

SAMPLE PAGE FROM  
TOD DATA DESCRIPTOR DICTIONARY

ELEMENT	NO.	LONG NAME	SHRT NAME	SUFFIX	UNITS	AF	DATA	TYPE	LIMIT	CHECKING	INITIALIZATION
P311		Guaiac (stool occult blood)	GUAIAC	_ON	+range			+RNG			
H 24	G I Sx	Character	GUTCHAR	_IM	none			CHAR			
H 27	G I Sx	Rate	GUTDATE	_IM	date			DATE			
P 84		Gynecologic symptom review	GYN	_ON	+range			+RNG			0 SAME
P 84		Hallucinations	HALLUCE	_IM	+			+RNG			0 SAME
P413		Hand X-Rays	HANDX	_IM	+			+RNG			0 SAME
P 17		Frontal Head Pain	HD_FRONT	_IM	+			+RNG			0 SAME
P 22		Frontal headache	HD_FRONT	_ON	+range			+RNG			0 SAME
P 16		Occipital Head Pain	HD_OCCIP	_IM	+			+RNG			0 SAME
P 21		Occipital headache	HD_OCCIP	_ON	+range			+RNG			0 SAME
P 15		Temporal Head Pain	HD_TEMP	_IM	+			+RNG			0 SAME
P 20		Temporal headache	HD_TEMP	_ON	+range			+RNG			0 SAME
P 19		Headache symptom review	HEADACHE	_ON	+range			+RNG			0 SAME
H 30		Neuropsych Sx Character	HEADCHAR	_IM	none			CHAR			
H 23		Neuropsychiatric Sx date	HEADDATE	_IM	date			DATE			
P 51		Heart symptom review	HEART	_ON	+range			+RNG			0 SAME
P386		Heel Pain	HEELPAIN	_IM	+			+RNG			0 SAME
P 94		Height	HEIGHT	_IM	Cm	L		HT VALU	10	200	
P115		height	HEIGHT	_ON	Cm	L		VALU	20	250	
P132		Heliotrope eyelids	HELI EYE	_IM	+			+RNG			0 SAME
P 55		Hematemesis	HEMATEM	_IM	+			+RNG			0 SAME
P 67		Hematemesis	HEMATEM	_ON	+range			+RNG			0 SAME
P 78		Hematuria, gross	HEMATUR	_ON	+range			+RNG			0 SAME
P227		Hemoglobin	HEMOGLOB	_IM	gm %	L		HT VALU	2	20	
P391		Hemoglobin	HEMOGLOB	_ON	gms%	L		T VALU	2	20	
P 3d		Hemoptysis	HEMOPTY	_IM	+			+RNG			0 SAME
P 45		Hemoptysis	HEMOPTY	_ON	+range			+RNG			0 SAME
P137		Hepatomegaly	HEPATO	_IM	+			+RNG			0 SAME
P156		Hepatomegaly	HEPATO	_ON	Cm	L		T VALU	5	50	
P187		Hip Joints	HIPS	_IM	+			+RNG			10 SAME
P402		HISTO	HISTO	_IM	none	L		VALU	0	5	
P478		Hospitalization	HOSPITAL	_IM	none	L		+RNG			0 SAME
H 10		Height	HO4_TALL	_IM	Cm			VALU	20	400	
P 51		Heartburn	HT_BURN	_IM	+			+RNG			0 SAME
P 63		Heartburn	HT_BURN	_ON	+range			+RNG			0 SAME
P126		Enlarged heart	HT_LARGE	_IM	+			+RNG			0 SAME
P142		Enlarged heart	HT_LARGE	_ON	+range			+RNG			0 SAME
P141		Heart physical exam	HT_PHY	_ON	+range			+RNG			0 SAME
P364		Low Back Sx	HXLOBACK	_IM	+			+RNG			0 SAME
P383		Neck Sx	HXNECK	_IM	+			+RNG			0 SAME
P378		Nodules, Hx	HXNOECULE	_IM	+			+RNG			0 SAME
P385		Temporomandibular Sx	HXTMARTH	_IM	+			+RNG			0 SAME
P379		Topli, Hx	HXTOPHI	_IM	+			+RNG			0 SAME
P190		Infra Clav. left lymphadenopathy	ICL_LYM	_ON	Cm	L		VALU	0	20	
P189		Infra Clav. right lymphadenopathy	ICR_LYM	_ON	Cm	L		VALU	0	20	
P 29		Icterus	ICTERUS	_ON	+range			+RNG			0 SAME
P255		IGA	IGA	_IM	mgm %	L		HT VALU	0	1000	
P256		IGG	IGG	_IM	mgm %	L		HT VALU	0	3000	
P257		IGM	IGM	_IM	mgm %	L		HT VALU	0	1000	
P106		Iliac left lymphadenopathy	ILL_LYM	_ON	Cm	L		VALU	0	20	
P195		Iliac right lymphadenopathy	ILL_LYM	_ON	Cm	L		VALU	0	20	
P312		Azathioprine	IMURAN	_IM	mg/day	L		HT VALU	0	300	
P265		Immune Electrophoresis	IM_ELECT	_IM	none	L		VALU	0	10	
P 16		Index number	INDEXNUM	_IM	none	L		VALU	0	2000	

The TOD Databank Description Language

A. The TOD Schema and DDL:

A TOD databank is constructed and accessed according to a carefully defined databank description called a schema. The schema is specified by a set of "high level" language statements which are stored on a textfile, much like a PL/ACME program. These statements are translated into an internal form by the schema translation program, TD\_TRA (ACME Note TDPT).

The high level schema language will henceforth be referred to as the databank description language, DDL. The syntax of DDL resembles PL/ACME syntax, except that it only provides for declarations. DDL is designed to require accuracy and completeness of definition, since these are essential for effective databank operation.

The schema serves two important functions: First, its DDL representation gives an explicit documentation of the content, unit of measurement, reference name, and type of each data item; and it indicates the data initialization, range checking, encoding, and retrieval file maintenance which must be performed. Second, the internal form of the schema provides generalized TOD programs with information about the structure of a specific user's databank (see example in ACME Note TODD, Part III, Section A.2.d.i.).

The necessary information for writing a DDL schema can be accumulated and proofed on a convenient printed form, copies of which may be obtained at the ACME office.

B. Basic Semantics of DDL:

A "single piece" of information is stored in a data item or just item. The information stored in an item is referred to as the value of that item.

The items of a databank can be partitioned into two general categories. Items in the first category are recorded only once and they store demographic information or background information associated with a patient. Items in the second category store the numeric values of several time-dependent parameters recorded for each patient-visit. A "visit" corresponds to a physician interview, or any other point in time at which information about a patient is logged.

DDL represents the two categories of data items as formal arrays. Each formal array element corresponds to a single data item, so that an array of elements is a list of related items. The category of demographic items is represented by the HEADER formal array. The category of time-dependent items is represented by the PARAMETER formal array.

The size of each formal array, and thus the size of each category of items, is established with a declaration statement. Then each formal array element may be assigned a 5-tuple of attributes. These attributes describe the data item associated with a formal array element. The attributes define content, a measurement unit, name, data type, and retrieval type for an item. Unassigned formal array elements are place-holders expediting the introduction of new data items into the databank.

The INITIAL statement allows a user to define the initial (i.e. default) value of each item prior to data entry. Careful assignment of default values can result in major savings of secretarial and processing time. Also, by reducing the amount of data which must be entered per patient-visit, the data entry error rate will be lowered.

