Honeywell Bull

TIME-SHARING APPLICATIONS LIBRARY GUIDE VOLUME III-INDUSTRY

SERIES 600/6000

APPLICATIONS

DA 45, Rev. 2

Ref.: 19.53.108 A

Honeywell Bull

TIME-SHARING APPLICATIONS LIBRARY GUIDE VOLUME III-INDUSTRY

SERIES 600/6000

SUBJECT:

Descriptions (with Sample Problems and Solutions) of Time-Sharing Programs Related to Management Science, Engineering, Demonstrations, and Other Classifications.

SPECIAL INSTRUCTIONS:

This manual supersedes its previous edition dated December 1971 (Order Number DA45, Revision 1). It is one of four volumes; the others are the Series 600/6000 Time-Sharing Applications Library Guide, Volume I - Mathematics (Order Number DA43); Volume II - Statistics (Order Number DA44); and Volume IV - Business and Finance (Order Number DA46).

New programs are listed in the Preface.

DATE:

December 1972

ORDER NUMBER:

DA45, Rev. 2

Printed in France

Ref.: 19.53.108 A

PREFACE

This manual describes and discusses the usage of the industry time-sharing programs available with Series 600 and 6000 information processing systems. Each program description includes the purpose of the program; language in which it is written; method of approach, if applicable; instructions for use; restrictions if any; and many contain sample problems and solutions. In the sample solutions, all information typed by the user is underlined.

The instructions provided assume that the programs are available in the user master catalog LIBRARY and accessible with READ or EXECUTE permission. In the sample solution printouts, the programs had already been accessed using the GET command, and/or copied onto the current file using the OLD or LIB command.

Time-sharing programs for other classifications are also available from Honeywell under the following titles:

Series 600/6000 Time-Sharing Applications Library Guide, Volume I - Mathematics, Order Number DA43

Series 600/6000 Time-Sharing Applications Library Guide, Volume II - Statistics, Order Number DA44

Series 600/6000 Time-Sharing Applications Library Guide, Volume IV - Business and Finance, Order Number DA46

This manual is organized into sections by type as follows:

MS - Management Science and Optimization

EN - Engineering

GP - Geometric and Plotting

ED - Education and Tutorial

DE - Demonstration

UM - Utility and Miscellaneous

Each section is paginated with the 2-letter identifier shown above and a number.

File No. 16R3, 17R3, IJZ3

A complete listing of the programs in the library is available by listing the Library program, CATALOG. A copy of this listing follows the Contents.

ADDITIONAL PROGRAMS INCORPORATED IN THE DECEMBER 1972 EDITION

Series 600/6000 Time-Sharing

MANAGEMENT SCIENCE

- ASSIGNIT
- COEFS
- COMBI
- CSM
- GEOSIM
- GPROG
- GPROG-SO
- JSSIM
- OPTIM
- TRANSPO
- UNDEQ

ENGINEERING

- LFILTR
- LFLDAT
- LFLTIN
- PAVEIT

DEMONSTRATION

- AMAZE
- POPING
- PRIME
- XMAS

UTILITY AND MISCELLANEOUS

- ADAPTER
- CONCLUDE
- DESEQ
- REFORM
- FLINE

Series 600/6000 Time-Sharing Applications Library programs are available to users of the DATANETWORK service. Please contact your local Honeywell representative for further details.

CONTENTS

CONTENTS (cont)

CATALØG ØF SERIES 6000/600 T-S LIBRARY PRØGRAMS

FILE TYPE INDICATØR:

LANGUAGE (FIRST LETTER)		MØDE (FØLLØWING LETTERS)	
A	ALGØL	P	(ØR BLANK) PRØGRAM
B	BASIC	S	SUBRØUTINE(S)
C	CARDIN	F	FUNCTIØN(S)
D	DATABASIC	P-S	PRØGRAM WITH EXTRACTABLE SUBRØUTINE(S)
E	TEXT EDITØR	R	RELØCATABLE ØBJECT (C*)
F	FØRTRAN	H	SYSTEM LØADABLE ØBJECT (H*)

ALL FILES ARE SØURCE MØDE UNLESS ØTHERWISE INDICATED.

SUBJECTS	DØCUMENTATIØN MANUAL
MATHEMATICS (MA)	..............ØRDER # DA43
INTEGRATIØN	
DIFFERENTIATIØN, DIFFERENTIAL EQ.	
INTERPØLATIØN	
PØLYNØMIALS	
LINEAR EQUATIØNS	
MATRICES	
NØN-LINEAR EQUATIØNS	
SPECIAL FUNCTIØN EVALUATIØN	
LØGIC AND NUMBER THEØRY	
STATISTICS (ST)	..............ØRDER # DA44
CURVE FITTING AND REGRESSIØN	
ANALYSIS ØF VARIANCE	
PRØBABILITY DISTRIBUTIØNS	
CØNFIDENCE LIMITS	
HYPØTHESIS TESTING	
DESCRIPTIVE STATISTICS	
RANDØM NUMBER GENERATIØN	
MISCELLANEØUS STATISTICS	
BUSINESS AND FINANCE (BF)	..............ØRDER # DA46
MANAGEMENT SCIENCE AND ØPTIMIZATIØN (MS)	....ØRDER # DA45
LINEAR PRØGRAMMING	
INTEGER PRØGRAMING	
NØN-LINEAR ØPTIMIZATIØN	
NETWØRK ANALYSIS	
FØRECASTING	
SIMULATIØN	
ENGINEERING (EN)	
GEØMETRIC AND PLØTTING (GP)	
EDUCATIØN AND TUTØRIAL (ED)	
DEMØNSTRATIØN (DE)	
UTILITY AND MISCELLANEØUS (UM)	

THE DØCUMENTATIØN FØR THESE PRØGRAMS IS AVAILABLE IN FØUR MANUALS:
SEE ØRDER # DA43 FØR PRØGRAMS IN MATHEMATICS
ØRDER # DA44 FØR PRØGRAMS IN STATISTICS
ØRDER # DA46 FØR PRØGRAMS IN BUSINESS AND FINANCE
ØRDER # DA45 FØR PRØGRAMS IN ALL ØTHER CATEGØRIES.

SUBRØUTINES THAT ARE CALLED BY A PRØGRAM AND MUST BE EXECUTED WITH IT ARE LISTED IN BRACKETS AT THE END ØF THE DESCRIPTIØN.

THESE PRØGRAMS HAVE ALL BEEN REVIEWED AND TESTED BUT NØ RESPØNSIBILITY CAN BE ASSUMED.

********************MA--MATHEMATICS**

INTEGRATIØN

CLCINT	FF	INTEGRATIØN BY SIMPSØN'S RULE
FINT	FF	EVALUATE FØURIER INTEGRALS BY FILØN'S FØRMULA
GAHER	FF	GAUSS-HERMITE QUADRATURE
GALA	FF	GAUSS-LAGUERRE QUADRATURE
GAUSSN	FF	EVALUATE DEFINITE DØUBLE ØR TRIPLE INTEGRALS
GAUSSQ	FF	GAUSSIAN QUADRATURE
NCØATES	FP-S	NEWTØN-CØATES QUADRATURE
NUMINT	B	GAUSSIAN QUADRATURE
RØMBINT	FP-S	RØMBERG INTEGRATIØN
SPLINE	B	INTEGRATE TABULATED FUNCTIØN BY SPLINE FITS

DIFFERENTIATIØN, DIFFERENTIAL EQ.

AMPBX	FS	ADAMS-MØULTØN FØR 1ST-ØRDER DIFF. EQNS [RKPBX]
FDRVUL	FF	DIFFERENTIATE TABULATED FUNCTIØN, UNEQUAL SPACING
HDRVEB	FF	DIFFERENTIATE TABULATED FUNCTIØN, EQUAL SPACING
RKPBX	FS	RUNGE-KUTTA FØR 1ST-ØRDER DIFF. EQNS

INTERPØLATIØN

SPLINT	B	SPLINE INTERPØLATIØN
TNT1	FF	SINGLE LAGRANGIAN INTERPØLATIØN [TLU1]
TNT2	FF	DØUBLE LAGRANGIAN INTERPØLATIØN [TLU1]
TNT2A	FF	VARIABLE DØUBLE LINEAR INTERPØLATIØN [TLU1]

PØLYNØMIALS

BICØF	FS	CALCULATE BINØMIAL CØEFFICIENTS
CLPLY	FF	EVALUATE REAL PØLY AT REAL ARGUMENT
CPØLY	FS	FINDS ZERØES ØF A CØMPLEX PØLYNØMIAL
CPØLY-DR	FP	FINDS ZERØES ØF A CØMPLEX PØLYNØMIAL [CPØLY]
DVALG	FS	PØLYNØMIAL DIVISIØN
EUALG	FS	G.C.D. ØF TWØ PØLYNØMIALS [DVALG]
MTALG	FS	MULTIPLY PØLYNØMIALS
PLMLT	FS	REAL PØLY CØEFFICIENTS RECØNSTRUCTED FRØM REAL RØØTS
PØLRTS	FP	SØLUTIØN ØF PØLY BY BAIRSTØWS METHØD
PØLYC	FS	REAL PØLY CØEFFICIENTS RECØNSTRUCTED FRØM CØMPLEX RØØTS
PØLYV	FS	EVALUATE REAL PØLY AT CØMPLEX ARGUMENT
QUADEQ	B	SØLUTIØN TØ QUADRATIC EQUATIØNS
RØØTER	B	SØLUTIØN ØF PØLY BY BAIRSTØWS METHØD
ZCØP	FP	RØØTS ØF PØLYNØMIAL WITH CØMPLEX CØEFF.
ZCØP2	FS	RØØTS ØF PØLYNØMIAL WITH CØMPLEX CØEF. [ZCØP2]
ZØRP	FP	RØØTS ØF REAL PØLY
ZØRP2	FS	RØØTS ØF REAL PØLY

LINEAR EQUATIØNS

GJSIMEQ	FS	SØLVE LINEAR SYSTEMS BY GAUSS-JØRDAN
GSEIDEL	FP-S	SØLVE LINEAR SYSTEMS BY GAUSS-SEIDEL
LINEQ	FS	SØLVE LINEAR SYSTEMS BY GAUSSIAN ELIMINATIØN
LINSR	FP	SØLVE LINEAR SYSTEMS BY GAUSSIAN ELIMINATIØN [LINEQ]
SIMEQN	B	SØLVE LINEAR SYSTEMS BY MATRIX INVERSIØN

MATRICES

DETE	FF	EVALUATE DETERMINANT ØF REAL MATRIX
DØMEIG	FP-S	DØMINANT EIGENVALUES ØF REAL MATRIX
EIG1	FS	EIGENVALUES ØF SYM MATRIX BY JACØBI METHØD
EIGNHC	FS	EIGENVALUES & VECTØRS ØF CØMPLEX NØN-HERMITIAN MATRICES
EIGNSR	FS	EIGENVALUES & VECTØRS ØF REAL NØN-SYMMETRIC MATRICES
EIGSR	FP	EIGENVALUES AND VECTØRS ØF REAL SYM. MATRIX [EIG1]
LINSD	FS	SØLVE LIN. SYS. W/ SYMMETRIC DØUBLE PREC. CØEF. MATRIX
LINSS	FS	SØLVE LIN. SYS. W/ SYMMETRIC SINGLE PREC. CØEF. MATRIX
MTINV	FS	MATRIX INVERSIØN BY PIVØTS
MTMPY	FS	MATRIX MULTIPLICATIØN
MTRAN	FS	TRANSPØSE A MATRIX
SPEIG1	FS	SPECIAL EIGEN PRØBLEMS [EIG1]
SYMEIG	FP	EIGENVALUES ØF SYM MATRIX BY JACØBI METHØD

NØN-LINEAR EQUATIØNS

BRØWN	FS	SØLN ØF SIMULTANEØUS SYSTEMS BY BRØWN METHØD
SECANT	FS	SØLN ØF SIMULTANEØUS SYSTEMS BY SECANT METHØD [MTINV]
SØLN	FF	ZERØ ØF AN ARBITRARY FUNCTIØN
ZERØES	B	ZERØ,MAX,MIN ØF FUNCTIØN

SPECIAL FUNCTIØN EVALUATIØN

ARCTAN	FF	ARCTANGENT IN RADIANS ØF Y/X
BESL	FS	BESSEL FUNCTIØN [GAMF]
CØMP1	FF	EVALUATES REAL HYPERBØLIC TRIG FUNCTIØNS
CØMP2	FS	CØMPLEX MULT. AND DIVISIØN
CØMP3	FS	EVALUATES VARIØUS FUNCTIØNS FØR CØMPLEX ARGUMENT [CØMP2]
ERRF	FF	ERRØR FUNCTIØN
ERRINV	FF	INVERSE ERRØR FUNCTIØN
FRESNL	FS	EVALUATES FRESNAL INTEGRALS
GAMF	FF	GAMMA FUNCTIØN
JACELF	FS	EVALUATES JACØBIAN ELLIPTIC FUNCTIØNS SN, CN, DN
ØRTHP	FF	EVALUATE ØRTHØGØNAL PØLYNØMIALS
STIRLING	FP-S	N FACTØRIAL BY STIRLINGS APPRØXIMATIØN
TMFCEV	B	EVALUATE DAMPED ØR UNDAMPED FØURIER SERIES

LØGIC AND NUMBER THEØRY

4SQRS	B	WRITES INTEGERS AS SUM ØF SQUARES ØF FØUR INTEGERS
BASE	FP	CØNVERTS NUMBERS FRØM ØNE BASE TØ ANØTHER
CØNCLUDE	B	DETERMINES LØGICAL CØNCLUSIØNS FRØM PRØPØSITIØNAL LØGIC
GCDN	FS	G.C.D. ØF N INTEGERS

********************ST--STATISTICS***

CURVE FITTING AND REGRESSIØN

CFIT	FP	LEAST SQRS. PØLY. WITH RESTRAINTS
CURFIT	B	FITS SIX DIFFERENT CURVES BY LEAST SQRS
FØRIR	FP	LEAST SQUARES ESTIMATE ØF FINITE FØURIER SERIES MØDEL
FØURIER	B	CØEFF ØF FØURIER SERIES TØ APPRØX A FUNCTIØN
LINEFIT	FS	LEAST SQRS LINE
LINREG	B	LST.SQRS. BY LINEAR, EXPØNENTIAL, ØR PØWER FUNCTIØN
LSPCFP	FP	LEAST SQRS PØLYNØMIAL FIT
LSQMM	FS	GENERALIZED PØLY FIT BY LEAST SQRS ØR MIN-MAX
MREG1	FP	MULTIPLE LINEAR REGRESSIØN
MULFIT	B	MULTIPLE LINEAR FIT WITH TRANSFØRMATIØNS
ØRPØL	FP	LEAST SQRS FIT WITH ØRTHØGØNAL PØLYS
PØLFIT	B	LEAST SQRS PØLYNØMIAL FIT
PØLFT	FP	LEAST SQRS PØLYNØMIAL FIT
SMLRP	FP	MULTIPLE LINEAR REGRESSIØN
SMLRPØBJ	FHP	SYSTEM LØADABLE FILE FØR SMLRP
STAT20	B	EFFRØYMSØN'S MULTIPLE LINEAR REGRESSIØN ALGØRITHM
STAT21	B	CØMPUTES MULTIPLE LINEAR REGRESSIØNS

