Construction of a Library of Computer Routines for Refrigeration Systems Analysis

T. C. Scott

G. L. Davis

CONSTRUCTION OF A LIBRARY OF COMPUTER ROUTINES FOR REFRIGERATION SYSTEMS ANALYSIS

T. C. SCOTT, SENIOR RESEARCH ENGINEER
G. L. DAVIS, MANAGER
ADVANCED ENGINEERING DEPARTMENT
SUNDSTRAND COMPRESSORS, BRISTOL, VIRGINIA

INTRODUCTION

Many powerful computer programs exist for the solution of difficult engineering problems. The aerospace industry has pioneered the development of many such programs for problems such as thermal and structural analyses of large systems. The major drawback of these advanced routines is that they are difficult for the average computer user to apply. To understand them requires a specialist in the particular field with additional experience in computer graphics. This restricts such programs to large problems whose complexity warrants the time required to learn how to use them.

At the other extreme are a large number of "scientific" program packages which are easy to use and available as standard software on most of today's computers. Such programs are generally devoted to mathematical and statistical problems (solution of polynomials, curve fitting, matrix manipulation, etc.).

In the middle lies a region which is highly populated but poorly organized. This region consists of intermediate level programs for the solution of specific engineering problems. The large number of such programs is perhaps nowhere more obvious than in the Purdue Compressor Technology Conferences themselves. Unfortunately, most intermediate level programs are created by and for research and development work. For the line or applications engineer to make use of them often requires an excessive amount of learning time.

To make these programs available in an easier to use form for the engineering profession in general is impractical due to their specialization. Within any particular industry, however, the possibilities are more promising provided that these programs can be properly organized, interfaced, and documented. These problems present a considerable challange to any industry willing to undertake the task of bringing more powerful computer techniques to its engineers.

In order to be successful in such a venture, many basic questions must be answered such as

1. What kinds of computer programs should be provided.
2. What are the characteristics of potential users.
3. How much effort should the users be expected to invest to learn how to use the programs.
4. How can misuse be prevented.

and the answers vary from industry to industry, company to company.

This paper describes one such effort to develop a system of computer programs applicable to the refrigeration industry. The scope of the system is large as shown in Figure (1) and many of the blocks are still in the developmental stage. The structure of this library of computer programs and how it addresses itself to some of the above questions will be discussed. The reader should keep in mind that this represents only one approach and that other workable schemes are possible. It is hoped that the discussion presented here will aid others engaged in similar projects.

PHILOSOPHY

Before beginning to develop a library of large scope, some basic premises must be set down governing how much is to be done and how much the user will be expected to invest.

In the library discussed here, the emphasis has been on minimizing only the required computer programming skills of the user. The level of engineering skill required, however, is not diminished to the point where the computer is a replacement for the engineering thinking process. This makes it difficult for an engineer to use the library unless he has a reasonable grasp of the principles upon which it is based. It requires the user to invest at least a minimal effort to understand what the programs do internally from a conceptual standpoint and it insures that only those users willing to continue learning can have access to the library. The engineer looking for an easy substitute for thinking on his own rapidly becomes discouraged and rarely makes more than a token use of the available programs. The engineer who actively works to incorporate what we might call "computer based thinking" can easily grasp the basic structure of the library and apply it successfully.

This philosophy was arrived at by the authors after many years of trying without success to get <u>everybody</u> computer oriented. Our personal conclusion is that there will only be a small percentage of the engineering staff willing to invest the effort required to use the library. To simplify the system in an attempt to gain more users leads to an exponential growth in the number of programs required and in their misuse. One must accept the fact that many engineers will never make use of the library. Engineers cannot be forced to use it nor can you afford to put it into the hands of someone who is not initially a competent engineer.

USER CHARACTERISTICS

Engineers who make use of computers may be put into a number of classifications. For this discussion we shall divide them into applications and research and development (R&D).

The applications engineer is primarily concerned with what is. His projects require that he select components from existing groups and combine them into systems or examine the performance of existing systems such as current products being manufactured by his company. For example, an applications engineer may desire the performance of a hermetic compressor at off-design conditions. This may develop from a customer's request for performance curves for a unit operating on 210 volts when the design voltage is 230. As another example, the effect of clearance on journal bearing behavior may be desired in order to set manufacturing tolerances. It is desirable in these situations for the engineer to have access to analytical tools for predicting the behavior or existing devices and systems.