C. Formal Syntax for DDL:

DDL sentences

/\* comments \*/

DECLAR	<table border="0"> <tr> <td rowspan="2"> <table border="0"> <tr> <td>HEADER</td> <td rowspan="2"> <table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_size);</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table></td></tr> </table></td></tr> </table>	<table border="0"> <tr> <td>HEADER</td> <td rowspan="2"> <table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_size);</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table></td></tr> </table>	HEADER	<table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_size);</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table>	PARAMETER	(array_size);	H	P
<table border="0"> <tr> <td>HEADER</td> <td rowspan="2"> <table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_size);</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table></td></tr> </table>			HEADER		<table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_size);</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table>		PARAMETER	(array_size);
	HEADER	<table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_size);</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table>	PARAMETER	(array_size);		H	P	
PARAMETER	(array_size);							
H		P						

|  |  |

<table border="0"> <tr> <td>HEADER</td> <td rowspan="2"> <table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_elt_no)=(content,unit,name,</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table></td></tr> </table>	HEADER	<table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_elt_no)=(content,unit,name,</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table>	PARAMETER	(array_elt_no)=(content,unit,name,	H	P
	HEADER		<table border="0"> <tr> <td>PARAMETER</td> <td rowspan="2">(array_elt_no)=(content,unit,name,</td> </tr> <tr> <td>H</td> <td>P</td> </tr> </table>		PARAMETER	(array_elt_no)=(content,unit,name,
PARAMETER	(array_elt_no)=(content,unit,name,					
H		P				

 |                                     |   |   |   |  |   |   |  |                                     |  | |-------------------------------------|---|---|---|--|---|---|--|-------------------------------------|--| | VALUE                               | <table border="0"> <tr> <td>+RANGE</td> <td rowspan="2"> <table border="0"> <tr> <td>[LIMIT(min,max)]</td> <td rowspan="2"> <table border="0"> <tr> <td>[CHECK][FIX],</td> <td rowspan="2"> <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table></td></tr> </table></td></tr> </table></td></tr> </table> | +RANGE  | <table border="0"> <tr> <td>[LIMIT(min,max)]</td> <td rowspan="2"> <table border="0"> <tr> <td>[CHECK][FIX],</td> <td rowspan="2"> <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table></td></tr> </table></td></tr> </table> | [LIMIT(min,max)]   | <table border="0"> <tr> <td>[CHECK][FIX],</td> <td rowspan="2"> <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table></td></tr> </table> | [CHECK][FIX],   | <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table> | [INDEX][RANGE][TRANSPosed][PRIVATE] | NONE   | | +RANGE                              |   | <table border="0"> <tr> <td>[LIMIT(min,max)]</td> <td rowspan="2"> <table border="0"> <tr> <td>[CHECK][FIX],</td> <td rowspan="2"> <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table></td></tr> </table></td></tr> </table> |   | [LIMIT(min,max)]   |   | <table border="0"> <tr> <td>[CHECK][FIX],</td> <td rowspan="2"> <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table></td></tr> </table> |  | [CHECK][FIX],                       | <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table> | | [LIMIT(min,max)]                    | <table border="0"> <tr> <td>[CHECK][FIX],</td> <td rowspan="2"> <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table></td></tr> </table>   |   | [CHECK][FIX],   | <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table> | [INDEX][RANGE][TRANSPosed][PRIVATE]   |   | NONE   |                                     |  | | [CHECK][FIX],                       |   | <table border="0"> <tr> <td>[INDEX][RANGE][TRANSPosed][PRIVATE]</td> </tr> <tr> <td>NONE</td> </tr> </table>  | [INDEX][RANGE][TRANSPosed][PRIVATE]   |  | NONE  |   |  |                                     |  | | [INDEX][RANGE][TRANSPosed][PRIVATE] |   |   |   |  |   |   |  |                                     |  | | NONE                                |   |   |   |  |   |   |  |                                     |  | || DISCRETE. |  |
|  |  |

INITIAL name [VALUE(initial\_value)][[SAME]];

For explanation of formal syntactic conventions, see ACME Note WAA.

Explanation of Terms:

Capitalized words are keywords recognized independently of capitalization. Lower case words are explained below.

"comments" is a string of any characters.

"array\_size", "array\_elt\_no"(array element number), "string\_length", "min", & "max" are numbers.

"content" and "unit" are literal strings enclosed in single quotes.

"name" is a PL/ACME variable name.

"initial\_value" may be literal string (enclosed in single quotes) or a number.

Textfile Format:

Multiple sentences may occur on a single textfile line. Extra non-embedded blanks are ignored. (Note: Comments are not permitted inside of sentences.)

Note on Semantics:

Some syntactically correct sentences are not semantically acceptable, and will be rejected by the translator.

See ACME Note TDPT for these exceptions.

D. Detailed Semantics for DDL Statements:

Comments, enclosed in the PL-conventional "/\*" and "\*/" are ignored by the schema translator and serve only as documentation.

A declaration statement specifies the number of elements in each formal array. Before data items can be assigned attributes, their associated formal arrays must be declared of adequate size. "H" and "P" are recognized abbreviations for "HEADER" and "PARAMETER".

The formal assignment statement defines for formal array elements the attributes of their associated data items. There are five attributes which must all be specified. We will consider each separately.

The first attribute of a data item is a character string of length  $\leq 40$  describing the content of that data item. For example, if a HEADER item stores the address of the referring physician for each patient, its content might be described as 'ADDRESS OF REFERRING PHYSICIAN'. The content attribute is included in the schema for documentation purposes. It may eventually be used for computer composition of Time Oriented Record forms.

The second attribute of a data item is a character string of length  $\leq 10$  describing a standard unit of measurement. For example, if a PARAMETER item stores the white blood cell count for each patient-visit, its units might be described as 'x1000'. A unit must always be specified and a null response will not be accepted. For the HEADER item suggested in the first example, its unit would have to be specified as 'address' or 'none'. The reason for always requiring units is to assure that databank planners fully specify the meaning of each data value, so that data can be shared.

The third attribute of a data item is a (short) name by which the data item may be uniquely identified. This name must be a valid PL/ACME variable name. Names will always be recognized irrespective of capitalization. In the second example above, "WBC" might name "white blood cell count".

Names are an important symbolic attribute of data items, and the use of standard names will greatly simplify communication of information between two data banks and merging of data banks. To assist in the standardization of data item names, a sorted list of names and the attributes of their associated data items will be maintained by the TOD manager.

Public databank procedures will make use of certain standard names for automatic update of data values. For example, the data item named "age" will automatically have the current age of a patient stored in it (as computed from the "date" for this visit and the "birthdat" item for this patient). These standard items are defined in Section E.

In the INITIAL statement, HEADER or PARAMETER elements are identified by their names.

Names will be used in tabular output programs for column headings.

The fourth attribute of a data item specifies the data type of information stored in this item. The following types are available:

- 1) VALUE data items can meaningfully assume continuous numeric values with six significant figures. They are stored as ACME single precision floating point numbers. VALUE data items are assumed by analysis programs to be measured on an interval scale.
- 2) +RANGE data items can assume the values 0, 1, 2, 3, 4 (of the "+RANGE scale"), which is treated by analysis programs as an ordinal scale.
- 3) DISCRETE data items can only meaningfully assume discrete numeric values. Analysis programs consider them to be measured on a nominal scale.
- 4) CHAR(string\_length) specifies that a data item has as its value a string of variable length  $\leq$  "string\_length".
- 5) DATE data items store dates in an internal form. Dates are entered in a standard format, DDMONYY. DD=two digits specifying the day, MON=first three letters of the month (irrespective of capitalization), and YY=last two digits of the year. There are no spaces. The standard form is automatically encoded into an internal "arithmetic date" for storage and numerical manipulation. Stored dates are converted back to external form by TOD retrieval modules.
- 6) CODE data items are stored as numeric values related by encoding-decoding procedures to a more legible representation. For example, the item "sex" might be coded as 0, 1 for F, M.
- 7) CONFIDENTIAL data items are encoded and decoded by private procedures providing keyword-protected scrambling. Only HEADER information can be confidential.
- 8) POINTER data items store pointers to information contained in an auxiliary data file, defined for a specific TOD databank. What gets pointed to by a POINTER items is determined when an individual TOD databank is defined. As an example, a HEADER element of type POINTER might point to the first of a group of textfile lines comprising a reference letter for each patient.

