; return_value < PRNUM; return_value++)
     {
     if (memcmp(str,src[return_value],3)) continue;
     if (!memcmp(src[return_value]+3,"EE",2)) return(return_value);
     if (!memcmp(src[return_value]+3,"***",2))
       {
       if (memcmp(str+24,"  ",2) && memcmp(str+24,"I ",2) && memcmp(str+24,"LT",2)
       && memcmp(str+24,"* ",2)  && memcmp(str+24,"% ",2) && memcmp(str+24,"? ",2))
            return(return_value);
       else continue;
       }
     if (!memcmp(src[return_value]+3,"  ",2) &&
     !memcmp(str+24,"  ",2) || !memcmp(str+24,"I ",2) || !memcmp(str+24,"LT",2))
       return(return_value);
     else continue;
     if (!memcmp(src[return_value],str+24,2)) return(return_value);
     }
  return(return_value);

  case 1:   /* presence of magnitude */
    if (!memcmp(str+53,"    ",3) && !memcmp(str+58,"    ",3)
        && !memcmp(str+64,"    ",3) && !memcmp(str+75,"    ",3)) return(1);
    return (0);

  case 2:   /* number of stations */
```

```
  if ((*(str+91) >= '0') && (*(str+91) <= '9')) mst=atoi(str+89,3);
      else mst=0;
  if (!memcmp(str+24,"XX",2)) mst=0;
  /* if (mst > 199) mst=0; */
  if (mst < 7) mst=0;
  return(200 - mst);

case 3:  /* mb or MB given, or source of Bbgl or Bbg? - data source */
  if (memcmp(str+53,"   ",3) || memcmp(str+38,"   ",3)
  || !memcmp(str,str+70,3) || !memcmp(str,str+75,3)) return(0);
  return(1);

case 4:  /* number of blanked position of the format */
  for (j=11;j<24;j++) if (*(str+j) == ' ') return_value++;
  for (j=46;j<49;j++) if (*(str+j) == ' ') return_value++;
  for (j=53;j<64;j++) if (*(str+j) == ' ') return_value++;
  for (j=75;j<81;j++) if (*(str+j) == ' ') return_value++;
  for (j=89;j<93;j++) if ((*(str+j) == ' ') || (*(str+j) == '.')) return_value++;
  return(return_value);
 }
}

/*************************************************************************/
cmp(str1,str2)
char *str1,*str2;
{
int j;
for (j=7;j<25;j++)
 {
 if (*(str1+j) < *(str2+j)) return(-1);
 else if (*(str1+j) > *(str2+j)) return(1);
 }
/*for (j=0;j<7;j++)
 {
 if (*(str1+j) < *(str2+j)) return(-1);
 else if (*(str1+j) > *(str2+j)) return(1);
 }*/
return(0);
}
```

APPENDIX III

The source code of the program to remove aftershocks.

Comment: This program determines the catalog of main shocks for a given catalog of earthquakes. It uses rather crude, but generally accepted, definition of aftershocks by a set of magnitude dependent temporal and spatial windows (e.g. Gardner and Knopoff, 1974, BSSA, 64, 1363-1367).

In the M8 test we use the following limits introduced by Keilis-Borok et al. (1980, JGR, 85, 803-811):

Magnitude, M	4.0	4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0+
R(M), km	40	40	50	50	50	100	100	150	200
T(M), days	23	46	91	183	183	365	730	913	1096

The program estimates R by the following approximation:

$$R = (111.1111 \cdot [(\varphi_1 - \varphi_2)^2 + (\lambda_1 - \lambda_2)^2 \cdot \cos^2((\varphi_1 + \varphi_2)/2)])^{1/2},$$

where (φ_1, λ_1) and (φ_2, λ_2) are coordinates in degrees.

* * *

```
/* This program determines aftershocks using values of parameters */
/* of standard algorithm fixed for the test of M-8 algorithm      */
/* Both input and output files are in 20-bytes format             */

#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <math.h>
#include <stdlib.h>

#define HUGE far
#define MALLOC _fmalloc
#define FREE _ffree

struct INPUT { long t; int lat; int lon; int h; int m[5]; } ;

struct MAIN_SHOCK { struct INPUT fmt20; char m_int; long offset;
                    struct MAIN_SHOCK HUGE *p; } ;

struct SELECT_PAR { int m_main; int m_aft; };

int m_m_int=9;
int  m_thresh[9]={ 649, 699, 749, 799, 649, 699, 749, 799, 900 };
long t_thresh[9]= { 331201, 662401, 1310401, 2635201, 2635201, 5256001,
                                                10512001, 13147201, 15768001 };
/*
long ti_thresh[2][9]= {
20160L, 20160L, 20160L, 20160L, 20160L, 20160L, 20160L, 20160L, 20160L,
331201L, 662401L, 1310401L, 2635201L, 2635201L, 5256001L, 10512001L, 13147201L, 15768001L };
*/
long ti_thresh=20160L;
double r_thresh[9] = { 40.0, 40.0, 50.0, 50.0, 50.0, 100.0, 100.0, 150.0, 200.0};
struct SELECT_PAR *select_par;

static double pi=3.14159265;

main(argc,argv)
int argc;
char **argv;
```

```
{
FILE *ipf,*opf;
struct INPUT *input;
struct MAIN_SHOCK HUGE *main_first, HUGE *main_next, HUGE *main_work;
int yr_start=0;
int m_int=0;
int m=0;
long ir=0L,im=0L;
int j=0,flag=0;
double dr();
pi/=18000.0;

if (argc < 3) {printf ("\nformat: aftshck <input file name> <output file name>\n");
exit(1); }
if ((ipf=fopen(argv[1],"rb")) == NULL) {printf ("\Unable to find %s",argv[1]); exit(1); }
if ((opf=fopen(argv[2],"wb+")) == NULL) {printf ("\Unable to open %s",argv[2]); exit(1); }

select_par=(struct SELECT_PAR *) malloc(sizeof (struct SELECT_PAR));
select_par->m_min=400;
select_par->m_aft =400;

input=(struct INPUT *) malloc(sizeof(struct INPUT));
fread((char *) input,20,1,ipf);
fseek(opf,0L,SEEK_SET);
fwrite((char *)input,20,1,opf); /* record showing number of records */
printf("\n%6ld  records in input file", input->t);
ir=1L; im=1L;
for (;;)
 { fread((char *) input, 20, 1, ipf); ir++; if (select(input,select_par,&m)) break; }
main_first=(struct MAIN_SHOCK HUGE *) MALLOC (sizeof (struct MAIN_SHOCK));
if (!main_first) {printf("\nSorry, memory is not enough to perform aftershocks removal\n");
  exit(1); }
im++;
add_main(main_first,input,opf,m,im);
printf("\n\n      0-th  record being processed");

while(fread((char *) input,20,1,ipf))
  {
  ir++;
  if (!(ir%100L)) printf("\r%6ld",ir);
  if (!select(input,select_par,&m)) continue;

/* remove first if necessary */
  while (main_first->p)
    {
    if ((main_first->fst20.t+t_thresh[main_first->m_int]) < input->t)
       {
       if (j=main_first->fst20.m[1])
          {
          fseek(opf,main_first->offset+12L,SEEK_SET);
          fwrite((char *)(&j),2,1,opf);
          }
       main_work=main_first;
       main_first=main_first->p;
       FREE(main_work);
       }
    else break;
    }

/* check magnitude of first */
  if (main_first->fst20.m[0] == m)
    {
    main_work=main_first; flag=0;
    while(main_work && (main_work->fst20.m[0] == m))
      {
      if (aftershock(main_work,input)) {flag=1; break;}
      main_work=main_work->p;
      }
    if (flag) continue;
    }
  if (m >= main_first->fst20.m[0])
    {
    im++;
    main_next=(struct MAIN_SHOCK HUGE *) MALLOC (sizeof (struct MAIN_SHOCK));
    add_main(main_next,input,opf,m,im);
    main_next->p=main_first;
    main_first=main_next;
    continue;
    }
/* check if it is aftershock of first */
  if (aftershock(main_first,input)) continue;

/* second main and further */
  main_next=main_first;
  while (main_next->p)
    {
/* check if to remove from list */
    if ((main_next->p->fst20.t+t_thresh[main_next->p->m_int]) < input->t)
       {
       if (j=main_next->p->fst20.m[1])
          {
          fseek(opf,main_next->p->offset+12L,SEEK_SET);
          fwrite((char *)(&j),2,1,opf);
          }
       main_work=main_next->p;
       main_next->p=main_next->p->p;
       FREE(main_work);
       if (!main_next->p) break;
       continue;
       }
```

```
/* check if magnitude is greater - add to list of main shocks */
    if (main_next->p->fmt20.m[0] == m)
       {
       main_work=main_next->p; flag=0;
       while(main_work && (main_work->fmt20.m[0] == m))
         {
         if (aftershock(main_work,input)) {flag=1; break;}
         main_work=main_work->p;
         }
       if (flag) break;
       }
    if (main_next->p->fmt20.m[0] <= m)
       {
       im++;
       main_work=(struct MAIN_SHOCK HUGE *) MALLOC (sizeof (struct MAIN_SHOCK));
       if (!main_work) { printf("\nSorry, memory is not enough to perform aftershocks removal"); exit(1); }
       add_main(main_work,input,opf,m,im);
       main_work->p=main_next->p; main_next->p=main_work;
       break;
       }

/* check if it is an aftershock */
    if (aftershock(main_next->p,input)) break;
    main_next=main_next->p;
    }
/* add to the end */
  if (!main_next->p)
    {
    main_next->p=(struct MAIN_SHOCK HUGE *) MALLOC (sizeof (struct MAIN_SHOCK));
    if (!main_next->p) { printf("\nSorry, memory is not enough to perform aftershocks removal"); exit(1); }
    main_next=main_next->p;
    im++;
    add_main(main_next,input,opf,m,im);
    }
  }
/* write values of parameters from the list of main shocks not removed yet */
while(main_first)
 {
 fseek(ipf,main_first->offset+12L,SEEK_SET);
 if (j=main_first->fmt20.m[1])
   {
   fseek(opf,main_first->offset+12L,SEEK_SET);
   fwrite((char *)(&j),2,1,opf);
   }
 main_first=main_first->p;
 }
fseek(opf,0L,SEEK_SET);
fwrite((char *) &im,4,1,opf);
fseek(opf,0L,SEEK_END);
fclose(opf);
printf("\n%4ld  records read from %s, %ld main shocks written to %s",
ir,argv[1],im-1,argv[2]);
}

add_main(main_str,inp_str,fl,magnitude,num_main)
struct MAIN_SHOCK HUGE *main_str;
struct INPUT *inp_str;
int magnitude;
long num_main;
FILE *fl;
{
int j;
main_str->fmt20.t=inp_str->t;
main_str->fmt20.lat=inp_str->lat;
main_str->fmt20.lon=inp_str->lon;
main_str->fmt20.h=inp_str->h;
main_str->fmt20.m[0]=magnitude;
inp_str->m[0]=magnitude;
for (j=1; j < 9; j++) { main_str->fmt20.m[j]=0; inp_str->m[j]=0; }
for (j=0;j<n_m_int;j++) if (magnitude <= m_thresh[j]) break;
main_str->m_int = (char) j;
main_str->offset=num_main*20L-20L;
main_str->p=(struct MAIN_SHOCK HUGE *) NULL;
fseek(fl,num_main*20L-20L,SEEK_SET);
if (!fwrite ((char *) inp_str, 20, 1, fl)) {printf("\nDisk full"); exit(1);}
}

aftershock(main_str,inp_str)
struct MAIN_SHOCK HUGE *main_str;
struct INPUT *inp_str;
{
if ((inp_str->t-main_str->fmt20.t) > t_thresh[main_str->m_int]) return(0);
/* if (dr(main_str->fmt20.lat,inp_str->lat,main_str->fmt20.lon,inp_str->lon)
                                  > r_thresh[main_str->m_int]) */
if (!dr1(main_str->fmt20.lat,inp_str->lat,main_str->fmt20.lon,inp_str->lon,r_thresh[main_str->m_int]))
                                                  return(0);

/* number of aftershocks and sigma */
/* if ((inp_str->t-main_str->fmt20.t) <= t1_thresh[0][main_str->m_int]) */
if ((inp_str->t-main_str->fmt20.t) <= t1_thresh)
main_str->fmt20.m[1]++;
/* if ((inp_str->t-main_str->fmt20.t) <= t1_thresh[1][main_str->m_int])
main_str->fmt20.m[2]++; */

return(1);
}
```

```
/*****************************************************************************/
/*                                   dr                                      */
/*****************************************************************************/

double dr(fi1,fi2,la1,la2)
int fi1,fi2,la1,la2;
{
float co;
long dl,df;
df=labs((long)fi1-(long)fi2);
dl=labs((long)la1-(long)la2); if (dl > 180000L) dl=360000L-dl;
co=cos (0.0000873 * (double)(fi1+fi2)); co*=co;
return((double)sqrt((float)(df*df) + (float)(dl*dl)*co)*1.111111);
}

select(inp_str,param,maq)
struct INPUT *inp_str;
struct SELECT_PAR *param;
int *maq;
{
if ((*maq=max(max(inp_str->m[0],inp_str->m[1]),max(inp_str->m[2],inp_str->m[3])))
   < min(select_par->m_main, select_par->m_aft)) return(0);
return(1);
}