ANALYSIS ØF VARIANCE

ANØVA	FP	ØNE ØR TWØ WAY ANALYSIS ØF VARIANCE
ANVA1	FP	ØNEWAY ANALYSIS ØF VARIANCE
ANVA3	FP	THREE WAY ANALYSIS ØF VARIANCE
ANVA5	FP	MULTIPLE VARIANCE ANALYSIS
KRUWAL	FP	KRUSKAL-WALLIS 2-WAY VARIANCE [XINGAM]
ØNEWAY	B	ØNEWAY ANALYSIS ØF VARIANCE
STAT13	B	ANALYSIS ØF VARIANCE TABLE, 1-WAY RANDØM DESIGN
STAT14	B	ANALYSIS ØF VARIANCE TABLE FØR RANDØMIZED BLØCK DESIGN
STAT15	B	ANALYSIS ØF VARIANCE TABLE FØR SIMPLE LATIN-SQ DESIGN
STAT16	B	ANALYSIS ØF VARIANCE TABLE, GRAECØ-LATIN SQUARE DESIGN
STAT17	B	ANØVA TABLE ØF BALANCED INCØMPLETE BLØCK DESIGN
STAT18	B	ANALYSIS ØF VARIANCE TABLE, YØUDEN SQUARE DESIGN
STAT33	B	ANALYSIS ØF VARIANCE TABLE, 1-WAY RANDØM DESIGN

```
***PRØBABILITY DISTRIBUTIØNS***
ANPF        FF    NØRMAL PRØBABILITY FUNCTIØN [ERRF]
BETA        FF    BETA DISTRIBUTIØN
BINDIS      B     BINØMIAL PRØBABILITIES
EXPLIM      B     EXPØNENTIAL DISTRIBUTIØNS
PØISØN      FF    PØISSØN DISTRIBUTIØN FUNCTIØN
PRØBC       FP    PRØBABLITIES ØF CØMBINATIØNS ØF RANDØM VARIABLES
PRØVAR      B     NØRMAL AND T-DISTRIBUTIØN
TDIST       FF    T-DISTRIBUTIØN [BETA]
XINGAM      FF    INCØMPLETE GAMA FUNCTIØN

***CØNFIDENCE LIMITS***
BAYES       B     DIFFERENCE ØF MEANS IN NØN-EQUAL VARIANCE
BICØNF      B     CØNF. LIMITS FØR PØPULATIØN PRØPØRTIØN (BINØMIAL)
BINØM       FP    BINØMIAL PRØBABILITIES AND CØNFIDENCE BANDS
CØLINR      B     CØNFIDENCE LIMITS ØN LINEAR REGRESSIØNS
CØNBIN      B     CØNF. LIMITS FØR PØPULATIØN PRØPØRTIØN (NØRMAL)
CØNDIF      B     DIFFERENCE ØF MEANS IN EQUAL VARIANCE
CØNLIM      B     CØNF. LIMITS FØR A SAMPLE MEAN
STAT05      B     CØNFIDENCE INTERVAL FØR MEAN BY SIGN TEST
STAT06      B     CØNFIDENCE LIMITS, WILCØXØN SIGNED RANK SUM TEST

***HYPØTHESIS TESTING***
BITEST      B     TEST ØF BINØMIAL PRØPØRTIØNS
CHISQR      FS    CHI-SQUARE CALCULATIØNS
CØRREL      FP    CØNTINGENCY CØEFFICIENT [XINGAM]
CØRRL2      FP    CØRRELATIØN CØEFFICIENT [TDIST;BETA]
KØKØ        FP    KØLMØGØRØV-SMIRNØV TWØ SAMPLE TEST [XINGAM]
SEVPRØ      B     CHI-SQUARE
STAT01      B     MEAN,STD ØF MEAN, ... , T-RATIØ,2 GRØUPS, PAIRED
STAT02      B     MEANS, VARIANCES, AND T-RATIØ 2 GRØUPS, UNPAIRED DATA
STAT04      B     CHI-SQUARE AND PRØBABILITIES, 2X2 TABLES
STAT08      B     CØMPARES TWØ GRØUPS ØF DATA USING THE MEDIAN TEST
STAT09      B     CØMPARE 2 DATA GRØUPS,MANN-WHITNEY 2-SAMPLE RANK TEST
STAT11      B     SPEARMAN RANK CØRRELATIØN CØEF. FØR 2 SERIES ØF DATA
STAT12      B     CØMPUTES CØRRELATIØN MATRIX FØR N SERIES ØF DATA
TAU         FP    KENDALL-RANK CØRRELATIØN

***DESCRIPTIVE STATISTICS***
MANDSD      B     FIND MEAN, VARIANCE, STD
STAT        FP    FIND SEVERAL STATISTICS FØR SAMPLE DATA [ANPF;ERRF]
STATAN      B     FIND VARIØUS STATISTICAL MEASURES
TESTUD      B     SAMPLE STATISTICS
UNISTA      B     DESCRIPTIØN ØF UNI-VARIANT DATA

***RANDØM NUMBER GENERATIØN***
FLATSØRC    C     CARDIN SØURCE FILE FØR FLAT
FLAT        FRF   UNIFØRM RANDØM NUMBER GENERATØR
RANDX       FF    RANDØM #'S, UNIFØRM DIST. BETWEEN 0 AND 1
RNDNRM      FF    CALCULATES NØRMAL RANDØM NUM. [FLAT]
UNIFM       FRF   UNIFØRM RANDØM NUMBER GENERATØR
UNIFMSØR    C     CARDIN SØURCE FILE FØR UNIFM
URAN        FRF   UNIFØRM RANDØM NUMBER GENERATØR
URANSØRC    C     CARDIN SØURCE FILE FØR URAN
XNØR1       FF    NØRMAL RANDØM NUMBERS, VARIABLE MEAN, STD [RANDX]
XNØRM       FF    NØRMAL RANDØM NUMBERS, MEAN 0, STD 1. [RANDX]

***MISCELLANEØUS STATISTICS***
FACTAN      FP    FACTØR ANALYSIS
STADES            EXPLANATIØN ØF CØLINR,CURFIT,MULFIT,UNISTA
```

*******************BF--BUSINESS AND FINANCE****************************

ANNUIT	B	ANNUITIES,LØANS,MØRTGAGES
BLDGCØST	B	ANALYZE BUILDING CØSTS
BØNDATA	B	ANALYSIS ØF A BØND INVESTMENT PØRTFØLIØ
BØNDPR	B	CØMPUTES PRICE AND ACCRUED INTEREST ØF A BØND
BØNDSW	B	CALCULATES THE EFFECT ØF A BØND SWITCH
BØNDYD	B	CØMPUTES BØND YIELDS
CASHFLØW	B	PREDICTS NEXT YEARS CASH FLØW
DEPREC	B	CALCULATES DEPRECIATIØN BY FØUR METHØDS
INSTLØ	B	CALCULATES MØNTHLY PAYMENT SCHEDULE ØN INSTALLMENT LØAN
INVANL	FP	RETURN ØN INVESTMENT ANALYSIS
LESSEE	B	CØMPARES A LEASE WITH PURCHASE ØF EQUIPMENT
LESSIM	B	SIMULATES LESSØR'S CASH FLØW AND RATE ØF RETURN
LESSØR	B	CALCULATES THE LESSØRS CASH FLØW & RATE ØF RETURN
MAKE-BUY	B	TØ MAKE ØR TØ BUY DECISIØNS
MGSIM	FHP	SIMULATES CØMPETITIVE INTERACTIØN ØF CØMPANIES
MGSIM-CS	FRP	ØBJECT DECKS FØR MGSIM
MGSIM-IN		ØN LINE INSTRUCTIØNS FØR MGSIM
MØRTCST	B	MØRTGAGE SCHEDULE FØR VARIØUS TERMS
MØRTGAGE	FP	CALCULATES A MØRTGAGE REPAYMENT SCHEDULE
RETURN	B	CØMPUTES ANNUAL RETURNS FØR A SECURITY FRØM ANNUAL DATA
SALDATA	B	CØMPUTES PRØFITABILITY ØF DEPARTMENTS ØF A FIRM
SAVING	B	SAVINGS PLAN CALCULATIØNS
SMLBUS	B	PAYMENT SCHEDULES FØR A SMALL BUSINESS ADMST. LØAN
TRUINT	B	INTEREST RATE CALCULATIØNS

*******************MS--MANAGEMENT SCIENCE AND ØPTIMIZATIØN************

LINEAR PRØGRAMMING

ASSIGNIT	B	THE ASSIGNMENT PRØBLEM
LINPRØ	B	LINEAR PRØGRAMMING
LNPRØG	FP	LINEAR PRØGRAMMING
TRANSPØ	B	THE TRANSPØRTATIØN PRØBLEM
UNDEQ	FS	FINDS A SØLUTIØN FØR AN UNDERDETERMINED LINEAR SYSTEM

INTEGER PRØGRAMMING

INTØ1	FP	ZIØNTS' MØDIFICATIØN ØF BALAS' ZERØ-ØNE ALGØRITHM
INTLP	FP	GØMØRY'S PURE AND MIXED INTEGER PRØGRAMMING

NØN-LINEAR ØPTIMIZATIØN

CSM	FS	ØPTIMIZE A LINEARLY CØNSTRAINED CØNVEX FUNCTIØN[UNDEQ]
DAVIDØN	B	DAVIDØN'S UNCØNSTRAINED ØPTIMIZATIØN
GEØSIM	B	HEURISTIC SCHEDULING ØF N JØBS IN A M MACHINE SHØP
GPRØG	FHP	SØLVES GEØMETRIC PRØGRAMMING PRØBLEMS
GPRØG-SØ	C	CARDIN SØURCE FILE FØR GPRØG [UNDEQ;CSM]
JSSIM	B	SCHEDULES N JØBS IN A SHØP WITH M MACHINES
LØGIC3	FP	UNCØNSTRAINED ØPTIMIZATIØN
MAXØPT	FP	UNCØNSTRAINED ØPTIMIZATIØN

NETWØRK ANALYSIS

CPM	FP	CRITICAL PATH METHØD
KILTER	FP	'ØUT ØF KILTER' ALGØRITHM (MINIMUM CØST CIRCULATIØN)
MAXFLØW	FP	MAXIMUM FLØW THRU NETWØRK
PERT	B	SIMPLE ANALYSIS ØF A PERT NETWØRK
SHØRTEST	FP	SHØRTEST PATH - MIN SPANNING TREE

FØRECASTING

CØEFS	B	DETERMINE SEASØNAL CØEFFICIENTS ØN TWØ CYCLES
CØMBI	B	DETERMINES ECØNØMIC ØRDER QUANITY FØR INVENTØRY ITEMS
ØPTIM	F	ØPTIMUM SERVICE LEVEL FØR ØNE INVENTØRY ITEM
TCAST	FP	TIME SERIES FØRECASTING [TCAST1;TCAST2]
TCAST1	FH	ØVERLAY MØDULE ØF TCAST
TCAST	FHP	TIME SERIES FØRECASTING
TCAST2	FH	ØVERLAY MØDULE ØF TCAST
TCAST1	FH	ØVERLAY MØDULE ØF TCAST
SMØØTH	FS	TRIPLE SMØØTHING ØF A TIME SERIES

SIMULATIØN

GASPDATA	E	DATA FILE FØR SAMPLE PRØGRAM GASPSAMP
GASPIIA	FS	'GASP' SIMULATIØN SYSTEM
GASPSAMP	FP	SAMPLE PRØGRAM FØR GASPIIA [GASPIIA;GASPDATA]

********************EN--ENGINEERING***

ACNET	FP	FREQUENCY RESPØNSE ØF A LINEAR CIRCUIT
BEMDES	B	STEEL BEAM SELECTIØN
GCVSIZ	B	GAS CØNTRØL VALVE CØEFF.
LCVSIC	B	LIQUID CØNTRØL VALVE CØEFF.
LFILTR	B	SYNTHESIZES ACTIVE LØW-PASS FILTERS [LFLDAT]
LFLDAT		DATA FØR LFILTR
LFLTIN		INSTRUCTIØNS FØR LFILTR
LPFILT	B	DESIGN LØW PASS FILTERS
NLNET	FP	GENERAL STEADY-STATE CIRCUIT ANALYSIS
ØTTØ	B	ØTTØ CYCLE ØF ENGINE
PAVEIT	B	CALCULATES $ CØST AND TØNS ØF MATERIAL TØ PAVE A RØAD
PVT	FP	FINDS MØLAR VØLUME ØF A GAS GIVEN TEMPERATURE AND PRES.
SCVSIZ	B	STEAM CØNTRØL VALVE CØEFF.
SECAP	B	STEEL SECTIØN CAPACITIES

********************GP-GEØMETRIC AND PLØTTING****************************

CIRCLE	B	DIVIDES A CIRCLE INTØ N EQUAL PARTS
PLØT	FS	PLØTS UP TØ 9 CURVES SIMULTANEØUSLY
PLØTTØ	B	SIMULTANEØUSLY PLØTS 1 TØ 6 FUNCTIØNS
PØLPLØ	FP	PLØTS EQNS IN PØLAR CØØRDINATES
SPHERE	B	SØLVES ANY SPHERICAL TRIANGLE
TRIANG	B	SØLVES FØR ALL PARTS ØF ANY TRIANGLE
TWØPLØ	B	SIMULTANEØUSLY PLØTS 2 FUNCTIØNS
XYPLØT	B	PLØTS SINGLE-VALVED FUNCTIØNS

********************ED--EDUCATIØN AND TUTØRIAL***************************

DRIVES	FHP	DRIVER FØR EXPER, A CØMPUTER ASSISTED INST. LANG.
EXPERN	E	EXPER TUTØRIALS IN EXPER (N=1 TØ 5) [PREPRS;DRIVES]
PREPRS	FHP	PREPRØCESSØR FØR EXPER, A CØMPUTER ASSISTED INST. LANG.

********************DE--DEMØNSTRATIØN**************************************

AMAZE	B	CØNSTRUCTS MAZES - EACH UNIQUE
BLKJAK	B	THE CØMPUTER DEALS BLACKJACK
PØPING	B	PØPULATIØN PRØJECTIØNS FØR AN AREA
PRIME	B	PRIME FACTØRIZATIØN ØF A NUMBER
XMAS	B	A HØLIDAY SING-ALØNG, CHRISTMAS CARD AND GREETINGS

********************UM--UTILITY AND MISCELLANEØUS*************************

ADATER	FP-S	A CALENDER DATING RØUTINE
CATALØG	E	CATALØG ØF SERIES 6000/600 T/S LIBRARY (THIS FILE)
CØNVRT	B	CØNVERTS MEASUREMENTS FRØM ØNE SCALE TØ ANØTHER
DBLSØRT	FS	SØRT TWØ ARRAYS
DESEQ	FP	STRIPS LINE SEQUENCE NUMBERS FRØM A FILE
REFØRM	FP	REFØRMATS A 'NFØRM' FØRTRAN SØURCE FILE TØ 'FØRM'
RLINE	FS	READS LINE, ØPTIØNALLY STRIPS LINE # & CØUNTS ENTRIES
SGLSØRT	FS	SØRT AN ARRAY
TLU1	FS	TABLE SEARCH
TPLSØRT	FS	SØRT THREE ARRAYS

END ØF CATALØG

This BASIC program uses the algorithm described by R. Silver in Communications of the ACM (Nov. 1960, pp. 605-606) to solve the classic assignment problem to compute a cost for the assignment.

The assignment problem may be formulated as follows: Given an n by n matrix (d_{ij}) of real numbers, find a permutation x of the integers 1 ... n that minimized $\sum_{i=1}^{n} d_{ixi}$. That is, for any permutation y of the integers 1 ... n, we have $\sum_{i=1}^{n} d_{ixi} = \sum_{i=1}^{n} d_{iyi}$.

If a permutation x has the property that $\sum_{i=1}^{n} d_{ixi} = \sum_{i=1}^{n} d_{iyi}$, for any permutation y, then we say that x minimized (d_{ij}).

The Assignment Problem is essentially a specialization of the Transportation Problem, and uses the same theory.

1. Assignment implies n resources to be assigned to n locations with a minimum cost answer desired.
2. Conversely n men may be rated (percentage of efficiency) to each of n jobs. In this case the answer desired is the maximum efficient assignment of man to job. This latter result is obtained in this version of the Assignment Problem by constructing the matrix with negative figures.

1\.

LOCATION / RESOURCE	1	2	3
1	150	130	160
2	110	90	135
3	120	150	95

2\.

JOB / MAN	1	2	3
1	-95	-92	-89
2	-93	-97	-98
3	-90	-91	-99

INSTRUCTIONS

Sample data is presently included in the program. To remove this data, use a DELETE statement. Then enter the number of matrix elements in line 2000 and the matrix itself on the following lines. Then type RUN. Further instructions can be obtained by typing *LIST 10 - 90

SAMPLE PROBLEM

Sample data showing the assignment matrix for a typical problem is presently included in the program. Solve the classic assignment problem and compute a cost for the assignment based on this sample data (see lines 2010 to 2050 in the sample solution).

SAMPLE SOLUTION

```
LIST 2000

2000 DATA 5
2010 DATA 144,74,46,81,68
2020 DATA 77,27,13,38,28
2030 DATA 107,55,34,60,47
2040 DATA 91,49,31,52,43
2050 DATA 106,38,19,53,44
2060 END



READY

*RUN


                THE ASSIGNMENT IS

MØDULE\LØCATIØN
           1       2       3       4       5

     1     0       0       1       0       0
     2     0       0       0       1       0
     3     0       0       0       0       1
     4     1       0       0       0       0
     5     0       1       0       0       0

THE CØST ØF THIS ASSIGNMENT IS  260
```

This BASIC program determines seasonal coefficients of an observation series of two cycles. Given the observations for each period of a cycle containing a minimum of two cycles, the program calculates the seasonal coefficients explaining the deviation of the observations from the mean and the trend. Each cycle must contain the same number of periods. The maximum number of periods per cycle is 13. The observations may be entered in either ascending order (i.e., January, February ...) or in descending order (i.e., December, November ...).

INSTRUCTIONS

Enter the data in DATA statements using line numbers 801 to 899. On the first line enter:

- number of periods per cycle (maximum of 13)
- period number for the first observation
- sorting code: 1 = observations in ascending order
2 = observations in descending order

Enter the observations for the cycles on the following lines. Then type RUN.

REFERENCES

This program is an adaptation of modules from the AIMS system. Further information on AIMS can be obtained from the following:

AIMS - Autoadaptive Inventory Management System Time-Sharing Demonstration Programs, Ref. #00.19.102A, Honeywell Bull Company; Paris, France

The AIMS System, Ref. #00.11.072A, Honeywell Bull Company; Paris, France

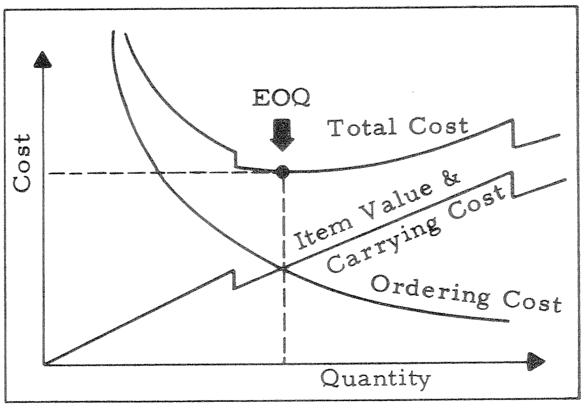
SAMPLE PROBLEM

Given an observation series for two years. The period is one month (i.e. number of periods per cycle = 12). The first observation is in January (i.e. period number for the first observation = 1). The observations are sorted in ascending order, i.e. January, February ... (i.e. sorting code of observations = 1). Enter the data and find the seasonal coefficients.