There are three data type qualifiers. At data entry time an item for which LIMIT(min,max) is specified will have its value checked. If the entered value falls outside of the specified limits, an error message will be given.

At data entry time an item for which CHECK is specified will have its value checked for validity by the user-provided encoding-decoding data check procedures (see ACME Note E2001). Items for which FIX is specified will have their values checked by a big data checking program run incrementally against the databank asynchronously with data entry.

The fifth attribute of a data item specifies the retrieval type for information stored in this item. Retrieval files will be maintained by a public UPDATE program facilitating the retrieval of data items in accordance with their retrieval types. Like the other attributes, retrieval type cannot be omitted from the formal assignment statement. Any subset of the types INDEX, RANGE, TRANSPOSED, or PRIVATE may be given for retrieval type, or else the user must specify the type NONE. PRIVATE indicates that a user maintains private files to expedite retrieval for this parameter.

Leaving several formal array elements unassigned with attributes greatly decreases the cost of adding new data items at databank re-compilation time, and is highly recommended. Assignment of previously unassigned formal array elements is a relatively inexpensive way of expanding a data bank, whereas redeclaration of formal arrays leads to costly reformatting of the entire databank.

The INITIAL statement causes the data items corresponding to a named formal array element to be set to an initial value given as "initial\_value" prior to data entry, checking and encoding. If "SAME" is specified for a HEADER element, it has no effect. If "SAME" is specified for a PARAMETER element, then it has the following effect: On the first visit for each patient, this parameter element is set to UNDEFINED, or to its initial value if VALUE (initial\_value) was indicated. Then for each successive visit this item's value is initialized to its value on the previous visit for this patient. If no initialization is specified, numeric values are automatically set to "UNDEFINED" = -0.0 and string values are set to null as a default.

Formal array elements must be assigned attributes before they can be assigned initial values.

Declarations of formal arrays must precede use of formal array elements, and an assignment of attributes to a formal array element must precede its initialization. If several assignments or initializations are specified for one formal array element, only the last will be effective.

E. Required Items:

The following three HEADER and two PARAMETER element assignments must be included in every databank definition if a user wishes to take advantage of public data entry and retrieval programs. The assignments are written in DDL with explanatory comments. Data type qualifiers and retrieval types other than NONE are acceptable throughout.

```
HEADER(1) = ('Name: last,first,middle initial','none',NAME,CHAR(30),NONE);
HEADER(2) = ('Stanford medical record number','none',MEDREC,VALUE,NONE);
          /* Stored as a six digit number. */
HEADER(3) = ('Birthdate','none',BIRTHDAT,DATE,NONE);
PARAMETER(1) = ('Date of visit','none',DATE,DATE,TRANPOSED);
PARAMETER(2) = ('Age of patient to date','years',AGE,VALUE,NONE);
          /* The value of age will be automatically computed from
            BIRTHDAT as a decimal number of years when the value
            of DATE is entered. */
```

F. Restrictions on Databank Schemas:

See ACME Note TDPT, Section 6.

G. Example of a DDL Schema, subset of the Oncology Databank (ACME Note TDUONE):

text-td\_schem

```

1.000/* ***** */
2.000 DECLARE HEADER(15);
3.000 DECLARE PARAMETER(50);
4.000 HEADER(1) = ('Name: last, first, middle initial', 'none', NAME, CF, VR(40), INDEX.);
5.000 HEADER(2) = ('Stanford medical record number', 'none', MEDREC, VALUE, INDEX.);
6.000 HEADER(3) = ('Birthdate', 'none', BIRTHDAT, DATE, NONE.);
7.000 HEADER(4) = ('Sex: M male, F female', 'sex', SEX, CODE, INDEX.);
8.000 HEADER(5) = ('Patient address', 'none', ADDRESS, CHAR(100), NONE.);
9.000 HEADER(6) = ('Zip code', 'none', ZIP_CODE, CHAR(5), NONE.);
10.000 HEADER(7) = ('Phone number', 'none', PHONE, CHAR(10), NONE.);
11.000 HEADER(8) = ('Diagnostic code 1', 'none', DCODE1, CHAR(11), CHECK, INDEX.);
12.000 HEADER(9) = ('Diagnostic code 2', 'none', DCODE2, CHAR(11), CHECK, INDEX.);
13.000 HEADER(10) = ('Diagnostic code 3', 'none', DCODE3, CHAR(11), CHECK, INDEX.);
14.000 HEADER(11) = ('Diagnostic code 4', 'none', DCODE4, CHAR(11), CHECK, INDEX.);
15.000 HEADER(12) = ('Diagnostic code 5', 'none', DCODE5, CHAR(11), CHECK, INDEX.);
16.000 HEADER(15) = ('Race: 1w, 2b, 3o', 'none', RACE, DISCRETE LIMIT(1, 3), INDEX.);
17.000 PARAMETER(1) = ('Date of visit', 'date', DATE, TRANSPOSED.);
18.000 PARAMETER(2) = ('Age of patient to date', 'years', AGE, VALUE, TRANSPOSED.);
19.000 PARAMETER(3) = ('Fatigue', '+range', FATIGUE, +RANGE, NONE.);
20.000 INITIAL FATIGUE SAME;
21.000 PARAMETER(4) = ('Fever', '+range', FEVER, +RANGE, NONE.);
22.000 INITIAL FEVER SAME;
23.000 PARAMETER(5) = ('Chills', '+range', CHILLS, +RANGE, NONE.);
24.000 INITIAL CHILLS SAME;
25.000 PARAMETER(6) = ('Nightsweats', '+range', SWEATS, +RANGE, NONE.);
26.000 INITIAL SWEATS SAME;
27.000 PARAMETER(7) = ('Weight loss', '+range', WT_LOSS, +RANGE, NONE.);
28.000 INITIAL WT_LOSS SAME;
29.000 PARAMETER(8) = ('Pain', '+range', PAIN, +RANGE, NONE.);
30.000 INITIAL PAIN SAME;
31.000 PARAMETER(9) = ('Pruritis', '+range', PRURIT, +RANGE, NONE.);
32.000 INITIAL PRURIT SAME;
33.000 PARAMETER(10) = ('Karnofsky status%', 'none', KARN(F, VALUE LIMIT(0, 100), TRANSPOSED.);
34.000 INITIAL KARNOF SAME;
35.000 PARAMETER(15) = ('Height', 'cm', HEIGHT, VALUE LIMIT(20, 250), NONE.);
36.000 PARAMETER(16) = ('Weight', 'kilograms', WEIGHT, VALUE LIMIT(5, 150), NONE.);
37.000 PARAMETER(17) = ('BP-systolic', 'mm hg', BP_SYST, VALUE LIMIT(60, 300), NONE.);
38.000 PARAMETER(18) = ('BP-diastolic', 'mm hg', BP_DIAS, VALUE LIMIT(0, 160), NONE.);
39.000 PARAMETER(19) = ('Temperature', 'deg C', TEMPERAT, VALUE LIMIT(35, 45), NONE.);
40.000 PARAMETER(20) = ('Respiration', 'per min', RESPIR, VALUE LIMIT(5, 60), NONE.);
41.000 PARAMETER(40) = ('Absolute lymphocytes (computed)', 'x1000/mmcu', LYMPH_AB, VALUE, TRANSPOSED.);
42.000/*--Absolute-lymphocyte-count-is-computed-at-data-entry-time-as-wbc-x-lymphs-in-percent-of-wbc--*/
43.000 PARAMETER(41) = ('Absolute neutrophils (computed)', 'x1000/mmcu', NEUT_AB, VALUE, TRANSPOSED.);
44.000/*--Absolute-neutrophils-are-computed-at-data-entry-time-as-wbc-x-neut-in-percent-of-wbc--*/

```

Core Research & Development (Continued)

Analyzing TOD Operational Costs

Early during the TOD project we found that physicians did not have the tools available to get an adequate picture of the operational costs of a databank. They knew their total costs and what they did, but the task of allocating costs to individual transactions had become complex. An attempt was made during the project to set up a cost analysis framework to aid the researcher in determining where his computer dollar went. ACME Note TODCST outlines a study underway; ACME Note TODPDA outlines the method of trapping basic cost and transaction data for the TOD system. Note TODCST follows.