int dr1(fi1,fi2,la1,la2,limit)
int fi1,fi2,la1,la2;
double limit;
{
long dla;
double q1,q2,q3,csc1,csc0;
if (abs(fi1-fi2) > (int)limit) return(0);
dla=labs((long)la1-(long)la2); if (dla > 180000L) dla=360000L-dla;
if ((double)(dla*cos((float)(pi*(fi1+fi2)/2.0))> limit) return(0);
q2=pi*fi1;
q1=cos((float)q2);
q2=sin((float)q2);
q3=cos((float)(pi*dla));
csc0=pi*fi2;
csc1=sin((float)csc0);
csc0=cos((float)csc0);
if ((q2*csc1+q1*q3*csc0) < (double)cos((float)(limit/6378.16))) return(0);
return(1);
}
```

APPENDIX IV

The M8 Algorithm: Description and Source Code

The prototype of the M8 algorithm was designed by V.I. Keilis-Borok and V.G. Kossobokov in 1984, and presented at the 27th Geological Congress in Moscow (Keilis-Borok and Kossobokov, 1984). The algorithm was post-predicting the occurrence of the magnitude 8.0-and-above earthquakes worldwide. In 1985 this algorithm was simplified and reassembled for application to predict smaller magnitude earthquakes, preserving the scheme and general definition of functionals (Keilis-Borok and Kossobokov, 1986). This modification was named the M8 algorithm when it was retroactively tested on 12 large earthquakes that occurred in seismic regions, mainly in the USSR territory (Gabrielov et al., 1986). During the next few years the M8 algorithm was tested in applications to postdict large earthquakes in other territories. The results of these studies are summarized in (Keilis-Borok and Kossobokov, 1990): 39 out of 44 large earthquakes, with magnitudes ranging from 4.9 to above 8.0, were postdicted in 19 seismic regions; the alarm times (TIP's) covered less than 20 percent of the total space-time considered. By 1990, two earthquakes had been predicted in advance: the Spitak (Armenia) earthquake on December 7, 1988, M = 6.9, and the Loma Prieta (California) earthquake on October 18, 1989, M = 7.1. The predictions were published prior to the earthquakes (see Gabrielov et al., 1986; Keilis-Borok and Kossobokov, 1988; Updike, 1989, page 10 of Appendix A).

In 1990, within the framework of Project III of the US-USSR Cooperative Program in Earthquake Prediction, a program directed by the U.S. Geological Survey under the auspices of the Environmental Protective Agency we started the monitoring of seismicity of the Circum-Pacific. Each half-year we updated the global catalog, and V.G. Kossobokov communicated the results obtained using the M8 algorithm to James W. Dewey of the USGS in Golden, Colorado, and to John H. Healy of the USGS in Menlo Park, California (Figures 1-3).

The update on January 1, 1991 issued the prediction of an earthquake of magnitude 7.5-and-above for the territory of Costa Rica and Panama (Figure 4). This prediction was discussed with Federico Guendel of the Observatorio Vulcanologico y Sismológico de Costa Rica and other participants of the Workshop on Prediction of Earthquakes held in Caracas, Venezuela (February 26-March 12, 1991). The April 22, 1991, M = 7.5, earthquake in Costa Rica confirmed this prediction.

Figure 1. TIP's in January 1, 1990 - July 1, 1990.

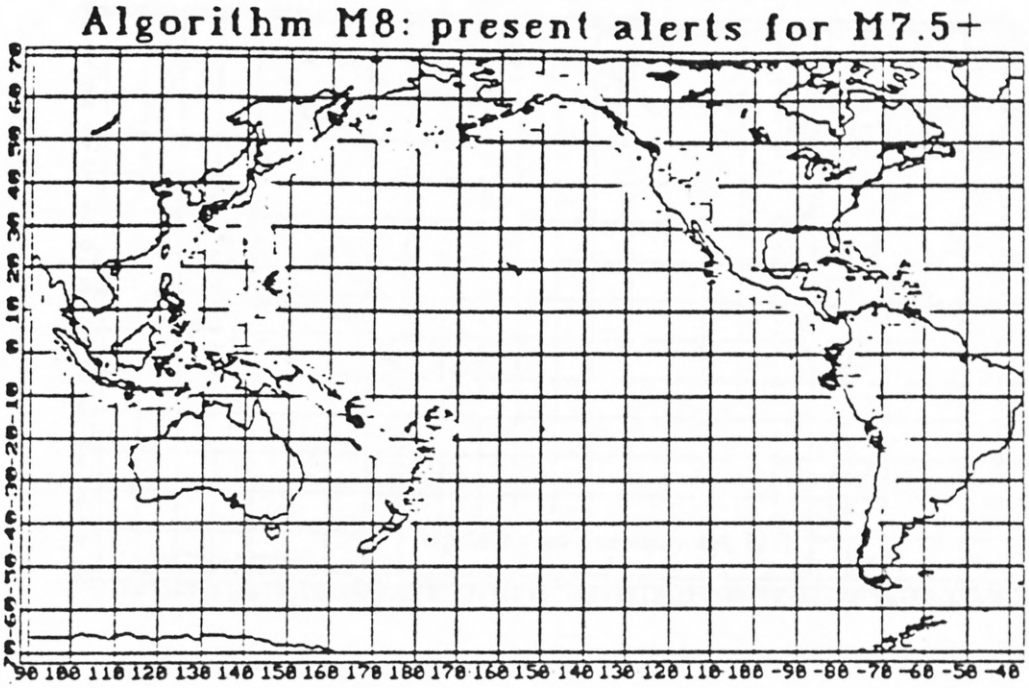


Figure 2. TIP's in July 1, 1990 - January 1, 1991.

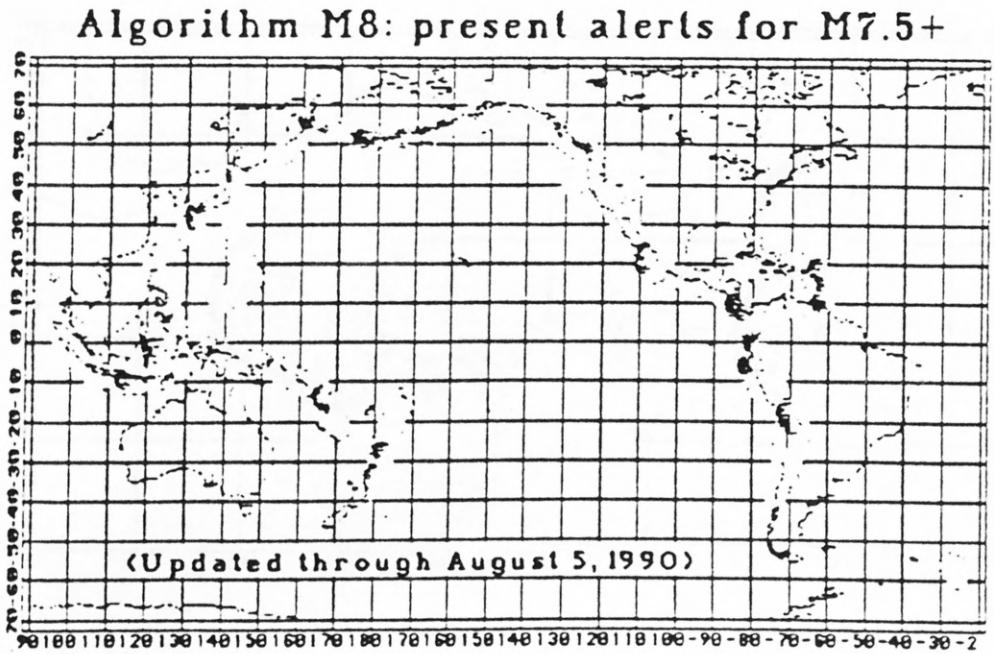


Figure 3. TIP's in January 1, 1991 - July 1, 1991.

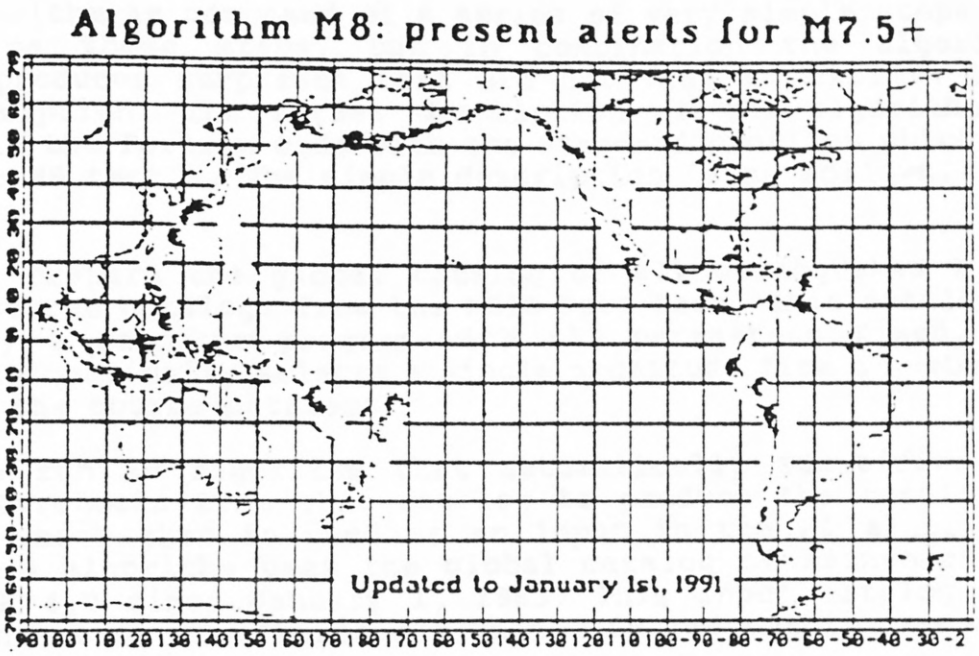
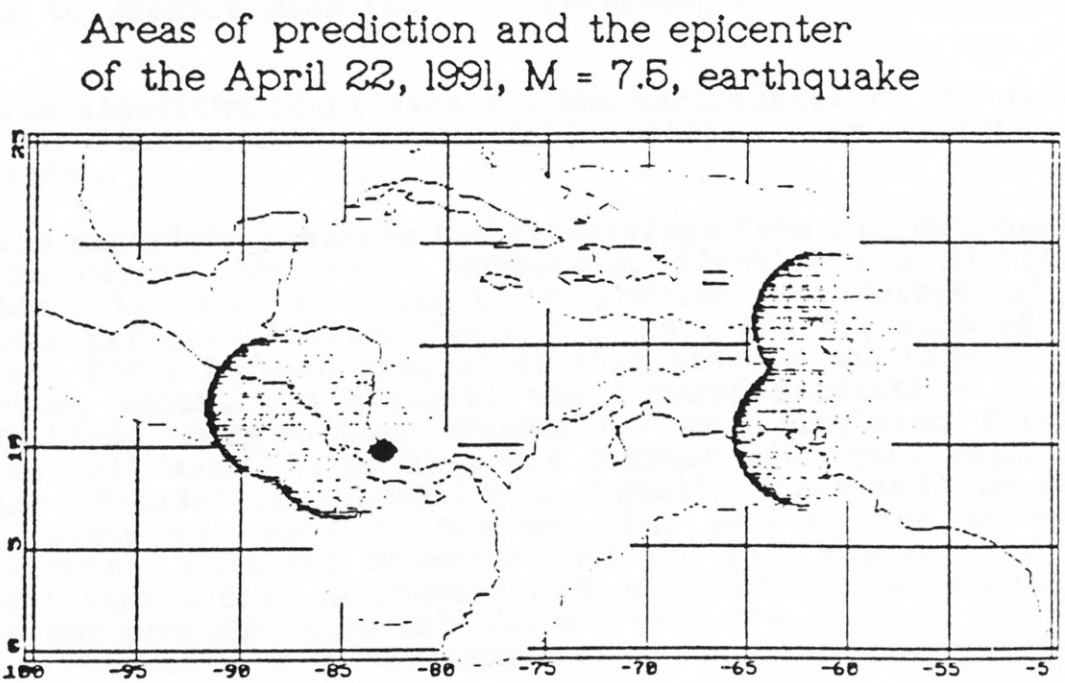


Figure 4. Central America: TIP's in January 1, 1991 - July 1, 1991.



A short description of the M8 algorithm

The M8 algorithm is composed of a series of very simple steps. We can describe those steps, but in combination the algorithm sometimes produces surprises that are not easy to explain in a short description. The formal description of the algorithm is contained in the Fortran code with the standard profiles which are used in the M8 test A. The simple description is as follows.

Step 1. We prepare the global catalog of all earthquakes ≥ 4.0 using the source catalogs from the NEIC Data Base. This catalog is prepared by a computer program with all parameters fixed. The algorithm automatically selects a single magnitude from a number of choices in the source catalogs.

Step 2. We run an algorithm that automatically identifies and removes aftershocks from this catalog to produce the catalog of mainshocks which then is used as an input in the M8 algorithm. Thus, the M8 algorithm uses the global catalog of mainshocks of magnitude ≥ 4.0 since January 1, 1963. This input catalog also contains the count of aftershocks that occurred within two weeks after each mainshock.

Step 3. We select a latitude and longitude which is the center of a circle of investigation. In this test this circle has a radius equal to 427 km. This number is appropriate to the size of the predicted earthquake and results from our decision to configure the algorithm to predict magnitude 7.5-and-above earthquakes in this test.

Step 4. The algorithm identifies all the earthquakes in the global catalog of mainshocks which fall within this circle of investigation.

Step 5. The algorithm prepares two subcatalogs from the earthquakes within the circle. The first subcatalog identifies a magnitude cutoff that is just sufficient to provide an average of 20 earthquakes per year between January 1, 1975 and the time of the prediction. For a forward prediction this time is the time of the last catalog update. To simulate the forward prediction in the past, the time is the time chosen for the simulated forward prediction. All earthquakes that are greater than this magnitude cutoff are included in subcatalog 1. Usually there will be more than 20 events per year in the catalog, because the selected magnitude cutoff does not relate exactly to this average. If the circle of investigation is located in the region of low seismicity that does not provide 20 mainshocks per year the program issues a warning. If there are fewer than 16 earthquakes per year the program will not run.

The second subcatalog is prepared in the same way but the magnitude

cutoff is chosen in order to produce at least 10 earthquakes per year.

Step 6. These subcatalogs are used to calculate three groups of functions with two functions in each group.

Group 1.

The first function in the first group is a six-year trailing average of the number of earthquakes in subcatalog 1. The second function in the first group is a six-year trailing average of the number of earthquakes in subcatalog 2.

For example we let the number of earthquakes in subcatalog 1 in the i-th half-year period equal X_i. We designate the value of the first function of the first group by $F_1(i)$ then:

$$F_1(i) = X_{i-11} + X_{i-10} + \ldots + X_i \qquad \text{(subcatalog 1)}$$

We designate the second function in the first group as $F_2(i)$. This function is calculated in a similar way using the data from subcatalog 2.

$$F_2(i) = X_{i-11} + X_{i-10} + \ldots + X_i \qquad \text{(subcatalog 2)}$$

Group 2.

This group calculates the difference between a predicted and an actual rate of seismicity. We designate the first function of the second group as $F_3(i)$. The function is calculated by taking the difference between the sum of all earthquakes between the start of the catalog on January 1, 1963 and including the time of the i-th half-year interval and the sum of all earthquakes in the catalog through the time of the i-12-th half-year multiplied by i/(i-12):

$$F_3(i) = (X_1 + X_2 + \ldots + X_i) - (X_1 + X_2 + \ldots + X_{i-12})*i/(i-12)$$

(subcatalog 1)

We designate the second function in the second group as $F_4(i)$. This function is calculated in a similar way using the data from subcatalog 2.

$$F_4(i) = (X_1 + X_2 + \ldots + X_i) - (X_1 + X_2 + \ldots + X_{i-12})*i/(i-12)$$

(subcatalog 2)

Group 3.

The functions in group 3 are similar to the functions in group 1, exept that we use a magnitude weighted sum. For this calculation,

in order to avoid domination by the largest magnitudes, we set zero weights for mainshocks of magnitude 7.0 or greater. We designate the two functions of group 3 as $F_5(i)$ and $F_6(i)$, and we define a function $S_i = \sum 10^{-0.46 M_j}$, where M_j is the magnitude of the j-th mainshock in he i-th half-year interval. The function $F_5(i)$ is calculated as follows:

$$F_5(i) = (S_{i-11} + S_{i-10} + \ldots + S_i)/(X_{i-11} + X_{i-10} + \ldots + X_i)^{2/3}$$

(subcatalog 1)

The second function in the third group, $F_6(i)$, is calculated in a similar way using the data from subcatalog 2.

$$F_6(i) = (S_{i-11} + S_{i-10} + \ldots + S_i)/(X_{i-11} + X_{i-10} + \ldots + X_i)^{2/3}$$

(subcatalog 2)

Group 4.

This group contains a single function calculated from the count of aftershocks made in Step 2, and included in the catalog of mainshocks. For each event in our global catalog of mainshocks we have a count of the number of aftershocks occurring in the first two weeks following the event. The algorithm prepares a third subcatalog including all events of magnitude ≥ 5.5 and < 7.3 in a circle of investigation. The 7-th function is designated as $F_7(i)$. We let A_j be equal to the number of aftershocks associated with the j-th event in subcatalog 3. Then we set $F_7(i)$ equal to the maximal value of A_j that occurs in the i-1-th and i-th half-years.

Step 7. In this step we identify the anomalously high values of each function. From the values of each function between January 1, 1975 and the time of the prediction (see Step 3 above) we select the highest 10 percent of the values and declare those values anomalous. During the periods when the function is anomalous it votes for a declaration of a TIP, or Time of Increased Probability, for the occurrence of the 7.5-and-above earthquake. Integer arithmetics complicates the application of this rule. Between January 1, 1975 and January 1, 1987 there are 22 one-half-year periods, so we select from the 22 values of the function the 2.2(!) highest values; obviously, by convention we take the two highest values. On July 1, 1989 we have 25 values of the functions and we select the 2.5(!) highest values; by convention we round it to 3.

Sometimes, more than 10 percent of the highest values are equal. For example, consider that on January 1, 1985 when we have 20 values of the functions and we wish to select the two highest values we find that the five highest values of the function are 731, 730, 730, 730, 730. In this case, we select only the single value 731 as anomalously high to avoid declaring five anomalous periods which would equal 25 percent of the total number of values. If the five highest values were 730, 730, 730, 730, 730, we then

must select 730 as the anomalous value in order not to throw away this function. This example shows the complexity of integer arithmetics.

We calculate the anomalous period of the first six functions following the above procedure.

For the 7-th function, we define the highest 25 percent of the values as anomalous.

Step 8. Declaration of TIP's.

We count the votes in the following way. When a function is anomalous in any period it votes for a declaration of a TIP in the following three years, or six one-half-year periods. When one function in any group votes the group votes for a TIP. To declare a TIP all four groups must vote, and six out of seven functions must vote in two consecutive half-year intervals.

References

Keilis-Borok, V.I., and V.G.Kossobokov, 1984, A complex of long-term precursors for the strongest earthquakes of the world. Proc. of 27 Geological Congress, Vol.61, Moscow: Nauka, pp.56-66.

Keilis-Borok, V.I., and V.G. Kossobokov, 1986 , Time of Increased Probability for the great earthquakes of the world. In Computational Seismology, 19, Moscow: Nauka, pp. 48-58.

Gabrielov, A.M., O.E. Dmitrieva, V.I. Keilis-Borok, V.G. Kossobokov, I.V. Kouznetsov, T.A. Levshina, K.M. Mirzoev, G.M. Molchan, S.Kh. Negmatullaev, V.F. Pisarenko, A.G. Prozoroff, W. Rinehart, I.M. Rotwain, P.N. Shebalin, M.G. Shnirman, and S.Yu. Schreider, 1986, Algorithms of long-term earthquakes' prediction. Lima(Peru): CERESIS, 61 p.

Keilis-Borok, V.I., and V.G. Kossobokov, 1988, Premonitory activation of seismic flow: Algorithm M8. Lecture Notes of the Workshop on Global Geophysical Informatics with Applications to Research in Earthquake Prediction and Seismic Risk (15 Nov. - 16 Dec. 1988), H4.SMR/303-10, Trieste: ICTP, 17p.

Keilis-Borok, V.I., V.G. Kossobokov, 1990, Premonitory activation of earthquake flow: algorithm M8. - Phys. Earth and Planet. Inter., Vol. 61, Nos.1-2, pp. 73-83

Updike, R.G., 1989, Proceedings of the National Earthquake Prediction Evaluation Council June 6-7, 1988, Reston, Virginia - Dept. Int. U.S.G.S.: Open-File Report 89-114.

program that realizes the M8 algorithm and its description was distributed at Workshops in Lima, Peru, 1986 (version for VAX under VMS), in Trieste, Italy, 1988, 1991, and in Caracas, Venezuela, 1991 (versions for IBM PC).

Source code