SAMPLE SOLUTION

```
*801 DATA 12,1,1
*802 DATA 469,193,191,145,276,417,675,549,816,889,563,510
*803 DATA 413,183,131,185,180,240,667,653,538,827,542,478
*RUN


CØEFS -- SEASØNAL CØEFFICIENTS


    CA :  SEASØNAL CØEFFICIENTS FØR CYCLE 1
    CB :  SEASØNAL CØFEFFICIENTS FØR CYCLE 2
    CS :  SEASØNAL CØEFFICIENTS FØR BØTH CYCLES

PERIØD  1     2     3     4     5     6     7     8     9     10    11    12

*  2.00                                                         B
*  1.90                                                        SAS
*  1.80
*  1.70                                                   A
*  1.60                                                  SSS
*  1.50                                      SBS    B     B
*  1.40                                       A
*  1.30                                            SSS               SBS
*  1.20                                             A                 A     B
*  1.10                                                                    SAS
*  1.00------------------------------------------------------------------------
*   .90 SAB                             A
*   .80
*   .70                                SSS
*   .60                          A      B
*   .50                         SSS
*   .40      SAB   A    SBS      B
*   .30           SBS    A
*   .20

*  SEASØNAL CØEFFICIENTS  *

*  CA    .94   .39   .39   .30   .57   .87 1.41 1.16 1.74 1.91 1.22 1.11
*  CB    .91   .41   .29   .42   .41   .55 1.55 1.53 1.51 1.97 1.30 1.15
*  CS    .92   .40   .34   .35   .49   .72 1.48 1.33 1.63 1.94 1.25 1.14

PERIØD  1     2     3     4     5     6     7     8     9     10    11    12



READY
*
```

This BASIC program determines the economic order quantity for a group of items under different supplier discounts.



Combined Orders EOQ

The choice of order quantity or order frequency is of great importance for the size of the working stock and thus for the investment of company money. If small quantities are ordered by a large number of orders, the working stock and thus the investment in stock is moderate, but the administrative cost related to the order procedure will be high. On the other hand, if large quantities are ordered in a limited number of orders, then the working stock and thus the investment will be high.

The problem is solved by determining the economic order quantity, also called the EOQ, by which the total annual cost is minimum.

The carrying cost is the cost of holding goods in inventory. It is expressed in percent of item price per year and comprises:

- cost of having capital tied up in inventory,
- cost of storage facilities, and
- taxes and insurance.

The annual carrying cost per item is normally between 8 and 30 percent of the item value.

The ordering cost is the cost of processing the replenishment order plus the cost of receiving the shipment. For users of COMBI, the ordering cost is normally between $5 and $125.

The supplier may offer quantity discounts in proportion to the total order volume, which may comprise different items. A unit index is attached to each item, and the total order volume is calculated as the sum of the number of each item, multiplied by the corresponding unit index.

Consider the following example:

Item	Unit Index	Item Price	Qty. ordered
Whisky 1/2 bottle	0.5	4	22
Whiskey 1/1 bottle	1.0	6	15
Whiskey King size	1.8	8	10

The total order volume:

0.5 x 22 + 1.0 x 15 + 1.8 x 10 = 44 units

And the order amount without discounts:

22 x 4 + 15 x 6 + 10 x 8 = 258

Let us assume, that the supplier offers a discount of 10% starting with 40 units. The amount to pay is then

0.9 x 258 = 232.20

REFERENCES

This program is an adaptation of modules from the AIMS system. Further information on AIMS can be obtained from the following:

AIMS - Autoadaptive Inventory Management System Time-Sharing Demonstration Programs, Ref. #00.19.102A, Honeywell Bull Company; Paris, France

The AIMS System, Ref. #00.11.072A, Honeywell Bull Company; Paris, France

INSTRUCTIONS

Enter the following information in DATA statements, using lines 1-999.

On the first line enter:

- number of items
- number of supplier discounts (0-8)
- carrying cost as percentage of item price (normally between 8 and 30 percent)
- ordering cost (normally between $5 and $125)

Then, on the following line for each item, enter:

- unit index
- item value
- annual demand

Finally, for each supplier discount (if any):

- minimum number limit
- discount in percent

NOTE: The program can be used to calculate the "simple" EOQ (economic order quantity) for a single item without supplier discounts. In this case, you will have:

number of items	= 1
number of supplier discounts	= 0
unit index for the item	= 1

SAMPLE PROBLEM

Calculate the EOQ for three items, when the carrying cost is 20% and the ordering cost $25. Unit index, item value and annual demand is as follows:

Item No.	Unit Index	Item Value	Annual Demand
1	1.0	22	76
2	0.5	40	48
3	3.0	68	20

For 32 units the supplier offers a discount of 20%.

For 80 units the supplier offers a discount of 33%.

SAMPLE SOLUTION

```
*100 DATA 3,2,20,25
*101 DATA 1,22,76
*102 DATA 0.5,40,48
*103 DATA 3,68,20
*111 DATA 32,20
*112 DATA 80,33
*RUN


THE CARRYING COST IS  20 PER CENT OF ITEM VALUE.
THE ORDERING COST IS 25.00 DOLLARS

: ITEM : UNIT  :  ITEM    : ANNUAL :
:  NO  : INDEX :  VALUE   : DEMAND :
:======:=======:==========:========:
:   1  :  1.00 :   22.00  :   76   :
:   2  :   .50 :   40.00  :   48   :
:   3  :  3.00 :   68.00  :   20   :

FROM  32 UNITS THE SUPPLIER OFFERS A DISCOUNT OF 20 PER CENT.
FROM  80 UNITS THE SUPPLIER OFFERS A DISCOUNT OF 33 PER CENT.

IN THE INTERVAL FROM   0 TO  32
 THERE IS NO EOQ.

IN THE INTERVAL FROM  32 TO  80
 THE ECONOMIC ORDER QUANTITY IS  40.19 INDEX UNITS
   CORRESPONDING TO A TOTAL COST OF 199.04 DOLLARS
   AND AN ORDER EVERY 13.1 WEEK.

IN THE INTERVAL FROM  80 TO 160
 THE ECONOMIC ORDER QUANTITY IS  80.00 INDEX UNITS
   CORRESPONDING TO A TOTAL COST OF 215.89 DOLLARS
   AND AN ORDER EVERY 26.0 WEEK.

HORIZONTAL: ORDER FREQUENCY   (IN WEEKS).
VERTICAL  : TOTAL ANNUAL COST (IN DOLLARS)

      675 : *
      650 :
      625 :
      600 :
      575 :
      550 :
      525 :
      500 :
      475 :
      450 :  *
      425 :
      400 :
      375 :   *
      350 :                                                ***-
      325 :                                            ****
      300 :    *                                  *****
      275 :     *                             ****
      250 :      **                *-    *****
      225 :        *-        ****** E****
      200 :         ***E*****
      175 :
==========:----------------------------------------------------
           1          13           26           39           52
```

```
DØ YOU WANT TØ ENLARGE THE CURVE - ANSWER YES ØR NØ ?YES

HØRIZØNTAL: ØRDER FREQUENCY   (IN WEEKS).
VERTICAL   : TØTAL ANNUAL CØST (IN DØLLARS)

      248 :                         -
      245 :                         I     *
      243 :                        *I
      240 :                         I    *
      238 :       *               * I
      235 :                         I   *
      233 :                      *  I
      230 :        *                I  *
      228 :                     *   I
      225 :         -               I *
      223 :         I          *    I
      220 :         I               I*
      218 :         I         *     I
      215 :         I               E
      213 :         I        *
      210 :         I       *
      208 :         I
      205 :         *      *
      203 :          *    *
      200 :           *E**
      198 :
==========:----------------------------------------------------
           1          13           26           39           52

HØW MANY WEEKS SHALL YØUR ØRDER CØMPRISE ?13

13.0 WEEKS CØRRESPØNDS TØ THE FØLLØWING ØRDER:
       19.00  ØF ITEM NUMBER   1
       12.00  ØF ITEM NUMBER   2
        5.00  ØF ITEM NUMBER   3


READY
*
```

This FORTRAN program will compute the critical paths of a project network model for the static case. Given the identification, duration, and cost for each project activity, the program computes: the direct project cost and total duration; the earliest (ES) and latest (LS) permissible start and the earliest (EF) and latest (LF) finish times for each activity, identification (**) of the activities on the critical path, the total float (TF) and free float (FF) for each activity.

INSTRUCTIONS

On execution, the program will ask for the name of the data file. All data should be entered in lines with line numbers. The data can be divided into two sections, activity data and if desired, calendar dating information. The following rules apply:

1. The first line is an alphanumeric problem identification.
2. The activity data follows with one activity per line in free format as below:

   line number, tail (I), head (J), duration, cost

3. If calendar dating is desired, the last activity line should have I, J, duration, and cost = 0.
4. For calendar dating the first line following the activities is the starting date in the following format:

   line number, month number, day of month, day of week number

5. Following the starting date, the nonworking days of the week and the holidays are entered for each year the project is expected to cover. This data is entered one item per line with line numbers in the following order:

   last two digits of year

   a nonworking day of week

   any other nonworking days of week

   -1 (end of nonworking days)

   holiday month number, day

   any other holiday months, days

   -1, -1 (end of holidays)

   last two digits of next year

   Repeat nonworking days of week and holiday months and days until entering:

   last holiday of last year

6. If the starting month number is negative, the calendar dating is ignored.

The program will test for the following errors and issue error messages:

1. I-J errors -- I or J greater than 999 or I=J.
2. Problem too big for allocated storage.

3. Multiple start or finish nodes.
4. A loop in the network. In this case, the activity identified will either be on the loop or on a sequence of jobs that passes through a node of the loop.

RESTRICTIONS

The maximum problem size is limited by:

2* (highest numbered node) + (# activities) + 2≤ 3000

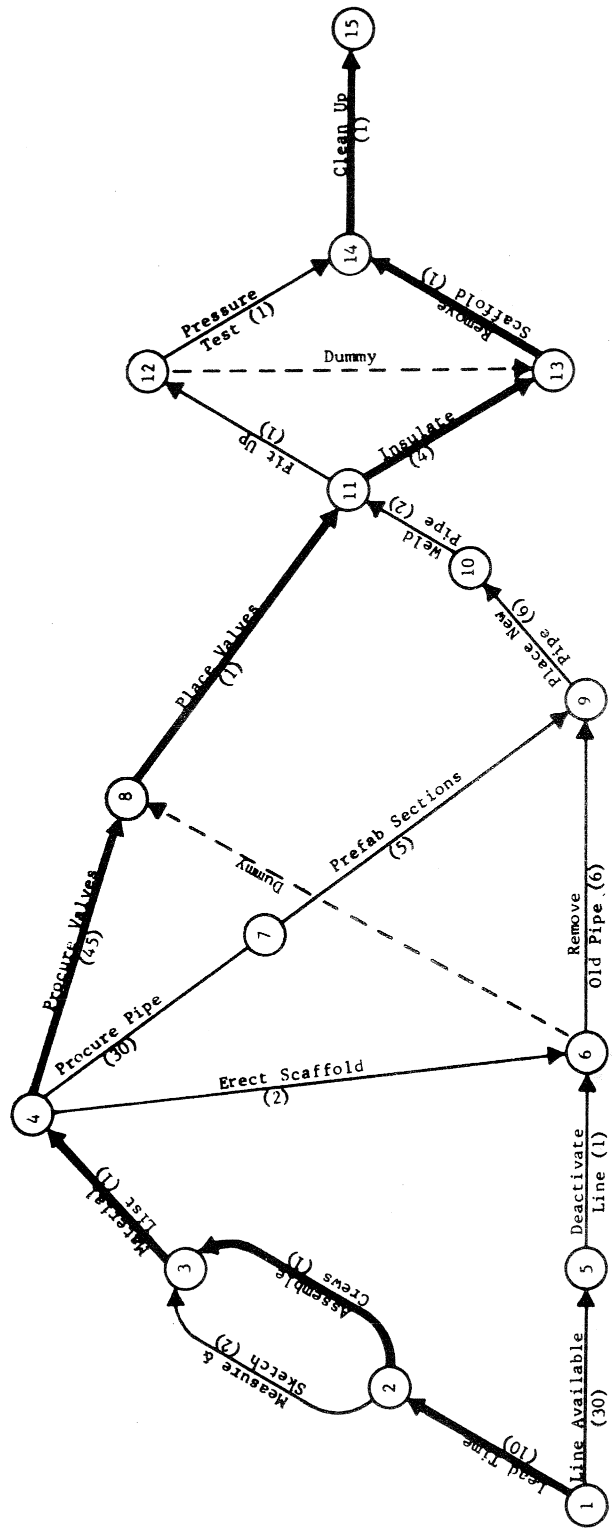
The numbering conventions for the activities are:

1. The nodes (events) of the arrow diagram may be numbered randomly with any number from 1 to 999.
2. The head of the job arrow (J) does not have to be numbered larger than the tail (I).
3. There may be more than one job arrow with the same I and J.

The activities can be input in any order.

SAMPLE PROBLEM

Figure 1 is an arrow diagram for a sample project. The problem is solved first **without and then with calendar dating.** CPM DATA was previously entered.



Sample Arrow Diagram for Renewal of Pipeline

SAMPLE SOLUTION

```
*LIST CPMDATA

100 TEST CPM PRØGRAM
101 1 2 10 0
102 1 5 28 0
103 2 3 1 25
104 2 3 2 300
105 3 40 1 100
106 40 6 2 300
107 40 7 30 850
108 40 8 45 300
109 5 6 1 100
110 6 8 0 0
111 6 9 6 400
112 7 9 5 1200
113 8 11 1 100
114 9 10 6 800
115 10 11 2 100
116 11 12 1 100
117 11 13 4 300
118 12 13 0 0
119 12 14 1 50
120 13 14 1 100
121 14 15 1 100

READY

*RUN CPM
G E - 6 0 0    C P M    P R Ø G R A M
DATA FILE NAME
= CPMDATA
   TEST CPM PRØGRAM

       I    J   DUR   CØST        ES    EF    LS    LF    TF    FF

**     1    2    10      0         0    10     0    10     0     0
       1    5    28      0         0    28    16    44    16     0
       2    3     1     25        10    11    11    12     1     1
**     2    3     2    300        10    12    10    12     0     0
**     3   40     1    100        12    13    12    13     0     0
      40    6     2    300        13    15    43    45    30    14
      40    7    30    850        13    43    16    46     3     0
**    40    8    45    300        13    58    13    58     0     0
       5    6     1    100        28    29    44    45    16     0
       6    8     0      0        29    29    58    58    29    29
       6    9     6    400        29    35    45    51    16    13
       7    9     5   1200        43    48    46    51     3     0
**     8   11     1    100        58    59    58    59     0     0
       9   10     6    800        48    54    51    57     3     0
      10   11     2    100        54    56    57    59     3     3
      11   12     1    100        59    60    62    63     3     0
**    11   13     4    300        59    63    59    63     0     0
      12   13     0      0        60    60    63    63     3     3
      12   14     1     50        60    61    63    64     3     3
**    13   14     1    100        63    64    63    64     0     0
**    14   15     1    100        64    65    64    65     0     0

 PRØJ TØTAL: CØST      5225.   DURATIØN      65

PRØGRAM STØP AT 3080
*
```

```
*LIST CPMDATA

100 TEST CPM PRØGRAM WITH CALENDER DATING
101 1 2 10 0
102 1 5 28 0
103 2 3 1 25
104 2 3 2 300
105 3 40 1 100
106 40 6 2 300
107 40 7 30 850
108 40 8 45 300
109 5 6 1 100
110 6 8 0 0
111 6 9 6 400
112 7 9 5 1200
113 8 11 1 100
114 9 10 6 800
115 10 11 2 100
116 11 12 1 100
117 11 13 4 300
118 12 13 0 0
119 12 14 1 50
120 13 14 1 100
121 14 15 1 100
199 0 0 0 0          STØP SCAN FØR NETWØRK DATA
200 12 29 2          STARTING MØNTH,DAY,DAY ØF WEEK
201 69               STARTING YEAR,WØRK SCHEDULE FØLLØWS FØR THIS YEAR
202 1                IS A NØN WØRK DAY ØF THE WEEK
203 7                DITTØ
204 -1               STØP SCAN FØR NØN WØRK DAY ØF WEEK
205 12 25            CHRISTMAS IS A HØLIDAY
206 -1 -1            STØP SCAN FØR HØLIDAY
207 70               WØRK SCHEDULE FØR YEAR '..' FØLLØWS
208 1                IS A NØN WØRK DAY ØF THE WEEK
209 7                DITTØ
210 -1               STØP SCAN
211 1,1              NEW YEARS IS A HØLIDAY
212 2,12             LINCØLN'S BIRTHDAY
213 3,27             GØØD FRIDAY

READY

*RUN CPM
G E - 6 0 0    C P M    P R Ø G R A M
DATA FILE NAME
= CPMDATA
   TEST CPM PRØGRAM WITH CALENDER DATING

       I    J   DUR   CØST       ES         EF         LS         LF      TF   FF

**     1    2    10     0     12/29/69    1/13/70   12/29/69    1/13/70    0    0
       1    5    28     0     12/29/69    2/ 6/70    1/21/70    3/ 3/70   16    0
       2    3     1    25      1/13/70    1/14/70    1/14/70    1/15/70    1    1
**     2    3     2   300      1/13/70    1/15/70    1/13/70    1/15/70    0    0
```