ACME Note

Analyzing the Costs of Running a TOD Databank

TODCST-1  
Frank Germano  
March 22, 1973

This note outlines a study to analyze the costs of running a TOD databank. Components of costs and procedures to compare them are identified, both for comparisons of cost components within individual databanks and among several databanks.

The outline described in this note will be used to compare TOD costs in Immunology and Oncology, two operational TOD databanks. In the future, as the number of TOD databanks grows, the benefits of cross-databank cost comparison, as outlined here, will increase. In any event, cost comparisons among components of a single databank are always useful.

Raw Data Sources:

The raw data for analyzing costs of a TOD databank can come from several sources:

- (1) ACME end-of-billing period accounting detail which shows LOGON-LOGOFF pageminute charges by account.
- (2) TOD program TD\_CSTCK output.
- (3) Independent logs kept by TOD users.

Each source yields information on some aspects of the cost picture. All have certain biases which should be understood.

ACME End-of-Billing Period Accounting Data:

The ACME end-of-billing period accounting data is the most critical because the monthly bill is developed from it. This is the number we are trying to analyze and justify. At present, an ACME user only gets a monthly bill showing totals for disk blocks, pageminutes, and terminal connect charges for the month. The logon-logoff detail is only provided on special request.

Since this detail is only identified by date, time, terminal, and pageminutes used during a session, and since a user can run many different programs during the time he is logged on, it is difficult to break down the costs of individual sessions to component operations. A component operation can be a program or some transaction(s) within a program.

The availability in ACME of a "pageminute used so far" function would be helpful in breaking down session costs. Even with this function, the responsibility of trapping the data and analyzing it falls to the application program. The availability of this command would eliminate the major error in accounting estimates made by the TD\_CSTCK program described below.

TOD TD\_CSTCK Program:

This program was designed to analyze the data generated by the TODOPEN-TODCLOSE procedures present in most TOD programs. These procedures estimated the page-

minutes used by individual TOD programs and logged the appropriate data. The pageminute function described above would turn these estimates into actual data.

The bulk of TD\_CSTCK is a TOD cost analysis system. Attempts are made to allocate costs by program, but more important, by transactions done by the programs. See ACME Note TODPDA for a discussion of the TD\_CSTCK program.

TD\_CSTCK is the first step in the data analysis because it is designed to allow the user to purge the data once a month coinciding with the accounting cycle. Data over time, for which an analysis framework will be developed later in this note, is not presently kept in computer form. Later as the kinds of analysis become more stable, TD\_CSTCK can be modified to keep data month-to-month and appropriately analyze it.

Presently there exists a program TD\_COMPR, which is a modified TD\_CSTCK. TD\_COMPR will summarize costs for any TOD databank whereas TD\_CSTCK can only access data from the name and project with which a user logged onto ACME. TD\_COMPR is only to be used by the TOD Manager.

#### Individual Logs:

Individual logs suffer from several disadvantages. They require everyone who operates the terminal to use it, a requirement which in practice is never adhered to. To get the detail provided by the method above would require an amount of time most users of the system would not be willing to spend. Individual logs are useful to keep data unrelated to the programs and transactions. For example, keeping a log of the names of people using a databank would allow better distribution of documentation to the people who need it.

#### COST Components for Individual TOD Databanks:

There are three major areas of cost in the operation of a TOD Databank. They are: personnel costs, computer run charges (including terminal rental), and computer disk storage. Personnel costs will not be directly studied here.

#### Computer Run Charges:

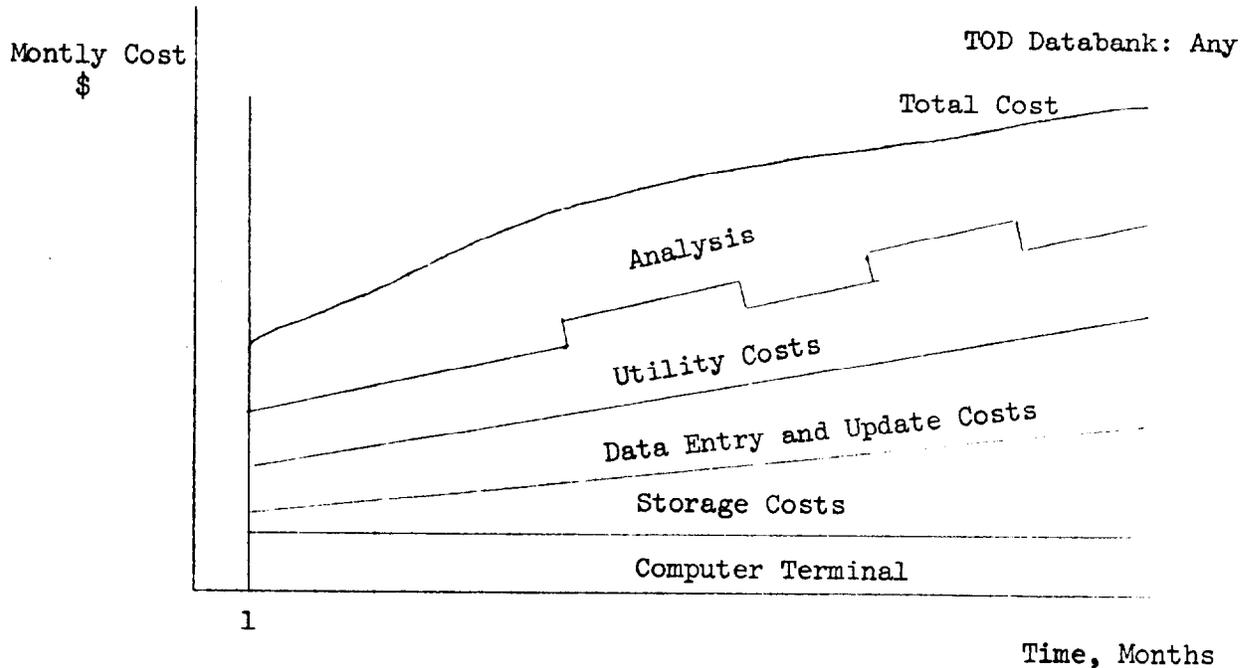
Computer run charges are composed of many components:

- The terminal rental per month is presently \$190 per month.
- The actual program charges for running a TOD databank can be broken down into several functional area: data entry and update; subset; analysis; and utility procedures. Many TOD programs can be found in each area. ACME Note TODI indicates this partitioning.
- The bulk of the computer run costs for a TOD databank fall into the data entry and update area. Dr. Jim Fries has estimated data entry and update charges to be 75% of the run time costs.

Individual Aggregate Analysis:

A useful analysis for an individual databank would be to follow the individual group mentioned above over time. Graph I illustrates this analysis. "Costs" can be measured in dollars, pageminutes, or time-units.

Graph I. Individual Aggregate Analysis



Note: Figure not necessarily to scale.

Graph I illustrates an aggregate analysis and includes all the factors that influence costs, such as more people using the databank, more patients in the databank, higher activity than normal in a functional area, etc.

In order to begin to understand the reasons why these aggregates change, a more detailed analysis is required. Data entry and update costs will be used as an illustration because they are the most critical. A similar analysis could be developed for the other functional areas.

In TOD, a patient's demographic information can be entered; this data can be updated; a patient-visit can be entered; and this visit data can be updated. These operations are done by the four programs TD\_ESTAB, TD\_EFIX, TD\_VISIT, TD\_VFIX, respectively. Costs in each of these programs are influenced by the same variables. The major variables are: the number of patients or visits entered or fixed at one use of the program; the number of elements per patient which were entered or fixed; and the complexity of the data entered. These variables influence computer charges for data entry. They also influence disk charges because the files increase in size as more data is entered.

