U.S. CUSTOMS LABORATORY METHODS

USCL METHOD 27-47

Guidelines for Country-of-Origin Determinations of Distillate Petroleum Products from Iraq

SAFETY PRECAUTIONS

This method does not purport to address all of the safety problems, if any, associated with its use. It is the responsibility of the user of this method to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to its use.

0 INTRODUCTION

There has been strong evidence for the organic origin of petroleum since 1936 when Treibs demonstrated a link between chlorophyll-a in living photosynthetic organisms and porphyrins in petroleum (9.1). His work marked the beginning of modern organic geochemistry by his demonstration of the biological origin of a specific compound found in crude oil. Such compounds are known as biomarkers. Formally, we define biomarkers as compounds found in petroleum that are derived from previously living organisms.

Biomarkers found in petroleum are useful for determining the origin of petroleum because their relative concentrations vary from oil field to oil field. This variation is due to differences in source organic input material, depositional conditions, age of the oil, and integrated oil temperature over geologic time (9.2, 9.3). The petroleum geochemistry and analytical chemical of pristane, phytane, and heavier molecular

weight biomarkers have been extensively discussed and developed in the literature (9.4 - 9.7). Additionally, compound specific isotope ratio mass spectrometry has recently been used to extend our geochemical understanding of petroleum development to include low molecular weight biomarkers in the n-C4 through n-C20 region (9.8 - 9.10). Moreover, new compounds continue to be identified, described in the literature, and established as biomarkers useful for oil source corrections (9.11, 9.12).

The cited literature and references therein have clearly established that biomarkers extend through the entire molecular weight range of petroleum. Therefore, for the purpose of determining the origin of a petroleum sample, it is only necessary that the compounds used for comparison purposes vary in their relative concentration with respect to each other from oil field to oil field.

1 SCOPE AND FIELD OF APPLICATION

This method describes the origin of a petroleum sample by the analysis of an entire high resolution capillary gas chromatogram from the instant sample and all consulted reference samples. Also discussed are a series of biomarker and other low molecular weight compounds whose concentrations vary widely among various oil fields.
In this regard, a preliminary screening of any complex petroleum mixture can be performed by the profile analysis of a wide variety of compounds, while a detailed analysis of specific regions of the chromatograms is necessary to distinguish between two very similar, yet different, petroleum oils. The specific problem of the determination of the origin of a petroleum sample via its profile is but one representative of the general class of problems known as Profiling Complex Mixtures (9.13). This profiling is rooted in mathematics, whereby, the fundamental problem is to determine if two objects are the same, a very difficult task when two objects are distinct, yet very similar (9.14).

The comparison of an instant petroleum sample to one or more reference petroleum samples is a mathematical problem in the specialized subject area of pattern recognition. Methods, by which pattern recognition occurs, range from a visual comparison, where the mathematics are performed in some unknown manner by the human brain, to a numerical comparison, where the mathematics are performed by a computer using a known and well described set of data in conjunction with a software program, compiler, and operating system with known and available source code.

This method describes a primary method for visual pattern recognition and a supplementary method for determining the relative mathematical similarity of a sample chromatogram to the members of a set of reference chromatograms.

2 REFERENCES

"Country-of-Origin Determinations of Distillate Petroleum Products from Iraq"

Neal D. Byington and Larry D. Fluty
Customs Laboratory Bulletin
(submitted)

3 REAGENTS AND APPARATUS

3.1 Hewlett/Packard Model 5890 Series II Plus capillary gas chromatograph with a split/splitless injector, flame ionization detector, liquid carbon dioxide sub-ambient cooling oven attachment, autosampler control module, and electronic pressure control.

3.2 Hewlett/Packard Model 7673A autosampler fitted with a 10 µL syringe

3.3 Hewlett/Packard Chem Station and software (PC)

3.4 Gas Chromatograph operating conditions

3.4.1 Split/splitless injector with a cup injector liner

3.4.2 J&W DB-1, 60-meter, capillary column, part number 122-1063. This non-polar column has a one-micron film thickness and an internal diameter of 0.25 mm.

3.4.3 Compressed helium carrier gas with a regulator for the capillary column.

3.4.4 Compressed air with a regulator for the FID detector.

3.4.5 Compressed hydrogen with a regulator for the FID detector.

3.4.6 Compressed carbon dioxide with a liquid dip tube feeder to provide liquid carbon dioxide cooling to the gas chromatograph oven.

3.4.7 Supelco High Capillary Gas Purifier

3.4.8 Supelco Thermogreen LB-2 septum.

3.5 Eppendorf pipets:

a) 100-ul pipet
b) variable 100-1000 ul pipet

3.6 Vortex Mixer

3.7 Cyclohexane, ACS reagent grade

3.8 Reference samples:

3.8.1 Gasoline

3.8.2 Kerosene

3.8.3 Crude Oil

3.8.4 Distillate Fuel Oil

3.8.5 Crude oil samples from oil fields of relevance to the instant analytical sample

4 SAMPLE PREPARATION

4.1 To prepare a sample of distillate petroleum for the gas chromatograph, 100 ul of distillate petroleum and 900 ul of cyclohexane are measured into a GC vial and capped.

4.2 Vortex the mixture for 15 seconds.

5 EXPERIMENTAL PROCEDURE

5.1 The autosampler is loaded to run a set of samples such that each of the first two vials, the last vial, and one vial between each sample vial is a blank vial of cyclohexane. This provides significant evidence that the gas chromatograph is operating properly both before and after the analysis of any specific sample.

5.2 A 1.0 µL sample of the mixture is injected into the gas chromatograph via an autosampler using a 10 µL syringe.

5.3 Gas Chromatograph run conditions

5.3.1 The helium column carrier gas is under electronic pressure control

with a column flow rate set at 1.5 mL/min +/- 0.1 mL/min at 30 degrees C. The helium carrier gas is purified through an electrically heated Supelco High Capacity Gas Purifier

5.3.2 Split/splitless injector (cup injector liner packed with 10% OV-1 on Chromosorb-W) is used in the split mode at a split vent volume of 140 mL/min +/- 2.0 mL/min

5.3.3 The temperature of the injection port is isothermally maintained at 275 ºC.

5.3.4 The oven temperature controller is set to hold the oven temperature for 5 minutes at 30 ºC via liquid carbon dioxide sub-ambient cooling of the oven and then to ramp the oven at 5 ºC/min to 275 ºC and hold it at that temperature for

a) 6 minutes for gasoline range material,
b) 40 minutes for mid-range distillate material,
c) 90 minutes for crude oil.

5.3.5 The temperature of the detector is isothermally maintained at 275 degrees C.

6 PRECISION

6.1 The resolution of the column is considered to be adequate when the system provides baseline resolution of the following compounds using the conditions described in the experimental procedure.

6.1.1 For gasoline and kerosine distillation range materials, trans-2-butene and cis-2-butene

6.1.2 For crude oil and distillate fuel oil materials, n-C17/pristane and n-C18/phytane peak pairs.

7 RESULTS

7.1 Figures I through IV are the high resolution capillary gas chromatograms of the profile and three key comparison sections of a distillate fuel oil obtained from Basrah Light crude oil from Iraq.

7.1.1 Figure I, Distillate Fuel Oil Obtained from Basrah Light Crude Oil, is the entire profile chromatogram of the sample.

7.1.2 Figure II, Distillate Fuel Oil Obtained from Basrah Light Crude Oil, is the partial chromatogram of the sample in the n-C8 through n-C9 hydrocarbon range.

7.1.3 Figure III, Distillate Fuel Oil Obtained from Basrah Light Crude Oil, is the partial chromatogram of the sample in the n-C9 through n-C11 hydrocarbon range.

7.1.4 Figure IV, Distillate Fuel Oil Obtained from Basrah Light Crude Oil, is the partial chromatogram of the sample in the n-C16 through phytane hydrocarbon range.

7.2 Figures V through VIII are the high resolution capillary gas chromatograms of the profile and three key comparison sections of Iranian Heavy crude oil from Iran.

7.2.1 Figure V, Iranian Heavy Crude Oil, is the entire profile chromatogram of the sample.

7.2.2 Figure VI, Iranian Heavy Crude Oil, is the partial chromatogram of the sample in the n-C8 through n-C9 hydrocarbon range.

7.2.3 Figure VII, Iranian Heavy Crude Oil, is the partial chromatogram of the sample in the n-C9 through n-C11 hydrocarbon range.

7.2.4 Figure VIII, Iranian Heavy Crude Oil, is the partial chromatogram of the sample in the n-C1 through phytane hydrocarbon range.

8 GUIDELINES FOR COUNTRY-OF-ORIGIN DETERMINATIONS

8.1 Introduction

8.1.1 The country-of-origin of a petroleum sample is determined by a chemist following an examination of all available data, including a detailed comparison of the entire gas chromatogram for the sample and all consulted reference samples with particular attention to selected compounds whose relative concentrations are very sensitive to origin-dependent factors.

The selection of how many and which reference samples and specific compounds within those samples to use for detailed origin dependent comparison purposes may change as examination of an instant sample proceeds. The detailed comparison of an instant sample with selected reference samples can be performed by any of a variety of individual techniques or a combination of techniques, which range from visual to mathematical comparisons of the relevant data.

Supplemental analytical methodology and other data of any type and from any source may be used to provide additional data to assist in the determination of the country-of-origin of an instant sample. The scope of this section encompasses two sets of guidelines, one visual and the other

mathematical, for the determination of the country-of-origin of distillate petroleum products from Iraq.

8.1.1 The relative ratios of the selected origin-dependent gas chromatographic peaks must vary between oils of different origin and remain relatively constant from tanker for oils of the same origin.

8.1.2 Each peak in a gas chromatogram of a complex mixture such as a crude oil or any distillate fraction probably consists of more than one organic compound. It is not necessary to know the identity of the major compound in each peak that is used for comparison purposes.

8.2 Visual Comparison Guidelines

8.2.1 In all cases, chromatographic data from an instant sample are visually compared with similar data from selected reference samples.

8.2.1.1 In all cases, raw chromatographic data obtained by the experimental procedure as described in section 5, using the sample preparation as described in section 4, and using the equipment as described in section 3, are processed into the formats of Figures I through IV and analyzed as described in sections 8.2.2 through 8.2.5.

8.2.2 Figure 1 provides the entire chromatogram which clearly illlustrates the very general chromatographic profile features that are common to this distillate product from Iraq.

8.2.2.1 Of particular note in Figure 1 is the very wide asymmetrical distillate range from n-C7 through n-C26 that is centered in the n-C13 to n-C14 region.

8.2.2.2 This fuel oil contains distillate material from the top of the naphtha region in the gasoline range, through the number one fuel oil kerosene range, through the number two fuel oil/diesel fuel range, and up into the bottom of the number four fuel oil range. It is a straight-run distillate fuel oil from a distillation tower that is maximized to produce distillate fuel oils suitable for use in either diesel engines or heating stoves.

8.2.3 Figure II provides an expansion of the entire chromatogram covering the n-C8 to the n-C9 range.

8.2.3.1 The light hydrocarbons between these two n-alkanes have very distinctive relative ratios which prominently reflect origin differences among many crude oils, particularly those very similar oils on the Western side of the Arabian Gulf.

8.2.3.2 The peaks at the retention indexes of 832, identified in the literature as n-propylcyclopentane, and 844, similarly identified in the literature as 1,1,3-trimethylcyclohexane, are very significant (9.10).

8.2.3.3 It has been observed that these compounds have consistently and uniquely equal in intensity for Basrah Light derived fuel oils.