**	3	40	1	100	1/15/70	1/16/70	1/15/70	1/16/70	0	0
	40	6	2	300	1/16/70	1/20/70	3/ 2/70	3/ 4/70	30	14
	40	7	30	850	1/16/70	3/ 2/70	1/21/70	3/ 5/70	3	0
**	40	8	45	300	1/16/70	3/23/70	1/16/70	3/23/70	0	0
	5	6	1	100	2/ 6/70	2/ 9/70	3/ 3/70	3/ 4/70	16	0
	6	8	0	0	2/ 9/70	2/ 9/70	3/23/70	3/23/70	29	29
	6	9	6	400	2/ 9/70	2/18/70	3/ 4/70	3/12/70	16	13
	7	9	5	1200	3/ 2/70	3/ 9/70	3/ 5/70	3/12/70	3	0
**	8	11	1	100	3/23/70	3/24/70	3/23/70	3/24/70	0	0
	9	10	6	800	3/ 9/70	3/17/70	3/12/70	3/20/70	3	0
	10	11	2	100	3/17/70	3/19/70	3/20/70	3/24/70	3	3
	11	12	1	100	3/24/70	3/25/70	3/30/70	3/31/70	3	0
**	11	13	4	300	3/24/70	3/31/70	3/24/70	3/31/70	0	0
	12	13	0	0	3/25/70	3/25/70	3/31/70	3/31/70	3	3
	12	14	1	50	3/25/70	3/26/70	3/31/70	4/ 1/70	3	3
**	13	14	1	100	3/31/70	4/ 1/70	3/31/70	4/ 1/70	0	0
**	14	15	1	100	4/ 1/70	4/ 2/70	4/ 1/70	4/ 2/70	0	0

PRØJ TØTAL: CØST 5225. DURATIØN 65

PRØGRAM STØP AT 3080
*

This FORTRAN subroutine uses the convex-simplex method[1] to find the non-negative vector X which maximizes an arbitrary convex function F(X) subject to the linear constraints.

$$\sum_{j=1}^{M} Aij \ Xj = Ci \quad , \quad i = 1, \ldots, \ N$$

$$X_j \geq 0 \ , \ j = 1, \ldots, \ M$$

INSTRUCTIONS

Before calling this routine, a basic feasible solution with its corresponding tableaux must be found. This may be done using the library program UNDEQ. The arguments are passed to the routine by blank common. The calling sequence is:

```
COMMON  M, N  X(50), A(50, 50), INBASE(50), MAXIT,
              ICONV, EPSLON
CALL   CSM
```

where

- M is the number of variables
- N is the number of constraint equations
- X on entry is a basic feasible solution and on exit is the best approximation found to the maximizing point
- A is the tableaux corresponding to the solution
- INBASE is an N-vector whose entries indicate which variables are in the basis for the tableaux A
- MAXIT on entry is an upper bound on the number of iterations to be performed. On exit is the actual number of iterations performed.
- ICONV is set by the routine to 1 if the convergence criteria was satisfied, to 2 if MAXIT iterations were exceeded and to 3 if an unbounded solution is indicated.
- EPSLON is a small number used in the convergence criteria.

REFERENCE

[1] Zangwill, W., Nonlinear Programming, a Unified Approach, Prentice-Hall, Englewood Cliffs, New Jersey

A subroutine FUNCT must also be supplied to evaluate the function F and its gradient. Calls to FUNCT take the form:

```
CALL        FUNCT (DELTA, FVAL, G)
```

where

- DELTA is the M-vector at which the function is to be evaluated.
- FVAL returns the value of F at DELTA.
- G is an M-vector that returns the gradient of F at DELTA.

NOTE: The blank common areas for both library programs UNDEQ and CSM are consistent.

SAMPLE PROBLEM

Find the non-negative vector X which maximizes:

$$\mathrm{In}\left[\left\{\frac{C_1(X_1+X_2+X_3)}{X_1}\right\}^{X_1}\left\{\frac{C_2(X_1+X_2+X_3)}{X_2}\right\}^{X_2}\left\{\frac{C_3(X_1+X_2+X_3)}{X_3}\right\}^{X_3}\left\{C_4\right\}^{X_4}\left\{\frac{C_5(X_5+X_6)}{X_5}\right\}^{X_5}\left\{\frac{C_6(X_5+X_6)}{X_6}\right\}^{X_6}\right]$$

where C = (200, .49138901, .49138901, 274285714, 1, .66666667E-6)

Note that the gradient vector for the function to be maximized is:

$$\left(\mathrm{In}\left\{\frac{C_1(X_1+X_2+X_3)}{X_1}\right\}, \mathrm{In}\left\{\frac{C_2(X_1+X_2+X_3)}{X_2}\right\}, \mathrm{In}\left\{\frac{C_3(X_1+X_2+X_3)}{X_3}\right\}, \mathrm{In}\left\{C_4\right\}\right.$$

$$\left.\mathrm{In}\left\{\frac{C_5(X_5+X_6)}{X_5}\right\}, \mathrm{In}\left\{\frac{C_6(X_5+X_6)}{X_6}\right\}\right)$$

subject to the constraints

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 1 & 2/3 & 2/3 & -1 & 0 & 1 \\ 0 & 3 & 3 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 1 \end{bmatrix} X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

SAMPLE SOLUTION

```
*LIST

100*....SAMPLE DRIVER FØR CSM....
110 CØMMØN M,N,X(50),A(50,50),INBASE(50),KEY1,KEY2,EPS,C(50)
120 DØ 5 I=1,5
130 C(I)=0.
140 DØ 5 J=1,6
150 5 A(I,J)=0.
160 A(1,1)=1.; A(1,2)=1.; A(1,3)=1.
170 A(2,1)=1.; A(2,2)=1.; A(2,3)=1.
180 A(3,1)=1.; A(5,3)=1.; A(5,5)=1.
190 A(3,6)=1.; A(4,6)=1.; A(5,6)=1.
200 A(3,2)=.66666667; A(3,3)=.66666667
210 A(4,2)=3.; A(4,3)=3.
220 A(2,4)=-1.; A(3,4)=-1.; A(4,4)=-1.; A(5,4)=-1.
230 C(1)=1.
240 M=6
250 N=5
260 EPS=1E-8
270 KEY2=0
280 CALL UNDEQ
290 PRINT 10,KEY1,(X(I),I=1,M)
300 10 FØRMAT("OFEASIBLE SØLUTIØN (KEY=",I1,")"/6F7.3)
310 PRINT 20, (INBASE(I),I=1,N)
320 20 FØRMAT("OBASIC VARIABLES ",6I3)
330 PRINT 30
340 30 FØRMAT("OTABLEAUX")
350 PRINT 40, ((A(I,J),J=1,6),I=1,5)
360 40 FØRMAT(6F7.3)
370 KEY1=50
380 EPS=1E-5
390 CALL CSM
400 PRINT 50, KEY2,(X(I),I=1,M)
410 50 FØRMAT("OØPTIMUM SØLUTIØN(ICØNV=",I2,")",/6F7.3)
420 PRINT 20,(INBASE(I),I=1,N)
430 PRINT 30
440 PRINT 40, ((A(I,J),J=1,6),I=1,5)
450 STØP
460 END
```

```
470*
480 SUBRØUTINE FUNCT(DELTA,FVAL,G)
490 DIMENSIØN DELTA(6),G(6)
500 DIMENSIØN JJ(4),X(6),C(6)
510 DATA JJ/0,3,4,6/
520 DATA C/200.,  .49138901,  .49138901,  2.74285714E8,  1.,
530&      .66666667E-6/
540 FVAL=0.
550 DØ 120 K=1,3
560 ISTART=JJ(K)+1
570 IEND=JJ(K+1)
580 RLAM=0.
590 DØ 100 I=ISTART,IEND
600 X(I)=AMAX1(DELTA(I),1E-8)
610 100 RLAM=RLAM+X(I)
620 DØ 110 I=ISTART,IEND
630 GG=ALØG(C(I)*RLAM/X(I))
640 FVAL=FVAL+GG*X(I)
650 110 G(I)=GG
660 120 CØNTINUE
680 RETURN
690 END
```

```
READY

*RUN *;UNDE@;CSM=(CØRE=20)

FEASIBLE SØLUTIØN (KEY=0)
 0.700  0.      0.300  1.000  0.600  0.100

BASIC VARIABLES    4  1  6  3  5

TABLEAUX
 0.      0.      0.      1.000   0.      0.
 1.000   0.      0.      0.      0.      0.
 0.      0.      0.      0.      0.      1.000
 0.      1.000   1.000   0.      0.      0.
 0.     -1.000   0.      0.      1.000   0.

ØPTIMUM SØLUTIØN(ICØNV= 1)
 0.700  0.159  0.141  1.000  0.759  0.100

BASIC VARIABLES    4  1  6  2  5

TABLEAUX
 0.      0.      0.      1.000   0.      0.
 1.000   0.      0.      0.      0.      0.
 0.      0.      0.      0.      0.      1.000
 0.      1.000   1.000   0.      0.      0.
 0.      0.      1.000   0.      1.000   0.

*
```

The BASIC program minimizes an unconstrained multivariate function. Partial derivatives are required. The Davidon optimization technique is a modified steepest descent method. The program is dimensioned for ten adjustable parameters.

INSTRUCTIONS

The user must enter the function to be minimized and its partial derivatives. The response function must have statement number 50. Statement numbers 51-60 are available for long functions. Statement numbers 61-100 are available for the partial derivatives. Enter:

```
50 LET F = [a function of the vector X]
61 LET D(1) = [1st partial derivative of F]
62 LET D(2) = [2nd partial derivative of F]
```

The variable F and the array D must be used for the function and its partial derivatives. Then type RUN.

Several quantities will be requested by the program. These input quantities have the following meaning:

- LIMIT is the maximum number of iterations to be performed
- EPSILON is the convergence test constant
- ALPHA is a constant, 0<ALPHA<1, try 0.001
- BETA is a constant, BETA>1, try 10
- NUM is the number of variables

When the program asks CONTINUE, an answer of 0 causes termination. An answer of 1 allows you to change parameters and/or perform more iterations.

REFERENCE

Computer Journal, Volume 10, 1968, pp. 406-410

SAMPLE PROBLEM

$$\text{Minimize } F = 100\,(x_2 - x_1^{\,2})^2 + (1 - x_1)^2$$

using as starting conditions $x_1 = -1.2$ and $x_2 = 1.0$. The surface is a banana-shaped ridge which has its minimum at (1, 1) with a value of zero.

SAMPLE SOLUTION

```
*50 LET F=100*(X(2)-X(1)↑2)↑2+(1-X(1))↑2
*61 LET D(1)=-400*X(1)*(X(2)-X(1)↑2)-2*(1-X(1))
*62 LET D(2)=200*(X(2)-X(1)↑2)
*RUN


WHAT ARE LIMIT,EPSILØN, ALPHA,BETA,NUM
?100,1E-6,.001,10,2
GIVE THE INITIAL GUESS FØR THE VECTØR X
X(     1 )= ?-1.2
X(     2 )= ?1.0

          90   ITERATIØNS HAVE BEEN PERFØRMED
THE CØNVERGENCE CRITERIA IS SATISFIED
THE CANDIDATE FØR THE MINIMUM IS:
  .9996785     .9993005
THE VALUE ØF THE FUNCTIØN THERE IS: 4.23746E-07
CØNTINUE ?0


READY
*
```

The file GASPIIA contains a set of Fortran subroutines that provide the programmer with a Fortran-based simulation language, used to simulate event-oriented systems. GASPSAMP and GASPDATA are sample problem files. GASPIIA provides routines for event control, statistical collection and computation, report generation, and random number generation. The user must supply the main program, the event routines and an optional output routine.

NOTE: GASPIIA calls a uniform random number generator in the form Y = UNIFM1(X). The library programs FLAT, URAN, or UNIFM may be used.

INSTRUCTIONS

Log onto time sharing under the Fortran subsystem and write a main program and subprograms as needed for the system to be simulated. Then give the command:

RUN*; GASPIIA; FLAT

The program will type:

NAME OF INPUT AND OUTPUT FILES?

Respond with the names of a previously prepared input file and an output file.

Input or output performed by the GASPIIA subroutines will be on these files. If blanks are entered for a filename, the file is the terminal device. The input file is in free-field format except for the first line, which is formatted as described in Simulation With GASPII. The input file should not have line numbers. The GASP storage map may be listed.

REFERENCE

A. Alen, B. Pritsker, and Philip J. Kiviat, Simulation with GASPII, Prentice-Hall, Englewood Cliffs, N.J., 1969.

SAMPLE PROBLEM

Sample problem 5 from Simulation With GASPII has been programmed. The user-supplied coding is stored in the file GASPSAMP. Problem data is stored in the file GASPDATA.

SAMPLE SOLUTION

The execution of GASPIIA is demonstrated by running the sample program GASPSAMP and GASPDATA. The GASPDATA file is also listed. The output is directed to a temporary file named GASPANS, which is then listed.

```
*LIST GASPDATA

PRITSKER A        504151968    1
 3 2 2 4 20 1 4 22 4
 20 20
 1 2 3 2
 1 1 1 1
 .4 0.0 10.0 1.
 .25 0.0 10.0 1.
 .5 0.0 10.0 1.
 0 1 0 0 0.0 400. 567
 -1 0
 1 1
 0.1 0.0 0.1 0.0
 1 2
 1.0 0.0 0.0 0.0
 1 3
 1.0 0.0 0.0 0.0
 2 0
  0.0 0 0 0
 2 0
  0.0 0 0 0
 2 0
  0 0 0 0
 1 4
 300. 0 0 0
 0 0

READY

*ØLD GASPSAMP
READY
*RUN *;GASPIIA;FLAT
NAME ØF INPUT AND ØUTPUT FILES?
= GASPDATA,GASPANS
DØ YØU WANT TØ SEE A GASP JØB STØRAGE DUMP?
0=NØ,1=YES
= 0

PRØGRAM STØP AT 220
*LIST GASPANS

  SIMULATIØN PRØJECT NØ.   5  BY  PRITSKER A

  DATE  4/ 15/ 1968            RUN NUMBER    1


   PARAMETER NØ.    1      0.4000      0.         10.0000     1.0000
   PARAMETER NØ.    2      0.2500      0.         10.0000     1.0000
   PARAMETER NØ.    3      0.5000      0.         10.0000     1.0000
    **INTERMEDIATE RESULTS**


  **GASP SUMMARY REPORT**

 SIMULATIØN PRØJECT NØ.   5  BY  PRITSKER A

 DATE  4/ 15/ 1968            RUN NUMBER    1


   PARAMETER NØ.    1      0.4000      0.         10.0000     1.0000
   PARAMETER NØ.    2      0.2500      0.         10.0000     1.0000
   PARAMETER NØ.    3      0.5000      0.         10.0000     1.0000
```

```
**GENERATED DATA**
CØDE     MEAN       S.DEV.      MIN.        MAX.      ØBS.

  1      3.2930     1.4775      0.3224      8.0680     554
  2      0.5481     0.5530      0.0004      4.3939     554

**TIME GENERATED DATA**
CØDE     MEAN       STD.DEV.      MIN.        MAX.    TØTAL TIME

  1      6.0080     1.8272      0.          8.0000    303.6451
  2      0.4942     0.5000      0.          1.0000    303.6451
  3      0.9324     0.2510      0.          1.0000    303.6451
  4     45.1561    49.7648      0.        100.0000    303.6451

**GENERATED FREQUENCY DISTRIBUTIØNS**
CØDE                    HISTØGRAMS

  1          2  16  29  56  76  81  97  44  45  34  20
            24   8   9  11   1   1   0   0   0   0   0

  2          0 154 105  83  60  44  29  19  16  14  10
             4   4   3   2   2   0   2   1   0   1   1


 FILE PRINTØUT, FILE NØ.  1

AVERAGE NUMBER IN FILE WAS,      2.4324
STD. DEV                         0.5505
MAXIMUM                             4

 FILE CØNTENTS


NSET


THE FILE IS EMPTY



 FILE PRINTØUT, FILE NØ.  2

AVERAGE NUMBER IN FILE WAS,      2.5281
STD. DEV                         1.3933
MAXIMUM                             4

 FILE CØNTENTS


NSET


THE FILE IS EMPTY



 FILE PRINTØUT, FILE NØ.  3

AVERAGE NUMBER IN FILE WAS,      1.5722
STD. DEV                         0.7200
MAXIMUM                             2

 FILE CØNTENTS
```

```
NSET

THE FILE IS EMPTY



 FILE PRINTOUT, FILE NO.  4

AVERAGE NUMBER IN FILE WAS,     0.4516
STD. DEV                        0.4976
MAXIMUM                            1

 FILE CONTENTS


NSET


THE FILE IS EMPTY



MEAN TIME BETWEEN ARRIVALS =0.40
MEAN SERVICE TIME FOR STATION 1 =0.25
MEAN SERVICE TIME FOR STATION 2 =0.50
PERCENT OF ITEMS SUBCONTRACTED = 32.22
NUMBER OF ITEMS SUBCONTRACTED = 261.
TOTAL ITEMS =  810.

READY

*BYE
   1 TEMPORARY FILES CREATED.

GASPANS  ?
```

This BASIC program schedules n jobs in a job shop with m machines using a heuristic geometric approach. The general job-shop scheduling problem is defined as follows: given a set of n jobs that are to be processed by m machines where the sequence and the process times are known, obtain a schedule that minimizes the total flow time. This program makes the following assumptions:

1. All jobs are available for scheduling at the same time.
2. A machine can process only one job at a time.
3. A job on a machine must be completed before the next job can enter the machine.
4. The sequence and processing time of each operation is known.
5. The sequence and processing times are independent of the schedule.
6. Only a finite number of jobs are considered without regard for future jobs.
7. No alternate sequences are permitted.
8. Process time includes all transportation and setup time required by an operation.