Of high priority in creating a system of computer programs for the applications engineer is ease of usage. But this must be tempered with the realization that ease of usage can lead to ease of misusage. It is very easy for computer users to let the computer do too much of the thinking and to accept computer generated results without question. Such a situation can arise if the user can solve a problem on the computer without being required to understand what the computer is doing.

One solution to this misusage is to provide only basic mathematical operations, thus requiring the user to develop the problem and write a major portion of the program logic himself. This usually leads to non-usage because the applications engineer does not have the time or programming skills to develop programs. It can also lead to the creation of another group of computer specialists whose job is to write programs for the engineer. All too often this solution is no solution at all because the computer specialists are not engineers. A communication gap then develops which leads to non-usage. Any workable package of computer programs for applications engineering must strike a balance between all these factors.

The R&D engineer deals more with what is not. His projects require him to develop new systems and analyse problems in more detail. The R&D engineer needs programs with more power and flexibility. Fortunately the R&D engineer also has more time to examine the basic fundamentals of the problem and do more of the programming himself. By necessity he must be a better computer programmer.

BASIC LIBRARY STRUCTURE

Figure (2) shows a general block diagram of the computer library. In this structure basic support programs back up a system of applications programs for use by the applications engineer. These basic support programs perform mathematical operations, calculate fluid properties, and store performance data on heat transfer surfaces, bearings, etc. The applications programs draw on these basic programs to provide particular services such as the performance of existing heat exchangers, compressors, and so forth.

The sophisticated support programs contain more powerful and more difficult programs. Thermal and structural analysis packages are examples of its content. These programs are called upon by R&D programs for specific problems. A finite element analysis of compressor reed valve behavior is a typical example. An R&D engineer might desire such a program to develop a new valve design but an applications engineer rarely requires such sophistication.

INPUT-OUTPUT FORMAT

All the programs in the library except the sophisticated support programs are either SUBROUTINE or FUNCTION subprograms written in FORTRAN. This allows the user to link routines together to create systems of programs for a variety of problems. All of the arguments for those routines which interface directly with the user are floating point or integer constants or arrays. The user never has to employ COMMON statements or possess a high degree of programming skill. Each subprogram also has its own formats so that the user rarely has to create write statements. These features allow the user to solve problems by creating only a short MAIN program. In some cases the routines are accessible to terminals directly so that the required input is requested by the routine. The objective is to reduce the required programming skills of the user to a minimum but, as later examples will show, basic engineering skills are an absolute necessity.

The argument list for all routines is arranged as follows

SUBROUTINE SAMPLE(C_1,C_2,...,X_1,X_2,...,Y_1, Y_2,...,IER,IPRT)

where

C_1,C_2,...	= Any constants or arrays of constants.
X_1,X_2,...	= Input variables.
Y_1,Y_2,...	= Output parameters.
IER	= An error code identifying the cause of any errors.
IPRT	= A print code.

The integer print code, IPRT, is input by the user as any number greater than zero. If IPRT is input greater than 2, no printing is done by the routine.

If IPRT=1, the routine will print an error message detailing the type and cause of any error which occurs.

If IPRT=0, the routine will print out the essential input and results as well as any error message.

If IPRT=2, the output associated with IPRT =0 will occur as well as the results of intermediate steps during computation.

As an illustration, consider a subprogram for the solution of a system of non-linear equations. Such a routine requires the user to make an initial guess of the roots. Systems of non-linear equations are, in many cases, highly unstable unless the initial guesses fall within certain limits. If the user sets IPRT=2 when calling this routine, the results of each iteration are printed. This output aids the user in discovering which initial guess is the source of a stability problem. With IPRT=0, only the final values of the roots or an error message would be printed.

As a second illustration, one of the applications programs evaluates hermetic compressor performance given the evaporating temperature, condensing temperature, line voltage, line frequency, type of refrigerant, inlet superheat, condenser subcooling, and a code identifying the particular model of unit. With IPRT=0, the output is system EER, killowatts, refrigerant flow rate, evaporator and condenser capacity. With IPRT=2, more detailed data such as RPM, oil temperature, volumetric efficiency, etc. is also printed.

Many subprograms automatically call upon others in the library during their execution. This can lead to a problem in determining the source of an error if it occurs. To overcome this difficulty, each subprogram sets IPRT=1 whenever it calls upon another. In this way, the user can quickly discover where an error occurs.