8.2.4 Figure III provides an expansion of the entire chromatogram covering the n-C9 to the n-C11 range.

8.2.4.1 The light hydrocarbons between these two n-alkanes have a general pattern that is very sensitive to origin differences.

8.2.4.2 In the n-C9 to n-C10 region, there are two sets of unidentified doublet peaks that are of particular interest.

8.2.4.2.1 The leftmost of the first set is at an uncorrected retention index of 964 while the leftmost of the second set is at an uncorrected retention index of 973.

8.2.4.2.2 For Basrah Light crude oil derived distillate fuel oils, the right member of each set is of a lower peak height than the leftmost member of each set.

8.2.4.3 In the n-C10 to n-C11 region, there are two sets of unidentified triplets that are of particular interest.

8.2.4.3.1 The rightmost member of the first set is at an uncorrected retention index of 1026, and similarly, the rightmost member of the second set is at 1066.

8.2.4.3.2 For Basrah Light crude oil derived distillate oils, the rightmost members of each set are higher than those to the left.

8.2.5 Figure IV provides an expansion of the entire chromatogram covering the n-C16 to the phytane region.

8.2.5.1 Using uncorrected retention indexes, of particular interest is the relationship between the peak for phytane at 1816, a peak that is predominantly pristane at 1712, and two unidentified peaks at 1674 and 1654.

8.2.5.1.1 For oils obtained from Basrah Light crude oil and some other Northwest Arabian gulf crude oils, a nearly straight line can be drawn across the top of the peaks at uncorrected retention indexes of 1816, 1712, and 1674 which will intersect the peak at 1654 at one half to three quarters of its height.

8.2.5.1.2 The very low pristane and phytane peak heights relative to

the adjacent n-alkanes is indicative of a very old organic source input material and characteristic of many oils in the Western Arabian Gulf, and this, among many other differences, distinguishes them from the Iranian oils.

8.3 Mathematical Comparison Guidelines

8.3.1 Visual pattern recognition may be assisted by mathematical techniques. A purely mathematical analysis is available through the use of the PASCAL program, PASCAL Pattern Similarity Program (PPSP). A detailed description of this program including full source code is provided in Annex I. This program, PPSP, provides a numerical measure of the similarity between sample and reference chromatograms. (9.13, 9.15-9.18).

8.3.1.1 In all cases where mathematical techniques are used, chromatographic data from an instant sample and all selected reference samples are visually compared before and after any mathematical comparisons. The conclusions from both the visual and the mathematical techniques are examined by the chemist prior to determining an origin.

8.3.2 The program, PPSP, will accept chromatographic data in the form of either peak heights or peak areas. Visual comparisons are more easily interpreted if peak height data are used in the mathematical program, but either is acceptable.

8.3.2.1 Select at least four peaks from the chromatogram whose height or area varies with the origin of the petroleum sample.

8.3.2.2 Select reference samples from suspect origin oils and other oils similar to the instant sample oil.

8.3.2.3 Develop the sample data set file as described in Annex 1,1.1, and 1.3.

8.3.2.4 Develop the reference data set file as described in Annex 1, 1.1, and 1.4.

8.3.2.5 Full source code for the program is provided in Annex 1.8. Further refinement may be obtained by modifying the source code to permit the program to use different Minkowsi Distances. A listing of alternate Minkowski Distances and why they may be useful is provided in Annex 1.5. A list of the procedures whose source code must be modified to enable alternate Minkowski Distances to be utilized is provided in Annex 1.6. A list of each procedure in the PPSP is provided in Annex 1.2.

8.3.2.6 Compile and run PPSP with the developed sample and reference data files according to the directions in Annex 1.1.

8.3.2.7 The results file from PPSP will provide a mathematical metric distance between the data points for the chromatographs from the instant sample to teach reference sample.

The mathematical distances between chromatographs are sorted by numerical distance and are computed for two different metrics.

8.3.2.8 The reference samples with the smallest Minkowski metric distances are considered to be more similar to the instant sample than those whose distances are larger.

9 BIBLIOGRAPHY

9.1 Treibs, A., *Angewandte Chemie*, 49, 682, 1936.

9.2 Seifert, W.K., *Fortshritte der Chemie organischer Naturstoffe*, 32, 2, 1975.

9.3 Peters, K.E. and Moldowan, J.M., *The Biomarker Guide*, Prentice Hall, New Jersey, 1993.

9.4 Philip, R.P., and Qung, J.N. *Analytical Chemistry*, 60(15), 887A, 1988.

9.5 Hostettler, F.D., et al, *Marine Pollution Bulletin*, 24(1), 15, 1992.

9.6 Hwang, R.J., *J. Chrom. Sci*., 28, 109, 1990.

9.7 McDonal, T.J. and Kennicutt II, M. C., *LC-GC*, 10(12), 935, 1992.

9.8 Bjoroy, M., et al, *Org. Geochem*., 19(1-3), 89, 1992.

9.9 Bjoroy, M., et al, *Org. Geochem*., 22(3-5), 355, 1994.

9.10 Bjoroy, M., et al, *Org. Geochem*., 21(6,7), 761, 1994.

9.11 Ellis, L., et al, *Geochemical et Cosmochimica Acta*., 60(23), 4747, 1996.

9.12 Ellis, L., et al, *Org. Geochem*., 24(1), 57, 1996.

9.13 Byington, N.D., Pacific Conference, San Francisco, 1996.

9.14 Pavel, M., *Fundamentals of Pattern Recognition*, Marcel Dekker, Inc., New York, 1989.

9.15 Byington, N.D. and Poon, E.Y., Pacific Conference, San Francisco, 1988.

9.16 Byington, N.D., Pacific Conference, Pasadena, 1989.

9.17 Byington, N.D. and England, R.E., Pacific Conference, San Francisco, 1990.

9.18 Byington, N.D., Rocky Mountain Conference on Analytical Chemical Denver, 1996.

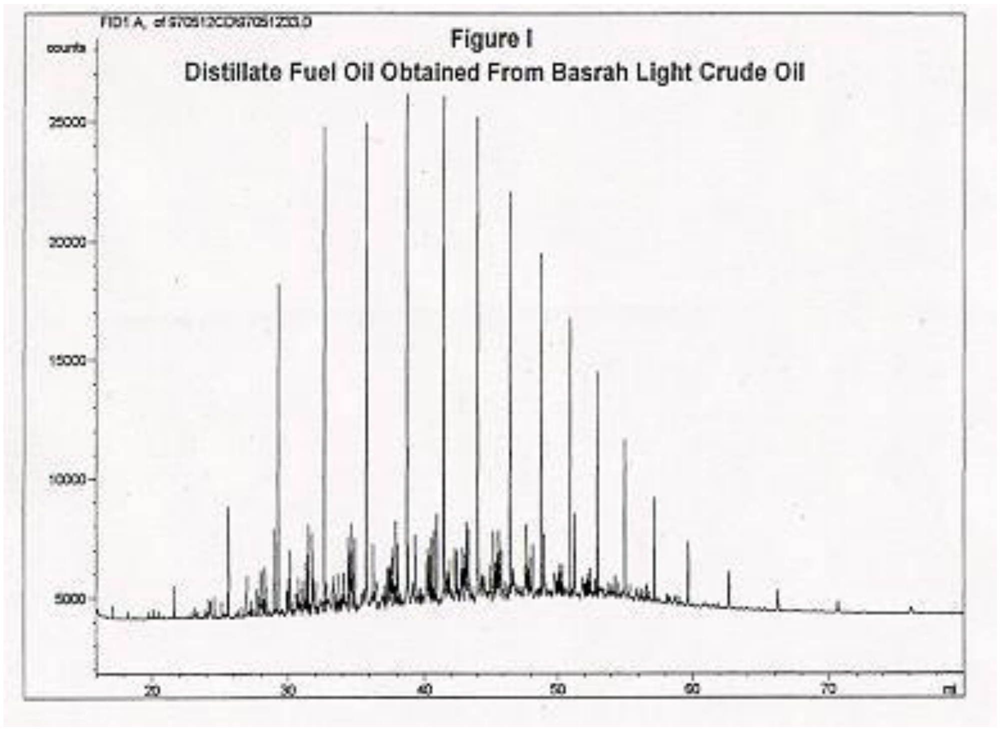
Annex 1 PASCAL Pattern Similarity Program (PPSP)

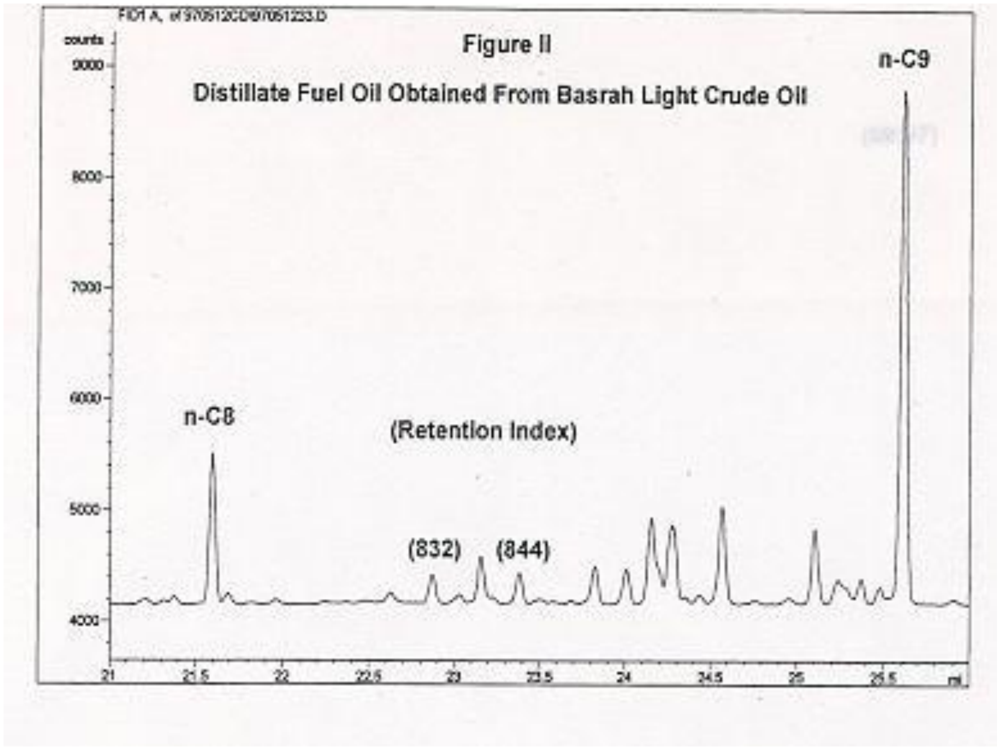
Annex 1.1 Directions on How to Use the PPSP

Annex 1.2 Listing and Description of Each Procedure in PPSP

Annex 1.3 Example of a Sample Data File

Annex 1.4 Example of a Reference Results File


FID1 A, of 970512CD\97051233.D
Figure I
Distillate Fuel Oil Obtained From Basrah Light Crude Oil
counts
25000
20000
15000
10000
5000
20
30
40
50
60
70
mi


FID1 A, of 970512CD\97051233.D
Figure II
Distillate Fuel Oil Obtained From Basrah Light Crude Oil
counts
9000
8000
7000
6000
5000
4000
n-C8
n-C9
(Retention Index)
(832)
(844)
21
21.5
22
22.5
23
23.5
24
24.5
25
25.5
mi