This program calculates two different schedules for each job and then uses eight different heuristic geometric approaches to improve the schedules. The schedule which has the minimum total time is then printed. Since a total of 16 schedules for each job are computed and each of these schedules is modified in an iterative loop, large computation times are required. Delays of several minutes or more may be encountered while running the program. The computation time increases rapidly with the size of the problem.

INSTRUCTIONS

The data is entered in DATA statements beginning in line 6000. Sample data is already imbedded in the program in lines 6000 through 6190. Delete the sample data by typing DELETE 6000, 6190; then enter your data in the following order:

- The number of jobs to be processed
- The number of machines
- The sequence matrix S by rows where S_{ij} is the sequence of the i^{th} job on the j^{th} machine.
- The process time matrix P by rows where P_{ij} is the process time required by the i^{th} job on the j^{th} machine.

REFERENCE

Blick, R.G. Heuristics for Scheduling the General n/m Job-Shop Problem. General Electric Company report No. 69-C-162, April 1969, GE Technical Information Exchange, P.O. Box 43, Bldg. 5, Schenectady, N.Y. 12301.

SAMPLE PROBLEM

A 6-job, 6-machine problem is imbedded in the sample data. The optimal total time for this problem is 55. Note that the best schedule found by this program has a total time of 57.

SAMPLE SOLUTION

```
*RUN


GEOMETRIC SIMULATOR

INSTRUCTIONS:
THIS PROGRAM CALULATES A JOB SHOP SCHEDULE USING
A HEURISTIC GEOMETRIC METHOD.
ENTER DATA IN THE FOLLOWING ORDER BEGINNING
IN LINE 6000:
    * THE NUMBER OF JOBS TO BE PROCESSED,
    * THE NUMBER OF MACHINES,
    * THE SEQUENCE MATRIX S(I,J) BY ROWS
      WHERE S(I,J) IS THE SEQUENCE OF THE I'TH
      JOB ON THE J'TH MACHINE,
    * THE PROCESS TIME MATRIX P(I,J) BY ROWS
      WHERE P(I,J) IS THE PROCESS TIME REQUIRED
      BY THE I'TH JOB ON THE J'TH MACHINE.

SAMPLE DATA IS ALREADY IN LINES 6000-6190.

TO EXECUTE THE PROGRAM:
    TYPE 'DELETE 6000-6190' (DELETES SAMPLE DATA)
    ENTER YOUR DATA
    TYPE 'RUN'

THIS PROGRAM REQUIRES LARGE AMOUNTS OF PROCESSOR TIME

THE FOLLOWING IS A SAMPLE EXECUTION USING THE
SAMPLE DATA.

SCHEDULING  6-JOBS ON  6-MACHINES
INPUT
SEQUENCE MATRIX-S

MACHINE  1      2      3      4      5      6
JOB
 1       2      3      1      4      6      5
 2       5      1      2      6      3      4
 3       4      5      1      2      6      3
 4       2      1      3      4      5      6
 5       5      2      1      6      3      4
 6       4      1      6      2      5      3
```

PRØCESSING TIME MATRIX-P

MACHINE	1	2	3	4	5	6
JØB						
1	3	6	1	7	6	3
2	10	8	5	4	10	10
3	9	1	5	4	7	8
4	5	5	5	3	8	9
5	3	3	9	1	5	4
6	10	3	1	3	4	9

SCHEDULE ØF JØB START TIMES

MACHINE	1	2	3	4	5	6
JØB						
1	1	16	0	26	51	38
2	38	0	8	48	13	28
3	18	27	1	6	44	10
4	13	8	18	23	26	45
5	48	32	23	52	35	41
6	28	13	44	16	40	19

TØTAL TIME = 57 AVERAGE TIME = 52

READY
*

This Fortran program solves geometric programming problems by using the convex-simplex method and Newton-Raphson iterations to maximize the dual problem. A geometric program is a special type of non-linear minimization problem. The general form is to minimize the objective function g(1, t) subject to the positivity constraints t(i)>0, i = 1, . . . , M and the forced constraints

$$0 < \omega(k) * g(k,t) ** \omega(k) \leq 1, \quad k = 2, \ldots, p$$

where the g(k, t) are functions given by

$$g(k,t) = \sum_{i=J(k)+1}^{J(k+1)} c(i) \prod_{j=1}^{M} t(j) ** a(i,j)$$

The a(i, j) values are called the exponent matrix values; the c(i)'s form the coefficient vector; the J(k)'s are monotonic increasing integers with J(1)=0 which partition the coefficient vector and the exponent matrix over the objective and (p-1) constraint functions; the ω (k)'s are + or - one; and the t(i)'s are the variables. This problem is called the generalized primal problem.

Many primal problems can be solved by first solving a related problem (called the dual problem). Let σ (i) = ± 1 as c(i) is positive or negative. Let ω (1) = ± 1 as the minimum to the primal problem is positive or negative. The dual problem is to maximize, as a function of δ , the dual objective function given by

$$\omega(1) \prod_{k=1}^{P} \prod_{i=J(k)+1}^{J(k+1)} \left[\frac{c(i) \quad \lambda(k)}{\delta(i)} \right] ** \left[\sigma(i) \quad \delta(i) \quad \omega(i) \right]$$

where

$$\lambda(k) = \omega(k) \sum_{i=J(k)+1}^{J(k+1)} \sigma(i) \quad \delta(i), \qquad k = 1, \ldots, p$$

subject to the normality constraint λ (1) = 1 and the orthogonality conditions

$$\sum_{i=1}^{J(p+1)} \sigma(i) \quad a(i,k) \quad \delta(i) = 0, \qquad k = 1, \ldots, M$$

and the non-negativity conditions

$$\delta(i) \geqq 0 , \qquad i = 1, \ldots, J(k+1)$$

$$\lambda(k) \geqq 0 , \qquad k = 2, \ldots, p$$

Example

Minimize the function $t(1)\ t(2)$ subject to $t(1),\ t(2) > 0$ and

$$3\ t(1)^{-1} - 2\ t(2)^{-2} \geqq 1.$$

To put the constraint into the proper form, we rewrite it as

$$0 < - \left[-3\ t(1)^{-1} + 2\ t(2)^{-2} \right]^{-1} \leqq 1$$

Then we have

$$a = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & -2 \end{bmatrix} \quad c = \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix} \quad \sigma = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \quad J = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \quad \omega = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The dual for this problem is to maximize

$$\left[\frac{1}{\delta(1)}\right]^{\delta(1)} \left[\frac{3\ \left[\delta(2) - \delta(3)\right]}{\delta(2)}\right]^{-\delta(2)} \left[\frac{2\ \left[\delta(2) - \delta(3)\right]}{\delta(3)}\right]^{\delta(3)}$$

subject to:

$$\delta(1) = 1$$

$$\delta(1) - \delta(2) = 0$$

$$\delta(1) - 2\ \delta(3) = 0$$

and $\delta(1)$, $\delta(2)$, $\delta(3)$, and $\left[\delta(2) - \delta(3)\right]$ all non-negative.

A special case of importance occurs when all the $c(i)$'s and $\omega(i)$'s are positive. This is called the posynomial case. In this case there is a strong and useful relation between the primal and dual problems: If the primal problem has a positive solution at a finite point in the primal feasible set, then the maximum of the dual problem is equal to the minimum of the primal program. Furthermore, given the solution to the dual problem, the solution to the primal problem may be found by solving a linear system of equations. In the posynomial case, the dual problem is a convex program with linear constraints; it may be solved using the convex-simplex method.

In the generalized case, the dual-primal relationship is not as strong since the dual and primal are not convex programming problems. This algorithm will find a local maximum for the dual problem. The primal point found corresponding to this point will be a local optimum (minimum or maximum) or a special point such as a saddle point.

In case the dual program is infeasible, then the primal program does not attain its minimum at a finite primal feasible point. In the generalized case this algorithm may fail when the local optimum of the dual is on the border of the dual feasible set (the Newton-Raphson iteration technique fails). This may occur if one of the primal constraint functions is not tight at the optimum. The total number of terms in all the g(i, t) functions minus the number of primal variables minus one is a measure of the degree of difficulty in solving the dual problem.

RESTRICTIONS

The total number of terms in all the g(i, t) functions must exceed the number of primal variables. None of the c(i)'s can be zero. There cannot be more than 50 primal variables, 50 total terms in the g(i, t) functions or 49 primal constraint functions.

REFERENCES

Wilde, D.J. and Beightler, C.S., Foundations of Optimization, Prentice-Hall, Englewood Cliffs, N.J., 1967.

INSTRUCTIONS

Data Input Format

The most convenient method of entering data into the program is to first build a data file; however, the data can be entered in a question/answer sequence while the program is executing. In either method, the data is entered in the same order and format. The data file can be built either with or without line numbers, but the use of line numbers greatly facilitates the changing and correcting of the file. The data file has the following format:

line #	description of values
10	any alphanumeric problem identification
20	# of primal variables, # of primal constraints
30	# of terms in the objective function and in each
.	constraint function
.	(use as many lines as needed)
.	
40	the coefficients of all the terms in the objective
.	and constraint functions
.	(use as many lines as needed)
.	
50	the exponent values of all the variables, for each
.	term, for the objective and constraint functions
.	(use as many lines as needed)
.	
60	the constraint numbers whose W's are negative
	(omit this line if they are all positive)

The data values are entered in free-field format; data values are separated from the line number by blanks and from each other by blanks or commas. If the data file does not have line numbers, then the first character of the first line must be non-numeric. When the program asks for the name of the data file, a null response initiates the question/answer sequence to read the data from the terminal directly. In this case, the first character of the problem identification must also be non-numeric.

The data file representing the problem in the example is

```
010 SAMPLE DATA FILE FOR A GENRLIZED GEO.PROG.
020  2,   1
030  1,  2
040  1,   3,   -2
050  1,   1
060 -1,   0
070  0,   -2
080  1
```

The question answer sequence for inputting this same data is illustrated in the following.

```
*RUN GPROG
ENTER NAME OF DATA FILE
=__
ENTER PROBLEM ID
=SAMPLE DATA FILE FOR A GENRILIZED GEO. PROG.
# VARIABLES, # CONSTRAINTS
=2,1
# TERMS IN OBJECTIVE AND EACH CONSTRAINT
=1,2
ENTER COEFFICIENTS FOR EACH TERM IN OBJECTIVE & CONSTRAINTS
=1, 3,-2
ENTER EXPONENTS FOR EACH TERM IN THE OBJECTIVE & CONSTRAINT EQS.
=1,1,    -1,0,    0,-2
ENTER CONSTRAINT #'S WHICH HAVE NEG. OMEGA'S
=1

ENTER COMAND
=
```

Program Execution Instructions

After the data has been read by the program using one of the methods described, the program requests a command verb. The following commands are implemented:

LIST — Causes the program to list the data as read by the program. This command is useful to verify that the data has been entered correctly.

SOLVE — Causes the following solution strategy to be initiated.

1. Find a dual feasible solution or a non-zero dual feasible according as the internal variable POS is set to .FALSE. or .TRUE.
2. Improve the dual solution estimate using the convex-simplex method (CSM). CSM iterations will stop after ITLIM iterations have been performed or when the improvement per iteration is less than EPS1. Find the corresponding primal solution. If this primal solution estimate is feasible within a tolerance of EPS2 and the primal objective value is within EPS3 of the dual objective value, take this estimate as the solution.
3. Perform additional dual iterations, using Newton-Raphson on the Lagrangian (LAGRAN) if possible. EPS1 is divided by 100 if the EPS1 criterion was the reason for halting iterations in step 2. The primal point is found and tested for optimality as in step 2. If optimality was not achieved, the current best estimates are printed and the program requests another command.

CONTINUE — Causes the program to continue dual-primal iterations as in steps 2 and 3 of SOLVE.

DATA — Causes the program to read data for a new problem.

PARAMETERS — Causes the program to print the current values of the internal parameters EPS1, EPS2, EPS3, ITLIM and POS as described in SOLVE. Also the value of the parameter TRACE. If TRACE = .TRUE. a step by step trace of the solution is printed. The default values for these parameters are:

EPS1 = 1E-3
EPS2 = 1E-5
EPS3 = 1E-5
ITLIM = 50
POS = .TRUE.
TRACE = .FALSE.

ALTER — Causes the program to read

NAMELIST/CHANGE/A, C, EPS1, EPS2, EPS3, ITLIM, POS, TRACE, DELTA from the terminal. This allows the user to alter the internal control parameters, the exponent data matrix, the coefficient data array, or the current best dual solution estimate (DELTA).

STOP — Causes the program execution to be terminated.

The file GPROG-SO is the source version and may be run via the CARDIN system to regenerate the object file GPROG.

SAMPLE PROBLEM

Minimize the objective function in two variables:

$$2x + xy + 3y$$

Subject to the two primal constraints:

$$x^{-2} + y^{-1} \leqq 3$$

$$x^{-1} + 2y^{-1} \leqq 4$$

and the positivity constraints: $x, y \geqq 0$.

SAMPLE SOLUTION

```
*LIST SAMPLE

010 SAMPLE PRØBLEM FØR GEØMETRIC PRØGRAMMING
020 2,2
030 3,2,2
040 2,1,3,    .33333333,.33333333,    .25,.5
050 1,0
060 1,1
070 0,1
080 -2,0
090 0,-1
100 -1,0
110 0,-1
```

```
READY

*RUN GPRØG
ENTER NAME ØF DATA FILE
=SAMPLE

ENTER CØMAND
=LIST


***INPUT DATA***

PRØB ID-
SAMPLE PRØBLEM FØR GEØMETRIC PRØGRAMMING
# VARIABLES     =     2
# CØNSTRAINTS =     2
DEGREE ØF DIF =     4
TYPE, PØSYNØMIAL
CØEFFICIENTS            EXPØNENT MATRIX
   2.00E 00     1.00E 00     0.
   1.00E 00     1.00E 00     1.00E 00
   3.00E 00     0.           1.00E 00
------------
   3.33E-01    -2.00E 00     0.
   3.33E-01     0.          -1.00E 00
------------
   2.50E-01    -1.00E 00     0.
   5.00E-01     0.          -1.00E 00
------------

ENTER CØMAND
=SØLVE


**SØLUTIØN**




**ØPTIMAL**




**DUAL PØINT
  3.59592922E-01  1.322444416E-01  5.0815 2662E-01  1.18139891E 01
  9.78839193E-02  2.55557559E-01  5.425231 53E-01
DUAL VALUE =     4.3423291E 00

CØRRESPØNDING PRIMAL-  FEASIBLE(MAX INFES=  0.53E-04)
  7.80713879E-01  7.35543177E-01
PRIMAL ØBJ & CØNSTRAINT VALUES (REL ERRØR = -0.53E-05)
  4.34230608E 00  1.00006348E 00  9.99989495E-01

ENTER CØMAND
=STØP

*
```

This Fortran program solves the zero-one integer programming problem using a modification of Balas' method of implicit enumeration

INSTRUCTIONS

To use this program, formulate the problem to be solved according to the following standard:

Minimize

$$\sum_{i=1}^{n} a_{oi}x_i$$

subject to

$$a_{jo} \geq \sum_{i=1}^{n} a_{ji}x_i \quad j=1, \ldots, m$$

and $x_i = 0$ or 1, $i = 1, \ldots, n$

so then n is the number of variables and m is the number of constraints. If m is greater than 11 or n is greater than 28, the dimensions in the program must be increased.

All of the coefficients, a_{ij}, should be integer. Establish a data file with line numbers in the following format:

line #, m, n

line #, 0, a_{o1}, ..., a_{on}

line #, a_{10}, a_{11}, ..., a_{17}

.

.

.

line #, a_{mo}, a_{m1}, ..., a_{mn}

Note that a_{i0} for i = 1 to m are the right-hand side constants. The first entry after the line number on the second line is ignored by this program. It is included to maintain consistency with the data file format used by the INTLP program.

The algorithm will ask for a maximum number of iterations to be performed. If the optimum has not been found and confirmed within the limit given, data will be written on a temporary file START. The user can then give a new limit on the number of iterations or can stop the program by giving 0 as the new limit. If the file START is saved, the user can continue the problem at a later time, picking up where he left off.

On executing, the program asks if the problem is a new or a restart using data on the file START. It also requests the name of the original data file. If the problem is a new one, the user can specify some variables to be set to one. The algorithm will never consider the cases with those variables set to 0 (see Note 2). If no variables are to be set to 1, enter 0 for the first index.

The program prints out the first feasible answer found and all subsequent feasible answers whose objective value is better than the last feasible answer.

NOTES:

1. It is usually the case that the optimum answer is found rather easily, but that it then requires many iterations to verify that it is optimum.
2. The option of setting variables to one may reduce the number of iterations needed to find the optimum; however, this optimum may not be the optimum to the original problem.

REFERENCE

Zionts, Stanley. Implicit Enumeration Using Bonds on Variables: A Generalization of Balas' Additive Alogrithm for Solving Linear Programs With Zero-One Variables presented at the Operational Research Society of India Annual Meeting, Calcutta, India, November 1968.