SUBPROGRAM BLOCK EXAMPLE

Each block of Figure (2) is broken down into several smaller blocks or groups of routines. We shall illustrate the content of one of these blocks as an example.

The study of thermo-fluids problems requires knowledge of the thermo-physical properties of fluids. A major block of subprograms in the basic support programs library is devoted to this effort. The individual routines and their functions are outlined in Table (1).

The only input required for any fluid is an array of 96 constants. These are the constants for the equation of state and other equations used for viscosity, conductivity, etc. Fluids whose behavior is modeled by any of the following equations of state

Beattie-Bridgeman
Benedict-Webb-Rubin
Martin-Hou
Van der Waals
Berthelot
Redlich-Kwong

as well as water may be examined. One of the 96 constants is a code telling the programs which equation of state to use as well as which of several equations to use for other properties. The program results duplicate the fluid property data in steam tables, ASHRAE handbook tables, and other sources exactly since the same equations were used to create these tables. The arrays of constants for a large number of fluids are stored internally so that the applications engineer need only input one number identifying the fluid type. The R&D user may examine many other fluids by inputing his own array directly.

As a simple example of the use of this program block, consider the problem of finding the outlet state of a compressor with a suction temperature of 80°F, suction pressure of 90 psia, discharge pressure of 300 psia, and 70% isentropic efficiency using refrigerant R-22. A manual solution would require the following steps.

1. Find entropy and enthalpy at 80°F, 90 psia.
2. Interpolate into R-22 tables at this entropy and 300 psia to find the isentropic outlet enthalpy, h_s.
3. Find the true outlet enthalpy

$$h = h_{in} + \frac{h_s - h_{in}}{0.7}$$

4. Interpolate into R-22 tables at this h and 300 psia to find the true outlet.

Using the computer routines of Table (1), the procedure is

1. Call up the array of constants for R-22
2. Call SHPVT, SHENTH, and SHENTR to find inlet state.
3. Call SSTV to find isentropic outlet.
4. Calculate

$$h = h_{in} + \frac{h_s - h_{in}}{0.7}$$

5. Call SHTV to find true outlet state.

Below is a MAIN program in FORTRAN for the solution of this problem. The arguments of each routine will not be discussed in detail but comment cards have been inserted to explain each step.

```
C   OPEN UP STORAGE FOR 96 CONSTANTS
       DIMENSION F(96)
C   CALL UP ARRAY OF CONSTANTS FOR R-22
       CALL FARRAY(22,F)
C   SET INLET AND OUTLET DATA
       PI=90.
       TI=80.
       PO=300.
C   FIND INLET SPECIFIC VOLUME
       CALL SHPVT(F,PI,VI,TI,1,IER,1)
C   FIND INLET ENTHALPY
       HI=SHENTH(F,PI,VI,TI,1)
C   FIND INLET ENTROPY
       SI=SHENTR(F,VI,TI,1)
C   FIND ISENTROPIC OUTLET STATE
       CALL SSTV(F,PO,SI,TS,VS,IER,1)
C   FIND ISENTROPIC OUTLET ENTHALPY
       HS=SHENTH(F,PO,VS,TS,1)
C   CALCULATE TRUE OUTLET ENTHALPY
       HO=HI+(HS-HI)/.7
C   FIND TRUE OUTLET STATE
       CALL SHTV(F,PO,HO,TO,VO,IER,0)
       STOP
       END
```

There are no write formats in this program. The last argument of each routine is the print code which is set at 1 for all but the last. The last routine will thus print the outlet state data.

TABLE 1: Content of Thermopysical Properties Block

ROUTINE	FUNCTION
SHPVT	Find pressure, specific volume, or temperature given the other two properties.
SHENTH	Find enthalpy of superheated vapor.
SHENTR	Find entropy of superheated vapor.
SPHTO	Find specific heat of ideal gases.
SPHRA	Find specific heat and sonic velocity of real gases.
VISG	Find viscosity of gases.
CONDG	Find conductivity of gases.
PSAT	Find saturation pressure given saturation temperature.
DPSAT	Find slope of saturation P-T curve given saturation temperature.
TSAT	Find saturation temperature given saturation pressure.
DENSF	Find density of liquids.
SATUR	Find all thermodynamic properties in saturation region.
SPHTF	Find liquid specific heat.
VISF	Find liquid viscosity.
CONDF	Find liquid conductivity.
SURF	Find surface tension.
SHTV	Find state given pressure and enthalpy.
SSTV	Find state given pressure and entropy.