```
c      program m8
c                        Revised version 2.30b, 1991/09/04
c-------------------------------------------------------------
c---- Changes of isize, niobj & nifun should be made ----
c----       both in m8.for &  m8subr.for  !       ----
c-------------------------------------------------------------
c
      character*5 shname(9),short,chfun(9)
      integer*2 v(132,9),lfn(9)
     *         ,vg(132),vh(132)
      integer*2
     *        c(9),gc(6),ntip(5)
      integer*2 ml(9,18),m2(9,18),nas(9),nuu(9),ntt0(9)
      real*4 ba(9),d(9),sn(132,9,18),s(132),as(100,18)
      integer*2 fn(132,9,18),mmax(132,18),f(132)
       integer t,mta(18),mtx(18),mtfb(18)
       character*1 fnmx(6),chvc(9)
       character*40 recf,cf(9)
      character*1 ccf(40,9)
       integer*2 nys(18),nas(18),nds(18)
       real*4 la(18),lo(18)
c------    isize=132 , niobj=18 , nifun=9    -----
c
        integer*4 nfirst(18),nfirsa(18),nlast(18),nlasta(18)
       integer*2 indexc(18)
       integer*2  k(7),m,s
       integer*2 lat,lon
       character*14 sign
          character*1 line(80),lin(4,18),scale(80)
          character*80 linel
       character*5 ckar
       character*1 record(20)
      character*20 reco, srecc(100)
c---tip
          character*10 treco(50),trc
          integer*2 tipnur
          integer*4 tti1,tti2
          equivalance (trc(1:2),tipnum),(trc(3:6),tti1),(trc(7:10),tti2)
c---tip
       equivalence (record(1),recc)
      equivalance (record(1),t),(record(5),k(1))
      equivalence (k(1),lat),(k(2),lon),(k(4),m),(k(5),s)
c
       character*80 flnm,msfl,fufl
       character*10 reqi(2),inte(2)
      equivalence (cf(1),ccf(1,1))
          equivalence (line(1),linel)
      common /mag/am0
       common itb,itx,v,fn
c
          data nfirst/18*0/,nfirsa/18*0/,nlast/18*0/,nlasta/18*0/
      data reqi/' squares  ',' circles  '/,inte/'T°','Te'/
      data fnmx/'n','k','l','s','v','b'/
      data msfl/'mWestUS.dat'/
      data ckar/'R(Mo)'/
c------    isize=132 , niobj=18 , nifun=9    -----
       qradka=40000./360.
       qkm100=qradka/100.
         inderr=0
         ml992=0
      isize=132
        niobj=18
        nifun=9
      m0=700
      mdt=0.500000
      am0=m0
      am0=am0/100.
      mtb=1975
          ntbmx=1
          ntbdd=1
      mte=1990
          ntemx=1
```

```
          ntmdd=1
      mf=7
      mr=1
      mtyn=2
      mtys=1
      kmr=0
      la(1)=36.00
      lo(1)=-120.00
      nys(1)=1989
      nms(1)=1
      nds(1)=1
      data cf(1)/'n act          10      12                '/
      data cf(2)/'n act          20      12                '/
      data cf(3)/'l act          10      12  1963          '/
      data cf(4)/'l act          20      12  1963          '/
      data cf(5)/'s act  shf     10 0.50 12        0.46 0.67'/
      data cf(6)/'s act  shf     20 0.50 12        0.46 0.67'/
      data cf(7)/'b shf  shf  3.00 0.20  2                 '/
      p=0.1
      pb=0.25
      mtau=6
      mtau=10
      mq=4
      mb=6
      manspr=1
      mansf =0
      fufl='LomaPri.eta'
      write(6,1001)
      write(6,1111)
      write(6,1001)
1001  format(1x,78(1h*))
1003  format(' Some profile?(y-n-name) =>'\)
102   continue
      write(6,1003)
      read(6,2001) flnm
2001  format(a)
      if(flnm(1:1).eq.'n') go to 98
      if(flnm(1:1).eq.'y') then
      write(6,1103)
      read(6,2001) flnm
      end if
1103  format(' Name the profile, please (name) =>'\)
c
      open(2,file=flnm,status='old',err=100)
1004  format(' There is no such file')
c------------- reading profile
      read(2,2002) masfl,m0
      read(2,2003) xdt,ntb,nte,nf,nr,ntyn,kmr,ntys
2002  format(a14,i3)
2003  format(f8.5,2i4,3i2,i4,i2)
      do 1 i=1,nr
1     read(2,2004) la(i),lo(i),nys(i),nms(i),nds(i)
2004  format(f6.2,f7.2,i4,2i3)
      do 2 i=1,nf
2     read(2,2001) cf(i)
      read(2,2005) p,pb,mtau,mtau,mq,mb,manspr,mansf,fufl
2005  format(2f4.2,2i3,2i2,2i1,a14)
20031     format(i4,2i2,i4,2i2)
      am0=m0/100.
          read(2,20031,err=20032,end=20032)
     *  ntb,ntbmx,ntbdd,nte,ntemx,ntedd
20032       close(2)
c ---------
98    write(6,1005) masfl,am0
1005  format(' 1. Catalog of main shocks : ',a14,'   Mo =',f5.2)

          mm9=ntbmx/10
          mm8=ntbmx-mm9*10
          mm7=ntbdd/10
          mm6=ntbdd-mm7*10
          mm5=ntemx/10
          mm4=ntemx-mm5*10
          mm3=ntedd/10
          mm2=ntedd-mm3*10
      write(6,1007) ntb,mm9,mm8,mm7,mm6,nte,mm5,mm4,mm3,mm2,
     *  xdt,imts(mtys)

      call mimts(ntb,ntbmx,ntbdd,0,0,mtb)
      call mimts(nte,ntemx,ntedd,0,0,mte)

1007  format(' 2. Time interval (Tb,Te) = (',i4,2('/',2i1),
     *',  ',i4,2('/',2i1),'):'/5x,'Step in time  dt =',f4.2,' years'/
     *5x,'Activity is estimated in a period from Tb up to ',a10)
      if(kmr.gt.0) write(ckmr,1104) kmr
1104  format(i5)
      write(6,1006) nr,regi(mtyn),ckmr
      if(kmr.eq.0) then
      xr=exp(am0-5.6)
      xr=(xr+1.)*gradkm
      kmr=nint(xr/2)
      write(6,19021) kmr
          else
          write(6,*)
      end if
c----------------
      write(6,1009)
          do 10081 i=1,nr
          if(mod(i,2).eq.1) write(6,*)
          mm9=nms(i)/10
          mm8=nms(i)-mm9*10
          mm7=nds(i)/10
          mm6=nds(i)-mm7*10
10081 write(6,1008) i,la(i),lo(i),nys(i),mm9,mm8,mm7,mm6
c-------------------------
```

```
1006  format(' 3.',i4,1x,a10,'  R = ',a5,' km '\)
1009  format(2(' No.   lat      lon    T*: yy-mm-dd: ')\)
1008  format(1x,i2,')',f6.2,2x,f7.2,i7,2('/',2i1),': '\)
          write(6,*)
      write(6,1010)
1010  format(' 4.',
     *2x,'  ____________________________________________________________________ '/
     *5x,' |   |    |Magnitudes M and MM|  Intervals  |  |    |  | '/
     *5x,' |No.|Func| m   mm   a   aa   |m/dt   n/dt  |To|beta|d | ')
1910  format(
     *5x,' |___|____|___________________|_____________|__|____|__| ')
1111  format(
     *9x,'                                          Version 2.30b, 1991 '/
     *9x,'                                                             '/
     *9x,'                                                             '/
     *9x,'                                                             '/
     *9x,'           ______________________________________           '/
     *9x,'          |                                      |          '/
     *9x,'          | Intermediate-Term Earthquake Prediction |       '/
     *9x,'          |                                      |          '/
     *9x,'          |             Algorithm M8             |          '/
     *9x,'          |______________________________________|          '/
     *9x,'                                                             '/
     *9x,'                       by V. Kossobokov                      '/
     *9x,'                                                             '/
     *9x,'  International Institute of Earthquake Prediction Theory'/
     *9x,'   and Mathematical Geophysics, Academy of Sciences USSR '/
     *9x,'      Warshavskoe Shosse, 79-2,   113556 Moscow, USSR    '/
     *9x,'                                                         '/
     *9x,'                                  Telex: 411 628 MITPA SU '/
     *9x,'                                  Fax #: 7 (095) 310 7032 '/
     *9x,'                                  Phone: 7 (095) 110 7795 '/
     *1x)
      write(6,1011) (i,(ccf(j,i), j=1,40),i=1,nf)
          write(6,1910)
1011  format(7x,'F',i1,':',3x,a1,1x,20a1,
     *2x,3a1,4x,2a1,2x,4a1,1x,5a1,1x,5a1)
      write(6,1012) pb,p,mtau,mg,mh,mtau
1012  format(' 5. Constants of diagnostics:'/
     *3x,'percentile p = (for b, for others) = (',f4.2,',',f4.2,' ):'/
     *3x,'time window for "simultaneous" extrema Tex =',i3,'*dt:'/
     *3x,'thresholds (G and H) = (',i2,' and',i2,' ):'/
     *3x,'duration of an alarm,  Tau =',i3,'*dt.')

          if(inderr.gt.0)     write(6,5101) msfl
          inderr=0

11013 write(6,1013)