SAMPLE PROBLEM

Solve the problem with 4 constraints and 20 variables which has as the coefficient vector of the objective function to be minimized

(3, 2, 5, 8, 6, 9, 11, 4, 5, 6, 11, 2, 8, 5, 8, 7, 3, 9, 2, 4)

The coefficient matrix of the constraints:

$$\begin{bmatrix} -6 & 5 & -8 & -3 & 0 & -1 & -3 & -8 & -9 & 3 & -8 & 6 & -3 & -8 & -6 & 7 & 6 & -2 & 3 & -7 \\ -1 & 3 & 3 & 4 & 1 & 0 & 4 & -1 & -6 & 0 & 8 & 0 & 1 & -5 & -4 & -1 & -9 & -7 & 2 & 2 \\ 3 & 6 & 1 & -3 & -5 & 6 & -9 & 6 & 3 & -9 & -6 & -3 & -6 & -6 & 6 & 2 & -7 & -6 & 0 & -7 \\ 1 & 4 & 2 & -1 & 1 & 1 & 1 & -7 & -8 & -9 & -8 & -7 & -9 & 1 & 1 & 0 & 1 & 2 & 1 & -3 \end{bmatrix}$$

and the right hand sides of the constraints

(-25, -13, -15, -13)

SAMPLE SOLUTION

The data was stored in the file IN01IN. The program was run with a maximum of 100 iterations, and then stopped. The restart capability was illustrated by running the program again with a maximum of 500 iterations. No new feasible solutions were found so the maximum was increased to 900 iterations. Notice that when signing off the user is given the option of saving the temporary file START.

```
*RUN

INT01

0=NEW PRØBLEM, 1=RESTART ØN ØLD PRØBLEM
= 0
NAME ØF THE DATA FILE
= IN01IN
MAX # ØF ITER BEFØRE DECISIØN PØINT
= 100
INDICES ØF VARIABLES TØ BE SET TØ 1, ØNE AT A TIME
(0 STØPS SCAN)
= 0

***FEASIBLE PØINT***
ALL VARIABLES ARE 0 EXCEPT:
X(  9)=1
X( 13)=1
X( 14)=1
X( 18)=1
X( 20)=1
Z=           31 ITER=           5

***FEASIBLE PØINT***
ALL VARIABLES ARE 0 EXCEPT:
X(  1)=1
X(  9)=1
X( 13)=1
X( 14)=1
X( 17)=1
X( 20)=1
Z=           28 ITER=          13

***FEASIBLE PØINT***
ALL VARIABLES ARE 0 EXCEPT:
X(  5)=1
X(  8)=1
X(  9)=1
X( 14)=1
X( 17)=1
X( 20)=1
Z=           27 ITER=          83

AT ITER          100 RESTART FILE BUILT
WHAT IS NEW ITMAX (0=STØP)
= 0

PRØGRAM STØP AT 700
*RUN

INT01

0=NEW PRØBLEM, 1=RESTART ØN ØLD PRØBLEM
= 1
NAME ØF THE DATA FILE
= IN01IN
MAX # ØF ITER BEFØRE DECISIØN PØINT
  500
```

```
BEST SØLUTIØN FØUND PREVIØUSLY

ALL VARIABLES ARE 0 EXCEPT:
X(  5)=1
X(  8)=1
X(  9)=1
X( 14)=1
X( 17)=1
X( 20)=1
Z=           27 ITER=          100

AT ITER          500 RESTART FILE BUILT
WHAT IS NEW ITMAX (0=STØP)
= 900

***ØPTIMUM***

ALL VARIABLES ARE 0 EXCEPT:
X(  5)=1
X(  8)=1
X(  9)=1
X( 14)=1
X( 17)=1
X( 20)=1
Z=           27 ITER=          890

PRØGRAM STØP AT 1670
*BYE
   1 TEMPØRARY FILES CREATED.

START    ?
```

The data was saved in the file IN01IN and is listed below:

```
*LIST IN01IN

010 4,20
020 0,    3,2,5,8,6,9,11,4,5,6,11,2,8,5,8,7,3,9,2,4
030 -25,  -6,5,-8,-3,0,-1,-3,-8,-9,3,-8,6,-3,-8,-6,7,6,-2,3,-7
040 -13,  -1,3,3,4,1,0,4,-1,-6,0,8,0,1,-5,-4,-1,-9,-7,2,2
050 -15,  3,6,1,-3,-5,6,-9,6,3,-9,-6,-3,-6,-6,6,2,-7,-6,0,-7
060 -13,  1,4,2,-1,1,1,1,-7,-8,-9,-8,-7,-9,1,1,0,1,2,1,-3

READY

*
```

This Fortran program uses Gomory's method to solve both the pure and mixed integer programming problem. The user can actively interface with the program by changing the method of picking the new constraints and by being able to add certain information to the problem in an effort to speed up convergence.

INSTRUCTIONS

To use this program, formulate the problem to be solved according to the following standard:

Minimize

$$a_{oo} + \sum_{i=1}^{n} a_{oi} x_i$$

subject to

$$a_{jo} \geq \sum_{i=1}^{n} a_{ji} x_i \quad , \quad j=1, \ldots, m$$

So then n is the number of variables and m is the number of constraints. All of the coefficients a_{ij} should be integer.

Establish a data file with line numbers in the following format:

line # , m, n

line # , a_{oo}, a_{o1}, ... , a_{on}

line # , a_{mo}, a_{m1}, ... , a_{mn}

On executing the program, the user is asked to supply the method to be used, any changes to be made to the data, and when to print out the iteration log. Every five times the log is printed out, the user can decide if he wants to stop, continue the solution, or restart the problem from the beginning. If he chooses to continue, he can change the printing of the iteration log and the method to be used. The following items should be supplied when requested by the program:

- The name of the data file
- LSTART — the iteration when the log should be first printed out
- LOFTEN — how often should the log be printed

- LMUCH — how much of the log does the user wish to see

  LMUCH = 1) print the entire log
  2) do not print the values of the variables

- METHOD — the method to be used in choosing the new constraints

  Methods 1 through 5 are for the pure integer case. Method 6 is the mixed case.

  METHOD = 1 generate the new constraint from the row with the greatest RHS fractional part.

  2 generate the new constraint from the row with the smallest average fractional part of the coefficients.

  3 generate the new constraint from the row with the smallest ratio of the RHS fractional part and the average fractional part of the coefficients

  4 Use the Euclidian algorithm to generate the new constraint from the linear combination of two rows that are likely to produce a deep cut in the axis along which the objective function decreases most slowly.

  5 Use the rows cyclicly to generate the new constraints

  6 This method is for the mixed case. Its selection will cause the program to ask for the number of variables constrained to integer values and their indices. Care should be used with this method to avoid changing the problem by specifying different variables to be integer at the various decision points.

- CHANGES — The number of changes to be made to the data before starting. If there are changes to be made, the user will be asked

  VARIABLE #, VALUE

  for each change. The permissible responses are:

  (a) a variable number (1 through N) and the value at which it is fixed. This option eliminates the variable and its column from any manipulation by the program.

  (b) 0, X. This option appends the constraint: objective value $\geq$ X to the problem.

  (c) -1, X. This option appends the constraint: objective value $\leq$ X to the problem.

- NEXT — after printing the iteration log five times, the user is given the choice of

  1) Stop
  NEXT = 2) Continue
  3) Begin the problem again

NOTE: If the objective function has not changed for several iterations, the program may be looping indefinitely through the same points. If this occurs, another method should be tried. Also, another method could profitably be tried whenever the change in the objective function becomes small.

RESTRICTIONS

The program currently cannot handle more than 14 variables and approximately 16 constraints. Also, Gomory's method is known to converge slowly, especially for large problems. However, by wisely using the interface capabilities of this program, many problems can be solved with reasonable effort.

REFERENCES

Gomory, R.E., "An Algorithm for Integer Solutions to Linear Programs," in Recent Advances in Optimization Theory, Ed: Graves & Wolfe, McGraw-Hill, 1963.

Trauth & Woosley, Mesa, An Heuristic Integer Programming Technique, Sandia Laboratories, Albuquerque, New Mexico.

SAMPLE PROBLEM

Minimize $-x_3 - x_4 - x_5$

subject to

$$180 \geq 20 x_1 + 30 x_2 + x_3 + 2 x_4 + 2 x_5$$

$$150 \geq 30 x_1 + 20 x_2 + 2 x_3 + x_4 + 2 x_5$$

$$0 \geq -60 x_1 + x_3$$

$$0 \geq -75 x_2 + x_4$$

$$1 \geq x_1$$

$$1 \geq x_2$$

SAMPLE SOLUTION

The data was saved in the file INTIN and is listed below:

```
*LIST INTIN

010 6 5
020 0 0 0 -1 -1 -1
030 180 20 30 1 2 2
040 150 30 20 2 1 2
050 0 -60 0 1 0 0
060 0 0 -75 0 1 0
070 1 1 0 0 0 0
080 1 0 1 0 0 0

READY
```

It was first specified to use Method 3 on the original problem and to print the entire log every iteration starting from iteration 40.

```
*RUN

INTLP

1=PRINT INSTRUCTIØNS, 0=DØNT
= 0
DATA FILE NAME
= INTIN
LSTART,LØFTEN,LMUCH,METHØD,CHANGES
= 40 1 1 3 0

ITERATIØN   40 ØBJ= -8.64583330E+01 DETERM.=  1.44E+03
X( 2)=   5.41666664E-01
X( 1)=   2.29166666E-01
X( 3)=   1.37500000E+01
X( 4)=   4.06250000E+01
X( 5)=   3.20833330E+01

ITERATIØN   41 ØBJ= -8.58305531E+01 DETERM.=  3.80E+03
X( 2)=   6.63108736E-01
X( 1)=   4.93662439E-01
X( 3)=   2.96197464E+01
X( 4)=   4.97331553E+01
X( 5)=   6.47765172E+00

ITERATIØN   42 ØBJ= -8.50000000E+01 DETERM.=  3.50E+03
X( 2)=   1.00000000E+00
X( 1)=   5.00000000E-01
X( 4)=   5.50000000E+01
X( 3)=   3.00000000E+01

ITERATIØN   43 ØBJ= -8.50000000E+01 DETERM.=  3.47E+03
X( 2)=   9.94236305E-01
X( 1)=   4.95677233E-01
X( 4)=   5.52593656E+01
X( 3)=   2.97406340E+01

ITERATIØN   44 ØBJ= -8.50000000E+01 DETERM.=  3.44E+03
X( 2)=   9.88372087E-01
X( 1)=   4.91279069E-01
X( 4)=   5.55232553E+01
X( 3)=   2.94767442E+01
```

After examing the results, it appeared that x_2 might end up to be 1 and that the optimal objective value might be -85. The problem was restarted with x_2 constrained to be 1 and the objective function constrained to be greater than or equal to -85. Again, Method 3 was used and the heading only of the log was to be printed every 20 iterations.

```
NEXT
= 3
LSTART,LØFTEN,LMUCH,METHØD,CHANGES
= 20,20 2 3 2
VARIABLE #,VALUE
= 0 -85
VARIABLE #,VALUE
= 2 1

ITERATIØN   20 ØBJ= -7.98113203E+01 DETERM.=   5.30E+01


ØPTIMUM


ITERATIØN   28 ØBJ= -7.60000000E+01 DETERM.=   1.00E+00
X( 4)=          54
X( 1)=           1
X( 3)=          22
```

A feasible integer solution (1, 1, 22, 54, 0) with an objective value of -76 was discovered. However, because of the constraint imposed on x_2, it might not be optimal. Any solution must have an objective value between -76 and -85. The problem was restarted using Method 3 and including these two constraints on the objective value. The heading only of the log was to be printed every 20 iterations.

```
NEXT
= 3
LSTART,LØFTEN,LMUCH,METHØD,CHANGES
= 20 20 2 3 2
VARIABLE #,VALUE
= 0 -85
VARIABLE #,VALUE
= -1 -76

ITERATIØN   20 ØBJ= -8.40000000E+01 DETERM.=   4.00E+01

ITERATIØN   40 ØBJ= -8.26955109E+01 DETERM.=   6.19E+03

ITERATIØN   60 ØBJ= -8.17365141E+01 DETERM.=   3.05E+04

ITERATIØN   80 ØBJ= -8.17358055E+01 DETERM.=   4.57E+05

ITERATIØN  100 ØBJ= -8.17354317E+01 DETERM.=   4.56E+05
```

Good progress was made for 40 iterations, but then little change occurred. The iterations were continued by using Method 5. The heading only of the log was printed every 15 iterations.

```
NEXT
= 2
LØFTEN,LMUCH,METHØD
= 15 2 5

ITERATIØN 115 ØBJ= -8.00000000E+01 DETERM.=  6.00E+01

ITERATIØN 130 ØBJ= -7.98245611E+01 DETERM.=  3.42E+03

ITERATIØN 145 ØBJ= -7.78947363E+01 DETERM.=  3.42E+03

ØPTIMUM


ITERATIØN 160 ØBJ= -7.60000000E+01 DETERM.=  1.00E+00
X( 2)=          1
X( 4)=         54
X( 3)=         22
X( 1)=          1

NEXT
= 1

PRØGRAM STØP AT 1710
*
```

It was verified that the optimal solution was in fact the feasible solution found earlier.

This BASIC program schedules n jobs in a job shop with m machines. Four schedules are produced using different heuristic dispatching rules. The general job-shop scheduling problem is defined as follows: given a set of n jobs that are to be processed by m machines where the sequence and the process times are known, obtain a schedule that minimizes the total flow time. This program makes the following assumptions.

1. All jobs are available for scheduling at the same time.
2. A machine can process only one job at a time.
3. A job on a machine must be completed before the next job can enter the machine.
4. The sequence and processing time of each operation is known.
5. The sequence and processing times are independent of the schedule.
6. Only a finite number of jobs are considered without regard for future jobs.
7. No alternate sequences are permitted.
8. Process time includes all transportation and setup time required by an operation.

This program uses four common dispatching rules to calculate schedules.

The popularity of dispatching rules arises from their direct application in the job-shop. A machine operator can readily make the decision as to which job he next works on. He chooses the job with the highest priority from only among the jobs waiting to be processed by his machine. As soon as one job is finished, he must immediately begin work on the next job designated by the dispatching rule. By definition, a dispatching rule does not allow the operator to hold his machine idle if his machine has a non-empty queue. This is a weakness of dispatching rules since, for certain conditions, it is best to leave a machine idle if a critical job will arrive at this machine before the job chosen by the dispatching rule can be completed. The greatest benefit of dispatching rules is that no paper work is required to generate an overall schedule for all jobs on all machines. Decisions can be made locally at the machine centers.

The dispatching rules used to select the next job to be processed on a particular machine are:

1. Select the job with minimum processing time required on that machine.
2. Select the job with minimum total remaining processing time for all unscheduled operations.
3. Select the job with maximum processing time required on that machine.
4. Select the job with maximum total remaining processing time for all unscheduled operations.

INSTRUCTIONS

The data is entered in DATA statements beginning in line 6000. Sample data is already imbedded in the program in lines 6000 through 6190. Delete the sample data by typing DELETE 6000, 6190; then enter your data in the following order:

- The number of jobs to be processed.
- The number of machines.
- The sequence matrix S by rows where S_{ij} is the sequence of the i^{th} job on the j^{th} machine.
- The process time matrix P by rows where P_{ij} is the process time required by the i^{th} job on the j^{th} machine.

REFERENCE

Blick, R.G., Heuristics for Scheduling the General n/m Job-Shop Problem. General Electric Company Report No. 69-C-162, April 1969, GE Technical Information Exchange, P.O. Box 43, Bldg. 5, Schenectady, N.Y. 12301

SAMPLE PROBLEM

The 6-job, 6-machine problem is imbedded in the sample data. The optimal total time for this problem is 55. Note that the best schedule found by this program has a total time of 61.

SAMPLE EXECUTION

```
*RUN


JOB-SHOP SIMULATOR

INSTRUCTIONS:
THIS PROGRAM SIMULATES A JOB SHOP FUNCTIONING UNDER
FOUR DIFFERENT DISPATCHER RULES.
ENTER DATA IN THE FOLLOWING ORDER BEGINNING
IN LINE 6000:
    * THE NUMBER OF JOBS TO BE PROCESSED,
    * THE NUMBER OF MACHINES,
    * THE SEQUENCE MATRIX S(I,J) BY ROWS
      WHERE S(I,J) IS THE SEQUENCE OF THE I'TH
      JOB ON THE J'TH MACHINE,
    * THE PROCESS TIME MATRIX P(I,J) BY ROWS
      WHERE P(I,J) IS THE PROCESS TIME REQUIRED
      BY THE I'TH JOB ON THE J'TH MACHINE.

SAMPLE DATA IS ALREADY IN LINES 6000-6190.
```

TØ EXECUTE THE PRØGRAM:
TYPE 'DELETE 6000-6190' (DELETES SAMPLE DATA)
ENTER YØUR DATA
TYPE 'RUN'

THE FØLLØWING IS A SAMPLE EXECUTIØN USING THE SAMPLE DATA.