The results of running this program are as follows

```
---OUTPUT OF SUBROUTINE SHTV---
AT P=  300.00  PSIA, H=  135.38  BTU/LBM
T=  226.38    DEG. F, V=  0.24363 CU.FT/LBM
```

This example illustrates how the user must know how to solve the engineering problem himself before he can utilize the library. This is in keeping with the philosophy of the library structure. It is possible, of course, to construct another program to solve this particular problem but to embark on such a course would result in an excessive number of different programs and isolate the user from requiring a basic understanding of what is being done. In fact, the above example would probably be solved manually, as it should be in this case since only one problem is involved. To use the computer for such a simple job is not efficient. If many computations were required, however, the engineer might want to construct such a program and it is better that he do this himself, using the library as an aid.

DOCUMENTATION

Because of the complexity of many intermediate level computer programs, there are many ways in which the user can get incorrect results or generate errors. For example, inputing a condensing temperature greater than the critical temperature to a refrigeration condenser computer program can lead to difficulties. In some cases checks can be put into the routine to prevent this but it is often impossible to check for every user mistake. This situation creates two important requirements.

1. Each routine must be checked carefully before being released for use. This makes it essential that an engineer, not a computer specialist either write the routine or be heavily involved.
2. Routines must be carefully documented.

All the routines in the library are documented on two levels. The first which one might call the "applications" level is illustrated by the sample below. A mathematical example has been chosen because the documentation is short. Routines for compressor analysis, etc. require more lengthy documentation and an example may be found elsewhere in these proceedings*

*Davis, G.L. and Scott, T.C., "Component Modeling Requirements for Refrigeration System Simulation"

(SAMPLE DOCUMENTATION)

EVALUATION OF DEFINITE INTEGRALS

The evaluation of

$$\int_{x_i}^{x_f} f(x)\,dx$$

when x_i and x_f are known, corresponds to finding the area under the curve described by f(x) between x_i and x_f. We present two computer routines for this depending on whether f(x) is a known function or is given as a set of tabulated data.

When f(x) can be expressed analytically, one can use Simpson's rule. The subroutine below uses Simpson's 1/3 rule with continuous halving of the step size until successive computations of the integral differ by less than a prescribed tolerance. The user must supply a function subprogram for evaluating f(x) at any given x.

To help improve the result, we include an extrapolation technique. If R_1 and R_2 are two approximations to the solution, then an improved solution is given by

$$R = \frac{16}{15} R_2 - \frac{1}{15} R_1$$

This is the Richardson extrapolation formula for the case of Simpson's 1/3 rule with step size halving. We shall apply it to the last two calculated values of the integral to get an improved result. The logic proceeds as follows:

1. Set step size.
2. Evaluate the integral to get R_1.
3. Halve step size.
4. Evaluate the integral to get R_2.
5. Find an improved result from
$$R_a = \frac{16}{15} R_2 - \frac{1}{15} R_1$$
6. Halve step size.
7. Evaluate the integral to get R_3.
8. Find an improved result from
$$R_b = \frac{16}{15} R_3 - \frac{1}{15} R_2$$
9. If $\frac{|R_b - R_a|}{|R_b|}$ is less than the specified tolerance, stop. If not, return to step 6 and repeat.

SUBROUTINE SIMPS (XI,XF,NH,EPS,C,ANS,UFSIM,IER,IPRT)

FUNCTION: Evaluates a definite integral by Simpson's 1/3 rule with repeated step size halvings and Richardson extrapolations

ARGUMENT	INPUT (I) or OUTPUT (O)	DESCRIPTION
XI	I	The lower limit of the integral.
XF	I	The upper limit of the integral.
NH	I	The maximum number of times the step size may be halved.
EPS	I	The degree within which successive calculations of the integral must agree $EPS = \frac{\lvert solution - last\ solution \rvert}{\lvert solution \rvert}$
C	I-O	An array which simply passes constants on to the user supplied function.
ANS	O	The final calculated value of the integral.
IER	O	An error code. IER = 0 : No error; IER = I : No convergence to within EPS after NH halving of the step size.
IPRT	I	A print code. If IPRT = 0, the results or an error message will be printed out. If IPRT = 1, only an error message will be printed if an error occurs.