1013  format(' Any changes?(n-y) =>'\)
2007  format(i1)
      read(6,2001) recf
      if(recf(1:1).eq.'e') stop
      if(recf(1:1).eq.'n') go to 99
        if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) go to 11113
      if(recf(1:1).ne.'y') goto 11013
11113       write(6,1105)
      read(6,*) numq
1105  format(' Number of block to be corrected?(1-5) =>'\)
      if(numq.gt.1) go to 3
51    write(6,1005) msfl,am0
      write(6,1016)
      read(6,2001) recf
1016  format(' Is it correct?(y,n,1-5) =>'\)
      if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) go to 98
      if(recf(1:1).eq.'n') go to 5
      read(recf,2007) numq
      go to 3
5     write(6,1014) msfl
1014  format(' Name of file with main shocks?(',a14,') =>'\)
      read(6,2001) recf
      if(recf(1:1).eq.'/') go to 4
      read(recf,2001) msfl
4     write(6,1015) am0
1015  format(' Mo =',f4.2,' ? =>'\)
      read(6,*) am0
      m0=nint(am0*100.)
3     if(numq.gt.2) go to 6
      if(nnmq.lt.2) go to 51
31     mm9=ntbmm/10
           mm8=ntbmm-mm9*10
           mm7=ntbdd/10
           mm6=ntbdd-mm7*10
           mm5=ntemm/10
           mm4=ntemm-mm5*10
           mm3=ntedd/10
           mm2=ntedd-mm3*10

      write(6,1007) ntb,mm9,mm8,mm7,mm6,nte,mm5,mm4,mm3,mm2,
     *   xdt,inte(ntye)
      write(6,1016)
      read(6,2001) recf
      if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) go to 98
      if(recf(1:1).eq.'n') go to 7
      read(recf,2007) numq
      go to 3
7     write(6,1017) ntb,ntbmm,ntbdd,nte,ntemm,ntedd
1017  format(' (Tb, Te) = (',i4,2(',',i2),',',i4,2(',',i2),') ? =>'\)
      read(6,*) ntb,ntbmm,ntbdd,nte,ntemm,ntedd
9     write(6,1019) xdt
1019  format(' dt = ',f7.5,' ? =>'\)
      read(6,*) xdt
10    write(6,1020) ntye
```

```
1020  format(' Time interval for activity estimation?'/
     *12x,'( 1 for (Tb,Te) or 2 for (Tb,Te) ) : (',i1,') =>'\)
      read(6,*) ntys
      if((ntys.lt.1).or.(ntys.gt.2)) go to 10
61    go to 31
6     if(numq.gt.3) go to 11
      if(numq.lt.3) go to 3
41    write(6,1006) nr, reqi(ntyn),ckar
      if(kar.eq.0) then
      r=exp(an0-5.6)
      r=(r+1.)*gradkm
      kmar=nint(r/2)
      write(6,19021) kmar
          else
          write(6,*)
      end if

      write(6,1009)

          do 10082 i=1,nr
          if(mod(i,2).eq.1) write(6,*)
          mm9=nms(i)/10
          mm8=nms(i)-mm9*10
          mm7=nds(i)/10
          mm6=nds(i)-mm7*10

10082 write(6,1008) i,la(i),lo(i),nys(i),mm9,mm8,mm7,mm6
          write(6,*)
      write(6,1016)
      read(6,2001) recf
      if(recf(1:1).eq.'y') go to 98
      if(recf(1:1).eq.'n') go to 40
      read(recf,2007) numq
      go to 3
40    write(6,1021) ntyn
1021  format(' Shape of area?('
     *,'1 - squares or 2 - circles) :  (',i1,') =>'\)
      read(6,*) ntyn
      if((ntyn.lt.1).or.(ntyn.gt.2)) go to 40
42    write(6,1022) ckar
1022  format(' R = ',a5,' km? (punch 0 for R=R(Ko)) =>'\)
      read(6,2001) recf
      if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) go to 43
      read(recf,1104) kar
      write(ckar,1104) kar
      if(kar.le.0) ckar='R(mo)'
      if(kar.eq.0) then
      r=exp(an0-5.6)
      r=(r+1.)*gradkm
      kmar=nint(r/2)
      write(6,19021) kmar
      end if
43    write(6,1009)

          do 10082 i=1,nr
          if(mod(i,2).eq.1) write(6,*)
          mm9=nms(i)/10
          mm8=nms(i)-mm9*10
          mm7=nds(i)/10
          mm6=nds(i)-mm7*10

10082 write(6,1008) i,la(i),lo(i),nys(i),mm9,mm8,mm7,mm6
      write(6,*)
      write(6,1023)
1023  format(' Correction, deletion or addition?(c-d-a-n) =>'\)
      read(6,2001) recf
      if(recf(1:1).eq.'c') then
44    write(6,1024)
1024  format(' Area No.? =>'\)
      read(6,*) nn
      if((nn.lt.1).or.(nn.gt.nr)) go to 44
      write(6,1025) la(nn)
1025  format(' Latitude?(',f6.2,') =>'\)
      read(6,*) la(nn)
45    write(6,1026) lo(nn)
1026  format(' Longitude?(',f7.2,') =>'\)
      read(6,*) lo(nn)
c-----------
          mm9=nms(nn)/10
          mm8=nms(nn)-mm9*10
          mm7=nds(nn)/10
          mm6=nds(nn)-mm7*10
46    write(6,1027) nys(nn),mm9,mm8,mm7,mm6

1027  format(' T*: year, month, day?(',i4,2(',',2i1),') =>'\)
      read(6,*) nys(nn),nms(nn),nds(nn)
      go to 43
      else if(recf(1:1).eq.'a') then
      k1=nr
      if(k1.lt.niobj) go to 481
      write(6,1125)
1125  format(' There are already enough areas ')
      go to 43
481   nr=nr+1
      la(nr)=la(k1)
      lo(nr)=lo(k1)
      nys(nr)=nys(k1)
      nms(nr)=nms(k1)
      nds(nr)=nds(k1)
      write(6,1025) la(k1)
      read(6,*) la(nr)
48    write(6,1026) lo(k1)
      read(6,*) lo(nr)
c-----------
```

```
 49         nm9=nms(kl)/10
            nm8=nms(kl)-nm9*10
            nm7=nds(kl)/10
            nm6=nds(kl)-nm7*10
            write(6,1027) nys(kl),nm9,nm8,nm7,nm6
      read(6,*) nys(nr),nms(nr),nds(nr)
      go to 43
      else if(recf(1:1).eq.'d') then
      write(6,1024)
      read(6,*) nn
      if((nn.lt.1).or.(nn.gt.nr)) go to 43
      if(nr.gt.1) go to 491
      write(6,1126)
 1126 format(' Please, do not delete the only remaining area,',
     *   ' just correct it')
      go to 43
 491  nr=nr-1
      if(nr.lt.nn) go to 43
      do 21 i=nn,nr
      ii=i+1
      la(i)=la(ii)
      lo(i)=lo(ii)
      nys(i)=nys(ii)
      nms(i)=nms(ii)
 21   nds(i)=nds(ii)
      go to 43
      else if(recf(1:1).eq.'n') then
      go to 41
      else
      go to 43
      end if
 11   if(numq.gt.4) go to 12
      if(numq.lt.4) go to 6
 55   write(6,1010)
      write(6,1011) (i,(ccf(j,i),j=1,40),i=1,nf)
          write(6,1910)
      write(6,1016)
      read(6,2001) recf
      if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) go to 98
      if(recf(1:1).eq.'n') go to 53
      read(recf,2007) numq
      go to 3
 53   write(6,1010)
      write(6,1011) (i,(ccf(j,i),j=1,40),i=1,nf)
          write(6,1910)
      write(6,1023)
      read(6,2001)recf
      if(recf(1:1).eq.'d') then
      write(6,1028)
 1028 format(' Function No.? =>'\)
      read(6,*) nn
      if((nn.lt.1).or.(nn.gt.nf)) go to 53
      if(nf.gt.1) go to 221
      write(6,1127)
 1127 format(' Please, do not delete the only remaining function,'
     *, ' just correct it')
      go to 53
 221  nf=nf-1
      if(nf.lt.nn) go to 53
      do 22 i=nn,nf
      ii=i+1
 22   cf(i)=cf(ii)
      go to 53
      else if(recf(1:1).eq.'a') then
      if(nf.lt.nifun) go to 54
      write(6,1029)
 1029 format(' There are already enough functions ')
      go to 53
 54   kl=nf
      nf=nf+1
      recf=cf(kl)
      call corfun(recf)
      cf(nf)=recf
      go to 53
      else if(recf(1:1).eq.'c') then
      write(6,1028)
      read(6,*) nn
      if((nn.lt.1).or.(nn.gt.nf)) go to 53
      recf=cf(nn)
      call corfun(recf)
      cf(nn)=recf
      go to 53
      else if(recf(1:1).eq.'n') then
      go to 55
      else
      go to 53
      end if
      go to 55
 12   if(numq.gt.5) go to 98
 121  write(6,1012) pb,p,mtsu,mq,mh,mtsu
      write(6,1016)
      read(6,2001) recf
      if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) go to 98
      if(recf(1:1).eq.'n') go to 13
      read(recf,2007) numq
      go to 3
 13   write(6,1129) pb,p
 1129 format(' Values of percentile p for b (pb) and other functions',
     *' (p)? ('
     *,f3.2,',',f3.2,') =>'\)
      read(6,*) pb,p
 14   write(6,1030) mtsu
 1030 format(' Value of Max (Tex-Mex*dt)?(',i3,') =>'\)
      read(6,*) mtsu
```

```
 15     write(6,1031) mg,mh
 1031   format(' Values of G and H?(',i2,1h,,i2,') =>'\)
        read(6,*) mg,mh
 16     write(6,1032) mtau
 1032   format(' Value of Mau (Tau=Mau*dt)?(',i3,') =>'\)
        read(6,*) mtau
 17     go to 98
 100    write(6,1004)
        go to 102
 99     nansf=0
        nanspr=0
        write(6,1035)
 1035   format('  Do you want to print functions?(y-n) =>'\)
        read(6,2001) recf
        if((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) nanspr=1
c--       write(6,1036)
c--1036   format('  Do you want to save values of functions?(y-n) =>'\)
c--       read(6,2001) recf
c--       if ((recf(1:1).eq.'y').or.(recf(1:1).eq.'/')) then
c--       nansf=1
c--       write(6,1037)
c--1037   format('  Name the file for functions, please =>'\)
c--       read(6,2001) fnfl
c--       end if
        write(6,1033)
 1033   format('  Do you want to save a',
       *' profile?(y-n) =>'\)
        read(6,2001) recf
        if(recf(1:1).eq.'y') then
        write(6,1034)
 1034   format('  Name the profile, please =>'\)
        read(6,2001)flnm
        open(2,file=flnm,status='unknown')
        write(2,2002) mxfl,mC
        write(2,2003) mdt,ntb,nte,nf,nr,ntyn,kmr,ntys
        do 18 i=1,nr
 18      write(2,2004) la(i),lo(i),nys(i),nms(i),nds(i)
        do 19 i=1,nf
 19      write(2,2001) cf(i)
        write(2,2005) p,pb,mtau,mtsu,mg,mh,nanspr,nansf,fufl
            write(2,10031) ntb,ntbxx,ntbdd,nte,ntexx,ntedd
        close(2)
        end if
        open(2,file='mf.pri')
         open(4,file=mxfl,access='direct',status='old',recl=20,err=509)
         read(4,rec=1) reco
         nnn=t
        write(2,1005) mxfl,amC

            xm9=ntbxx/10
            xm8=ntbxx-xm9*10
            xm7=ntbdd/10
            xm6=ntbdd-xm7*10
            xm5=ntexx/10
            xm4=ntexx-xm5*10
            xm3=ntedd/10
            xm2=ntedd-xm3*10

        write(2,1007) ntb,xm9,xm8,xm7,xm6,nte,xm5,xm4,xm3,xm2,
       *  mdt,inte(ntys)

c--       write(2,1111)

        write(2,1006) nr,reqi(ntyn),ckxr
        write(2,1009)
            do 10085 i=1,nr
            xm9=nms(i)/10
            xm8=nms(i)-xm9*10
            xm7=nds(i)/10
            xm6=nds(i)-xm7*10
            write(lin(1,i),10086) xm9
            write(lin(2,i),10086) xm8
            write(lin(3,i),10086) xm7
            write(lin(4,i),10086) xm6
10085       continue
        write(2,10084) (i,la(i),lo(i),nys(i),(lin(jk,i),jk=1,4), i=1,nr)
10084   format(bz,2(1x,i2,')',f6.2,2x,f7.2,i7,2('/',2a1),': '))
10086       format(i1)
        write(2,1010)
        write(2,1011) (i,(ccf(j,i),j=1,40),i=1,nf)
            write(2,1910)
        write(2,1012) pb,p,mtau,mg,mh,mtau
        write(6,1992)
            n1992=1
1992    format(50x,' *****  Wait, a bit...  *****')
c------------------------  end of dialogue :  start ------------
c-------------
            nquant=100
            if(nnn.gt.7900) nquant=nnn/79+1
            length=nnn/nquant
            do 5259 i=2,length+1
5259        scale(i)='*'
            scale(1)=' '
            write(6,20871) (scale(i),i=1,length+1)
            scale(1)='+'
c-------------
        mdt=mdt*525960
                            myefs=nte-mdt*mtau
        nn=0
        do 71 i=1,nf
        if((ccf(3,i).eq.'a').or.(ccf(8,i).eq.'a'))nn=nn+1
        do 71 j=1,6
        if(ccf(1,i).eq.fnrm(j)) lfn(i)=j
71      continue
```

```
      if(kmr.eq.0) then
      r=exp(ss0-5.6)
      r=(r+1.)*gradkm
      kmr=nint(r/2)
      write(2,1902) regi(ntyn),kmr
1902  format(' Radius of ',a10,i5,' km ')
19021 format('=',i5,' km')
      end if
c ------------------------------------------------------------
      if(nn.le.0) go to 799
c ---------- Rate of activity, beginning

      do 73 j=1,mlobj
      do 73 i=1,100
73    ms(i,j)=0

      do 74 i=1,nr
      mta(i)=mtb
      if(ntya.eq.1) then
      nysi=nys(i)
      nmsi=nms(i)
      ndsi=nds(i)
      call mjsta(nysi,nmsi,ndsi,0,0,mta(i))
      end if
      if((mta(i).le.mtb).or.(mta(i).gt.mte)) mta(i)=mtb
74    continue
c ----------
          iscal=2

       do 721 i=2,nnn
       read(4,rec=i) reco
c ----------
          if(mod(i,nquant).eq.0) then
          scale(iscal)='*'
          iscal=iscal+1
          write(6,20871) (scale(j),j=1,length+1)
          end if
c ----------
      do 72 j=1,nr
       if((dist(ntyn,gkm100,lat,lon,la(j),lo(j)).gt.kmr).or.
     *    (t.lt.mtb).or.(t.gt.mta(j))) go to 72
                    if(nfirsa(j).le.0) nfirsa(j)=i
                    nlasta(j)=i
      mm=r/10+1
      ms(mm,j)=ms(mm,j)+1.