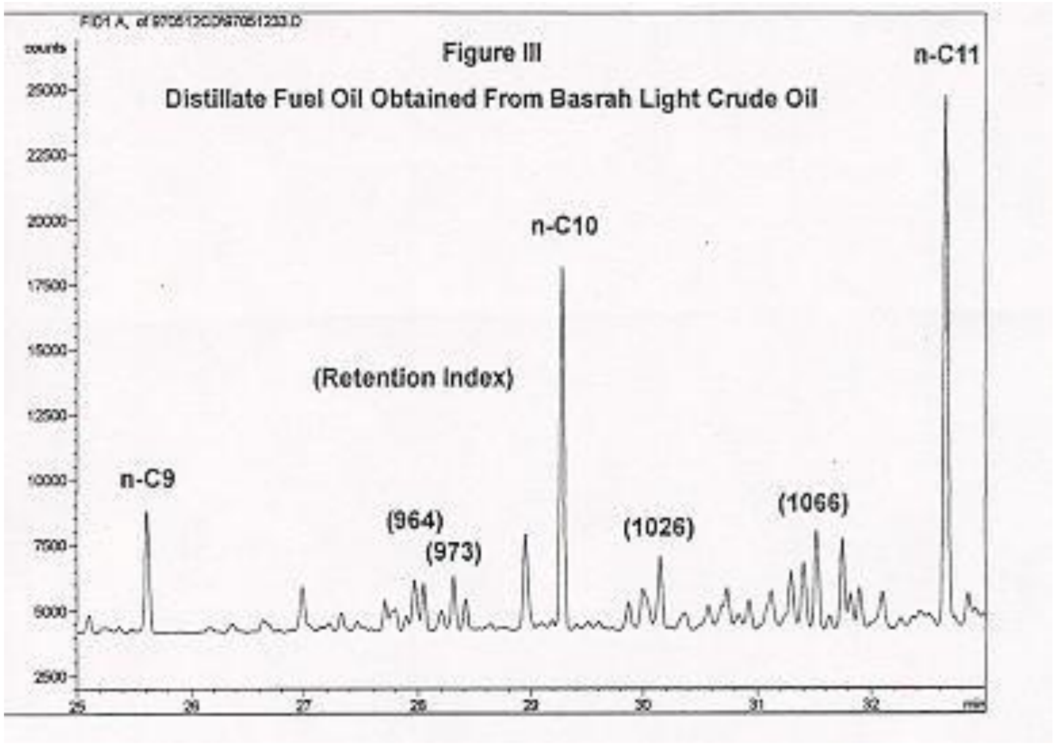


Figure III

Distillate Fuel Oil Obtained From Basrah Light Crude Oil

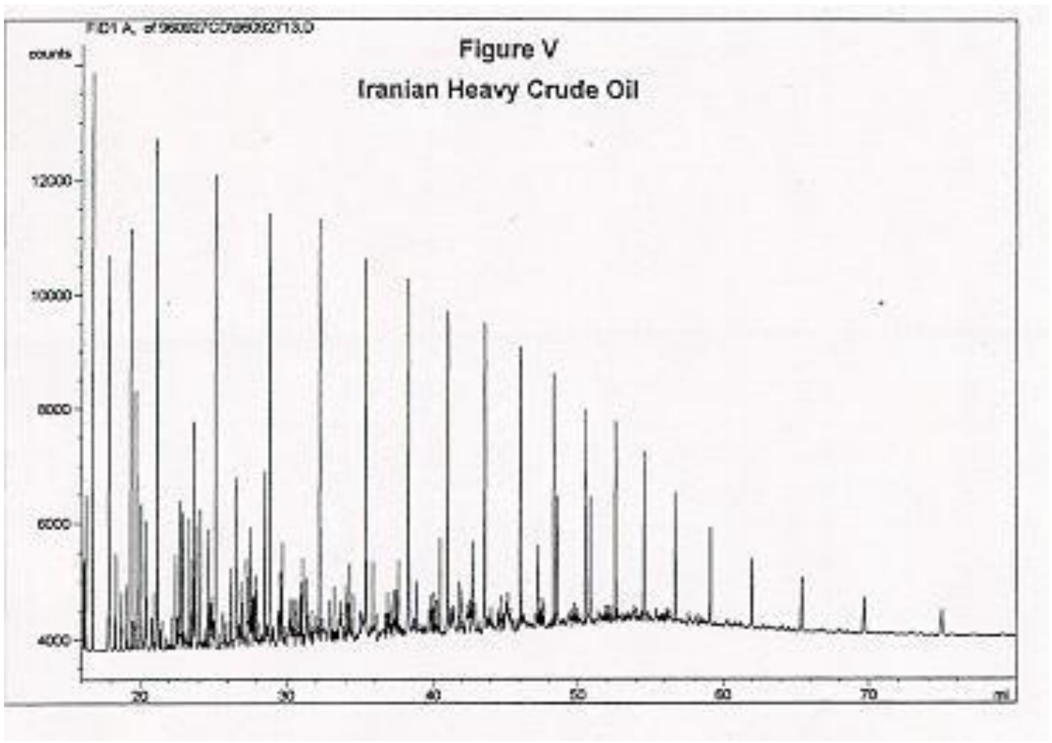
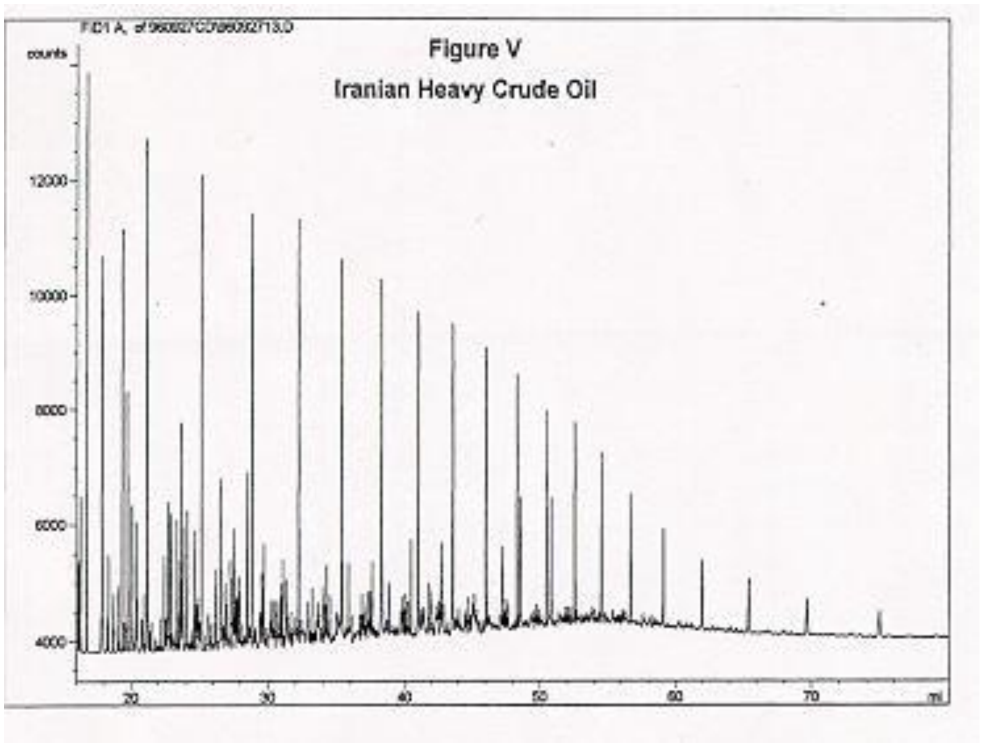
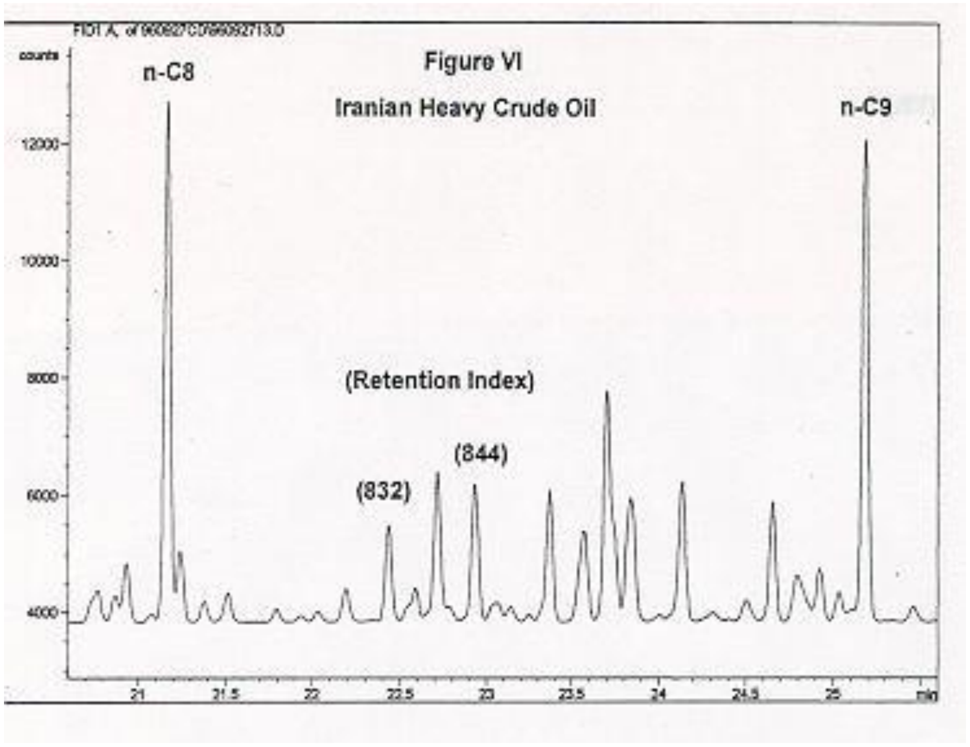


Figure V

Iranian Heavy Crude Oil


FID1 A, of 960927C\960092713.D
Figure V
Iranian Heavy Crude Oil
counts
12000
10000
8000
6000
4000
20
30
40
50
60
70
min


FID1 A, of 960927C\960092713.D
Figure VI
Iranian Heavy Crude Oil
n-C8
n-C9
(Retention Index)
(832)
(844)
counts
12000
10000
8000
6000
4000
21
21.5
22
22.5
23
23.5
24
24.5
25
min