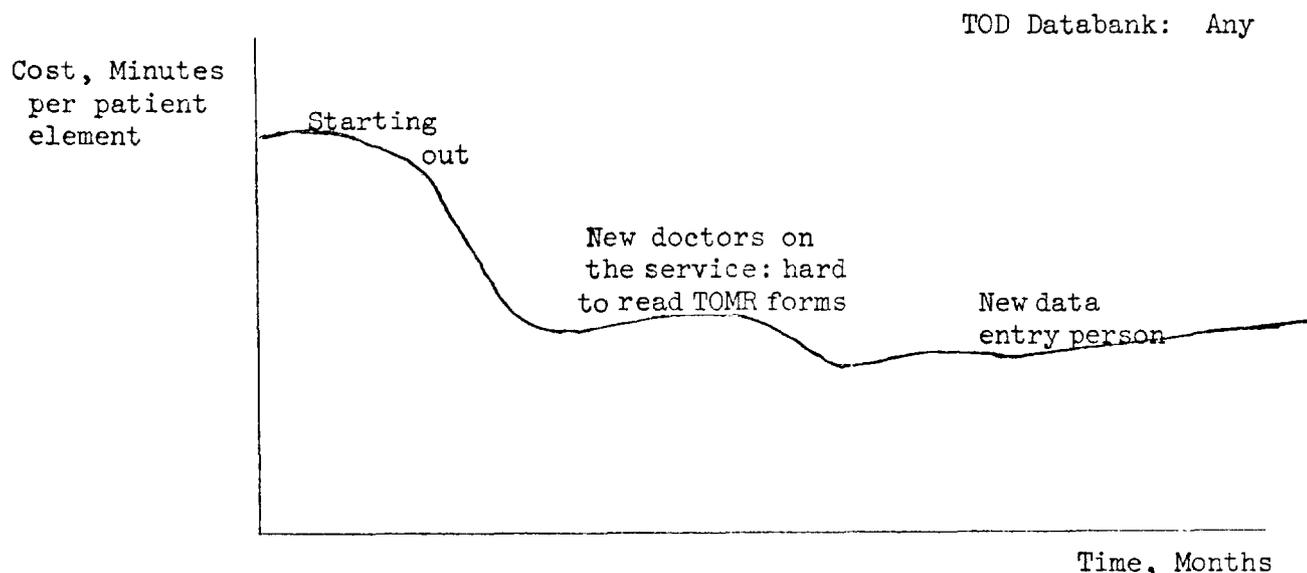
In looking at the costs of using these programs, the example of entering a patient's demographic information using program TD\_ESTAB will be studied. Similar statements can be made for the other programs. In TD\_ESTAB, the cost (time, dollars, or pageminutes) per patient entered is useful because it removes the direct effect of the number of patients entered to yield a more basic cost coefficient. For the same reason, the cost per patient element entered is taken as the elementary cost unit. One could argue that the complexity of the element measured in key-strokes to enter would be the true basic cost unit. At present we are only analyzing down to the element level.

Watching the cost per patient and the cost per element provides useful insight into the data entry costs. If the number of elements added per patient was always about the same, then we wouldn't have to use the cost per element; the cost per patient would do. In Oncology and Immunology, the number of elements entered per patient does vary widely. TD\_CSTCK does not presently include compile time.

One must remember that the cost of compiling a program is finite and could shadow the effect of entering data. The effects are relative in the sense that 20 elements for a patient might be comparable to one quarter of compile time. In this case, ignoring compile time, which is not accounted for by TD\_CSTCK, introduces a sizable error. If 200 elements were entered, the error of not counting compile time becomes negligible.

Having picked cost per element as the elementary item, Graph II illustrates a method of studying it over time. Cost will be measured in minutes per patient element.

Graph II. Elementary Cost Components Vs. Time



One would expect variability in this elementary item for several reasons. A badly written TOMR chart with items improperly marked will slow down the data entry operation. Interruptions of the data entry person such as phone calls, visitors, etc., will increase the data entry cost. A new data entry person will take more time until he or she has learned the system. Assuming a consistent chart, no interruptions, and the same data entry person, the costs still will vary (e.g. data entry person can become bored or fatigued).

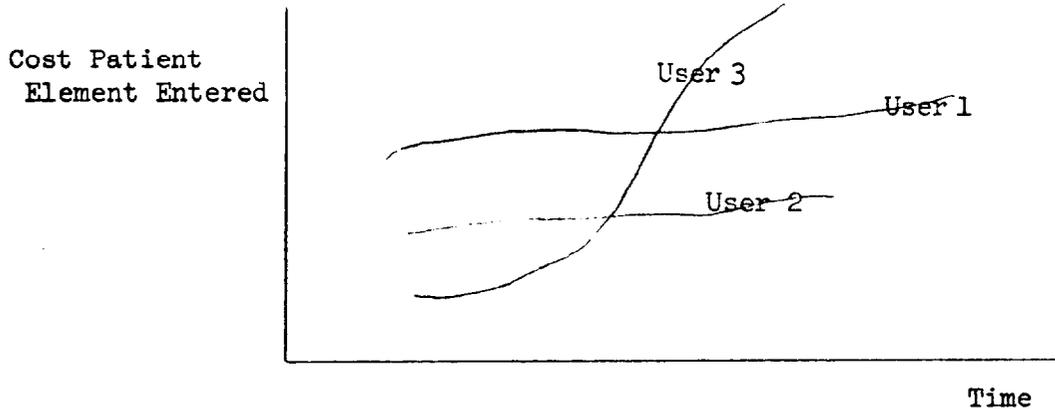
Cost Comparisons Among Databanks:

Aside from intra-databank comparisons discussed earlier, inter-databank comparisons can be made.

Studying the costs of running databanks using the same programs allows us to establish norms or standards for the operations involved in running a databank. The data is also useful in estimating future costs of planned databanks. A better understanding of costs will allow the detection of high cost areas and, hopefully, the subsequent improvement of these areas. For example, if a databank has an elementary cost component twice that of another databank, that tells us something about the first databank's operation. Moreover, if we can determine what the second databank is "doing right" and can teach the first databank the same procedures, it too can lower its costs.

Each month, elementary cost items can be tabulated and summarized for all the operational TOD databanks. Graph III illustrates a data entry cost comparison for several users.