OTHER ROUTINES REQUIRED: A user supplied function UFSIM for calculating the function at any x.

COMMENTS: Observe that the user may pass any number of constants on to the user supplied function through the C array.

REFERENCES: Conte, S.D., Elementary Numerical Analysis, McGraw-Hill, 1965

McCormick, J.M., and Salvadori, M.G., Numerical Methods in Fortran, Prentice-Hall, 1964

LXAMPLL: Evaluate the Integral

$$\int_1^5 \left[\exp(c_1 x) - c_2 x^2 - x^{c_3}\right] dx$$

with $c_1 = 1.$, $c_2 = 2.$, $c_3 = 3.5$

We shall use the C array to transfer the three constant values to the user supplied function. The main program for the job is

```
DIMENSION C(3)
EXTERNAL FUN
DATA C/1.,2.,3.5/
CALL SIMPS(1.,5.,20,. 0001,C,ANS,FUN,IER,0)
STOP
END
```

where we have set the tolerance at 0.0001

The user supplied function is

```
FUNCTION FUN(X,C)
DIMENSION C(1)
FUN=EXP(C(1)*X)+C(2)*X*X-X**C(3)
RETURN
END
```

and, the resulting output is

```
---OUTPUT OF SUBROUTINE SIMPS---
integral= -83.981       TO EPS= .10000E-03 IN    3 HALVINGS
```

Observe that this sample documentation does not go into the mathematics of Simpsons rule. The user is expected to be familiar with this or take the initiative to familiarize himself through the listed references. This is in keeping with the philosophy of the library.

In some cases, of course, documentation must be more instructive. This is particularly true when the subject goes beyond what is available in basic references. As an example, one of the support programs is a compilation of a large number of relations for free and forced convection heat transfer coefficients. The user inputs the Reynolds number and Prandtl number or Grashof and Prandtl number along with a code specifying the geometry and a series of significant length dimensions. The program returns the Nusselt number. The documentation for this program defines these dimensionless groups and lists the formulas for each geometry as well as the source. The documentation thus serves two purposes; it is a guide to the program as well as a heat transfer manual. This also becomes a clever way to "sell" computer usage. The engineer who uses the heat transfer manual for hand calculations is continuously lured towards giving the computer program a try.

The second level of documentation is designed for the R&D user who may want to incorporate portions of the routines in new programs or alter existing routines to perform other functions. This level of documentation includes a listing of the routine, a flow chart, and instructions for obtaining a copy on tape or punched cards.

The complete derivation of the equations used in the routine is included in many cases as well. Take, for example, the subprogram block listed in Table (1). If a user has a fluid modeled by a different equation of state, he may want to simply call up the routines and add the new equation. To do so he will need to know how thermodynamic properties are found from an equation of state and what each of the 96 constants stands for. The second level of documentation for this block contains all this information as well as examples showing the user how to add new equations and determine each of the 96 constants. This kind of detailed documentation is an absolute necessity when the original creator of the program is no longer available. Someone else must be able to take over the job of updating the program with minimum difficulty.

In addition to this basic documentation is the need for instructing new users of the library and/or providing reviews of basic programming. Some programming review is given in the first section of each first level documentation manual. This also contains the particulars of the computer being used such as control card formats. Other instruction must often take the form of group instruction.

THE MODULAR APPROACH

One of the keys to the library's usefulness is its flexibility. This is achieved by a pyramid structure in which complex programs are created by combining several simpler "modules".

As an example, one block of programs analyses the behavior of rotary sliding vane compressors. As with many such analyses, this program integrates the governing equations numerically from suction to discharge to give pressure and temperature profiles. It may also include vane dynamics and tip friction if the user desires. The program is not, however, one single large routine but consists of a sequence of logic which calls upon a number of modules. Each module performs one of the necessary functions.

Figure (3) shows the relation between some of the major modules serving this program.

This modular approach serves several purposes. First of all, many of the slave modules are not unique to the rotary vane problem. They are called upon by a number of other programs for slider bearing problems, fluid flow problems, etc. as well. By making these modules general, duplication of effort is reduced.

Secondly, since each module is complete in itself, the user may run individual modules separately to study smaller problems without requiring the entire system of programs.

Finally, the beginning user is not forced to confront one large program. Instead, he may look at the serving modules one at a time, see how they work, and try running them separately. The documentation aids here by starting with the routine for housing contours and working up through the other modules.