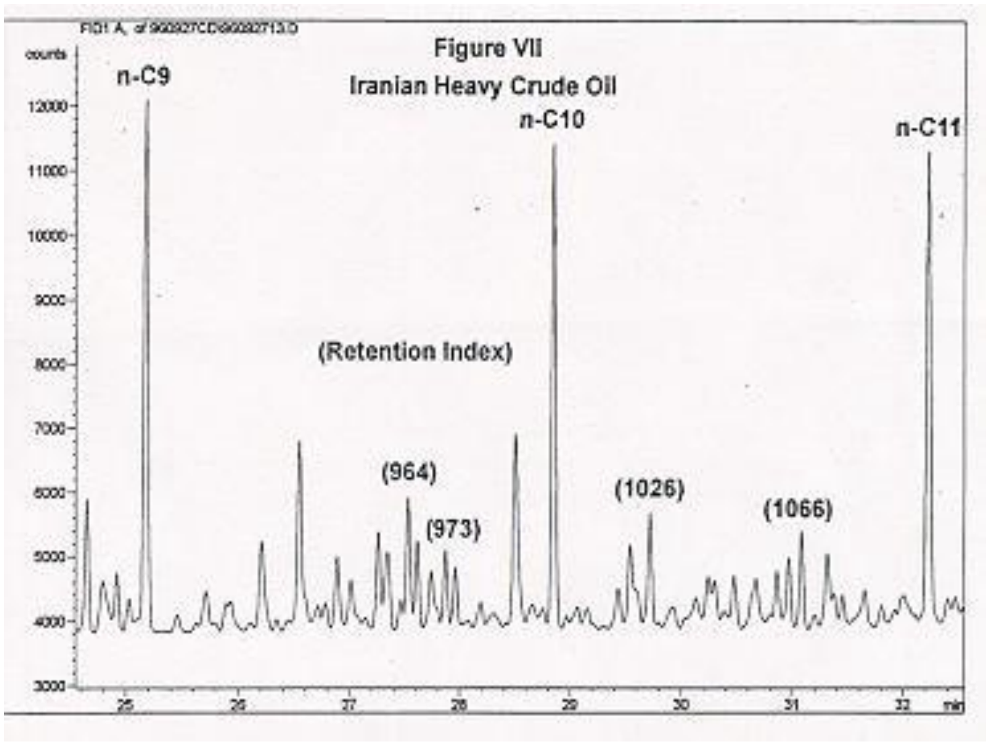


Figure VII
Iranian Heavy Crude Oil
counts
n-C9
n-C10
n-C11
(Retention Index)
(964)
(973)
(1026)
(1066)
min

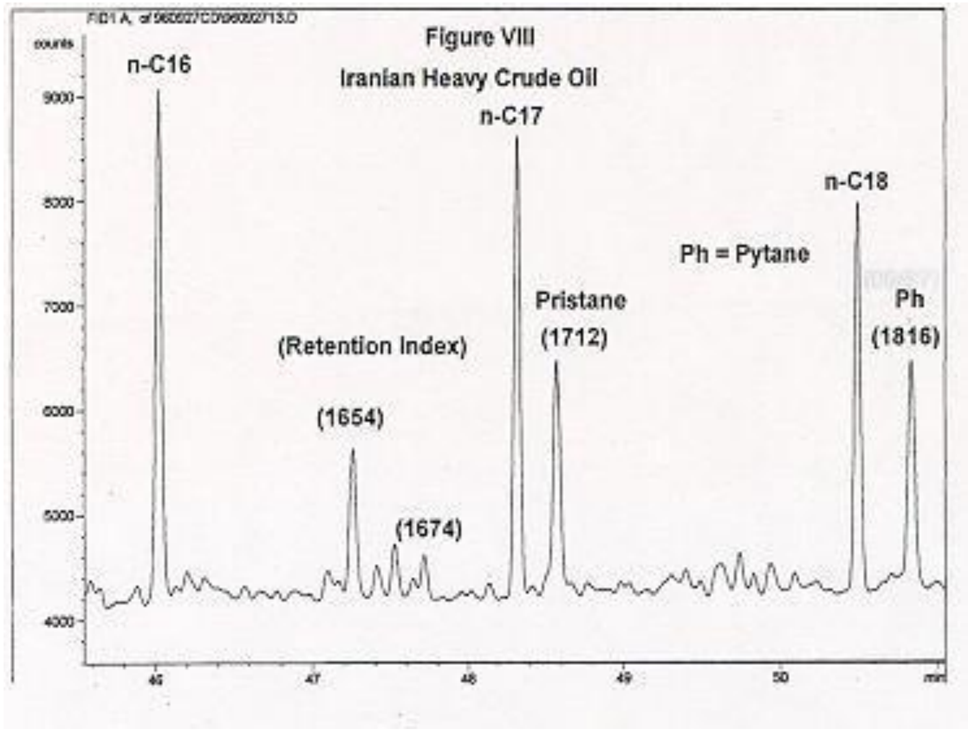


Figure VIII
Iranian Heavy Crude Oil
counts
n-C16
n-C17
n-C18
Ph = Pytane
Pristane
(1712)
Ph
(1816)
(Retention Index)
(1654)
(1674)
min

Annex 1 PASCAL Pattern Similarity Program (PPSP)

Annex 1.1 Directions on How to Use the PPSP

1.1.2 Directions on how to run the PASCAL Pattern Similarity Program (PPSP) using TURBO PASCAL version 4.0

All keyboard entries for this program are enclosed in curly brackets {}. If a return key is needed, CR is indicated after each entry. If the screen is cleared after an entry, cls is indicated. Responses from the computer are typed as they appear on the screen and are enclosed in [brackets]. Comments are enclosed in (parentheses).

1.1.3 The two example input data files, sample.txt and ref.txt, have been set-up and provided to the reader for the purpose of validating a local implementation of the PPSP program whose source code is supplied in **Annex 1.8**. The example sample input data file is located in **Annex 1.3**. The example reference input data file is located in **Annex 1.4**.

1.1.4 The source code for this program as provided in **Annex 1.8** is configured to compile and run without modification from the C:\ drive. The C:\ drive must contain the following four items for this program to properly run:

A. Turbo Pascal Version 4.0
B. This program, PPSP.PAS
C. A sample input data file in the proper format
D. A reference input data file in the proper format

1.1.5 The same data format is used for both the sample data file and the reference data file. Each data file is an ASCII text file with a precise definition of just what data go in which columns. The reference and sample data input files can be created by using any ASCII text editor or by using the integrated programming environment editor in Turbo Pascal. In the later case, type {turbo}CR at the C:\ prompt and then {alt E} to access the text editor. When data entry is complete, the file is saved by pressing {F2} and providing a name for the file. The data format definition is as follows:

A. Columns 1-15 -- serial no. (e.g. iraq_sample_001)
B. Columns 16-21 -- run no. (e.g. 210792)
C. Next, enter the retention time and peak area in pairs, separating each entry by a space as this is a space delimited format scheme -- (e.g. 22.87 312 23.39 327 49.00 2998 51.26 3800 -1 -1).
D. Continue to enter the data into the same one line until the whole line (256 characters) is filled.
E. When the last entry has been entered, follow this by two negative numbers (e.g. 22.87 312 23.39 327

49.00 2998 51.26 3800 -1 -1) as this is a flag to indicate the last entry is reached.

F. In the sample and reference data files, do not leave any blank spaces either before the first character space or after the last two characters, which are -1.

G. When data entry is finished, press {F2} and the file name of your choice to save the file and leave the edit mode.

Annex 1.1.6 A. If you are still in TURBO PASCAL, skip this step and proceed to **Annex 1.17**.

B. If you are in MS-DOS and not in the C:\ directory, type {CD C:\}CR. You should now be at the C:\ prompt.

C. At the C:\ prompt, type {turbo}CR to start Turbo Pascal.

D. To load the program PPSP.PAS into the Turbo Pascal compiler, press {alt + F} and then {L}, and then at the input line the filename of the program {PPSP.PAS} followed by a CR.

Annex 1.1.7 To compile and run the program PPSP.PAS press {alt + C} then {C} and then {CR} to compile the program and {Esc} to get back to the command portion of the Turbo Pascal compiler. The program is now ready to run.

Annex 1.1.8 To run the program, type {alt + R}. The program will start to run. The following instructions are suggested preliminary responses to questions that the program will ask the user. The entries here are only examples for the user to consider as possible responses.

A. [Do you need instructions on how to prepare the input files?]
[Enter Y for yes and N for no,]
Enter {n}CR

B. [Are the input files ready to be processed?]
[Enter Y for yes and N for no.]
Enter {y}CR

C. [The input files are ready and program will continue.]

D. [Enter the size of the window for the retention time.]
(The recommended size is between 0.05 to 0.1)
Enter {0.05}CR

E. [The size of the window is 0.05]

F. [Enter the number of the closest patterns you want to list.]
(The suggested values are between 10 to 30)
Enter {10}CR

[10 closest patterns to the sample pattern will be listed.]

G. [Enter the name of the sample file.]
(A copy of trial sample data is stored in iraqsmp.txt)
Enter {iraqsmp.txt}CR

H. [Enter the name of the reference file.]
(A copy of trial reference data is stored in iraqref.txt)
Enter {iraqref.txt}CR

I. [Enter the name of the output file for the summary.]
(A suggested name is c:\iraqres.txt)
Enter {c:iraqres.txt}CR

L. [Do you want to store the details of the result?]
[Enter Y for yes and N for no.]
Enter {y}CR

M. [Enter the name of the output file for details.]
(A suggested name is c:\iraqdet.txt)
Enter {c:\iraqdet.txt}CR

N.
```
[>>>>>iraq_sample_001 210792<<<<<]
[Peak Number 1        22.87312]
[Peak Number 2        23.39327]
[Peak Number 3        49.002998]
[Peak Number 4        51.263800]

[Enter the peak number of the peak you want to
normalize.]
```
Enter {1}CR

O.
```
[Peak  1Retention Time22.87  Peak Area  312 is]
[chosen to be normalized.]

[Program is running....]

[The summary is stored in the file c:iraqres.txt]
[The details are stored in the file c:\iraqdet.txt]
```

P. [Do you want to continue with another set of data?]
[Enter Y for yes or N for no.]
Enter {n}CR

Q. [User does not wish to continue.]
[Program will exit and return to Turbo Pascal.]

Annex 1.1.9 The program output results are in a file on the C drive at c:\iraqres.txt.This file is an ASCII text file that may be read by any text editor, the editor in Turbo Pascal, or any word processor.

Annex 1.1.10 The details of the program output results are in a file on the C drive at c:\iraqdet.txt. This file is an ASCII text file that may be read by any text editor, the editor in Turbo Pascal, or any word processor.

Annex 1.2 Listing and Description of Each Procedure in the PPSP

The name for each of the 21 procedures (**Annex 1.2.1** through **Annex 1.2.21**) and the main program (**Annex 1.2.22**) of the pattern recognition program, PASCAL Pattern Similarity Program (PPSP), is listed along with a brief description. The source code listing for this Pascal program consists of 1206 lines of version 4.0 of Turbo Pascal. Following the listing of the name of each procedure is a pair of numbers joined with a dash listing the beginning and ending source code line numbers for that procedure. This is followed by a brief description of what that procedure accomplishes. An asterisk (*) between the number of the procedure and the name of the procedure indicates that the procedure is one of the three procedures (**Annex 1.2.12**, **Annex 1.2.15**, and **Annex 1.2.20**) and the main program (**Annex 1.2.22**) whose source code must be modified when the comparison metric is changed.

Annex 1.2.1 CheckReal 61-93

This procedure checks to see if the input data is a real number. If it is not a real number, the program will come back and ask the user for the data until the appropriate type of data has been entered.

Annex 1.2.2 CheckInteger 94-123

This procedure checks to see if the input data is an integer. If it is not an integer, the program will come back and ask the user for the data until the appropriate type of data has been entered.

Annex 1.2.3 CheckSameName 124-140

This procedure checks to see if the file name for the details of the calculations is the same as the file name for the summary of the calculations.

Annex 1.2.4 CheckNoName 141-159

The procedure checks to see if a name has been entered.

Annex 1.2.5 CheckBlank 160-188

This procedure checks to see if there is a blank in the entered name, if so it will ask for the name to be entered again.

Annex 1.2.6 CheckFile 189-235

This procedure checks to see if the input file exists, if not, it will ask for it until it exists.

Annex 1.2.7 CheckFileExist 236-294

This procedure checks to see if the output file exists, if so, it asks if you want to write over it or provide the name of a new output file.

Annex 1.2.8 CheckSummaryName 295-336

This procedure checks the name of the file entered by the user to store the summary of the calculations. If no name is entered, or the same name is entered, or the name has blanks, the user will be asked to enter the name again.