Graph III. TOD Data Entry Cost Comparison



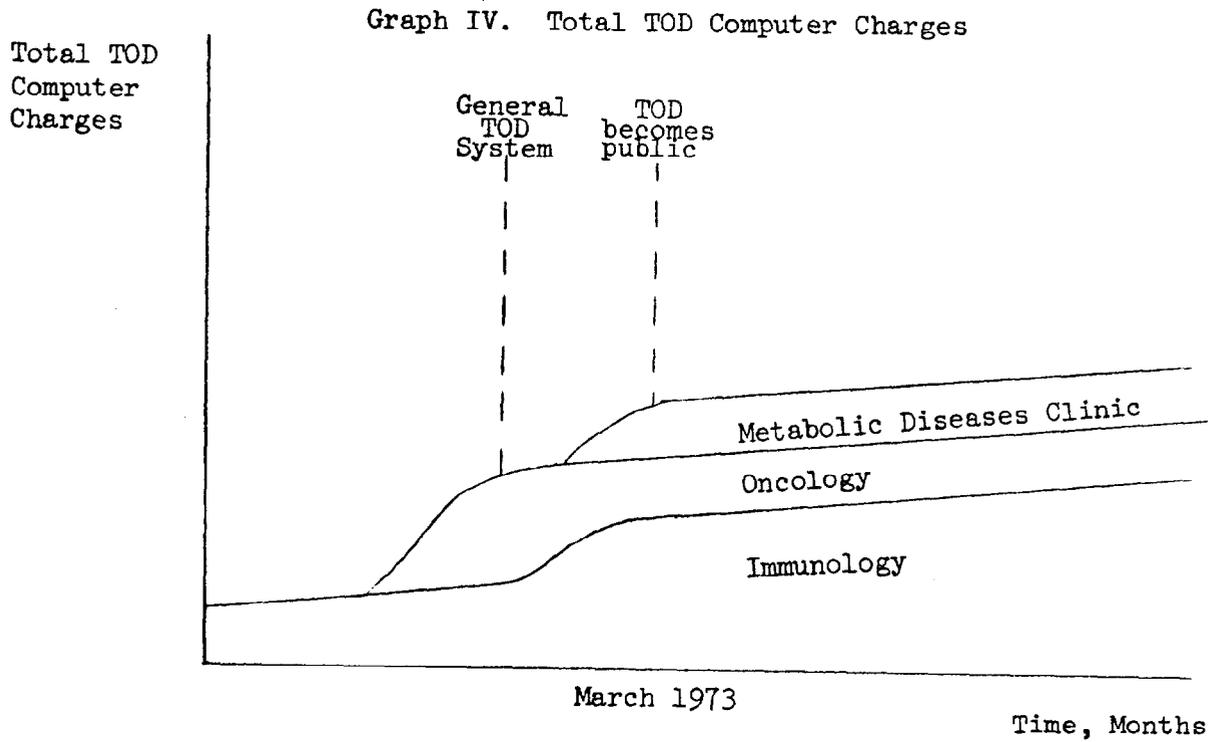
Aggregate items can also be compared among TOD users. Some of the more important items are listed in Table I. All items are per month.

Table I. TOD Costs Comparison for all TOD Users

Month of: x x x x x x

<u>ITEM</u>	<u>USER 1</u>	<u>USER 2</u>	<u>USER 3</u>	...
<u>From ACME Accounting:</u>				
Terminal Cost.				
Computer Cost.				
Number of Pageminutes.				
Number of Minutes.				
Number of Blocks.				
<u>From TD_CSTCK:</u>				
Minutes.				
\$ Estimated.				
Pageminutes Estimated.				
Cost per Program.				
Cost per Functional area.				
<u>Miscellaneous:</u>				
Ratio(\$ from ACME to \$ from TD_CSTCK).				
Ratio(Time from ACME to time from TD_CSTCK).				
Ratio(Pageminutes from ACME to page. from TD_CSTCK).				
% of cost falling in each functional area.				
Number of patients in file.				
Number of patients entered.				
Number of patient-visits in file.				
Number of patient-visits entered.				

A final analysis which would illustrate the use of TOD is shown in Graph IV.



Dist: Staff/TOD/All

Core Research & Development (Continued)

C. New Analysis Programs for Time Series Data

Project: ACME  
Realtime

Investigator: Will Gersch

Dept. of Neurology

1. Description

ARSPEC is an interactive research tool for automatic spectral density analysis of time series data. The program performs four interrelated tasks: (1) display of time series data; (2) filter design and application to data; (3) data reduction; and (4) spectral density computation. The user instructs ARSPEC to perform tasks in a desired order by issuing supervisory commands.

Unlike PUBLIC program TIMESER, ARSPEC uses an automatic decision procedure to produce accurate spectral estimates. This technique is applicable to all time series data.

The computational procedures are defined in ACME Note EARSPE-2.

2. Historical Note

The recursive procedure for computing autoregressive coefficients is due to Levinson (A)(1947). This procedure was "re-discovered" by Durbin (B)(1960). The automatic decision procedure for deciding the order of the autoregressive model is due to Akaike (C,E)(1970). An alternate statistical procedure based on earlier statistical procedures for deciding autoregressive model order appears in Gersch (D)(1970).

3. A Short Explanation of Spectral Analysis by Autoregressive Model Technique

Conventional spectral analysis procedures compute spectral density estimates using a weighted Fourier transform of the empirical autocorrelation function of observed time series. The use of conventional techniques is complicated by the requirement that a user assign the values of parameters determining statistical tradeoffs in the spectral representation. This requires expertise with spectral estimation theory. As a case in point, conventional spectral analysis techniques are employed by ACME PUBLIC program TIMESER.

The autoregressive model technique fits a model to be observed data. This model is autoregressive in that each observation of the time series is expressed as a linear combination of its own past (hence it is regressive on itself) plus a term drawn from an uncorrelated sequence (Equation 1). The coefficients of the model are determined by a least squares fit to the empirical autocorrelation function. The order of the model is determined by the Akaike (E) final predictor error criterion.

## Core Research & Development (Continued)

Spectral density estimates are calculated from the coefficients of the autoregressive model (Equation 2).

The statistical performance of spectral estimation using the autoregressive technique has the properties that (i) the spectral estimates are unbiased, and (ii) the variance of the spectral estimates is approximately given by

$$\text{var } \hat{S}(f) = \frac{2p}{N} \hat{S}^2(f)$$

where  $p$  = the order of the autoregressive model. The statistical performance of this procedure is at least as good as the performance of the best conventional spectral estimate. Finally, the fact that the procedure is unbiased eliminates the need for expert determination of the statistical trade-offs, and hence the procedure can be made automatic.

### 4. References

- A. Levinson, N. (1947). The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction. *J. Math. Physics*, XXV, pp. 261-278.
- B. Durbin, J. (1960). The fitting of time-series models. *Rev. Int. Stat. Inst.* 28, pp. 233-244.
- C. Akaike, H., Information theory and an extension of the maximum likelihood principle, presented at the Second International Symposium on Information Theory, Tsahkadsor, Armenian SSR, 2-8 of September, 1971. (To be published in Problems of Control and Information Theory, USSR 1972.)
- D. Gersch, W. (1970), "Spectral Analysis of EEG's by Autoregressive Decomposition of Time Series", Mathematical Biosciences, 7, 1970, pp. 205-222.
- E. Akaike, H., Statistical Predictor Identification, *Ann. Inst. Stat. Math.*, 21, 1970.

### 5. Sample Runs

Raw data is a sine wave imbedded in noise. Noise is Gaussian noise with mean zero and standard deviation an order of magnitude greater than the sine wave amplitude. Noise has been passed through a high pass filter before addition to the sine wave.

Core Research & Development (Continued)

Data set number of graphics device=?13  
Ψ YOU ARE NOW PERMITTED 01 LINES ON THE 1800  
Number of raw data samples=?1000  
Name of raw data file =?TEST  
Key of record containing raw data=?1  
Enter a command =?display  
Enter 0 to operate on raw data, nonzero to operate on processed data=?0

DFILE= TEST, DREC= 1

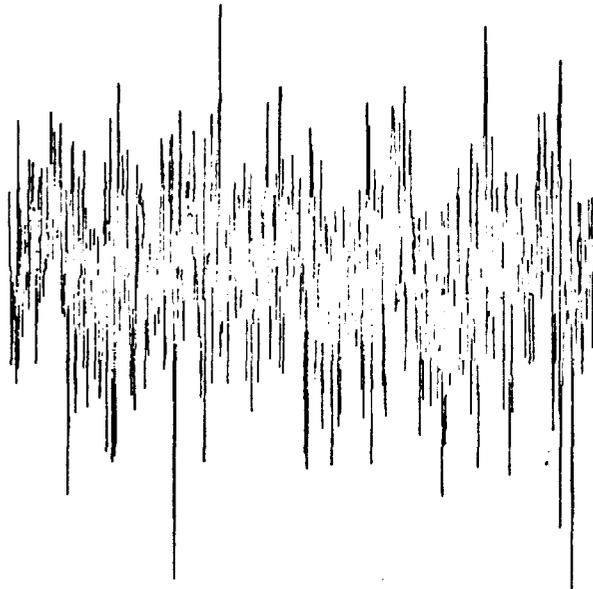


FIGURE 1

Enter a command =?spectrum  
Enter 0 to operate on raw data, nonzero to operate on processed data=?0  
='Autoregressive model is of order',= 40;  
Optional description of spectrum =?sine wave in filtered noise  
Enter 0 if done, nonzero to save spectrum on data file.=?0