SCHEDULING 6-JØBS ØN 6-MACHINES

INPUT
SEQUENCE MATRIX-S

MACHINE JØB	1	2	3	4	5	6
1	2	3	1	4	6	5
2	5	1	2	6	3	4
3	4	5	1	2	6	3
4	2	1	3	4	5	6
5	5	2	1	6	3	4
6	4	1	6	2	5	3

PRØCESSING TIME MATRIX-P

MACHINE JØB	1	2	3	4	5	6
1	3	6	1	7	6	3
2	10	8	5	4	10	10
3	9	1	5	4	7	8
4	5	5	5	3	8	9
5	3	3	9	1	5	4
6	10	3	1	3	4	9

SCHEDULE ØF JØB START TIMES

DISPATCHING RULE: CHØSE JØB FØR MINIMUM PRØCESSING TIME ØN CURRENT MACHINE

MACHINE JØB	1	2	3	4	5	6
1	1	8	0	14	41	23
2	74	14	22	84	54	64
3	25	34	1	6	47	15
4	8	3	15	21	24	32
5	45	22	6	48	36	41
6	15	0	36	3	32	6

TØTAL TIME = 88 AVERAGE TIME = 52.66667

SCHEDULE ØF JØB START TIMES

DISPATCHING RULE: CHØSE JØB FØR MINIMUM
TØTAL TIME FØR ALL REMAINING PRØCESSING

MACHINE / JØB	1	2	3	4	5	6
1	13	19	9	25	46	41
2	56	8	20	66	36	46
3	27	36	10	15	52	19
4	8	3	15	20	24	32
5	36	16	0	39	19	27
6	16	0	36	3	32	6

TØTAL TIME = 70 AVERAGE TIME = 49.83333

SCHEDULE OF JØB START TIMES

DISPATCHING RULE: CHØSE JØB FØR MAXIMUM
PRØCESSING TIME ØN CURRENT MACHINE

MACHINE / JØB	1	2	3	4	5	6
1	25	28	24	34	61	58
2	47	0	14	57	19	35
3	28	37	9	14	42	18
4	13	8	19	24	29	45
5	58	16	0	61	37	54
6	37	13	53	18	49	26

TØTAL TIME = 67 AVERAGE TIME = 57.83333

SCHEDULE OF JØB START TIMES*

DISPATCHING RULE: CHØSE JØB FØR MAXIMUM
TØTAL TIME FØR ALL REMAINING PRØCESSING

MACHINE / JØB	1	2	3	4	5	6
1	6	16	5	22	50	29
2	42	0	15	52	20	32
3	18	27	0	5	43	9
4	13	8	20	29	35	46
5	52	22	6	56	30	42
6	28	13	60	16	56	19

TØTAL TIME = 61 AVERAGE TIME = 55.83333

READY
*

This Fortran algorithm solves the minimal cost circulation network problem using the out-of-kilter algorithm.

INSTRUCTIONS

For each arc in the network the following data is required:

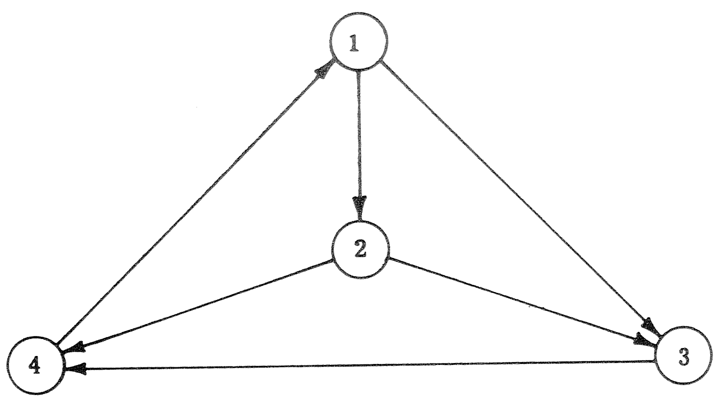
- I, the starting node,
- J, the ending node,
- the unit cost, and
- an upper and lower bound for the arc flow.

The algorithm calculates the integral circulation that satisfies the bounds and minimizes the total cost, or indicates the infeasibility of the problem. The problem is bounded by dimension statements to no more than 100 arcs.

On execution the program will ask for the name of the data file. The data should be entered in this file, one line for each arc in the following order:

line number, I, J, cost, upper bound, lower bound

SAMPLE PROBLEM



arc	cost	upper bound	lower bound
1-2	1	4	3
1-3	2	1	1
2-3	2	4	2
2-4	6	2	1
3-4	5	3	2
4-1	0	∞	-∞

SAMPLE SOLUTION

The data was entered in a file, DATA.

```
*LIST DATA

10 1 2 1 4 3
20 1 3 2 1 1
30 2 3 2 4 2
40 2 4 6 2 1
50 3 4 5 3 2
60 4 1 0 999999 -999999

READY

*RUN KILTER
ØUT ØF KILTER ALGØ.
DATA FILE NAME
= DATA
NETWØRK STATUS AND MINIMUM CØST FLØW

     I      J       CØST      U.BND      L.BND       FLØW

     1      2          1          4          3          3

     1      3          2          1          1          1

     2      3          2          4          2          2

     2      4          6          2          1          1

     3      4          5          3          2          3

     4      1          0     999999    -999999          4

PRØGRAM STØP AT 2470
*
```

This BASIC program maximizes objective function using the 2-phase method. It automatically supplies slack, surplus, and artificial variables as required for the solution.

INSTRUCTIONS

To use this program enter the data starting at line 10, in this order:

Coefficients of each of the problem variables (including zeroes for variables not appearing) in each restriction, starting with the first restriction and proceeding in order until all coefficients of all restrictions have been entered in data statements, then the elements of the "B" vector (the constants comprising the right side of all restrictions) in the same order as the restrictions; finally, the coefficients of the (linear) objective function, in the same order as used in the restrictions, including zeroes if needed; then type RUN. Additional instructions are found in the listing.

The program will ask for M and N where

- M is the number of constraints
- N is the number of variables

The program will then ask for "less than," "equals," and "greater thans" where

- "less thans" are the number of < = restrictions
- "equals" are the number of = restrictions
- "greater thans" are the number of > = restrictions

RESTRICTIONS

1. Maximum size is 18 x 30. Large problems may result in excessive running time and large roundoff error.
2. Before using this program, arrange all constraints (i.e. linear restrictions on the problem variables) as follows:
   A. The "less than or equal" inequalities.
   B. The strict equalities.
   C. The "greater than or equal" inequalities.

SAMPLE PROBLEM

Maximize the function 30 x X1 + 45 x X2 while satisfying the following constraints:

```
            X1            <   6000
                      X2  <   4000
      2.5 x X1 + 2.0 x X2 < = 2400
            X1            >= 1000
                      X2  >= 1000
      3.0 x X1 -      X2  > = 0
      2.5 x X1 + 2.0 x X2 > = 10000
```

The data would be entered starting at line number 10, in the following order.

First:	Coefficients of each of the problem variables in each constraint.
Ex :	10 Data 1, 0, 0, 1, 2.5, 2, 1, 0, 0, 1, 3, -1, 2.5, 2
Then:	The values on the right side of the constraints.
Ex :	11 Data 6000, 4000, 24000, 1000, 1000, 0, 10000
Last :	The coefficients of the objective function.
Ex :	12 Data 30, 45

As the program runs, the values 7, 2 would be supplied for M, N. When less thans, equals, and greater thans are called for, 3, 0, 4 would be supplied.

NOTE: The problem solution is X1 = 6000, X2 = 4000

SAMPLE SOLUTION

```
*10 DATA 1,0,0,1,2.5,2,1,0,0,1,3,-1,2.5,2
*11 DATA 6000,4000,24000,1000,1000,0,10000
*12 DATA 30,45
*RUN


L I N P R Ø


TYPE '2' FØR ØUTPUT ØF ØF TABLEAUS AND BASIS AT EACH ITERATIØN,
     '1' FØR THE BASIS ØNLY, ØR
     '0' FØR JUST THE SØLUTIØN.
WHICH ?2

WHAT ARE M AND N ØF THE DATA MATRIX ?7,2

HØW MANY 'LESS THANS','EQUALS','GREATER THANS' ?3,0,4

       YØUR VARIABLES     1 THRØUGH    2
    SURPLUS VARIABLES     3 THRØUGH    6
      SLACK VARIABLES     7 THRØUGH    9
 ARTIFICIAL VARIABLES    10 THRØUGH   13


TABLEAU AFTER    0 ITERATIØNS
          1          0          0          0          0
          0          1          0          0          0
          0          0          0       6000

          0          1          0          0          0
          0          0          1          0          0
          0          0          0       4000

        2.5          2          0          0          0
          0          0          0          1         -0
          0          0          0      24000

          1          0         -1          0          0
          0          0          0          0          1
          0          0          0       1000

          0          1          0         -1          0
          0          0          0          0          0
          1          0          0       1000

          3         -1          0          0         -1
          0          0          0          0          0
          0          1          0          0

        2.5          2          0          0          0
         -1          0          0          0          0
          0          0          1      10000

        -30        -45          0          0         -0
          0          0          0          0          0
          0          0          0          0

       -6.5         -2          1          1          1

          1          0          0          0          0-
          0          0          0          0
```

```
        BASIS BEFORE ITERATION     1
        VARIABLE          VALUE
            7              6000
            8              4000
            9             24000
           10              1000
           11              1000
           12                 0
           13             10000

OBJECTIVE FUNCTION VALUE                 0

        BASIS BEFORE ITERATION     2
        VARIABLE          VALUE
            7              6000
            8              4000
            9             24000
           10              1000
           11              1000
            1                 0
           13             10000

OBJECTIVE FUNCTION VALUE                 0

        BASIS BEFORE ITERATION     3
        VARIABLE          VALUE
            7          5666.667
            8              3000
            9          21166.67
           10          666.6667
            2              1000
            1          333.3333
           13          7166.667

OBJECTIVE FUNCTION VALUE          4166.667

        BASIS BEFORE ITERATION     4
        VARIABLE          VALUE
            7              5000
            8              1000
            9             15500
            4              2000
            2              3000
            1              1000
           13              1500

OBJECTIVE FUNCTION VALUE             10500

        BASIS BEFORE ITERATION     5
        VARIABLE          VALUE
            7          4823.529
            8          470.5882
            9             14000
            4          2529.412
            2          3529.412
            1          1176.471
            3          176.4706
```

ØBJECTIVE FUNCTIØN VALUE 194117.6

BASIS BEFØRE ITERATIØN 6

VARIABLE	VALUE
7	4666.667
6	1333.333
9	12666.67
4	3000
2	4000
1	1333.333
3	333.3333

ØBJECTIVE FUNCTIØN VALUE 220000

ANSWERS:

VARIABLE	VALUE
5	14000
6	13000
9	999.9993
4	3000
2	4000
1	6000
3	5000

ØBJECTIVE FUNCTIØN VALUE 360000

DUAL VARIABLES:

CØLUMN	VALUE
3	0
4	0
5	0
6	0
7	30
8	45
9	0

TABLEAU AFTER 6 ITERATIØNS

0	0	0	0	1
0	3	-1	0	0
0	-1	0	14000	
0	0	0	0	0
1	2.5	2	0	-8.44399 E-8
0	1.49012 E-8	-1	13000	
0	0	0	0	0
0	-2.5	-2	1	2.03649 E-7
0	3.72529 E-8	0	999.9993	
0	0	0	1	0
0	-3.35276 E-8	1	0	2.98023 E-8
-1	1.11759 E-8	0	3000	
0	1	0	0	0
0	-3.35276 E-8	1	0	2.98023 E-8
0	1.11759 E-8	0	4000	
1	0	0	0	0
0	1	0	0	1.24176 E-8
0	0	0	6000	

```
0          0           1          0          0
0          1           0          0         -1
0          0           0       5000

0          0           0          0          0
0         30          45          0   2.17557 E-6
0  9.53674 E-7  4.76837 E-7  360000

READY
*
```

This Fortran program computes the optimum solutions for linear programming problems. Specifically, a linear objective function is maximized (or minimized) subject to a set of linear constraints. Letting X_j refer to the structural variables, S_i to slack variables, and C_j and D_i to the objective function coefficients of the structural and slack variables, the objective function is expressed as:

$$Z = \sum_{J=1}^{n} C_j X_j + \sum_{i=1}^{m} D_i S_i$$

where:

- n is the number of structural variables
- m is the number of constraints

Each constraint is of the form:

$$\sum_{j=1}^{n} A_{ij} X_j \leq (\text{or} = \text{or} \geq) \; B_i$$

where A_{ij} refers to the structural variable coefficients and B_i refers to the requirements column or right-hand side values.

The results of the LP solution consist of an optional iteration log, the basic variable results and the non-basic variable results. The structural variables are identified by the number. 1 through N. The slack variables associated with each constraint are identified by the numbers 101 through 100+M. Variable identifications 999 and 200+(M+1) are added by the program to handle the "greater than" constraints and normally do not appear in the basic and non-basic variable results.

The iteration log, which is selected at RUN time, consists of the iteration count, the identifications of the variables entering and leaving the basis, and the current value of the objective function.

The basic variable results consist of the variable identification, the objective function coefficient and the answer (value of the variable) for each variable in the final solution.

The non-basic variable results consist of the variable identification, the objective function coefficient, and the answer (reduced cost or shadow price) for each variable not in the final solution.

An infeasible condition is indicated by a slack variable associated with an "equal to" constraint and/or the slack variable identified as 200+(M+1) appearing in the basic variable results at other than a zero value. This means that the problem contains two or more constraints that cannot be simultaneously satisfied.

The appearance of a negative identification for a structural variable in the non-basic results indicates that this variable is unbounded. None of the problem constraints restrict this variable from entering the solution at an infinite value.

INSTRUCTIONS

At execution, the program asks for the data file name. It is assumed this file has been built without line numbers. The first line of data is used for problem identification. The second line contains m (the number of constraints), n (the number of structural variables) and either MAX or MIN indicating whether the problem is a maximization or minimization problem. The next line is the objective function coefficients, C_j. The data for each constraint follows in the following order:

$$A_{ij} \quad A_{i2} \; . \; . \; . \; A_{in} < \; B_i \; D_i$$

where " < " can be replaced by " > " or "=" to indicate the sense of the constraint.

The data for additional problems may also be included sequentially in the same file.

RESTRICTIONS

$m \leq 30$
$n \leq 50$

where

- m is the number of constraints
- n is the number of structural variables

SAMPLE PROBLEM

Maximize the function

$$Z = -X - Y - Z + W$$

subject to the constraints

$$\begin{aligned} -X - 1Y + 4Z - 1W &\geq 5 \\ 2Y - Z - W &\geq 4 \\ X - 3Y + 4Z - 4W &\leq 5 \\ X - 2Y &\leq 3 \end{aligned}$$

SAMPLE SOLUTION

The data for this problem was stored in the file LPDATA.

```
*LIST LPDATA

TEST PRØBLEM FØR LP
 4  4  MAX
-1 -1 -1  1
-1 -1  4 -1  >  5  0
 0  2 -1 -1  >  4  0
 1 -3  4 -4  <  5  0
 1 -2  0  0  <  3  0

READY

*RUN
G E - 6 0 0   L P   P R Ø G R A M
DATA FILE NAME
= LPDATA
TEST PRØBLEM FØR LP
    4 RØWS X  4CØLS
1=PRINT ITERATIØN LØG, 2=SUPPRESS
= 2
ØBJ. FUNCT.     =    -5.00000E+00

  BAS VAR     ØBJ. CØEFF.         RESULT
        2     -1.00000E+00     3.00000E+00
        3     -1.00000E+00     2.00000E+00
      103<     0.              6.00000E+00
      104<     0.              9.00000E+00
N. BAS VAR     ØBJ. CØEFF.         RESULT
        1     -1.00000E+00     1.42857E+00
        4      1.00000E+00     1.42857E-01
      101>     0.              4.28571E-01
      102>     0.              7.14286E-01

PRØGRAM STØP AT 2490
*
```

This Fortran program maximizes an unconstrained multivariate function. This program is a simple hill-climbing strategy based on a method due to Fletcher and Reeves using conjugate gradients. The program is dimensioned for five variables and requires the partial derivatives of the function to be available.

The program will find the optimum of a quadratic function of n variables in n unidirectional searches. It can also be used to advantage if the function can be approximated by a quadratic function near the optimum.

INSTRUCTIONS

After the user logs onto the system, he must type in the function to be maximized and its partial derivatives. The function must have statement number 1100. Statement numbers 1101-1105 are available for long functions. Statement numbers 1120-1140 are available for the partial derivatives. For example:

```
1100 F      = [a function of the vector X]
1120 DR(1) = [First partial derivative of F]
1130 DR(2) = [Second partial derivative of F]
```

The variable F and the array DR must be used for the function and its partial derivatives.

Several quantities must be input to the program from the terminal. These input quantities have the following meaning:

- N is the number of variables (up to five)
- SX is the minimum allowable stepsize
- CSS is the initial (and current) stepsize
- NSTEP is the maximum number of evaluations allowed
- DSMIN is the minimum allowable RMS value of gradient

Two questions will be asked by the program. The user's answers will determine the type of output. The questions are:

DO YOU JUST WANT THE FINAL RESULTS? (YES OR NO)

=

HOW OFTEN DO YOU WANT TO SEE THE RESULTS?

=

An answer of YES to the first question will result in just the final results being printed. An answer of NO will result in the second question being asked. The answer to the second question should merely consist of a number. For example, if the user wants to see the results after every fourth iteration, the number 4 should be entered as the answer.

The following statements will be issued by the program informing the user when to enter the input quantities.

ENTER N, SX, CSS, NSTEP, DSMIN

=

ENTER THE INITIAL VALUES FOR THE VARIABLES

=

REFERENCE

Fletcher, R. and Reeves, "Function Minimization by Conjugate Gradients," Computer Journal, Vol. 7, 1964, 149-154

SAMPLE PROBLEM

$$\text{Maximize } F = -\left[\frac{(x_1 - 1)^2}{2} + \frac{(x_2 - 2)^2}{4} + \frac{(x_3 - 3)^2}{8}\right]$$

using as starting conditions $X_1 = 0$, $X_2 = 0$, and $X_3 = 0$.

The surface described by F is a 3-dimensional ellipse. The maximum of F is at (1, 2, 3) with a value of 0.