Annex 1.2.9 CheckDetailName 337-379

This procedure checks the name of the file entered by the user to store the details of the calculations. If no name is entered, or the same name as the one for the summary, or the name has blanks, the user will be asked to enter he name again.

Annex 1.2.10 Instruction 380-482

This procedure provides instructions on how to set up the input files and asks, in separate questions, for the user to input the window retention time size and the number of closest references to list.

Annex 1.2.11 GetFiles 483-553

This procedure asks, in separate questions, for the user to input the filename for the sample data file (an example is provided in **Annex 1.3**) which contains the sample input data, the filename for the reference data file (an example is provided in **Annex 1.4**) which contains the reference input data, the filename into which the program should put a summary of the program results, and the filename into which the program should put the program details for the results. If no filename is provided for the details for the results, they will automatically be sent to the file junkfile.txt on the c:\ drive.

The Turbo Pascal Version 4.0 compiler, this program, the sample data file, and the reference data file must all be in the same directory on the computer. The output files will be created and placed into this directory automatically. This program is configured to operate in the C:\ directory without modification.

Annex **1.2.12*** WriteResult 554-610

This procedure writes a summary of the results of the pattern comparison into the file specified in the procedure GetFiles (**Annex 1.2.11**).

Annex **1.2.13** WriteSample611-646
This procedure writes the sample input data for the sample tobecompared from the sample data file (an example is provided in **Annex 1.3**) to the file specified in the procedure GetFiles (**Annex 1.2.11**).

Annex **1.2.14** WriteRef 647-675

This procedure writes the reference input data for the reference samples to be compared from the reference data file (an example is provided in **Annex 1.4**) to the file specified in the procedure GetFiles (**Annex 1.2.11**). The similarity coefficient is also written.

Annex **1.2.15*** WriteLongResult 676-716

This procedure writes the program details for the results of the comparison calculation into the output file as specified by the user in the procedure Getfiles (**Annex 1.2.11**).

Annex **1.2.16** GetData 717-775

This procedure reads into the program data from a data file.

Annex **1.2.17** NormalizeSample 776-827

This procedure asks the user to choose the peak to be normalized to one and to input that peak number.

Annex **1.2.18** GetRefPeak 828-924

This procedure picks out the peaks in the reference pattern that correspond to the peaks in the sample pattern so that the sample pattern can be compared to the reference pattern. The procedure to select the most appropriate reference peak is to find all reference peaks that are within the user supplied gas chromatographic retention time resolution of the sample peak and then to select from those peaks the reference peak with the smallest retention time difference between the retention time of the sample peak and the candidate reference peak.

Annex 1.2.19 NormalizeReference 925-961

This procedure will normalize the peaks in the reference set of data according to the peak chosen by the user for the sample set of data.

Annex 1.2.20* GetCoefficient 962-992

This procedure computes two coefficients for each reference sample based upon two methods, the absolute difference distance and the Minkowski distance for k = 2, the classic Euclidean distance.

Annex 1.2.21 OrderArray 993-1074

This procedure enters the reference sample into the list of best reference samples in the ascending order of similarity coefficients.

Annex 1.2.22* Main Program 1075-1206

This is the main program. It calls all of the preceeding procedures.

Annex 1.3 Example of a Sample Data File

```
iraq_sample_001210792 22.87 312 23.39 327 49.00 2998 51.26 3800 -1 -1
```

Annex 1.4 Example of a Reference Data File

```
iraq_sample_001210792 22.87 312 23.39 327 49.00 2998 51.26 3800 -1 -1
basl_reference1210758 23.17 495 23.69 503 49.19 4821 51.44 4769 -1 -1
kuwa_reference2210834 22.88 1365 23.39 944 49.00 1005 51.27 1393 -1 -1
irlt_reference3210166 22.43 1855 22.93 2348 48.58 2393 50.84 2615 -1 -1
irhv_reference4210168 22.43 1625 22.93 2362 48.58 2568 50.84 2585 -1 -1
```

Annex 1.5 Example of a Sample Results File

```
      >>>>>>>>>>>>>>>>>>>>SAMPLE<<<<<<<<<<<<<<<<<<<

SAMPLE Serial Number:    iraq_sample_001      SAMPLE Run Number:  210792

The Resolution is   1.00    and Peak    1  is chosen to be normalized.

>>Result of Absolute Difference Test<<

The Coefficient is     0.00    Serial no.:  iraq_sample_001           Run no.:  210792
The Coefficient is     2.71    Serial no.:  basl_reference1           Run no.:  210758
The Coefficient is   20.39    Serial no.:  kuwa_reference2           Run no.:  210834
The Coefficient is   20.65    Serial no.:  irhv_reference4            Run no.:  210168
The Coefficient is   20.70    Serial no.:  irlt_reference3            Run no.:  210166

>>Result of Minkowski Distance Test, k = 2<<

The Coefficient is     0.00    Serial no.:  iraq_sample_001           Run no.:  210792
The Coefficient is     2.55    Serial no.:  basl_reference1           Run no.:  210758
The Coefficient is   14.02    Serial no.:  irhv_reference4            Run no.:  210168
The Coefficient is   14.05    Serial no.:  irlt_reference3            Run no.:  210166
The Coefficient is   14.26    Serial no.:  kuwa_reference2            Run no.:  210834
```

Annex 1.6 Discussion and List of Alternate Minkowski Distance Parameters

In the general case, the absolute value Minkowski Distance, using the exponent k, is equal to the 1/k power of the sum, from n =1 to n, of the absolute value of the difference between the intensity (here, in units of either peak height or peak area) of each peak in the instant sample and the corresponding peak in the reference sample to the power of k. We use absolute values to maintain our conventional notion of distance as being greater than or equal to zero.

The use of Minkowski distance parameters, other than the two provided (k = 1, the absolute value of the difference distance and k = 2, the Euclidean distance), may be of assistance in two rare and extreme cases. The first case is where an instant sample and a set of reference samples are distinct, yet very similar. Essentially, as smaller fractional Minkowski distance parameters are used, the apparent scale expands, and it is easier to identify very small differences between an instant sample and a set of very similar reference samples. Minkowski distance parameters useful in this case are in the range of k = 0.1 to k = 0.5. The second case is where an instant sample and a set of reference samples are clearly distinct and not at all similar. As larger Minkowski distance parameters are used, the apparent scale contracts, and it is easier to identify very large differences among very different reference samples. Minkowski distance parameters useful in this case are in the range of k = 2.5 to k = 5.0.

Annex 1.7 List of Procedures Whose Source Code Must Be Modified To Enable PPSP to Use Different Minkowski Distance Parameters

Three procedures (**Annex 1.2.12**, **Annex 1.2.15**, and **Annex 1.2.20**) and the Main Program (**Annex 1.2.22**) require source code modifications to enable Minkowski distances to be computed using alternate Minkowski distance parameters. The PPSP program source code listing provided in **Annex 1.9** provides for the computation of two distance metrics for each reference sample, the absolute value of the difference distance (Minkowski distance parameter of k = 1) and the Euclidean distance (Minkowski distance parameter of k = 2).

Annex 1.8 Pascal Compiler Requirements

The PPSP source code provided in **Annex 1.8** as PPSP.PAS will directly compile as listed on the Turbo Pascal, Version 4.0, compiler from Borland International, Inc., copyright 1987.