It is important to realize that unless an engineer has previous experience with the mathematical analysis of rotary vane machines he will be completely lost if presented with a large program which does everything. You cannot go up to an engineer working on a new rotary vane compressor and say

"The R&D people have a computer program which might help you."

and then drop a 2000 card deck in his lap. The engineer will be overwhelmed and afraid to tackle the job of figuring out how to use the program.

With the modular approach the engineer can spend one hour, learn how to run several of the supporting routines, and generate some volume or vane tip friction data. With his confidence and interest captured, he is more inclined to continue.

CONCLUSIONS

Many research and development engineers and computer specialists are continually bewildered when the programs they write and use daily are not picked up by the rest of the engineering staff. One reason may be that they overestimate the ability of the average engineer. Unless one uses the computer regularly it is very difficult to follow someone elses program. The average engineer may require days of study to grasp what the specialist sees immediately.

Libraries of computer programs also go unused because they are too complex, too specialized, or documentation fails to point out simple, basic information (such as where the "on" switch for the terminal is). The library described in this paper represents one method of trying to deal with these difficulties.

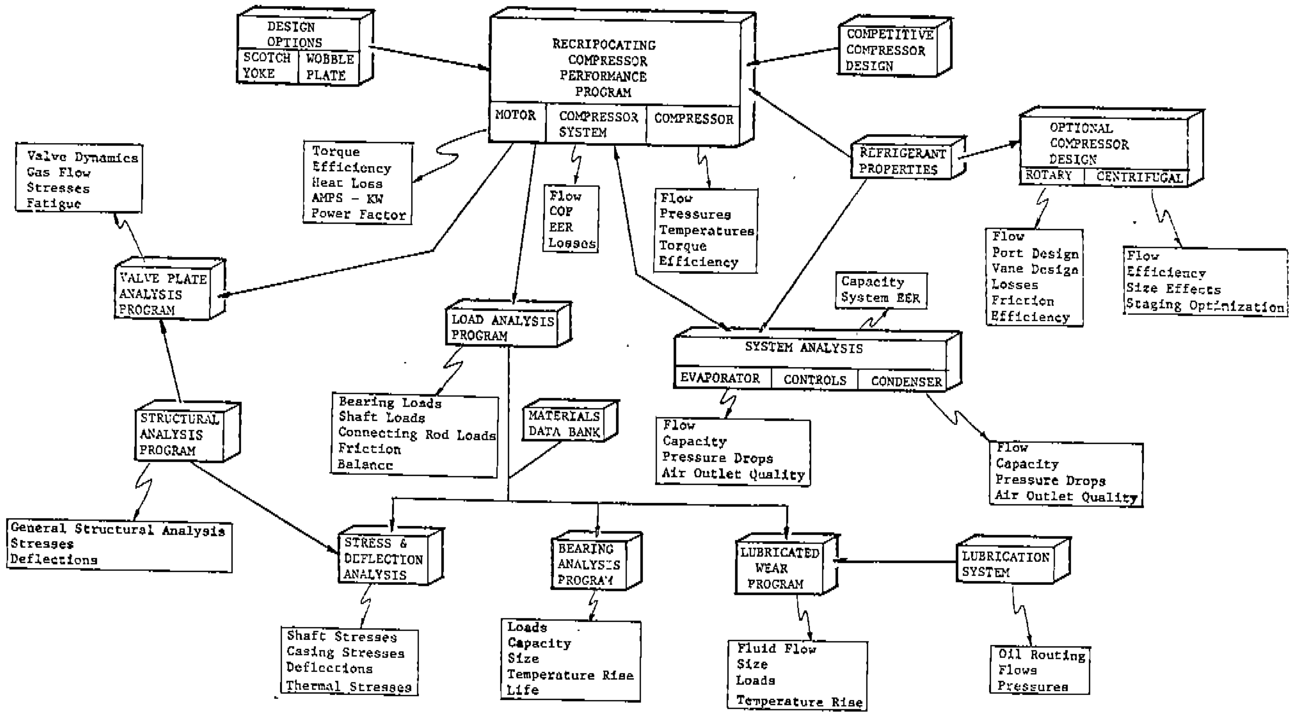

FIG. 1: General View of Library Contents

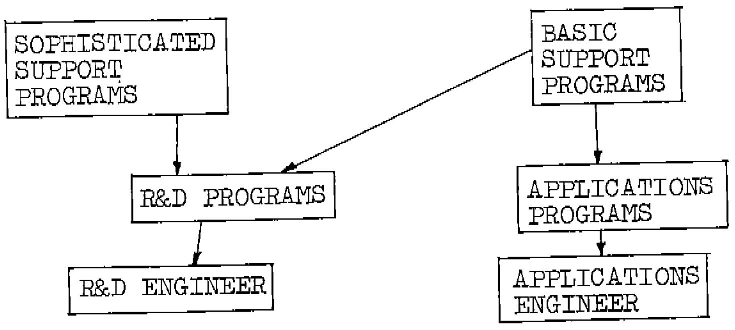

FIG. 2: Basic Library Structure

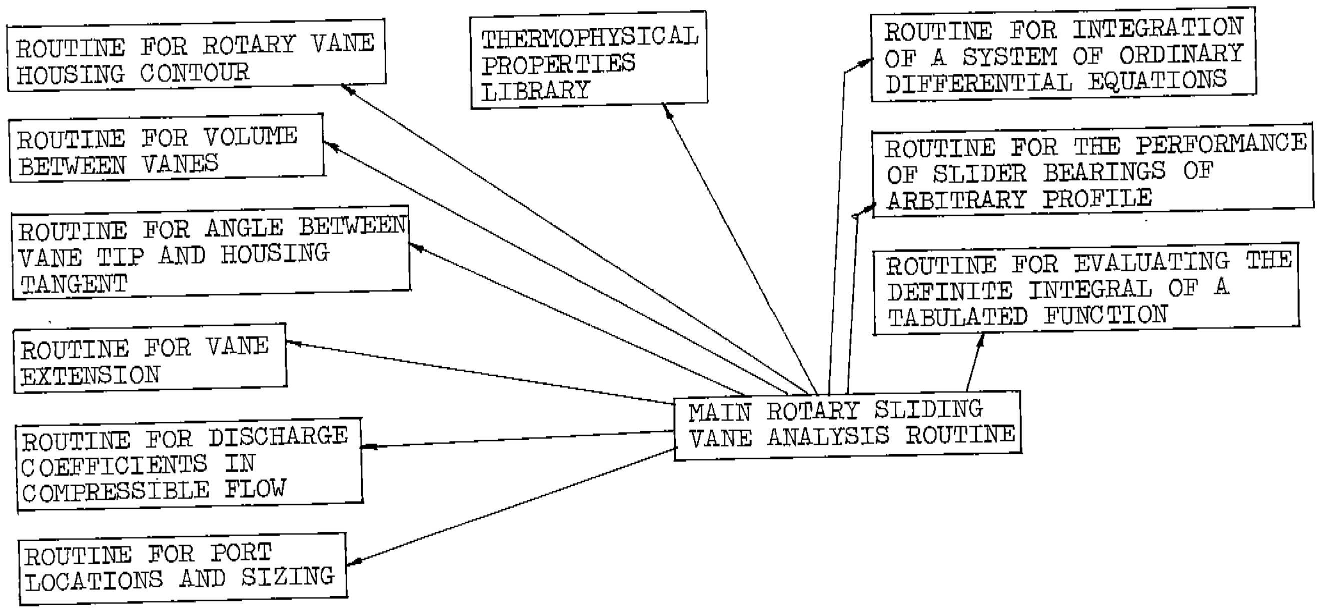
ROUTINE FOR ROTARY VANE HOUSING CONTOUR
THERMOPHYSICAL PROPERTIES LIBRARY
ROUTINE FOR INTEGRATION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
ROUTINE FOR VOLUME BETWEEN VANES
ROUTINE FOR THE PERFORMANCE OF SLIDER BEARINGS OF ARBITRARY PROFILE
ROUTINE FOR ANGLE BETWEEN VANE TIP AND HOUSING TANGENT
ROUTINE FOR EVALUATING THE DEFINITE INTEGRAL OF A TABULATED FUNCTION
ROUTINE FOR VANE EXTENSION
MAIN ROTARY SLIDING VANE ANALYSIS ROUTINE
ROUTINE FOR DISCHARGE COEFFICIENTS IN COMPRESSIBLE FLOW
ROUTINE FOR PORT LOCATIONS AND SIZING

FIG. 3: Partial Content of Rotary Vane Compressor Routine Group