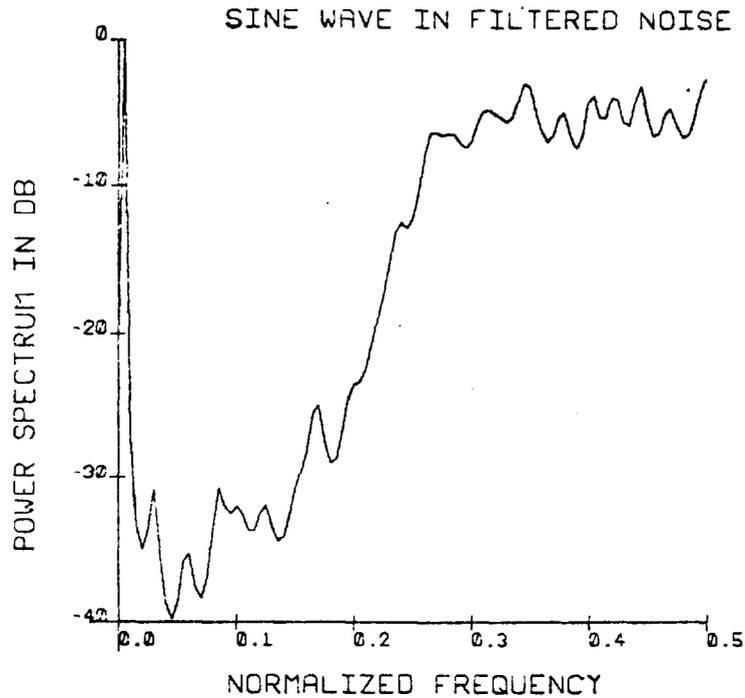


Figure 2

Enter a command =?filter  
High frequency cutoff point=?0.1  
Low frequency cutoff=?0.05  
Enter 0 for low pass, nonzero for high pass filter=?0  
Enter 0 for unsmoothed, nonzero for smoothed filter=?1  
Enter 0 to display, nonzero to apply filter=?0  
Enter 0 to operate on raw data, nonzero to operate on processed data=?0

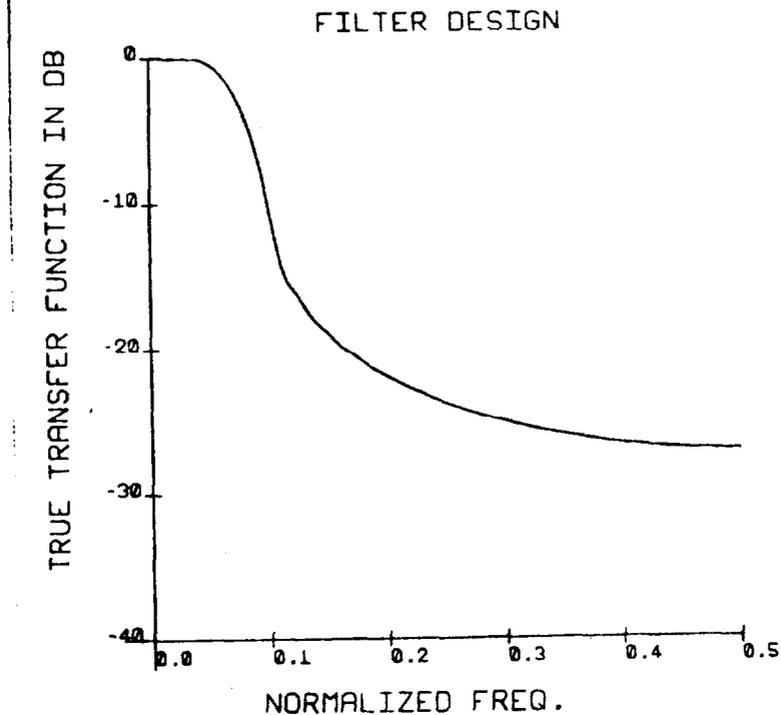


FIGURE 3

After filtering, we can see the original sine wave.

Enter a command =?display

Enter 0 to operate on raw data, nonzero to operate on processed data=?1

PROCESSED DATA

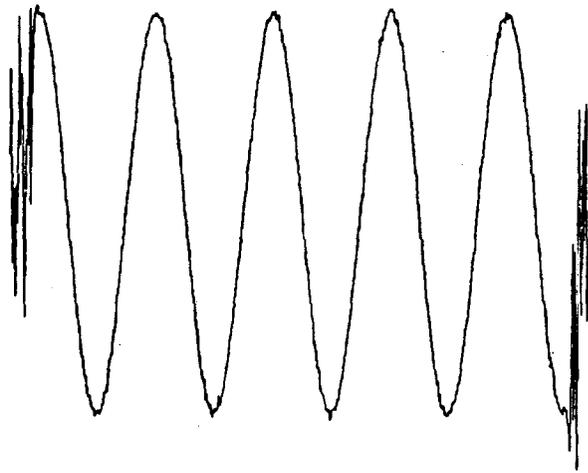


FIGURE 4

Note the end effects caused by filtering

2. The second example is a spectral analysis of actual EEG data.

Data set number of graphics device=?91

Number of raw data samples=?2000

Name of raw data file =?EEGDATA

Key of record containing raw data=?100

Enter a command =?display

Enter 0 to operate on raw data, nonzero to operate on processed data=?0

DFILE= EEGDATA, DREC= 100

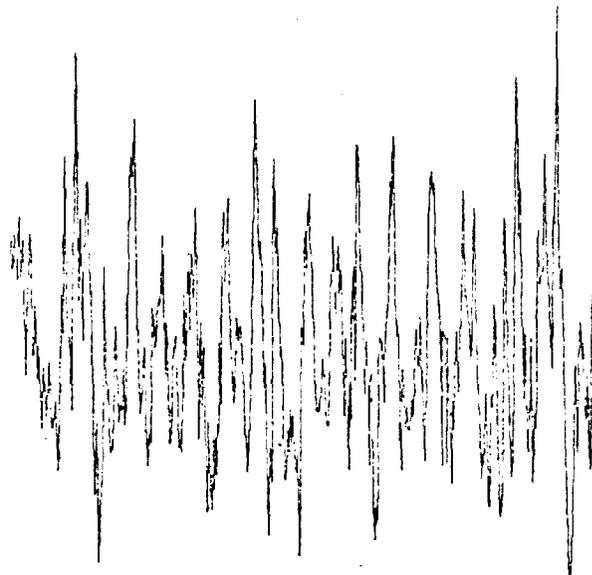


Figure 5

Enter a command =?spectrum  
Enter 0 to operate on raw data, nonzero to operate on processed data=?0  
='Autoregressive model is of order',= 22;  
Optional description of spectrum =?EEG SPECTRUM  
Enter 0 if done, nonzero to save spectrum on data file.=?0

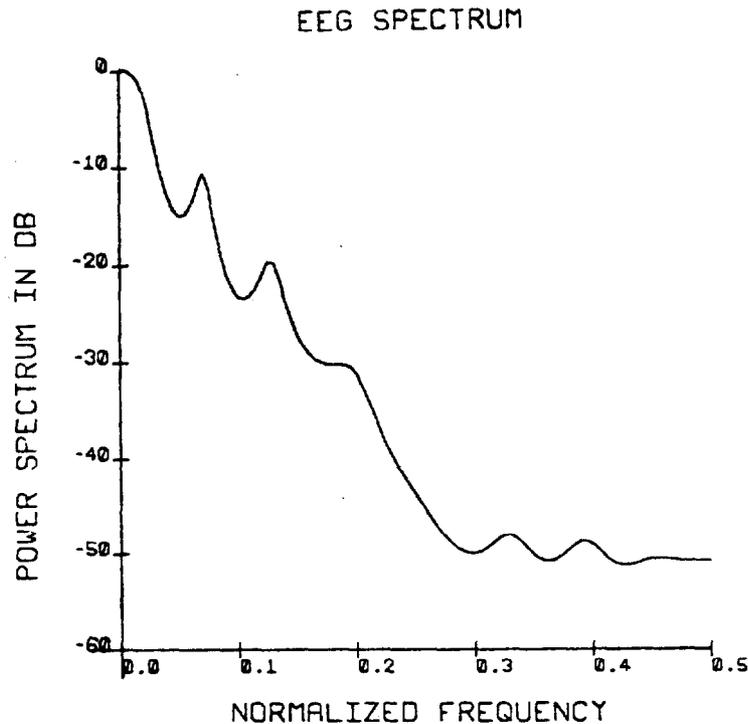


Figure 6

A low pass filter is now applied; then data is reduced to 1000 points.