Annex 1.9 PASCAL Pattern Similarity Program (PPSP.PAS) Source Code

```
{$R-}    {Range checking off}
{$B+}    {Boolean complete evaluation on}
{$S+}    {Stack checking on}
{$I+}    {I/O checking on}
{$N-}    {No numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}
PROGRAM PASCAL_Pattern_Similarity_Program(INPUT,OUTPUT);

{This program compares the GC pattern of an unknown sample}
{with the GC patterns of some other known samples.}
{The absolute difference between the peaks and the Euclidean distance}
{are the two methods used to determine the similarity of the patterns.}
{The sample data is stored in a file specified by user and}
{the reference data in another file also specified by user}
{The program reads in the sample data first.}
{Then it reads in the reference data one at a time.}
{The user can choose whichever peak he wants to normalize against.}
{A lower value in the absolute difference or the Euclidean distance}
{will indicate the patterns are more similar.}
{The summary of results are recorded in an output file given by user.}


CONST
  MaxPeak = 50;
  MaxArray = 40;


TYPE
  String15 = STRING[15];
  String14 = STRING[15];
  String13 = STRING[13];
  String6 = STRING[6];
  GcPattern = RECORD
              SerialNo : String15;
              RunNo : String6;
              Time : ARRAY [1..MaxPeak] OF REAL;
              Peak : ARRAY [1..MaxPeak] OF REAL;
              NumPeak : INTEGER;
              Coeff : REAL
            END;
  ArrayOfPattern = ARRAY [1..MaxArray] OF GcPattern;
  MatchType = ARRAY [1..MaxPeak] OF BOOLEAN;
```

```
VAR
  Reference,Sample,NormSample,RefSelect : GcPattern;
  GoodRef1,GoodRef2 : ArrayOfPattern;
  RefFile,SmpFile,Result,LongResult : TEXT;
  RefName,SmpName,Summary,Details : String14;
  Match : MatchType;
  Ready,Overflow,BadData,RefOK,NoDetail,Continue : BOOLEAN;
  Resolution,Coeff1,Coeff2 : REAL;
  Best,NormPeak,Count1,Count2,index : INTEGER;
  Answer : CHAR;


PROCEDURE CheckReal (VAR InData : REAL;
                     VAR IO : INTEGER);

{This procedure checks if the typed-in data is a real number.}
{If it is not, then there will be an I/O error and the program will bomb.}
{This procedure will prevent the program from bombing from bad input data.}
{Instead, it will come back to ask for the data once more}
{until the appropriate type has been entered.}


VAR
  GoodData : BOOLEAN;


BEGIN

  GoodData := FALSE;
  WHILE NOT GoodData DO
    BEGIN
      WRITELN;
      WRITELN('Bad Data.  Enter the number again.');
      {$I-} READLN(Indata); {$I+}
      IO := IOResult;
{! 1. IOResu^lt now returns different values corresponding to DOS error codes.}
      IF (IO<>0) OR (Indata<=0.0) THEN
        GoodData := FALSE
      ELSE GoodData := TRUE
    END

END;


PROCEDURE CheckInteger (VAR Indata, IO : INTEGER);

{This procedure checks if the input data is an integer.}
{If not, it prevents the program from bombing}
{but keeping requesting for more until the input is an integer.}
```

```
VAR
  GoodData : BOOLEAN;


BEGIN

  GoodData := FALSE;
  WHILE NOT GoodData DO
    BEGIN
      WRITELN;
      WRITELN('Bad Data.  Enter the number again.');
      {$I-} READLN(Indata); {$I+}
      IO := IOResult;
{! 2. IOResu^lt now returns different values corresponding to DOS error codes.}
      IF (IO<>0) OR (Indata<=0) THEN
        GoodData := FALSE
      ELSE GoodData := TRUE
    END

END;



PROCEDURE CheckSameName(VAR InName : String14;
                        ExistName : String14;
                        VAR SameName : BOOLEAN);

{This procedure checks if the name for details is the same as}
{the one for summary.  If this is so, SameName is true.}


BEGIN

  SameName := FALSE;
  IF InName = ExistName THEN SameName := TRUE

END;



PROCEDURE CheckNoName (Filename : String14;
                       VAR NoName : BOOLEAN);

{This procedure checks if a name has been entered.}
{If not, NoName will be true.}


VAR
  LengthOfName : INTEGER;
```

```
BEGIN
  LengthOfName := LENGTH(Filename);
  IF LengthOfName <= 0 THEN NoName := TRUE
  ELSE NoName := FALSE
END;



PROCEDURE CheckBlank (Filename : String14;
                      VAR HasBlank : BOOLEAN);

{This procedure checks if there is any blank in the name entered.}
{If there is any, the program will flag this as a mistake.}
{The user will then asked to enter the name again.}


CONST
  Blank = ' ';


VAR
  i, LengthOfName : INTEGER;


BEGIN
  LengthOfName := LENGTH(Filename);
  i := 0;
  HasBlank := FALSE;
  REPEAT
    i := i+1;
    IF Filename[i] = Blank THEN HasBlank := TRUE
  UNTIL HasBlank OR (i>=LengthOfName)

END;



PROCEDURE CheckFile (VAR Infile : TEXT;
                     VAR Filename : String14;
                     ExistName : String14;
                     VAR IO : INTEGER);

{This procedure checks if the input file exists.}
{If it doesn't, the procedure will prevent the program from bombing.}
{Instead, it will ask for the filename again until one exists.}


VAR
  GoodData,SameName,NoName,HasBlank : BOOLEAN;
```

```
BEGIN

  SameName := FALSE;
  NoName := FALSE;
  HasBlank := FALSE;
  GoodData := FALSE;
  WHILE NOT GoodData DO
    BEGIN
      CheckNoName(Filename,NoName);
      IF Not NoName THEN CheckBlank(Filename,HasBlank);
      IF (Not NoName) AND (Not HasBlank) THEN
      CheckSameName(Filename, Existname, SameName);
      IF (IO<>0) OR NoName OR HasBlank OR SameName THEN
        GoodData := FALSE
      ELSE GoodData := TRUE;
      IF Not GoodData THEN
        BEGIN
          WRITELN;
          IF SameName THEN
            WRITELN('Same name as sample file. Enter another name for reference file.')
          ELSE WRITELN('Input file does not exist.  Enter the filename again.');
          READLN(Filename);
          ASSIGN(Infile,Filename);
          {$I-} RESET(Infile); {$I+}
          IO := IOResult
{! 3. IOResu^lt now returns different values corresponding to DOS error codes.}
        END
    END

END;


PROCEDURE CheckFileExist (VAR Filename : String14;
                          VAR OK : BOOLEAN);

{This procedure checks if the name of the output file already exists.}
{If so, then FileExist will be true.}
{The program will inform the user that file already exists.}
{The user may decide to write over the existing file.}
{If not, the program will ask the user to enter another name for the output file.}

VAR
  DummyFile : TEXT;
  IO : INTEGER;
  FileExist : BOOLEAN;

BEGIN
  OK := TRUE;
  FileExist := FALSE;
  ASSIGN(DummyFile,Filename);
  {$I-} RESET(DummyFile); {$I+}
  IO := IOResult;
{! 4. IO^Result now returns different values corresponding to DOS error codes.}
  IF IO = 0 THEN
    BEGIN
 FileExist := TRUE;
WRITELN;
      WRITELN('Output file already exists. Enter Y to write over the file.');
      READLN(Answer);
      IF Answer IN ['y','Y'] THEN OK := TRUE
      ELSE
        BEGIN
          OK := FALSE;
          WRITELN('User does not wish to write over existing file.');
          WRITELN('Enter another name for the output file.');
          READLN(Filename)
        END
    END
  ELSE
    BEGIN
      {$I-} REWRITE(DummyFile); {$I+}
      IO := IOResult;
{! 5. IOResu^lt now returns different values corresponding to DOS error codes.}
      IF IO <> 0 THEN
      BEGIN
        WRITELN;
        WRITELN('Bad Name.  Enter again.');
        READLN(Filename);
        OK := FALSE
      END
    END;
  {$I-} CLOSE(DummyFile); {I+}
  IO := IOResult;
{! 6. IO^Result now returns different values corresponding to DOS error codes.}
  IF IO <> 0 THEN OK := FALSE

END;



PROCEDURE CheckSummaryName (VAR Summary : String14;
                            SmpName,RefName : String14);

{This procedure checks the name of the file entered by user.}
{If no name is entered or there are blanks in the name,}
{the program will flag this as a mistake.}
{The user will be asked to enter the name again.}


VAR
  SameName,NoName,HasBlank,OK : BOOLEAN;
  Answer : CHAR;
```

```
BEGIN



  SameName := TRUE;
  NoName := TRUE;
  HasBlank := TRUE;
  OK := TRUE;
  REPEAT
    WHILE NoName OR HasBlank OR SameNAme DO
      BEGIN
        CheckSameName(Summary,SmpName,SameName);
        IF NOT SameName THEN CheckSameName(Summary,RefName,SameName);
        IF NOT SameName THEN CheckNoName(Summary,NoName);
        IF (NOT NoName) AND (NOT SameName) THEN
CheckBlank(Summary,HasBlank);
        IF NoName OR HasBlank OR SameName THEN
          BEGIN
            WRITELN;
            WRITELN('Bad name.  Enter the filename again.');
            READLN(Summary)
          END
      END;
    CheckFileExist(Summary,OK);
    IF NOT OK THEN NoName := TRUE {Make sure new name is checked again}
  UNTIL OK;

END;



PROCEDURE CheckDetailName (VAR Details : String14;
                          Summary,SmpName,RefName : String14);

{This procedure checks the name given to store the details.}
{If no name is given, or there is blanks in the name, or}
{if the name is the same as the one for the summary.}
{If this is so, the program will flag this as a mistake and}
{the user will be asked to enter the name again.}


VAR
  SameName,NoName,HasBlank,OK : BOOLEAN;


BEGIN

  SameName := TRUE;
  NoName := TRUE;
  HasBlank := TRUE;
  OK := TRUE;
  REPEAT
    WHILE NoName OR HasBlank OR SameName DO
      BEGIN
        CheckSameName(Details,Summary,SameName);


        IF NOT SameName THEN CheckSameName(Details,RefName,SameName);
        IF NOT SameName THEN CheckSameName(Details,SmpName,SameName);
        IF NOT SameName THEN CheckNoName(Details,NoName);
        IF (NOT NoName) AND (NOT SameName) THEN CheckBlank(Details,HasBlank);
        IF NoName OR HasBlank OR SameName THEN
          BEGIN
            WRITELN;
            WRITELN('Bad Name.  Enter the filename again.');
            READLN(Details)
          END
      END;
    CheckFileExist(Details,OK);
    IF NOT OK THEN NoName := TRUE {Allow new name to be checked again.}
  UNTIL OK

END;


PROCEDURE Instruction (VAR Ready : BOOLEAN;
                      VAR Resolution : REAL;
                      VAR Best : INTEGER);

{This procedure gives the instructions on how to set up input files.}
{If the input files are not ready yet, the program will terminate.}
{Otherwise, the user will give the size of the window for the retention time}
{and the number of closest references he would like to list.}


VAR
  Answer : CHAR;
  IO : INTEGER;

BEGIN

  WRITELN;
  WRITELN('Do you need instructions on how to prepare the input files?');
  READLN(Answer);
  WRITELN;
  IF Answer IN ['Y','y'] THEN
    BEGIN
      WRITELN('Before this program can be run, two input files need to be set up:');
      WRITELN('  one for sample data and one for reference data.');
      WRITELN;
      WRITELN('The format for the data in both files are the same:');
      WRITELN;
      WRITELN('**  columns 1-15 -- serial no. (e.g. iraq_sample_001)');
      WRITELN;
      WRITELN('**  columns 16-21 -- run no. (e.g. 210792)');
      WRITELN;
      WRITELN('**  next, enter the retention time and peak area in pairs,');
      WRITELN('    separating each entry by a space.');
      WRITELN('    (e.g. 22.87 312 23.39 327 49.00 2998 51.26 3800 -1 -1)');
      WRITELN;
      WRITELN('**  continue enter the data in one line');
      WRITELN('    until the whole line (256 characters) is filled.');
      WRITELN;
      WRITELN('**  when the last entry has been entered,');
      WRITELN('    follow this by two negative numbers.');
      WRITELN('    (e.g. 22.87 312 23.39 327 49.00 2998 51.26 3800 -1 -1)');
      WRITELN('    this is a flag to indicate the last entry is reached.');
      WRITELN
    END;

    WRITELN('Are the input files ready to be processed?');
    WRITELN('Enter Y for yes and N for no.');
    READLN(Answer);
    WRITELN;
    IF Answer IN ['Y','y'] THEN
      BEGIN
        Ready := TRUE;
        WRITELN('The input files are ready and program will continue.');