72    continue
721   continue

      do 76 j=1,nr
      do 75 i=1,99
      i1=100-i
      i2=i1+1
      ms(i1,j)=ms(i1,j)+ms(i2,j)
      if(ms(i2,j).eq.0) mmm=i1
75    continue
      xnm=(mta(j)-mtb)/525960.
      do 77 i=1,mmm
77    ms(i,j)=ms(i,j)/xnm
76    continue

c ---------- Rate of activity, end

799   continue

      do 80 j=1,nr
                namax=0
                indexc(j)=0
      do 880 i=1,nf
      if(ccf(3,i).eq.'c') then
c ------ M = const

      read(cf(i),800) ami
800   format(12x,f4.2)
802   format(17x,f4.2)
      m1(i,j)=nint(ami*100.)

      else if(ccf(3,i).eq.'s') then
c ------ M = Mo - shift

      read(cf(i),800) ami
      m1(i,j)=m0-nint(ami*100.)

      else if(ccf(3,i).eq.'a') then
c ------ M = M(rate of activity)
      read(cf(i),801) na
        if(namax.lt.na) namax=na

801   format(12x,i4)
807   format(17x,i4)
      i1=0
81    i1=i1+1
      if(na.le.ms(i1,j)) go to 81
      m1(i,j)=(i1-1)*10
      end if

      m2(i,j)=999

      if(ccf(8,i).eq.'c') then
      read(cf(i),802) ami
      m2(i,j)=nint(ami*100.)
      else if(ccf(8,i).eq.'s') then
      read(cf(i),802) ami
      m2(i,j)=m0-nint(ami*100.)
```

```
      else if(mcf(8,i).eq.'a') then
      read(cf(i),807) aa
      ii=0
 83   ii=ii+1
      if(na.le.aa(ii,j)) go to 82
      a3(i,j)=(ii-2)*10
      end if
                xxma=aa(1,j)
                if(nanax.gt.xxma) indexo(j)=1
                if(nanax.gt.xxma*1.25) indexo(j)=2
 880   continue
 80   continue
c-------------------------------------------------------------------
c---------------  Printout of n and nn ---
      do  808 j=1,nr
      do 8081 i=1,nf
      an(1,i,j)=(float(a1(i,j))+0.1)/100.
8081  an(2,i,j)=(float(a2(i,j))+0.1)/100.
        if(indexo(j).eq.1) write(2,8092) j
        if(indexo(j).eq.2) write(2,8093) j
      if(nn.gt.0) then
          write(2,809) j,aa(1,j)
                  read(4,rec=nfirsa(j)) reco

        call ymdhm(iy,imo,id,ih,im,t)
        xxla=float(lat)/100.
        xxlo=float(lon)/100.
        xxmag=float(m)/100.
          mm9=imo/10
          mm8=imo-mm9*10
          mm7=id/10
          mm6=id-mm7*10
        write(2,77222) iy,mm9,mm8,mm7,mm6,ih,im,xxla,xxlo,k(3),xxmag
77222   format(' First quake:',i5,3('/',2i1),i3,':',i2,2f8.2,i4,f5.2)

                  read(4,rec=nlasta(j)) reco

        call ymdhm(iy,imo,id,ih,im,t)
        xxla=float(lat)/100.
        xxlo=float(lon)/100.
        xxmag=float(m)/100.
          mm9=imo/10
          mm8=imo-mm9*10
          mm7=id/10
          mm6=id-mm7*10
        write(2,79222) iy,mm9,mm8,mm7,mm6,ih,im,xxla,xxlo,k(3),xxmag
79222   format(' Last quake :',i5,3('/',2i1),i3,':',i2,2f8.2,i4,f5.2)

          write(2,8094) (i,an(1,i,j),an(2,i,j),i=1,nf)
          end if
      if(nn.le.0)write(2,8091) j,(i,an(1,i,j),an(2,i,j),i=1,nf)

808      continue
c-------------------------------
8092     format(/' *** Warning ! The annual rate of seismicity may be',
     *' insufficient'/1x,'**************'
     *,' for application in Region No.',i2)
8093     format(/' *** The annual rate of seismicity is insufficient',
     *' for application'/' *** in Region No.',i2)
8091  format(/' Magnitude thresholds in Region No.',
     * i2/3(5(1x,i2,':',2f5.2,':')/))
809   format(/' Magnitude thresholds in Region No.',
     * i2,3x,'(',f6.2,' quakes per year )')
8094  format(3(5(1x,i2,':',2f5.2,':')/))
c-------------------------------------------------------------------
      ntx=0
      nt0x=ntb
      do 89 i=1,nf
      read(cf(i),88) nax(i),auu(i),ntt0(i),be(i),d(i)
      if((ccf(1,i).eq.'l').and.(nt0x.gt.ntt0(i))) nt0x=ntt0(i)
      nt=nax(i)+auu(i)
      if(nt.gt.ntx) ntx=nt
89    continue
88    format(21x,i3,i2,i4,2f5.2)
      nt=ntb-ntx*ndt
      call minta(nt0x,1,1,0,0,mt0)
      if(mt0.gt.nt) mt0=nt
      do 90 j=1,nr
      nysj=nyx(j)
      nmsj=nmx(j)
      ndsj=ndx(j)
      call minta(nysj,nmsj,ndsj,0,0,mtx(j))
      mtfb(j)=mod(mtx(j)-mt0,ndt)
90    mtfb(j)=mt0+mtfb(j)
      is=isize
      do 92 i=1,isize
      do 92 j=1,niobj
92    amax(i,j)=0
      do 93 i=1,isize
      do 93 j=1,nifun
      do 93 j1=1,niobj
      fn(i,j,j1)=0
      if(ccf(1,j).eq.'b') fn(i,j,j1)=9999
93    an(i,j,j1)=0
        icm=0
                numnt=0
                    numti=0
                    numct=0
                    numft=0
                    numec=0
c -----------------------------------------------------------------
         do 52596 i=2,length+1
52596    scale(i)='|'
         scale(1)=' '
```

```
c-          write(6,*)
                  if((nn.le.0).and.(nl992.le.0)) write(6,1992)
                  if(nn.gt.0) write(6,7217)
7217        format('+','[illegible]hat was half of a bit. Wait for another one.')

            write(6,20871) (scale(j),j=1,length+1)
            scale(1)='+'
            iscal=2
c-----------
      do 91 ir=2,nnn
      read(4,rec=ir) reco

          if(mod(ir,nquant).eq.0) then
          scale(iscal)='[illegible]'
          iscal=iscal+1
          write(6,20871) (scale(j),j=1,length+1)
          end if

      if((t.lt.st0).or.(t.gt.ste)) go to 91

c ssssssssssssssssssssssssssssssss
                  if(m.lt.m0) go to 9172
                  numst=numst+1
                  sreco(numst)=reco
c ssssssssssssssssssssssssssssssss

9172    continue

      do 9102 j=1,nr
c iiiiiiiiiiiiiiiiiiiiiiiiiiiiii
                  if(indexc(j).ge.2) go to 9102

      if(dist(ntyn,gtm100,lat,lon,la(j),lo(j)).gt.kmr) go to 9102

                     if(nfirst(j).le.0) nfirst(j)=ir
                     nlast(j)=ir

      io=(t-stfb(j))/ndt+1
      if(io.gt.ie) go to 9102
        if(io.gt.ior) ior=io
      if(m.gt.mmax(io,j)) mmax(io,j)=m
      do 9103 i=1,nf
      if((m.lt.m1(i,j)).or.(m.gt.m2(i,j))) go to 9103
      if(ccf(1,i).ne.'b') go to 9101
      if((fn(io,i,j).ge.9999).or.(fn(io,i,j).lt.a)) fn(io,i,j)=a
      go to 9103
9101  fn(io,i,j)=fn(io,i,j)+1
      if(ccf(1,i).ne.'s') go to 9103
      sn(io,i,j)=sn(io,i,j)+10**(be(i)*m*0.01)
9103    continue
9102    continue
91    continue

c ---------------------- Counting functions -----------------
      if(ior.lt.isize) ie=ior

      do 94 i=1,nf
      do 94 ir=1,nr
c iiiiiiiiiiiiiiiiiiiiiiiiiiiiii
                  if(indexc(ir).ge.2) go to 94

      nsi=nsz(i)
      if((ccf(1,i).ne.'b').and.(ccf(1,i).ne.'l')) then
      f(1)=fn(1,i,ir)
      do 95 j=1,nsi-1
      j1=j+1
95    f(j1)=f(j)+fn(j1,i,ir)
      do 951 j=nsi,ie-1
      j1=j+1
      js=j1-nsi
951   f(j1)=f(j)+fn(j1,i,ir)-fn(js,i,ir)
      if(ccf(1,i).eq.'n') then
      do 958 j=1,nsi-1
958   fn(j,i,ir)=9999
      do 952 j=nsi,ie
952   fn(j,i,ir)=f(j)
      go to 94
      end if
      if(ccf(1,i).ne.'v') go to 956
      nu=nuu(i)
      nv=nsi+nu-1
      do 959 j=1,nv
959   fn(j,i,ir)=9999
      fn(nv,i,ir)=0
      do 953 j=nsi,nv-1
      j1=j+1
      jf=f(j1)-f(j)
953   fn(nv,i,ir)=fn(nv,i,ir)+iabs(jf)
      do 954 j=nv,ie-1
      j1=j+1
      js=j1-nu
      js1=js+1
      jfs=f(js)-f(js1)
      jf=f(j)-f(j1)
954   fn(j1,i,ir)=fn(j,i,ir)-iabs(jfs)          +
     *    iabs(jf)
      go to 94
956   if(ccf(1,i).ne.'k') go to 957
      nv=nsi+nsi
      do 960 j=nv,ie
      js=j-nsi
960   fn(j,i,ir)=f(j)-f(js)
      nv=nv-1
```

```
      do 961 j=1,nv
  961 fn(j,i,ir)=9999
      go to 94
  957 s(1)=sn(1,i,ir)
      do 962 j=1,nmi-1
      j1=j+1
  962 s(j1)=s(j)+sn(j1,i,ir)
      do 963 j=nmi,ie-1
      j1=j+1
      je=j1-nmi
  963 s(j1)=s(j)+sn(j1,i,ir)-sn(je,i,ir)
      do 964 j=1,nmi-1
  964 fn(j,i,ir)=9999
      smax=0
      do 965 j=nmi,ie
      if(f(j).gt.0)
     *s(j)=s(j)/f(j)**d(i)
      if(f(j).le.0)
     *s(j)=0.
      if(smax.lt.s(j)) smax=s(j)
  965 continue
      k10=0
  955 if(smax.lt.10000) go to 9551
      k10=k10+1
      do 9552 j=nmi,ie
 9552 s(j)=s(j)/10
      smax=smax/10
      go to 955
 9551 if(k10.gt.0) write(2,9553) i,ir,k10
 9553 format(' Function No.',i2,' in Region No.',i2,
     *' is multiplied by factor 1/10',i3,' times')
      do 9554 j=nmi,ie
 9554 fn(j,i,ir)=s(j)
      go to 94
      end if
      if(ccf(1,i).ne.'b') go to 966
      do 9671 j=1,ie
      je=j-nmi+1
      f(j)=9999
      if(je.lt.1) je=1
      do 967 j1=je,j
      if(((fn(j1,i,ir).gt.f(j)).and.(fn(j1,i,ir).ne.9999))
     *          .or.(f(j).eq.9999)) f(j)=fn(j1,i,ir)
  967 continue
 9671 continue
      do 968 j=1,ie
  968 fn(j,i,ir)=f(j)
      go to 94
  966 ntt0i=ntt0(i)
      call minta(ntt0i,1,1,0,0,mtti)
      imti=(mtti-mtfb(ir))/ndt+1
      s(imti)=fn(imti,i,ir)
      do 969 j=imti+1,ie
      j1=j-1
  969 s(j)=s(j1)+fn(j,i,ir)
      do 970 j=1,ie
      if(j.le.nmi-imti) then
      fn(j,i,ir)=9999
      else
      je=j-nmi
      fn(j,i,ir)=s(j)-(j*s(je))/je
      end if
  970 continue
   94 continue
c------------------- End of counting functions ----------

      do 69 k1=1,5
   69   ntip(k1)=0
      numar=0
      if(nanzf.le.0) go to 9090
      open(3,file=fufl)
      nobj=0
      do 500 ir=1,nr
c iiiiiiiiiiiiiiiiiiiiiiiiiiiiii
              if(indexo(ir).ge.2) go to 500

       nobj=nobj+ie-(mtb-mtfb(ir))/ndt
  500 continue
      write(3,*) nf,nobj
  501 format(18a5)
  503 format(3a1,i2)
      do 502 i=1,nf
      write(ehert,503) ccf(1,i),ccf(23,i),ccf(24,i),i
  502 shname(i)=ehort
      write(3,501) (shname(i),i=1,nf)
 9090 continue

      do 402 ir=1,nr
                if(nanspr.gt.0) write(2,2084)
c iiiiiiiiiiiiiiiiiiiiiiiiiiiiii
              if(indexo(ir).ge.2) then
              write(2,8093) ir
      go to 402
              end if
              if(indexo(ir).ge.1) write(2,8092) ir

      itb=(mtb-mtfb(ir))/ndt+1
      itx=(mtx(ir)-mtfb(ir))/ndt
      if(itx.lt.itb) itx=ie
      do 510 j=1,nf
      q=p
      if(lfn(j).eq.6) q=pb
      call qmax(j,q,j,ir)
  510   continue
```