Enter a command =?filter  
High frequency cutoff point=?0.25  
Low frequency cutoff=?0.2  
Enter 0 for low pass, nonzero for high pass filter=?0  
Enter 0 for unsmoothed, nonzero for smoothed filter=?1  
Enter 0 to display, nonzero to apply filter=?1  
Enter 0 to operate on raw data, nonzero to operate on processed data=?0  
Enter a command =?reduce  
Enter 0 to operate on raw data, nonzero to operate on processed data=?1  
Number of reduced data samples=?1000  
Enter a command =?spectrum  
Enter 0 to operate on raw data, nonzero to operate on processed data=?1  
='Autoregressive model is of order',= 21;  
Optional description of spectrum =?  
Enter 0 if done, nonzero to save spectrum on data file.=?0  
Enter a command =?done  
● 127: RUN STOPPED  
AT LINE 3.100 IN PROCEDURE ARSPEC

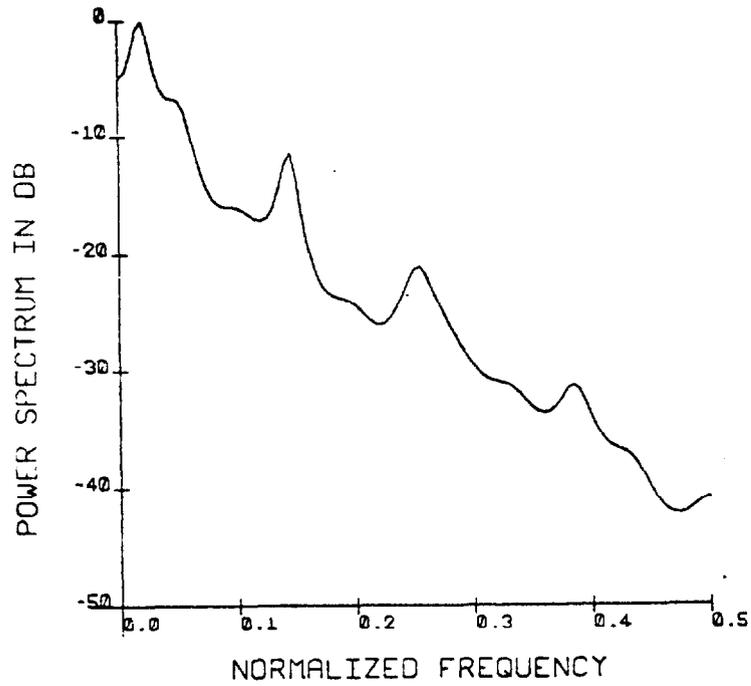


Figure 7

The spectral outline of the reduced and filtered data has characteristics parallel to the original data. Note that 50% reduction of data causes the normalized frequency scale to double.

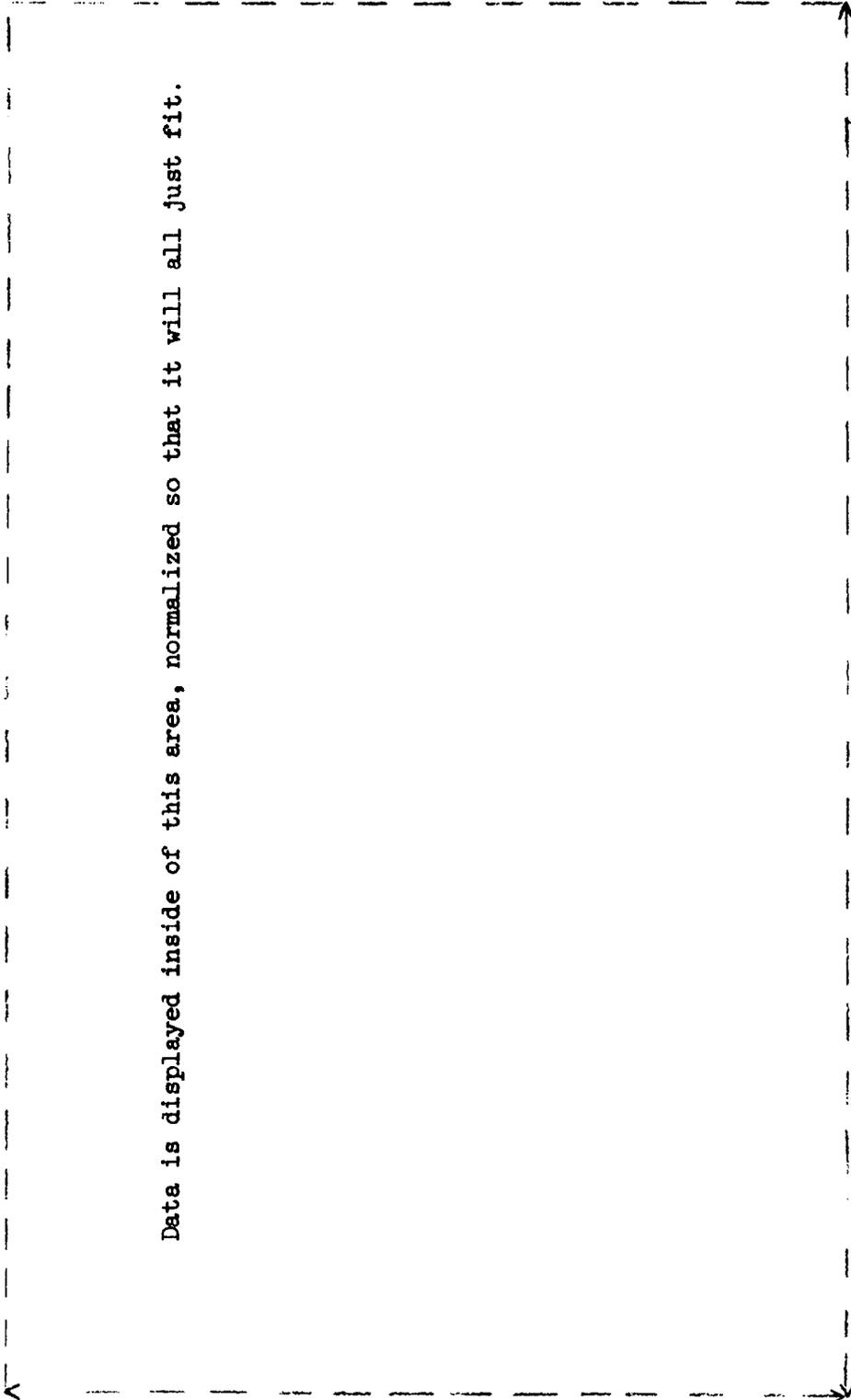
APPENDIX - 2

Format for displaying raw data

```

*charh=30
|<dfile=raw data filename> <drec=key of raw data record >
* (tcharx, tchary) = (center, 935)

```



Data is displayed inside of this area, normalized so that it will all just fit.

\* gridheight=800

\* gridleft=150

\* gridlength = 800

\* gridbot=100

\*These are the actual program names of parameters defining the output format, specified with the number of rasters they are initialized to. To modify these settings, change the first declaration at the beginning of ARSPEC.