        WRITELN
      END
    ELSE
      BEGIN
        Ready := FALSE;
        WRITELN('Input files are not ready to be processed.');
        WRITELN('Refer to the REFERENCE GUIDE on how to set up files.');
        WRITELN
      END;

    IF Ready THEN
      BEGIN
        Resolution := -1.0;
        WRITELN;
        WRITELN('Enter the size of the window for the retention time.');
        WRITELN('(The recommended size is between 0.05 to 0.1)');
        {$I-} READLN(Resolution); {$I+}
        IO := IOResult;
{! 7. IOResult^ now returns different values corresponding to DOS error codes.}
        IF (IO<>0) OR (Resolution<=0.0) THEN CheckReal(Resolution,IO);
        WRITELN;
        WRITELN('The size of the window is ',Resolution:5:2);
        WRITELN;
        Best := -1;
        WRITELN;
        WRITELN('Enter the number of the closest patterns you want to list.');
        WRITELN('(The suggested values are between 10 to 30)');
        {$I-} READLN(Best); {$I+}
        IO := IOResult;
{! 8. IOResult^ now returns different values corresponding to DOS error codes.}
        IF (IO<>0) OR (Best<=0) THEN CheckInteger(Best,IO);
        IF Best > MaxArray THEN
          REPEAT
            WRITELN('The number you entered exceeds the allocations space which is
',MaxArray:3);
            WRITELN('Enter another number which is smaller than ',MaxArray:3);
            {$I-} READLN(Best); {$I+}
            IO := IOResult;
{! 9. IOResult now^ returns different values corresponding to DOS error codes.}
            IF IO <> 0 THEN CheckInteger(Best,IO);
          UNTIL Best <= MaxArray;
        WRITELN;
        WRITELN(Output,Best:3,' closest patterns to the sample pattern will be listed.');
        WRITELN
      END

END;


PROCEDURE GetFiles (VAR SmpFile,RefFile,Result,LongResult : TEXT;
                    VAR SmpName,RefName,Summary,Details : String14;
                    VAR NoDetail : BOOLEAN);

{This procedure reads in the names of the input files and output files.}
{The two input files are sample file and reference file.}
{The two output files are one for summarization of results and one for details.}
{The user may choose not to include the details for saving space.}


VAR
  Answer : CHAR;
  IO : INTEGER;


BEGIN

  WRITELN;
  WRITELN('Enter the name of the sample file.');
  WRITELN('(A copy  of trial sample data is stored in iraqsmp.txt)');
  READLN(SmpName);
  ASSIGN(SmpFile,SmpName);
  {$I-} RESET(SmpFile); {$I+}
  IO := IOResult;
```

```
{! 10. I^OResult now returns different values corresponding to DOS error codes.}
  CheckFile(SmpFile,SmpName,' ',IO);
  WRITELN;
  WRITELN('Enter the name of the reference file.');
  WRITELN('(A copy of trial reference data is stored in iraqref.txt)');
  READLN(RefName);
  ASSIGN(RefFile,RefName);
  {$I-} RESET(RefFile); {$I+}
  IO := IOResult;
{! 11. I^OResult now returns different values corresponding to DOS error codes.}
  CheckFile(RefFile,RefName,SmpName,IO);
  WRITELN;
  WRITELN('Enter the name of the output file for the summary.');
  WRITELN('(A suggested name is c;\iraqres.txt)');
  READLN(Summary);
  CheckSummaryName(Summary,SmpName,RefName);
  ASSIGN(Result,Summary);
  REWRITE(Result);
  WRITELN;
  WRITELN('Do you want to store the details of the result?');
  WRITELN('Enter Y for yes and N for no.');
  READLN(Answer);
  IF Answer IN ['Y','y'] THEN
    BEGIN
      NoDetail := FALSE;
      WRITELN;
      WRITELN('Enter the name of the output file for details.');
      WRITELN('(A suggested name is c:\iraqdet.txt)');
      READLN(Details);
      CheckDetailName(Details,Summary,SmpName,RefName);
      ASSIGN(LongResult,Details);
      REWRITE(LongResult);
      WRITELN
    END
  ELSE
    BEGIN
      NoDetail := TRUE;
      Details := 'c:\junkfile.txt';
      ASSIGN(LongResult,Details);
      REWRITE(LongResult);
      WRITELN
    END

END;


PROCEDURE WriteResult (VAR Result : TEXT;
                       Sample : GcPattern;
                       GoodRef1,GoodRef2 : ArrayOfPattern;
                       Resolution : REAL;
                       NormPeak, Count1,Count2 : INTEGER);
```

```
{This procedure summarizes the result of this pattern comparison}
{in the file as indicated by the user in the procedure GetFiles.}
{Only the sample numbers and the run numbers and the coefficients}
{of the closest references are included.}
{The details of the results are stored in another file which is also}
{indicated by the user in the procedure GetFiles.}


VAR
  i : INTEGER;


BEGIN

  IF Count1 > Best THEN Count1 := Best;
  IF Count2 > Best THEN Count2 := Best;
  WRITELN (Result);
  WRITELN (Result);
  WRITELN (Result,'     >>>>>>>>>>>>>>>>>>>>SAMPLE<<<<<<<<<<<<<<<<<<<<
');
  WRITELN (Result);
  WRITE (Result,'SAMPLE Serial Number :   ',Sample.SerialNo);
  WRITELN (Result,'      SAMPLE Run Number :   ',Sample.RunNo);
  WRITELN (Result);
  WRITE (Result,'The Resolution is  ',Resolution:5:2);
  WRITELN (Result,'    and Peak  ',NormPeak:3,'  is chosen to be normalized.');
  WRITELN (Result);
  WRITELN (Result);
  WRITELN (Result,'>>Result of Absolute Difference Test<<');
  WRITELN (Result);
  FOR i := 1 TO Count1 DO
    BEGIN
      WRITE (Result,'The Coefficient is  ',GoodRef1[i].Coeff:6:2);
      WRITE (Result,'    Serial no.:  ',GoodRef1[i].SerialNo);
      WRITELN (Result,'    Run no.:  ',GoodRef1[i].RunNo)
    END;
  WRITELN (Result);
  WRITELN (Result,'>>Result of Euclidean Distance Test<<');
  WRITELN (Result);
  FOR i := 1 TO Count2 DO
    BEGIN
      WRITE (Result,'The Coefficient is  ',GoodRef2[i].Coeff:6:2);
      WRITE (Result,'    Serial no.:  ',GoodRef2[i].SerialNo);
      WRITELN (Result,'    Run no.:  ',GoodRef2[i].RunNo)
    END;
  WRITELN (Result);
  WRITELN (Result)

END;
```

```
PROCEDURE WriteSample (VAR Result : TEXT;
                       SmpData : GcPattern;
                       Resolution : REAL;
                       NormPeak : INTEGER);

{This procedure writes the data of the sample to be compared,}
{to the output file whose name is given by the user in procedure GetFiles.}
{The data includes serial number, run number, the retention time and}
{the peak area of each peak.}
{The resolution of the retention time and the peak chosen to be}
{normalized are also included.}


VAR
  i : INTEGER;


BEGIN

  WITH SmpData DO
    BEGIN
      WRITELN(Result);
      WRITELN(Result);
      WRITELN(Result,'     >>>>>>>>>>>>>>>>>>>>SAMPLE<<<<<<<<<<<<<<<<<<<<
');
      WRITELN(Result);
      WRITELN(Result,'SAMPLE Serial Number :   ',SerialNo,'      SAMPLE Run Number :
',RunNo);
      FOR i := 1 TO NumPeak DO
        WRITELN(Result,Time[i]:6:2,Peak[i]:12:4);
      WRITELN(Result,'The resolution is ',Resolution:5:2,'    and Peak ',NormPeak:3,' is
chosen to be normalized.');
      WRITELN(Result);
    END

END;



PROCEDURE WriteRef (VAR Result : TEXT;
                     Indata : GcPattern);
{This procedure writes the data of a GC pattern to the output file}
{which stores the result and its name is given by the user.}
{The data includes serial number, run number, the retention time and}
{the peak area of each peak.}
{A coefficient which indicates the similarity of the patterns is also included.}



VAR
  i : INTEGER;
```

```
BEGIN

  WITH Indata DO
    BEGIN
      WRITE(Result,'The coefficient is ',Coeff:6:2);
      WRITELN(Result,'    Serial no.: ',SerialNo,'   Run no.: ',RunNo);
      FOR i := 1 TO NumPeak DO
        WRITELN(Result,Time[i]:6:2,Peak[i]:12:4);
      WRITELN(Result)
    END

END;



PROCEDURE WriteLongResult (VAR LongResult : TEXT;
                          Sample,NormSample : GcPattern;
                          GoodRef1,GoodRef2 : ArrayOfPattern;
                          Resolution : REAL;
                          NormPeak,Count1,Count2 : INTEGER);

{This procedure writes the details of the results to the output file}
{whose name is given by the user.}
{The output data include the serial number, the run number,}
{the retention time, the normalized peak area and the coefficient.}
{The references are listed in the ascending order of the coefficients.}


VAR
  i : INTEGER;


BEGIN

  IF Count1 > Best THEN Count1 := Best;
  IF Count2 > Best THEN Count2 := Best;
  WRITELN (LongResult);
  WRITELN (LongResult);
  WriteSample (LongResult,Sample,Resolution,NormPeak);
  WriteRef (LongResult,NormSample);
  WRITELN (LongResult);
  WRITELN (LongResult,'>>Result of Euclidean Distance Test<<');
  WRITELN (LongResult);
  FOR i := 1 TO Count1 DO
    WriteRef (LongResult,GoodRef1[i]);
  WRITELN (LongResult);
  WRITELN (LongResult,'>>Result of Euclidean Distance Test<<');
  WRITELN (LongResult);
  FOR i := 1 TO Count2 DO
```

```
    WriteRef (LongResult,GoodRef2[i]);
  WRITELN (LongResult);

END;


PROCEDURE GetData (MaxPeak : INTEGER;
                   VAR Infile : TEXT;
                   VAR Indata : GcPattern;
                   VAR Overflow : BOOLEAN;
                   VAR BadData : BOOLEAN);

{This procedure reads the data from the data file one at a time.}
{Each set of data consists of serial no., run no., retention time}
{and peak area for each peak.}
{The procedure will indicate if there is not enough space to read}
{all the data in the variable Overflow.}


VAR
  i, IO : INTEGER;


BEGIN

  WITH Indata DO
    BEGIN
      READ(Infile,SerialNo,RunNo);
      i := 0;
      REPEAT
        i := i+1;
        {$I-} READ(Infile,Time[i],Peak[i]); {$I+}
        IO := IOResult;
{! 12. IOResul^t now returns different values corresponding to DOS error codes.}
        IF IO <> 0 THEN
          BEGIN
            WRITELN;
            WRITELN('Bad Data in the Input File');
            WRITELN('Program will terminate for this set of data.');
            WRITELN;
            BadData := TRUE
          END;
      UNTIL (Time[i]<0.0) OR (i>=MaxPeak) OR BadData;
      IF NOT BadData THEN
        BEGIN
          IF Time[i] < 0.0 THEN
            NumPeak := i - 1
          ELSE
            BEGIN
              WRITELN;
              WRITELN('>>>>>ARRAY OVERFLOW<<<<<');
              WRITELN(SerialNo,' ',RunNo,' has more data than the space allocated.');
              WRITELN('The dimension of the array is given by MaxPeak whose present
value is ',MaxPeak:3);
              WRITELN('The program will terminate for this set of data.');
              WRITELN;
              Overflow := TRUE
            END
        END
    END;
    READLN(Infile)

END;


PROCEDURE NormalizeSample (Sample : GcPattern;
                    VAR NormSample : GcPattern;
                    VAR NormPeak : INTEGER);

{This procedure allows the user to choose one of the peaks}
{in the sample pattern to be normalized to one.}


VAR
  i,IO : INTEGER;


BEGIN

  WITH Sample DO
    BEGIN
      WRITELN;
      WRITELN('>>>>>',SerialNo,' ',RunNo,'<<<<<');
      FOR i := 1 TO NumPeak DO
        WRITELN('Peak Number ',i:3,Time[i]:10:2,Peak[i]:10:0);
      REPEAT
        NormPeak := 0;
        WRITELN('Enter the peak number of the peak you want to normalize.');
        {$I-} READLN(NormPeak); {$I+}
        IO := IOResult;
{! 13. IOResul^t now returns different values corresponding to DOS error codes.}
        IF (NormPeak<=0) OR (NormPeak>NumPeak) THEN IO := 10;
        WHILE IO <> 0 DO
          BEGIN
            CheckInteger(NormPeak,IO);
            IF NormPeak > NumPeak THEN IO := 10
          END;
      UNTIL Peak[NormPeak] > 0.0;
      WRITELN;
      WRITE('Peak ',NormPeak:3,' Retention Time ',Time[NormPeak]:6:2,' Peak Area
',Peak[NormPeak]:6:0);
      WRITELN(' is chosen to be normalized.');
      WRITELN('Program is running....');
      WRITELN;
      NormSample.SerialNo := SerialNo;
      NormSample.RunNo := RunNo;
      FOR i := 1 TO NumPeak DO
        BEGIN
          NormSample.Time[i] := Time[i];
          NormSample.Peak[i] := Peak[i]/Peak[NormPeak]
        END;