```
cccccc
      if((m7.gt.0).and.(m8.le.0)) then
         write(2,3001) xmaxx
        if(sign.eq.'CTIP') numct=numct+1
        if(sign.eq.'FTIP') numft=numft+1
        if(sign.eq.'e.c.') numec=numec+1
      end if
3001     format(5x,'* An earthquake of magnitude',f5.2,' occurred ',
     *'during this time')
c---tip

      if(sign.ne.'e.c.') then
        numti=numti+1
        tipnum=ir
        tti1=it1
        tti2=it2
        treco(numti)=trc
        end if

c---tip
c       end if
c300      format(1x,a6,1x,i3,f7.2,f8.2,2i5,4014)
c       if(sign.ne.'e.c.') write(2,300) sign,ir,la(ir),lo(ir),iy1,iy2,
c     *       (c(k1),k1=1,nf),(xmax(k1,ir),k1=i4,i3),ivr,ivs,i7,i8
       if(sign.eq.'STIP') ntip(1)=ntip(1)+1
c---       if(sign.eq.'TTIP') ntip(2)=ntip(2)+1
       if(sign.eq.'CTIP') ntip(3)=ntip(3)+1
       if(sign.eq.'FTIP') ntip(4)=ntip(4)+1
       if(sign.eq.'e.c.') ntip(5)=ntip(5)+1
c----------     i+(step to next alarm)
       i=i+4
       i1=i-wtsu+1
       do 67 k1=1,nf
       c(k1)=0
       do 67 i2=i1,i
67       c(k1)=2*c(k1)+v(i2,k1)
65       ind=0
       if(i-ie) 610,222,222
66       ind=1
       i7=ivr
       i8=ivs
       if(i-ie) 610,222,222
222     continue

c ------- xxxxxxxxxxxxxxxxxxxxxxxxxxx
       do 2222 istr=1,numst

              reco=treco(istr)

     if(dist(nsyn,gkm100,lat,lon,la(ir),lo(ir)).gt.kmr) go to 2222

       call ymdhx(iy,imo,id,ih,im,t)
       xxla=float(lat)/100.
       xxlo=float(lon)/100.
       xxmag=float(m)/100.
         mm9=imo/10
         mm8=imo-mm9*10
         mm7=id/10
         mm6=id-mm7*10
       write(2,7222) iy,mm9,mm8,mm7,mm6,ih,im,xxla,xxlo,k(3),xxmag,ir
7222     format(' Strong quake:',i5,2('/',2i2),i3,':',i2,2f8.2,i4,
     *f5.2,' in Region No.',i2)

2222     continue

c xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          if(nanspr.le.0) go to 4021
         write(2,*)
       write(2,3101) ir,nyn(ir),la(ir),lo(ir)

               read(4,rec=nfirst(ir)) reco

       call ymdhx(iy,imo,id,ih,im,t)
       xxla=float(lat)/100.
       xxlo=float(lon)/100.
       xxmag=float(m)/100.

         mm9=imo/10
         mm8=imo-mm9*10
         mm7=id/10
         mm6=id-mm7*10

       write(2,77222) iy,mm9,mm8,mm7,mm6,ih,im,xxla,xxlo,k(3),xxmag

3101     format(10x,'Region No.',i3,i12,2f12.2)
2081     format(100x,'( continued )')
3084     format(/)
20871      format(80a1)
         mff=1

         if(nf.gt.10) mff=2

         do 103 j2=1,mff

          nfb=10*(j2-1)+1
          nfe=j2*10
          if(nfe.gt.nf) nfe=nf
          if(j2.gt.1) write(2,2081)
        write(line1,2082) (j, j=nfb,nfe)
         lng=22+4*(nfe-nfb+1)
         write(2,*)
         write(2,20871) (line(j),j=1,lng)
         write(2,*)
```

```
      do 103 j1=itb,ie

      it1=j1*ndt+mtfb(ir)
c --last line
        if(it1.gt.nte+7200) go to 103
      call ymdhs(iy1,imo,id,ihc,imi,it1)
      axmm=amax(j1,ir)/100.
      do 1131 j=nfb,nfe
       chvo(j)=' '
       chfun(j)='    -'
      if(fn(j1,j,ir).ne.9999) write(chfun(j),33133) fn(j1,j,ir)
        if(v(j1,j).gt.9) chvo(j)='*'
1131   continue

         mm9=imo/10
         mm8=imo-mm9*10
         mm7=id/10
         mm6=id-mm7*10
       write(2,208) vq(j1),vh(j1),axmm,iy1,mm9,mm8,mm7,mm6,
     *       (chfun(j),chvo(j),j=nfb,nfe)
103     continue

              read(4,rec=nlast(ir)) reco

      call ymdhs(iy,imo,id,ih,im,t)
      xxla=float(lat)/100.
      xxlo=float(lon)/100.
      xxmag=float(m)/100.
        mm9=imo/10
        mm8=imo-mm9*10
        mm7=id/10
        mm6=id-mm7*10
       write(2,79222) iy,mm9,mm8,mm7,mm6,ih,im,xxla,xxlo,k(3),xxmag
4021  if (namsf.le.0) go to 402
      do 508 j1=itb,ie
       it1=(j1-1)*ndt+mtfb(ir)
       call ymdhs(iy1,imo,id,ihc,imi,it1)
      numnam=numnam+1
       iy1=iy1-1900
      if(imo.lt.10) write(short,507) imo
      if(imo.eq.10) write(short,505) 'o'
      if(imo.eq.11) write(short,505) 'n'
      if(imo.eq.12) write(short,505) 'd'
        write(short(3:4),504) ir,iy1
504   format(2i2)
33133 format(i5)
507   format(4x,i1)
505   format(4x,a1)
      shname(numnam)=short
      if(numnam.lt.18) go to 508
      write(3,501) (shname(j),j=1,18)
      numnam=0
508   continue

402      continue

                                    write(2,2084)
       write(2,212) ntip(1),ntip(4),ntip(3),ntip(5)
212       format(1x,'Number of STIP''s =',i3,': FTIP''s =',i3,
     *       ': CTIP''s =',i3,': e.c.''s =',i3)
3002      format(1x,'* The quakes of  M ≥ Mo-.5, < Mo  occurred in ',
     *i2,' FTIP''s and ',i2,' CTIP''s')
          if((numct.gt.0).or.(numft.gt.0)) write(2,3002) numft,numct
          if(numec.gt.0) write(2,3003) numec
3003      format(1x,'* The mainshocks of M ≥ Mo-.5 occurred within ',
     *'5 years after ',i2,' strong')

cxx       write(2,212) ntip(1),ntip(4),ntip(2),ntip(3),ntip(5)
cxx
cxx212       format(1x,'Number of STIP''s =',i3,': FTIP''s =',i3,
cxx     *       ': TTIP''s =',i3,': CTIP''s =',i3,': e.c.''s =',i3)

      if(namsf.le.0) go to 9999
      if((numnam.gt.0).and.(numnam.lt.18))
     *    write(3,501) (shname(j),j=1,numnam)
      do 506 ir=1,nr
c iiiiiiiiiiiiiiiiiiiiiiiiiiiii
                if(indexc(ir).ge.2) go to 506
      itb=(mtb-mtfb(ir))/ndt+1
      do 5061 j1=itb,ie
5061   write(3,*) (fn(j1,j,ir),j=1,nf)
506    continue

       close(3)
9999  write(6,1993)
        close(1)
        close(2)
c+++tip
         open(2,file='m8.ers')
         open(3,file='m8.tip')
         int=0

44222    format(i5,i7)
         do 91222 ir=1,nr

         if(indexc(ir).eq.2) then
         indexc(ir)=-1
         else
         indexc(ir)=0
         end if
         ila=nint(la(ir)*100.)
         ilo=nint(lc(ir)*100.)
         write(2,44222) ila,ilo
91222   continue
```

```
          do 91333 itip=1,numti
          trc=trsco(itip)
        call ymdhm(iy1,imo1,id1,ih1,im,tti1)
        call ymdhm(iy2,imo2,id2,ih,im,tti2)
          ir=tipnum
          indstr=0
       istr=0
91339       istr=istr+1
          if(istr.gt.numst) go to 91338

              reco=areco(istr)
          if((t.lt.tti1).or.(t.gt.tti2)) go to 91339
      if(dist(rtyn,gtml00,lat,lon,la(ir),lo(ir)).gt.kmr) go to 91339
          indstr=istr
91338     if((tti2.gt.ste+7200).and.(indstr.le.0)) indexo(ir)=1
91333     write(3,44333) ir,iy1,imo1,id1,iy2,imo2,id2,indstr
44333     format(i2,i4,2i2,i4,2i2,i3)

          close(3)
          close(2)
          open(3,file='st.str')
          open(3,file='st.res')
        do 92222 istr=1,numst
              reco=areco(istr)
        call ymdhm(iy,imo,id,ih,im,t)
        write(2,41222) iy,imo,id,ih,im,(k(i),i=1,7),int
92222    continue
41222    format(i4,4i2,' 0',i5,i6,5i3,i1)

44293     format(' On ',i4,2('/',2i1),' current TIPs for profile: ',a80)
44292     format(i5,'.',2f10.2,i4)

          nm5=ntemm/10
          nm4=ntemm-nm5*10
          nm3=ntedd/10
          nm2=ntedd-nm3*10

          write(3,44293) nta,nm5,nm4,nm3,nm2,flnm
          do 91292 ir=1,nr
          write(3,44292) ir,la(ir),lo(ir),indexo(ir)
91292   continue

c---tip
          close(3)
          close(2)
        close(4)
1993  format(' ------ Output''s are in files st.* ------')
          go to 99999
509   indarr=1
      go to 98
5101  format(
     *' <<<<<<<<<< There is no file of main shocks named ',a14,
     *' >>>>>>>>>>')
99999       stop
201       format(i3,1h:,i1,f5.1,i5,2('/',2i1),2x,10(a5,a1))
2082      format(2x,'g:h  Mmax   Date     ',10('    P',i1))
301       format(5i12)
      end


c*****************************************************************

      subroutine qmax(r,q,i10,ir)
      integer*2 v(132,9),n(132)
      integer*2 fn(132,9,18)
      common itb,itx,v,fn
c-----  isize=132 , niobj=18 , nifun=9  -----
      isize=132
      max=9999
      ns=0
      nmax=0
      do 1 i=itb,itx
      if(fn(i,m,ir)-9999) 4,1,1
4      ns=ns+1
      if(fn(i,m,ir)-max) 1,2,3
2      nmax=nmax+1
      go to 1
3      max=fn(i,m,ir)
      nmax=1
1      continue
      r=ns
      r=r+q+0.5
      iq=r
      if(nmax-iq) 6,6,100
c      if(nmax-iq) 6,6,101
6      do 5 i=1,isize
5      n(i)=fn(i,m,ir)
20      do 7 i=itb,itx
      if(n(i)-max) 7,8,8
8      n(i)=9999
7      continue
      ms=9999
      nma=0
      do 10 i=itb,itx
      if(n(i)-9999) 19,10,10
19      if(n(i)-ms) 10,11,12
11      nma=nma+1
      go to 10
12      ms=n(i)
      nma=1
10      continue

      nmax=nmax+nma
```

```
      if(mmax-iq) 13,15,14
13      max=ma
        go to 20
14      mmax=mmax-ma
        go to 100
15      max=ma
100       do 30 i=1,ixire
        v(i,110)=0
        if((fn(i,m,ir).lt.max).or.(fn(i,m,ir).ge.9999)) go to 30
        v(i,110)=10
30        continue
        go to 102
101       max=max+1
        mmax=0
        go to 100
102       return
        end


c*************************************************************

      subroutine cerfun(cf)
      character*40 cf,ff
      character*1 cff(40),fnm(6),ansv,fnn,bl,nm1,nm2
      character*4 m(4),nm(3),bla
      common /maq/ma0
      equivalence (cff(1),ff),(nm1,m(1)),(nm2,m(2))
      data fnm/'m','k','l','s','v','b'/,bl/' '/,
     +nm/'act ','shf ','con '/

      bla=bl
      ff=cf
      read(ff,212) fnn,m
212   format(1a1,4(1x,a4))
      read(ff,211) ns,nu,mt0,beta,dd
211   format(21x,i3,i2,i4,2f5.2)
40    format(f4.2)
41    format(i4)
      read(m(3),40) ami
      read(m(4),40) ama
200    write(6,10) cf(1:1)
10       format(' Please,select function from the list given below ->'
     *  /'  n - n(t|M,s) - number of main shocks with Mi >= M in'
     *  ,' (t-s,t) time interval'
     *  /'  k - k(t|M,s) = n(t|M,s) - n(t-s|M,s)'/'  l - l(t|M,s) = ',
     *       'n(t|M,t-To) - [(t-To)/(t-s-To)]*n(t-s|M,t-s-To)'
     *       /'  s - s(t|M,MX,s) =',
     *    ' sum(exp10(beta*(Mi-alfa))/[n(t|M,s)-n(t|MX,s)]**d'
     *       /15x,'for main shocks (i) with M <= Mi < MX'
     *       ,' in (t-s,t) time interval.'
     *       /'  v - v(t|M,s,u) = var(t,t-u) n(t|M,s)'/'  b - b',
     *  '(t|M,MX,Ma,s,e) - max number of aftershocks with M'' >= Ma'/
     *  14x,'for the main shocks from (M,MX) magnitude and (t-s,t) time'
     *  /     14x,'intervals counted for s days'//
     *    15x,'Function ',a1,'? ->'\)
       read(6,20) ansv
20       format(a)
       fnn=ansv
      if((ansv.eq.'/').or.(ansv.eq.'y')) fnn=cf(1:1)
      nam=0
      do 2 i=1,6
      if(fnn.eq.fnm(i)) nam=i
2     continue
      if(nam.le.0) go to 200
      write(6,100) ns
100   format(' Value of ns (s=ns*dt)?(',i2,') ->'\)
21    format(bz,a1,4(1x,a4),i3,i2,i4,2f5.2)
      read(6,*) ns
      if(nam.eq.5) then
      write(6,101) nu
101   format(10x,'Value of nu (u=nu*dt)?(',i2,') ->'\)
      read(6,*) nu
      else if(nam.eq.3) then
      write(6,102) mt0
102   format(10x,'Beginning of the catalog, To = ',i4,' ->'\)
      read(6,*) mt0
      end if
      write(6,103)
3     write(6,107) nm1
103   format(1x,' Magnitude thresholds :')
107   format(5x,'Type of M?(',
     +' c- M = a: s- M = Mo - a: e- M = M(s) )(',a1,') ->'\)
      read(6,20) ansv
      if((ansv.eq.'y').or.(ansv.eq.'/')) ansv=nm1
      if(ansv.eq.'c') then
      write(6,104) ami
104   format(10x,'M = a = ',f4.2,'? ->'\)
      read(6,*) ami
      write(m(3),40) ami
      m(1)=nm(3)
      else if(ansv.eq.'s') then
      write(6,105) ami
105   format(10x,'a = Mo - M =',f5.2,'? ->'\)
      read(6,*) ami
      m(1)=nm(2)
      write(m(3),40) ami
      else if(ansv.eq.'e') then
      ns=(ami+.001)*100
      write(6,106) ns
106   format(3x,'M = M(s), where s is the average ',
     +'annual number of main shocks:  s =',i3,'? ->'\)
      read(6,*) na
      m(1)=nm(1)
      write(m(3),41) na
```

```
      else
      go to 3
      end if

      if(nanz.eq.4) go to 33
      if(nanz.eq.6) go to 33
      go to 42
 33   write(6,117) nm2
 117  format(1x,'Type of MM?'
     *,'( c- constant, s- MM = Mo - aa , e- MM = M(aa) )(',a1,') =>'\)
      read(6,20) answ
      if((answ.eq.'y').or.(answ.eq.'/')) answ=nm2
      if(answ.eq.'c') then
      write(6,114)
 114  format(10x,'MM = aa - ',f4.2,'? =>'\)
      read(6,*) aaa
      write(m(4),40) aaa
      m(1)=nm(3)
      else if(answ.eq.'s') then
      write(6,115) aaa
 115  format(10x,'aa = Mo - MM = ',f4.2,'? =>'\)
      read(6,*) aaa
      m(3)=nm(2)
      write(m(4),40) aaa
      else if(answ.eq.'e') then
      aa=(aaa+.001)*100
      write(6,116) aa
 116  format(3x,'MM = M(aa), where aa is the average annual ',
     *'number of main shocks:  aa =',i3,'? =>'\)
      read(6,*) aa
      m(3)=nm(1)
      write(m(4),41) aa
      else
      go to 33
      end if
 42   if(nanz.eq.4) then
 5    write(6,108) beta
 108  format(10x,'beta =',f5.2,'? =>'\)
      read(6,*,err=5) beta
 6    write(6,109) dd
 109  format(10x,'d =',f5.2,'? =>'\)
      read(6,*,err=6) dd
 7    end if
      if((nanz.le.3).or.(nanz.eq.5)) then
      m(2)=bla
      m(4)=bla
      end if
      write(ff,21) fnam(nanz),m,na,nu,mt0,beta,dd
      if (nanz.ne.5) then
      cff(25)=bl
      cff(26)=bl
      end if
      if(nanz.ne.3) then
      do 34 i=27,30
 34   cff(i)=bl
      end if
      if(nanz.eq.4) go to 999
      do 32 i=31,40
 32   cff(i)=bl
 999  cf=ff
      return
      end


c*************************************************************

      real function dist(n,gradkm,lat,lon,la,lo)
      real*4 la,lo
      integer*2 lat,lon
      x=100*lo-lon
      y=100*la-lat
      z=la
      x=x*3.1415926536/180.
      x=cos(x)
      x=abs(x)
      if(x.gt.18000.) x=36000.-x
      if(n.eq.1) then
      x=x*z
      y=abs(y)
      if(x.lt.y) x=y
      else
      x=x*x*z*z+y*y
      x=sqrt(x)
      end if
      dist=x*gradkm
      return
      end


c*************************************************************
c        year,month,day,hour,minute / minute

      subroutine minta(iyear,imon,iday,ih,imin,mins)
      integer iyear,imon,iday,ih,imin,igday
      integer mins
      call grejul(iyear,imon,iday,igday)
      call ydhmin(iyear-1,igday-1,ih,imin,mins)
      return
      end
```

```
c*****************************************************************
c          )minute / year,month,day.hour,minute

      subroutine ymdhm(iyear,imon,iday,ih,imin,mina)
      integer iyear,imon,iday,ih,imin
      integer mina
      call minydh(iyear,iqday,ih,imin,mina)
      iyear=iyear+1
      call julgre(iyear,imon,iday,iqday+1)
      return
      end


c*****************************************************************

      subroutine minydh(iyear,iday,ihour,min,minute)
c minydh takes the number of minutes since the beginning
c        of the century and convert it into a julian date.
c        )2103840=(3*365+366)*24*60
c        )525600=365*24*60
c        )1440=24*60
      integer minute,minutr,n4yrgr,id,nsyear
      n4yrgr=minute/2103840
      minutr=minute-2103840*n4yrgr
      nsyear=minutr/525600
      if(nsyear.eq.4) nsyear=3
      minutr=minutr-525600*nsyear
      iyear=4*n4yrgr+nsyear
      id   =minutr/1440
      iday=id
      minutr=minutr-1440*id
      ihour =minutr/60
      minutr=minutr-60*ihour
      min   =minutr
      return
      end


c*****************************************************************

      subroutine ydhmin(iyear,iday,ihour,min,minute)
c inverse of minydh
c        ))n4yrgr is the number of 4 year groups.
c        ))1461=3*365+366
      integer minute,n4yrgr,ipr
      n4yrgr=iyear/4
      ipr=n4yrgr*1461+iday
      ipr=ipr+(iyear-4*n4yrgr)*365
      ipr=ipr*24+ihour
      ipr=ipr*60+min
      minute=ipr
      return
      end


c*****************************************************************
      subroutine julgre(year,month,day,julian)
      integer year,day,tndpmo
      dimension tndpmo(24)
      data tndpmo /0,31,59,90,120,151,181,212,243,273,304,334,
     1   0,31,60,91,121,152,182,213,244,274,305,335/
      nmonpl=13
      if (mod(year,4).eq.0) nmonpl=25
      do 14 jf1=1,12
      month=nmonpl-jf1
      if (julian.le.tndpmo(month)) goto 14
      day=julian-tndpmo(month)
      if (month.gt.12) month=month-12
      return
14        continue
      return
      end


c*****************************************************************
c*******   year,month,day/year,day