      NormSample.NumPeak := NumPeak
    END

END;


PROCEDURE GetRefPeak (Reference,Sample : GcPattern;
                      Resolution : REAL;
                      NormPeak : INTEGER;
                      VAR RefSelect : GcPattern;
                      VAR Match : MatchType;
                      VAR RefOK : BOOLEAN);

{This procedure picks out the corresponding peaks in the reference pattern}
{so that the sample pattern can be compared to it.}
{The procedure to match the peaks is as follows:}
{  1. Find the peaks that are within the resolution.}
{  2. Pick the one with the smallest time differences.}
{  3. Store this matching peak under the variable RefSelect.}
{If a peak cannot be matched, the indicator Match will be FALSE.}


VAR
  TimeDiff1, TimeDiff2 : REAL;
  i,j,k : INTEGER;
  SmallerDiff : BOOLEAN;


BEGIN

{Initialize variables}
  FILLCHAR(RefSelect,SIZEOF(RefSelect),CHR(0));
  FOR i := 1 TO MaxPeak DO
    Match[i] := FALSE;
  j := 0;
  k := 0;

  RefSelect.SerialNo := Reference.SerialNo;
  RefSelect.RunNo := Reference.RunNo;

  WITH Sample DO
    BEGIN
```

```
      FOR i := 1 TO NumPeak DO
        BEGIN
{Set j to the peak that is last picked}
          j := k;
{Pick first peak that falls within the resolution}
          REPEAT
            Match[i] := FALSE;
            j := j+1;
            TimeDiff1 := Reference.Time[j] - Time[i];
            TimeDiff1 := ABS(TimeDiff1);
            IF TimeDiff1 <= Resolution THEN
              Match[i] := TRUE
          UNTIL Match[i] OR (j>=Reference.NumPeak);

{Next, pick peak with smallest time difference within resolution}
          IF Match[i] THEN
            BEGIN
              k := j;
              SmallerDiff := TRUE;
              WHILE (j<Reference.NumPeak) AND SmallerDiff DO
                BEGIN
                  j := j+1;
                  TimeDiff2 := Reference.Time[j] - Time[i];
                  TimeDiff2 := ABS(TimeDiff2);
                  IF TimeDiff2 < TimeDiff1 THEN
                    BEGIN
                      SmallerDiff := TRUE;
                      TimeDiff1 := TimeDiff2;
                      k := j
                    END
                  ELSE
                    SmallerDiff := FALSE
                END;

{Put matching peak in RefSelect}
              RefSelect.Time[i] := Reference.Time[k];
              RefSelect.Peak[i] := Reference.Peak[k];
              RefSelect.NumPeak := RefSelect.NumPeak + 1;
              Reference.Time[k] := 0.0
            END
{When no match, set peak area to zero for that retention time}
          ELSE
            BEGIN
              RefSelect.Time[i] := Time[i];
              RefSelect.Peak[i] := 0.0;
              RefSelect.NumPeak := RefSelect.NumPeak + 1
            END
        END

    END;
```

```
{If peak to be normalized is zero, reference is automatically  discarded}
{as indicated by the variable RefOK}
  IF Match[NormPeak] = TRUE THEN
    RefOK := TRUE
  ELSE RefOK := FALSE

END;


PROCEDURE NormalizeReference (NormPeak : INTEGER;
                              Match : MatchType;
                              VAR RefSelect : GcPattern;
                              VAR RefOK : BOOLEAN);

{This procedure will normalize the peaks in the reference}
{according the peak chosen by the user for the sample.}
{If there is no corresponding peak, reference is considered not similar}
{and its coefficient is not computed at all.}


VAR
  i : INTEGER;
  NormValue : REAL;


BEGIN
WITH RefSelect DO
    BEGIN

      NormValue := Peak[NormPeak];
      IF Match[NormPeak] AND (Peak[NormPeak]>0.0) THEN
        BEGIN
          RefOk := TRUE;
          FOR i := 1 TO NumPeak DO
            Peak[i] := Peak[i]/NormValue
        END
      ELSE
        RefOk := FALSE

    END

END;


PROCEDURE GetCoefficient (NormSample,RefSelect : GcPattern;
                          VAR Coeff1,Coeff2 : REAL);

{This procedure will compute two coefficients for each reference}
{based on two methods.}
{The first one is to take the absolute difference of the peak area}
{between the sample and the reference.}
```

```
{The second one is the Euclidean distance, that is}
{the square root of the sum of the squares of the difference.}
{In this way we can compare the two methods.}


VAR
  i : INTEGER;


BEGIN

  Coeff1 := 0.0;
  Coeff2 := 0.0;
  FOR i := 1 TO NormSample.NumPeak DO
    BEGIN
      Coeff1 := Coeff1 + ABS(NormSample.Peak[i] - RefSelect.Peak[i]);
      Coeff2 := Coeff2 + SQR(NormSample.Peak[i] - RefSelect.Peak[i])
    END;
  Coeff2 := SQRT(Coeff2)

END;


PROCEDURE OrderArray (RefSelect : GcPattern;
                      Coefficient : REAL;
                      VAR GoodRef : ArrayOfPattern;
                      VAR Count : INTEGER);

{This procedure will enter the reference into the list of the best references}
{in the ascending order of coefficients.}


VAR
  i,j,k : INTEGER;


BEGIN

  i := 1;
{Compare new reference to the list of best references}
{Locate the appropriate position for this new reference in the list}
  WHILE (i<Count) AND (Coefficient>GoodRef[i].Coeff) DO
    i := i+1;
  IF Coefficient <= GoodRef[i].Coeff THEN
    BEGIN
{Push all references with larger coefficients down the list}
      Count := Count + 1;
      IF Count > Best THEN Count := Best;
      FOR j := Count DOWNTO (i+1) DO
        BEGIN
          WITH GoodRef[j] DO
          BEGIN
            SerialNo := GoodRef[j-1].SerialNo;
            RunNo := GoodRef[j-1].RunNo;
            FOR k := 1 TO GoodRef[j-1].NumPeak DO
              BEGIN
                Time[k] := GoodRef[j-1].Time[k];
                Peak[k] := GoodRef[j-1].Peak[k]
              END;
            NumPeak := GoodRef[j-1].NumPeak;
            Coeff := GoodRef[j-1].Coeff
          END
      END;
{Insert new reference in the appropriate position of the list}
      WITH GoodRef[i] DO
        BEGIN
          SerialNo := RefSelect.SerialNo;
          RunNo := RefSelect.RunNo;
          FOR k := 1 TO RefSelect.NumPeak DO
            BEGIN
              Time[k] := RefSelect.Time[k];
              Peak[k] := RefSelect.Peak[k]
            END;
          NumPeak := RefSelect.NumPeak;
          Coeff := Coefficient
        END
    END

{Add new reference to end of list}
  ELSE
    BEGIN
      Count := Count + 1;
      IF Count <= Best THEN
        BEGIN
          WITH GoodRef[Count] DO
            BEGIN
              SerialNo := RefSelect.SerialNo;
              RunNo := RefSelect.RunNo;
              FOR k := 1 TO RefSelect.NumPeak DO
                BEGIN
                  Time[k] := RefSelect.Time[k];
                  Peak[k] := RefSelect.Peak[k]
                END;
              NumPeak := RefSelect.NumPeak;
              Coeff := Coefficient
            END
        END
      ELSE
        Count := Best
    END

END;
```

```
{>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<}
{>>>>>>>>>>>>>>>>          MAIN  PROGRAM          <<<<<<<<<<<<<<<<}
{>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<}


BEGIN

{Initialize all variables}
  FILLCHAR(Reference,SIZEOF(Reference),CHR(0));
  FILLCHAR(Sample,SIZEOF(Sample),CHR(0));
  FILLCHAR(NormSample,SIZEOF(NormSample),CHR(0));
  FILLCHAR(RefSelect,SIZEOF(RefSelect),CHR(0));
  FILLCHAR(GoodRef1,SIZEOF(GoodRef1),CHR(0));
  FILLCHAR(GoodRef2,SIZEOF(GoodRef2),CHR(0));
  FILLCHAR(Match,SIZEOF(Match),CHR(0));
  Ready := TRUE;
  Overflow := FALSE;
  BadData := FALSE;
  RefOK := TRUE;
  Continue := TRUE;
  Resolution := 0.1;
  NormPeak := 1;
  Count1 := 0;
  Count2 := 0;
  Coeff1 := 0.0;
  Coeff2 := 0.0;

{End of Initialization}

  WHILE Continue DO
    BEGIN
      Instruction(Ready,Resolution,Best);
      IF Ready THEN
        BEGIN
{Open the input files containing sample data and reference data}
{and the output files to store the results of this program}

GetFiles(SmpFile,RefFile,Result,LongResult,SmpName,RefName,Summary,Details,No
Detail);

{Compare each set of sample to the list of references}
          REPEAT
            FOR index := 1 TO Best DO
              BEGIN
                FILLCHAR(GoodRef1,SIZEOF(GoodRef1),CHR(0));
                FILLCHAR(GoodRef2,SIZEOF(GoodRef2),CHR(0))
              END;
          Overflow := FALSE;
          BadData := FALSE;
{Reads in sample data one set at a time}
          GetData(MaxPeak,SmpFile,Sample,Overflow,BadData);
          IF (NOT Overflow) AND (NOT BadData) THEN
            BEGIN
{Normalize the peaks of the sample data}
              NormalizeSample(Sample,NormSample,NormPeak);
{Count1 keeps track of best references for absolute difference method}
{Count2 keeps track of best references for euclidean distance method}
              Count1 := 0;
              Count2 := 0;

{Compare each reference to sample, compute the 2 coefficients according to}
{the two methods used: absolute difference and euclidean distance}
{Then place the reference on the best references list accordingly.}
              WHILE NOT EOF(RefFile) AND (NOT Overflow) AND (NOT BadData) DO
                BEGIN

                  Overflow := FALSE;
                  BadData := FALSE;
                  GetData(MaxPeak,RefFile,Reference,Overflow,BadData);
                  IF (NOT Overflow) AND (NOT BadData) THEN
                    BEGIN
{Get the matching peaks in the reference}

GetRefPeak(Reference,NormSample,Resolution,NormPeak,RefSelect,Match,RefOK);
{If the peak to be normalized to one has zero area, then reference is}
{automatically discarded. RefOK is then set to false.}

                      IF RefOK THEN
                        BEGIN
{Normalize the peaks of the reference}
                          NormalizeReference(NormPeak,Match,RefSelect,RefOK);
                          IF RefOK THEN
                            BEGIN
{Compute the 2 coefficients according to the two methods used:}
{  absolute difference and euclidean distance}
                              GetCoefficient(NormSample,RefSelect,Coeff1,Coeff2);
{Place the best references in the two lists: GoodRef1, GoodRef2}
{The best references will be listed in the ascending order of the coefficients.}
                              OrderArray(RefSelect,Coeff1,GoodRef1,Count1);
                              OrderArray(RefSelect,Coeff2,GoodRef2,Count2)
                            END
                        END
                    END
                END;

              CLOSE(RefFile);
{Write the summary of the results to the output file on the diskette}
```

```
                WriteResult
(Result,Sample,GoodRef1,GoodRef2,Resolution,NormPeak,Count1,Count2);
                IF NOT NoDetail THEN
{Write the details of the results to another output file on the diskette}
                  WriteLongResult
(LongResult,Sample,NormSample,GoodRef1,GoodRef2,Resolution,NormPeak,Count1,
Count2);
{Reset the reference file to compare another set of sample data}
                ASSIGN(RefFile,RefName);
                RESET(RefFile)
              END
          UNTIL EOF(SmpFile) OR Overflow OR BadData;

          CLOSE(SmpFile);
          CLOSE(Result);
          CLOSE(LongResult)
        END;
      IF (NOT Overflow) AND (NOT BadData) THEN
        BEGIN
          WRITELN;
          WRITELN('The summary is stored in the file ',Summary);
          IF NOT NoDetail THEN
          WRITELN('The details is stored in the file ',Details);
          WRITELN
        END;
      WRITELN;
      WRITELN('Do you want to continue with another set of data?');
      WRITELN('Enter Y for yes and N for no.');
      READLN(Answer);
      IF Answer IN ['Y','y'] THEN Continue := TRUE
      ELSE
        BEGIN
          Continue := FALSE;
          WRITELN;
          WRITELN('User does not wish to continue.');
          WRITELN('Program will exit and return to Turbo Pascal.');
          WRITELN
        END

    END;


END.

{          END  OF  PROGRAM              }
```

[END OF METHOD