      subroutine grejul(year,month,day,julian)
      integer year,day,tndpmo
      dimension tndpmo(24)
      data tndpmo/0,31,59,90,120,151,181,212,243,273,304,334,
     1   0,31,60,91,121,152,182,213,244,274,305,335/
      i=month
      if (mod(year,4).eq.0) i=i+12
      julian=tndpmo(i)+day
      return
      end
c*****************************************************************
```

APPENDIX V

A sample of the semiannual update for the M8 Test

Regions of Increased Probability of magnitude 7.5+ Earthquakes for the Period from January 1, 1992 to July 1, 1992

The attached map and table represent the results to date of an ongoing test of the earthquake-prediction algorithm M8. Algorithm M8 is described by Kossobokov and Keilis-Borok (1990). We refer interested parties to the paper by Healy and others (1992) for a more complete description of the test and its rationale.

The map and Table 1 represent Times of Increased Probability (TIP's) that have been identified by the algorithm M8 as configured for the M8 Test. The purpose of the test is to estimate the statistical significance of M8 TIP's and to determine if an M8 TIP provides information that improves our ability to estimate the probability of large earthquakes. The TIP's should therefore be viewed as "research TIP's". The U. S. Geological Survey does not, at this time, endorse this algorithm as being capable of predicting earthquakes and does not recommend that any special action be taken because of this information.

The warnings or TIP's declared by this algorithm occur in regions where large earthquakes are possible and where appropriate steps to mitigate the earthquake hazard are desirable in any case. Any recommendation for public action would be based on a synthesis of all the available information for each region performed by the specialists and authorities who are responsible for that region.

Reference

Keilis-Borok, V. I., and Kossobokov, V. G., b, Times of increased probability of strong earthquakes (M => 7.5) diagnosed by algorithm M8 in Japan and adjacent territories: Journal of Geophysical Research, v. 95, p. 12,413-12,422.

Healy, J. H., Kossobokov, V. G., and Dewey, J. W., 1992, The Design of a test to evaluate the Soviet earthquake prediction algorithm, M8, U. S. Geological Survey Open-File Report **, ** p.

Table 1.

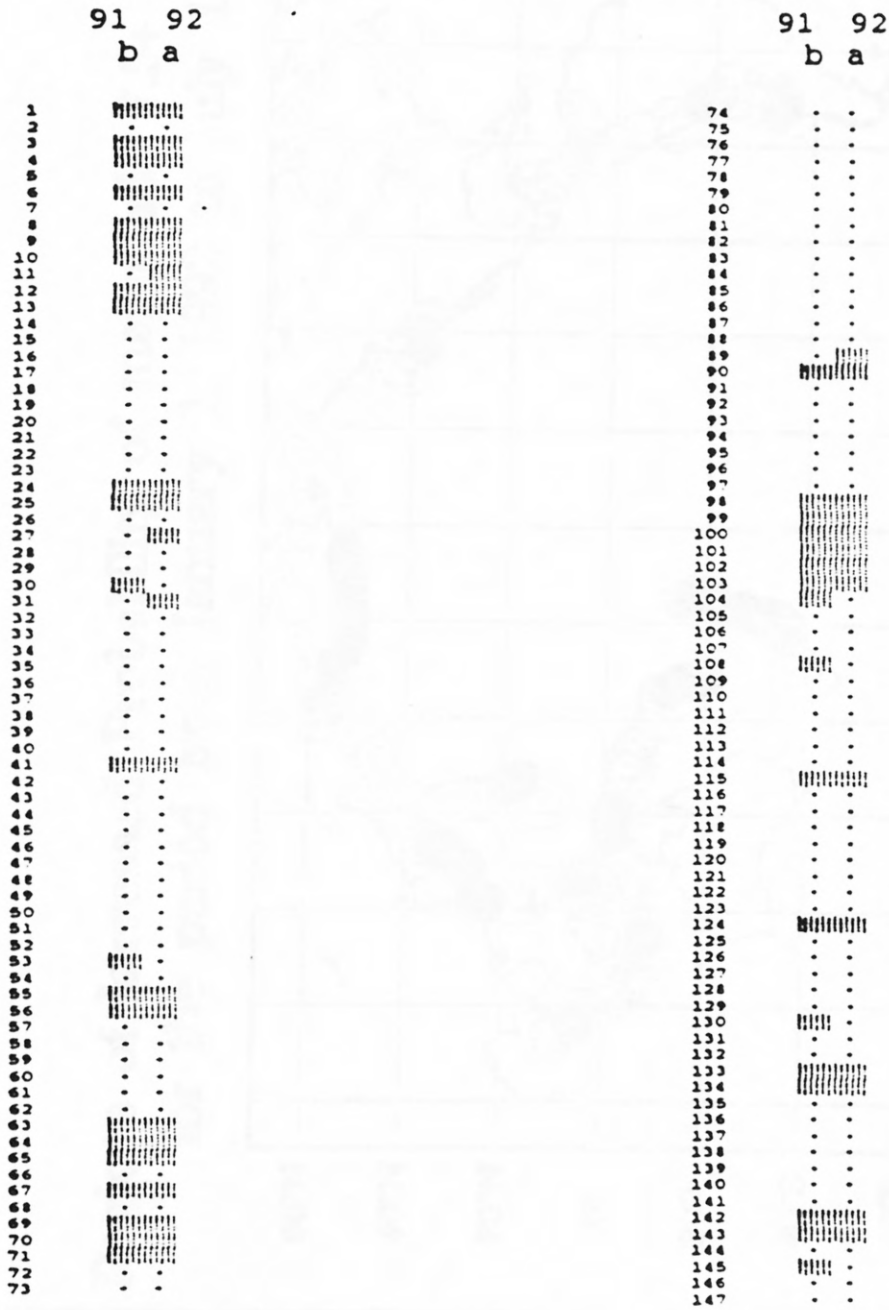
THE DISTRIBUTION OF TIMES OF INCREASED PROBABILITY (TIP) FOR EARTHQUAKES OF MAGNITUDE ≥ 7.5 IN THE CIRCUM PACIFIC SEISMIC ZONE.
The location of circles of investigation.

Reg No	Lat	Lon	Geographic Location
1	-15.00	-175.00	Tonga Trench
2	-17.50	-174.00	Tonga Trench
3	-20.00	-175.00	Tonga Trench
4	-22.50	-176.00	Tonga Trench
5	-25.00	-177.00	Tonga Trench
6	-27.50	-177.50	Tonga Trench
7	-30.00	-178.00	Kermadec Trench
8	-32.50	-179.00	Kermadec Trench
9	-35.00	180.00	Kermadec Trench
10	-37.00	178.00	Kermadec Trench
11	-2.00	136.00	New Guinea
12	-2.25	138.50	New Guinea
13	-2.50	141.00	New Guinea
14	-3.75	143.50	New Guinea
15	-5.00	146.00	New Guinea
16	-5.00	149.00	New Guinea
17	-5.00	152.00	Solomon Islands
18	-6.25	154.00	Solomon Islands
19	-7.50	156.00	Solomon Islands
20	-8.75	158.00	Solomon Islands
21	-10.00	160.00	Solomon Islands
22	-10.50	162.50	Solomon Islands
23	-11.00	165.00	Solomon Islands
24	-13.00	166.25	New Hebrides
25	-15.00	167.50	New Hebrides
26	-17.50	168.25	New Hebrides
27	-20.00	169.00	New Hebrides
28	-21.25	170.75	New Hebrides
29	9.50	93.75	Java Trench
30	7.00	94.50	Java Trench
31	5.00	95.75	Java Trench
32	3.00	97.00	Java Trench
33	2.00	98.50	Java Trench
34	-1.00	100.00	Java Trench
35	-3.00	101.50	Java Trench
36	-5.00	103.00	Java Trench
37	-6.50	105.00	Java Trench
38	-8.00	107.00	Java Trench
39	-8.50	109.50	Java Trench
40	-9.00	112.00	Java Trench
41	-9.25	114.50	Java Trench
42	-9.50	117.00	Java Trench
43	-9.50	119.50	Java Trench
44	-9.50	122.00	Java Trench
45	-8.25	124.50	Banda Sea
46	-7.00	127.00	Banda Sea
47	-6.00	129.00	Banda Sea
48	-5.00	131.00	Banda Sea
49	-3.50	129.25	Banda Sea
50	-2.00	127.50	Banda Sea
51	-1.00	125.25	Celebes Basin
52	.00	123.00	Celebes Basin
53	.00	120.50	Celebes Basin
54	1.50	125.00	Celebes Basin
55	3.00	127.00	Philippine Islands
56	5.25	126.50	Philippine Islands
57	7.50	126.00	Philippine Islands
58	9.75	125.50	Philippine Islands
59	12.00	125.00	Philippine Islands
60	13.50	123.00	Philippine Islands
61	15.00	121.00	Philippine Islands
62	35.00	136.00	Southern Japan
63	34.00	134.00	Southern Japan
64	33.00	132.00	Southern Japan
65	31.00	132.00	Southern Japan
66	29.00	131.00	Southern Japan
67	27.00	129.00	Ryukyu Islands
68	26.00	127.00	Ryukyu Islands
69	25.00	124.50	Ryukyu Islands
70	25.00	122.00	Ryukyu Islands
71	22.50	121.50	Taiwan
72	20.00	121.50	Taiwan
73	17.50	121.00	Taiwan
74	45.50	150.50	Kuril Islands
75	44.00	148.00	Kuril Islands
76	43.50	145.50	Kuril Islands
77	43.00	143.00	Japan
78	41.00	141.00	Japan
79	39.00	142.00	Japan
80	36.50	141.00	Japan
81	35.00	139.00	Japan
82	33.00	141.00	Izu Trench
83	31.00	142.00	Izu Trench
84	29.00	142.50	Izu Trench
85	27.00	143.00	Bonin Trench
86	25.00	143.00	Bonin Trench
87	23.00	143.00	Bonin Trench
88	21.00	144.50	Bonin Trench
89	19.00	146.00	Mariana Trench
90	16.50	147.00	Mariana Trench
91	14.00	146.00	Mariana Trench
92	12.00	144.00	Mariana Trench
93	12.00	141.00	Mariana Trench
94	46.00	152.00	Kuril Islands
95	48.50	155.50	Kuril Islands
96	51.00	158.00	Kamchatka
97	53.50	160.00	Kamchatka
98	56.50	161.50	Kamchatka
99	55.00	166.50	Aleutian Trench
100	51.50	176.00	Aleutian Trench
101	51.00	-178.50	Aleutian Trench
102	51.50	-173.00	Aleutian Trench
103	52.50	-167.50	Aleutian Trench
104	54.00	-162.50	Aleutian Trench
105	55.50	-157.50	Aleutian Trench
106	56.50	-152.00	Aleutian Trench
107	60.00	-153.00	Alaska
108	63.00	-151.00	Alaska
109	62.00	-145.00	Alaska
110	44.50	-130.00	Blanco Fault Zone
111	43.00	-126.00	Gorda Ridge
112	40.50	-128.00	Mendocino Fault Zone
113	40.50	-123.00	Mendocino Fault Zone
114	38.00	-119.00	Nevada
115	37.50	-122.00	San Andreas Fault Zone
116	35.00	-118.50	San Andreas Fault Zone
117	37.00	-118.00	Nevada
118	16.00	-88.00	Central America
119	16.00	-97.00	Central America
120	15.00	-94.00	Central America
121	14.00	-91.00	Central America
122	12.00	-88.00	Central America
123	10.00	-85.00	Central America
124	8.00	-82.50	Central America
125	5.00	-82.50	South America
126	6.00	-76.00	South America
127	8.00	-73.00	South America
128	.00	-80.00	South America
129	-5.00	-77.00	South America
130	-11.00	-74.00	South America
131	-12.00	-77.50	South America
132	-15.00	-75.00	South America
133	-17.50	-71.00	South America
134	-20.50	-69.00	South America
135	-22.00	-67.00	South America
136	-23.50	-70.00	South America
137	-25.00	-68.00	South America
138	-27.00	-71.00	South America
139	-28.00	-69.00	South America
140	-30.00	-71.50	South America
141	-31.00	-70.00	South America
142	-33.00	-72.50	South America
143	-34.00	-71.00	South America
144	-36.00	-73.00	South America
145	-56.00	-27.00	South Sandvich Islands
146	-57.00	-25.00	South Sandvich Islands
147	-58.50	-25.50	South Sandvich Islands

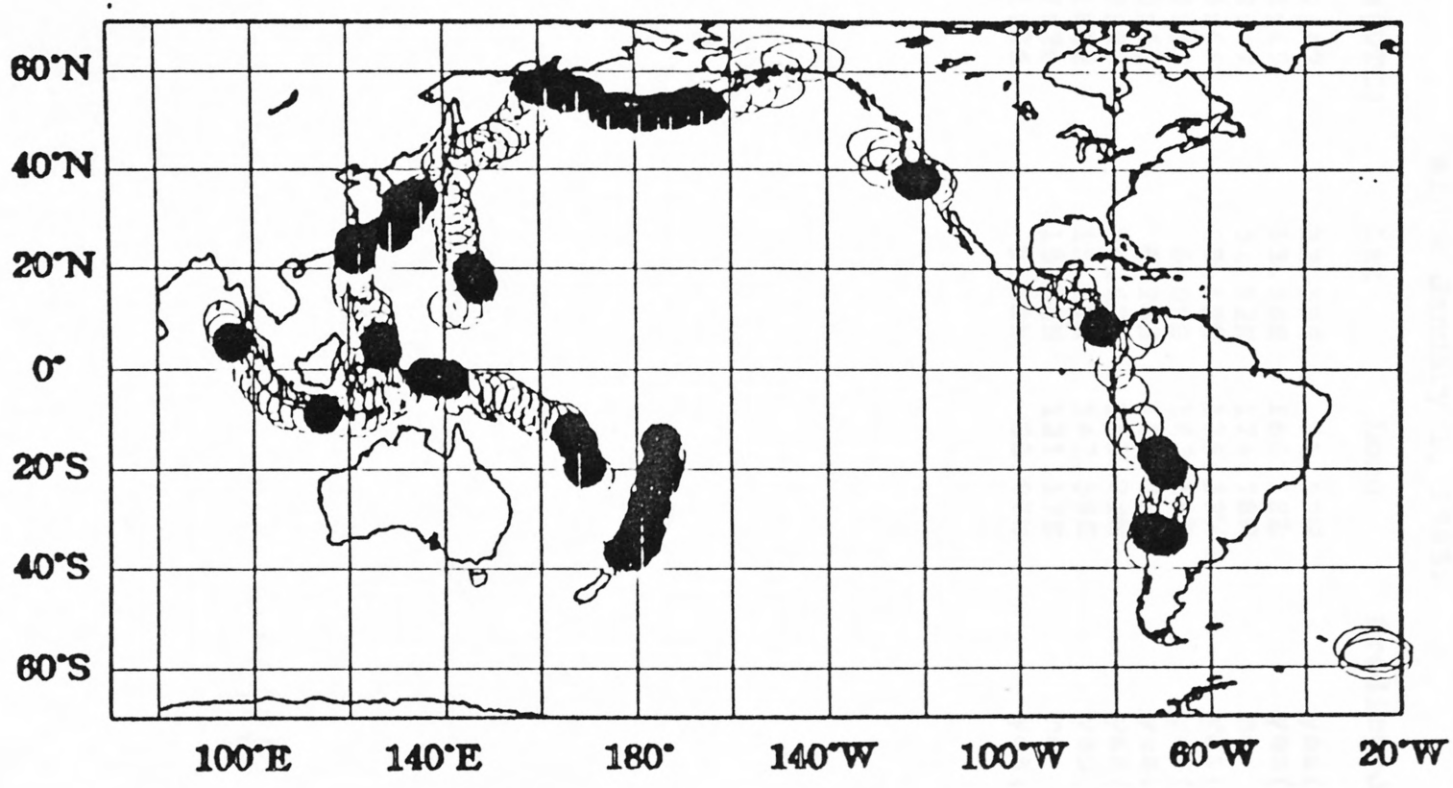
The symbols in the following tables indicate the status of each circle of investigation in six-month intervals from January 1, 1985 to January 1, 1992. ▥ indicates a TIP, · indicates no TIP, ■ indicates a successful prediction, (·) earthquake occurred at this location, and a blank indicates insufficient data to run the algorithm. Because the circles of investigation overlap an earthquake usually occurs in more than one circle.

M8 test B results for years from July 1, 1991.

▒ - TIP
. - no TIP
■ - successful prediction
(•) - earthquake located in this circle
- blank - insufficient data



Regions of Increased Probability of magnitude 7.5+ earthquakes for the period from January 1, 1992 to July 1, 1992



APPENDIX VI

An Alternate Null Hypothesis

James H. Dieterich
U. S. Geological Survey
Menlo Park, California

As part of the review process, the following test was developed to compare the results of the M8 test with a method of random assignment of TIPs based on the Poisson model of earthquake occurrence. At the request J. Healy, the following memorandum was prepared to describe the method employed for the test and results obtained.

April 21, 1992

Memo
To: J. Healy, V. Kossobokov and J. Dewey
From: Jim Dieterich
Subject: Comparison of M8 test results with 'Poisson' model

Thank you for the opportunity to look at and comment upon the draft Open-File Report, and for supplying the data needed to undertake the comparison that I describe below. I fully endorse the goals of this study to develop an objective test of the M8 algorithm. This is an important topic and I believe what you have done is both reasonable and constructive.

As I discussed previously, I have been interested in the question: How does M8 compare with a simple scheme of assignment of TIPs based on a Poisson model for earthquake occurrence? The Poisson model is widely regarded as an acceptable zero-order model of earthquake occurrence. Hence, departures of the M8 test results from a Poisson-based method of TIP assignment, would be useful for evaluating the value and performance of M8.

I have written a computer program for a simple implementation of this comparison. The test assigns TIPs randomly to the circles, weighting the probability of a TIP by a measure of the long-term average rate of earthquake activity in the circle. I think this is an appropriate null hypothesis for the M8 test. It is a variant of your comparison of M8 that is based on unweighted assignment of TIPs. However, the unweighted assignment of TIPs has the problem that all circles will not have the same rate of seismic activity and consequently, may not have the same chance of randomly producing an earthquake in some interval of time.

The measure of average rate of activity for this test is taken to be the number of

earthquakes M≥6.5 in each circle since 1900. For this I used the data you provided to me from the earlier draft version of the report.

The weighting parameter, F_i, for random assignment of a TIP for circle, i is

$$F_i = \frac{\text{number of earthquakes M≥6.5 in circle } i \text{ (since 1900)}}{\text{sum of number of earthquakes M≥6.5 over all circles (since 1900)}} .$$

The test is run in six month increments for the period covered by the M8 test. For direct comparison with the performance of M8, the total number of randomly assigned Poisson TIPs should be the same as produced by the M8 algorithm over the duration of the test. Hence, the average number of TIPs/6month, as declared by M8, over the entire 13 six-month intervals was used. For the weighted random TIP assignment, the probability, Pi, of a TIP being declared in circle, i for any six month interval is

$$P_i = F_i N ,$$

where N is the average number of TIPs per six month interval declared by M8 (*i.e.*, N = 409/13).

COMPUTATIONAL PROCEDURE

Iterate though time the intervals and circles i

1) For each interval and circle, draw a random number R (range 0 to 1)
2) If, $P_i > R$, then, declare a TIP for circle i

At end of simulation, count number of earthquakes M≥7.5 that had at least one TIP declared for the circles and interval in which it occurred. Such earthquakes are defined by M8 as a successful prediction.

Repeat the simulation

(Note: in practice it is only necessary to iterate over the circles and interval in which each M≥7.5 earthquake occurred)

CIRCUM PACIFIC TEST: GENERAL CHARACTERISTICS

Average number of earthquakes/circle for any half-year interval:

$$\frac{\text{26 circles affected by M≥7.5 EQs}}{\text{(13 intervals)(147 circles)}} = 0.014 .$$

TEST A - THE BASIC MODEL

The first comparison is with the original test of the M8 algorithm. The M8 algorithm generated a total of 409 TIP intervals. Table 1 gives the results of the random assignment of TIPs to circles for 100,000 simulations. The columns labeled 'Weighted' are the results obtained for assignment of TIPs using the weighting procedure, described above, based on historical rate of seismicity in each circle. The columns labeled 'Unweighted' employ a simple unweighted random assignment of TIPs that should be equivalent to the random test given in the report. The results of 'Unweighted' agree quite well with the random test of the report.

Table 1: Random Model A
(100,000 simulations)

	Weighted		Unweighted	
Success	Number	Cumulative percent	Number	Cumulative percent
0	30	100.0	197	100
1	365	100.0	1920	99.8
2	2411	99.6	8406	97.9
3	8862	97.2	18566	89.5
4	20759	88.3	25344	70.9
5	27541	67.6	23217	45.6
6	23930	40.0	14796	22.4
7	11983	16.1	5643	7.6
8	3585	4.1	1622	1.9
9	519	0.5	274	0.3
10	15	0	15	0

Results of M8 algorithm (From Healy, et al.)

Successful TIPs = 6

(success is defined as earthquake, M≥7.5, occurring in a circle with active TIP)

Successes/TIP interval: 6/409 = 0.015

Results for weighted random assignment of TIPs:

Average successes/simulation = 5.13

Successes/TIP interval: 5.13/409 = 0.013

Weighted random assignment of TIPs does as well, or better, than M8, 40% of the time.

TEST B - THE MODIFIED M8 TEST

This comparison is with the test of the M8 algorithm, modified to not automatically turn off a TIP if an earthquake, M≥7.5, occurs. The modified M8 algorithm generated a 446 TIP intervals in the 13 six-month intervals. Table 1 gives the results of the random assignment of TIPs for 100,000 simulations. Again, the results of 'Unweighted' agree quite well with the random test of the report.

Table 2: Random Model B
(100,000 simulations)

	Weighted		Unweighted	
Success	Number	Cumulative percent	Number	Cumulative percent
0	0	100.0	75	100.0
1	72	100.0	1254	99.9
2	1401	99.9	5909	98.7
3	5812	98.5	15158	92.8
4	16776	92.7	23607	77.6
5	26833	75.9	24844	54.0
6	27469	49.1	17747	29.2
7	15472	21.6	8301	11.4
8	5082	6.2	2557	3.1
9	1010	1.1	518	0.5
10	73	0.1	30	0

Results of M8 algorithm (From Healy et al.)

Successful TIPs = 8

Successes/TIP interval: 8/446* = 0.018

(* 480 TIPS declared in 7 years, estimated as 446 TIPs in 13 intervals)

Results for weighted random assignment of TIPs:

Average successes/simulation = 5.45 (4.67 for unweighted random)

Successes/TIP interval: 5.45/446 = 0.012

Weighted random assignment of TIPs does as well, or better, than M8, 6% of the time.

DISCUSSION

From the results of Table 1, it appears that the original M8 algorithm was not significantly better at declaring successful TIPs than the simple method of random

assignment based on the Poisson probability model.

The modified M8 algorithm (Test B) is more problematic. The results of Table 2 indicate that the weighted random model could do as well or better that the modified M8 only about 6 percent of the time. There is however, the troublesome issue of altering the M8 algorithm after the results of the first test were seen. While the arguments in support of the change seem plausible, it also seems evident that the modified algorithm cannot receive an independent test from the data set employed for Test A. For the circum-Pacific, I feel the test begins now and will be decided by future earthquakes. I look forward with interest to see how it comes out.

Finally, I would like to take this opportunity to offer an opinion, based on these results, on the present status and appropriate uses of M8. I think it is at least plausible that future earthquakes could support the modified M8 algorithm and yield a success rate comparable to that indicated by the B test. If for the sake of argument we accept that speculation, the probability of an earthquake, M≥7.5, occurring in a circle during a 6 month TIP interval would be about 0.018. This remains is a rather low probability, not much greater than the observed average number of earthquakes/circle for any half-year interval of 0.014. Also, the method does not predict specific earthquakes because any earthquake, M≥7.5, located within the large area of a circle will satisfy a TIP. In view of these results, I believe at present, the method is principally of scientific interest to a) possibly improve our understanding of underlying physics of earthquake occurrence, and b) attempt to develop improved algorithms that could someday serve as a basis for recommendations of public response.