SAMPLE SOLUTION

```
*1100 F=-((.5*(X(1)-1)**2)+(.25*(X(2)-2)**2)+(.125*(X(3)-3)**2))
*1120 DR(1)=-(X(1)-1)
*1130 DR(2)=-((X(2)-2)/2)
*1140 DR(3)=-((X(3)-3)/4)
*RUN
DØ YØU JUST WANT THE FINAL RESULTS? (YES ØR NØ)
= NØ
HØW ØFTEN DØ YØU WANT TØ SEE THE RESULTS?
= 1
ENTER N,SX,CSS,NSTEP,DSMIN
= 3,.01,1,20,.001
ENTER THE INITIAL VALUES FØR THE VARIABLES
= 0,0,0
```

```
***MAXIMIZATION USING CONJUGATE GRADIENTS***

N= 3     SX= .010    CSS= 1.00    NSTEP=  20    GRAD> 0.00100

INITIAL VALUES
F =-0.262500E+01     X(1) = 0.                  X(2) = 0.
   X(3) = 0.

STEP NO.    1
F =-0.134434E+01     X(1) = 0.624695E+00       X(2) = 0.624695E+00
   X(3) = 0.468521E+00

STEP NO.    2
F =-0.703926E+00     X(1) = 0.124939E+01       X(2) = 0.124939E+01
   X(3) = 0.937043E+00

STEP NO.    3
F =-0.703754E+00     X(1) = 0.187409E+01       X(2) = 0.187409E+01
   X(3) = 0.140556E+01

STEP NO.    4
F =-0.134383E+01     X(1) = 0.249878E+01       X(2) = 0.249878E+01
   X(3) = 0.187409E+01

STEP NO.    5
F =-0.623810E+00     X(1) = 0.156190E+01       X(2) = 0.156190E+01
   X(3) = 0.117143E+01

STEP NO.    6
F =-0.104881E+00     X(1) = 0.102286E+01       X(2) = 0.226680E+01
   X(3) = 0.216658E+01

STEP NO.    7
F =-0.372542E+00     X(1) = 0.483823E+00       X(2) = 0.297140E+01
   X(3) = 0.316173E+01

STEP NO.    8
F =-0.948478E-01     X(1) = 0.936768E+00       X(2) = 0.237939E+01
   X(3) = 0.232553E+01

STEP NO.    9



F =-0.487946E-01     X(1) = 0.104535E+01       X(2) = 0.172788E+01
   X(3) = 0.348377E+01

STEP NO.   10
F =-0.562145E+00     X(1) = 0.115394E+01       X(2) = 0.107637E+01
   X(3) = 0.464201E+01

STEP NO.   11
F =-0.277556E-15     X(1) = 0.100000E+01       X(2) = 0.200000E+01
   X(3) = 0.300000E+01
THIS RUN IS TERMINATED BECAUSE OF SMALL GRADIENT    1.8250121E-08

FINAL RESULTS OCCUR AT

STEP NO.   11
F =-0.277556E-15     X(1) = 0.100000E+01       X(2) = 0.200000E+01
   X(3) = 0.300000E+01
***END***

PROGRAM STOP AT 0
*
```

SAMPLE PROBLEM

Maximize $F = -100\ (x_2 - x_1)^2 \quad - \quad (1 - x_1)^2$

using as starting conditions $x_1 = -1.2$ and $x_2 = 1.$

The surface described by F is a banana-shaped ridge and has properties which make the attainment of the optimum very difficult. There are large changes in the gradient between iterations, and the parameters of the logic have to be carefully selected so that the logic does not stop prematurely or take too many steps.

The maximum of F is at (1, 1) with a value of 0.

SAMPLE SOLUTION

```
*1100 F=-100*(X(2)-X(1))**2-(1-X(1))**2
*1120 DR(1)=200*(X(2)-X(1))+2*(1-X(1))
*1130 DR(2)=-200*(X(2)-X(1))
*RUN
DØ YØU JUST WANT THE FINAL RESULTS? (YES ØR NØ)
= NØ
HØW ØFTEN DØ YØU WANT TØ SEE THE RESULTS?
= 30
ENTER N,SX,CSS,NSTEP,DSMIN
= 2,.0001,.01,300,.001
ENTER THE INITIAL VALUES FØR THE VARIABLES
= -1.2,1


   ***MAXIMIZATIØN USING CØNJUGATE GRADIENTS***

N= 2     SX= .000    CSS= 0.01    NSTEP= 300    GRAD> 0.00100

INITIAL VALUES
F =-0.488840E+03     X(1) =-0.120000E+01     X(2) = 0.100000E+01

STEP NØ.  30
F =-0.319273E+03     X(1) =-0.986815E+00     X(2) = 0.788926E+00

STEP NØ.  60
F =-0.185796E+03     X(1) =-0.773631E+00     X(2) = 0.577852E+00

STEP NØ.  90
F =-0.884093E+02     X(1) =-0.560446E+00     X(2) = 0.366777E+00

STEP NØ. 120
F =-0.271125E+02     X(1) =-0.347261E+00     X(2) = 0.155703E+00

STEP NØ. 150
F =-0.190559E+01     X(1) =-0.134077E+00     X(2) =-0.553707E-01
THIS RUN IS TERMINATED BECAUSE ØF SMALL GRADIENT    4.3097395E-04

FINAL RESULTS ØCCUR AT

STEP NØ. 163
F =-0.231593E-09     X(1) = 0.100000E+01     X(2) = 0.999999E+00
***END***

PRØGRAM STØP AT 0
*
```

This Fortran program finds the maximum flow from the source node to the sink node in a directed network using a maximum flow, minimum cut algorithm.

INSTRUCTIONS

On execution, the program will ask for the data file name. The first line of this file is taken for problem identification. The data follows with one line containing the data for each arc in the following order:

line number, starting node, ending node, capacity, starting flow

The network source node should be numbered 1, and the network sink node should be the highest numbered node. Dimension statements limit the highest numbered node to be less than or equal to 25. The program provides the maximum network flow and the flow in each arc that yields the maximum flow.

SAMPLE PROBLEM

Analyze the following network:

Starting Node	Ending Node	Capacity	Starting Flow
1	2	4	0
1	4	1	0
2	3	2	0
3	4	1	0
3	6	1	0
4	3	1	0
4	5	2	0
5	6	3	0

The maximum flow in this network is 3.

SAMPLE SOLUTION

The data was stored in the file DATA.

```
*LIST DATA

0010 MAXFLØW 6 PØINT TEST CASE
0020 1 2 4 0
0030 1 4 1 0
0040 2 3 2 0
0050 3 4 1 0
0060 3 6 1 0
0070 4 3 1 0
0080 4 5 2 0
0090 5 6 3 0

READY

*RUN MAXFLØW
G E - 6 0 0   M A X - F L Ø W / M I N - C U T
DATA FILE NAME
= DATA
TYPE 0 FØR TRACE ØF ITERATIØNS, 1 FØR ØPTIMAL SØL. ØNLY
= 1
   MAXFLØW 6 PØINT TEST CASE

 STATUS ØF NETWØRK AT ØPTIMAL SØLUTIØN

STARTING NØDE  ENDING NØDE  CAPACITY FLØW IN ARC
            1            2      4.00         2.00
            1            4      1.00         1.00
            2            3      2.00         2.00
            3            4      1.00         1.00
            3            6      1.00         1.00
            4            3      1.00         0.
            4            5      2.00         2.00
            5            6      3.00         2.00
MAX FLØW IS    3.0000000E+00

PRØGRAM STØP AT 1890
*
```

This Fortran program maximizes an unconstrained multivariate function. Derivatives are not required. This program consists of two simple maximizing codes, LOGIC1 and LOGIC2, both written as subroutines and an analysis program which calculates the value of the response function and issues a call to one of the subroutines. Both strategies are simple hill-climbers: LOGIC1 implements an Optimum Gradient technique when used in MODE = 1 or a steepest ascent method when MODE = 0; LOGIC2 is based on a Direct Search procedure without pattern moves. The program is dimensioned for 10 adjustable parameters.

INSTRUCTIONS

The user must enter in line numbers 1915-1925 the statements necessary to evaluate the response function R as a function of the vector XV. For example, to maximize $X_1X_2 - X_2^2$ the user could enter

```
1915        R = XV(1) * XV(2) - XV(2) **2
```

During execution the program will ask for the following input quantities:

- NV, the number of variables (up to 10)
- LOGIC, 1 for LOGIC1 or 2 for LOGIC2
- SX, the minimum normalized step and perturbation size
- CSS, the initial normalized step size
- NSTEP, the maximum number of evaluations allowed
- MODE, 0 for Steepest Ascent or 1 for Optimum Gradient
- RANGE, the span of expected variation for i^{th} variable

NOTE: If LOGIC2 is chosen, MODE will not be an input quantity.

The following statements will be issued by the program informing the user when to enter the input quantities:

```
ENTER NV AND LOGIC
=
ENTER THE INITIAL VALUES FOR THE VARIABLES
=
ENTER SX, CSS, NSTEP, AND MODE
=
ENTER THE VALUES FOR THE ARRAY RANGE
=
```

Two other questions will be asked by the program. The user's answers will determine the type of output. The questions are:

DO YOU JUST WANT THE FINAL RESULTS?
ANSWER YES OR NO
=
HOW OFTEN DO YOU WANT TO SEE THE RESULTS?
=

An answer of YES to the first question will result in just the final results being printed. An answer of NO will result in the second question being asked. The answer to the second question should merely consist of a number. For example, if the user wants to see the results after every fourth iteration, the number 4 should be entered as the answer.

SAMPLE PROBLEM

$$\text{Maximize } R = \exp\left(- \left(\frac{B^2 * X_1^2 + X_2^2}{B^2} \right) \right)$$

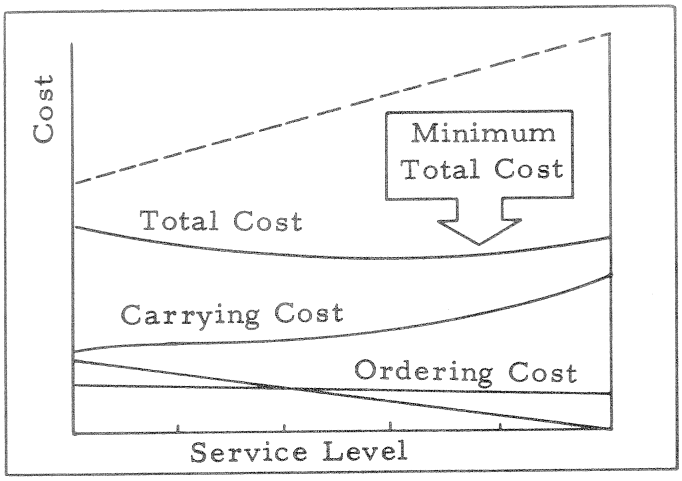
using the starting conditions $X_1 = .25$ and $X_2 = .55$.

The surface described by R is bell-shaped and its contours depend on the parameter B. If B = 1, the contours are circular, otherwise they are ellipses. For this example B = .75. The maximum of R is at (0, 0) with a value of unity.

SAMPLE SOLUTION

```
*1915 B=0.75
*1916 R=EXP(-(B**2*XV(1)**2+XV(2)**2)/B**2)
*RUN
ENTER NV AND LØGIC
= 2,1
ENTER THE INITIAL VALUES FØR THE VARIABLES
= .25,.55
ENTER SX, CSS, NSTEP, AND MØDE
= .01,.1,20,1
ENTER THE VALUES FØR THE ARRAY RANGE
= 1,1
DØ YØU JUST WANT THE FINAL RESULTS?
ANSWER YES ØR NØ
= NØ
HØW ØFTEN DØ YØU WANT TØ SEE THE RESULTS?
= 2

            ØPTIMIZATIØN ØF RESPØNSE USING LØGIC1

INITIAL VALUES
    R = 0.548659E+00     XV(1) = 0.250000E+00     XV(2) = 0.550000E+00
ITERATIØN NØ.   2
    R = 0.537938E+00     XV(1) = 0.250000E+00     XV(2) = 0.560000E+00
ITERATIØN NØ.   4
    R = 0.766650E+00     XV(1) = 0.199612E+00     XV(2) = 0.356451E+00
ITERATIØN NØ.   6
    R = 0.932911E+00     XV(1) = 0.149223E+00     XV(2) = 0.162903E+00
ITERATIØN NØ.   8
    R = 0.988627E+00     XV(1) = 0.988351E-01     XV(2) =-0.306455E-01
ITERATIØN NØ.  10
    R = 0.986576E+00     XV(1) = 0.108835E+00     XV(2) =-0.306455E-01
ITERATIØN NØ.  12
    R = 0.996992E+00     XV(1) = 0.530649E-01     XV(2) =-0.105187E-01
ITERATIØN NØ.  14
    R = 0.996952E+00     XV(1) =-0.384755E-01     XV(2) = 0.297350E-01
ITERATIØN NØ.  16
    R = 0.999264E+00     XV(1) = 0.729471E-02     XV(2) = 0.196081E-01
ITERATIØN NØ.  18
    R = 0.999991E+00     XV(1) = 0.194552E-02     XV(2) =-0.168948E-02
ITERATIØN NØ.  20
    R = 0.999873E+00     XV(1) = 0.194552E-02     XV(2) = 0.831052E-02

                                        *** END ***

PRØGRAM STØP AT 330
*
```

This Fortran program finds the optimum service level for one inventory item. It considers the carrying cost, ordering cost, and loss of profit.



One of the most precious assets for an inventory manager is customer satisfaction. An important motive for the customer's choice of department store is his belief that the merchandise he wants to buy will be in stock, and therefore, department stores normally set high service objectives.

The service level is defined as the ratio between the satisfied demand and the total demand:

$$\text{service level} = \frac{\text{satisfied demand}}{\text{total demand}}$$

A high service level requires an extensive safety stock to handle exceptionally high demands. This results in few lost sales and thus a small loss of profit, but the price is a high investment in stock. On the other hand, a lower service level means a smaller safety stock and a smaller stock investment, but will also result in lost sales and therefore a less profit.

The problem is to find the service level by which the total cost is minimum. The total cost is composed of carrying cost, ordering cost and loss of profit.

The carrying cost is the cost of holding goods in inventory. It is expressed in percent of item value per year and comprises

- Cost of having capital tied up in inventory
- Cost of storage facilities
- Taxes and insurance

The annual carrying cost per item is normally between 8 and 30% of the item value.

The ordering cost is the cost of processing the replenishment order plus the cost of receiving the shipment. The ordering cost is normally between 5 and 100 dollars.

The loss of profit per item in case of shortage is expressed in percent of the selling price. It is normally between 2 and 7% of the selling price.

REFERENCES

This program is an adaptation of modules from the AIMS system. Further information on AIMS can be obtained from the following:

AIMS - Autoadaptive Inventory Management System Time-Sharing Demonstration Programs, Ref. # 00.19.102A, Honeywell Bull Company, Pans, France

The AIMS System, Ref. # 00.11.072A, Honeywell Bull Company, Pans, France

INSTRUCTIONS

The program will request the following input parameters:

- average monthly demands (in number of items)
- deviation of monthly demand
- forecast period in months; it comprises:
  - review time (time between replenishment decisions)
  - lead time (time that elapses between preparation of the order and receipt of merchandise)
- number of orders in the year
- item value (buying price)
- carrying cost as percentage of item value (normally between 8 and 30%)
- lost of profit per item as percentage of selling price (normally between 2 and 7%)
- ordering cost (normally between 5 and 100 dollars)
- selling price per item

The program calculates safety stock and expected shortage as well as the corresponding costs and the annual turnover for six different service levels:

84, 87, 90, 93, 96 and 99%

The lowest total cost determines the optimum service level.

SAMPLE PROBLEM

Determine the optimum service level for an item with an average monthly demand of 100 ± 15. Review time is 1 month and lead time is 2 months. The stock is replenished by 12 orders in the year, the ordering cost is 20 dollars per order and the carrying cost is 25% of item value. The buying price is 30 dollars and selling price is 40 dollars per item. The loss of profit per item in case of shortage is 5% of the selling price.

SAMPLE SOLUTION

The program yields an optimum service level of 93%, for which the total annual cost is 933 dollars. With this service level, the program has calculated the following average stock state that you will find just before receipt of a replenishment order:

Working stock	=	46
Safety stock	=	24
Mean stock	=	70
Shortage	=	7

Just before receipt of a new order, you will find, on an average, an unused stock of 24 items, which is the safety stock.

Sometimes you will find a safety stock of zero, namely when the demand has been greater than the quantity in stock. The average number of items in shortage is 7.

```
*RUN


OPTIMIZATION OF SERVICE LEVEL
INPUT THE FOLLOWING PROBLEM PARAMETERS:

AVERAGE MONTHLY DEMAND (ITEMS)
=100

DEVIATION OF AVERAGE MONTHLY DEMAND (ITEMS)
=15

FORECAST PERIOD (MONTHS)
=3

NUMBER OF ORDERS PER YEAR
=12

BUYING PRICE PER ITEM (DOLLARS)
=30

CARRYING COST (% OF BUYING PRICE)
=25

LOSS OF PROFIT (% OF SELLING PRICE)
=5

ORDERING COST ($/ORDER)
 20

SELLING PRICE PER ITEM (DOLLARS)
=40
*** AVERAGE STOCK STATE JUST BEFORE RECEIPT OF A REPLENISHMENT ORDER ***
SERVICE
LEVEL           0.84      0.87      0.90      0.93      0.96      0.99
-----------------------------------------------------------------------

WORKING
 STOCK          42.       43.       45.       46.       48.       49.

SAFTEY
 STOCK          12.       15.       18.       24.       33.       54.

MEAN
 STOCK          54.       58.       63.       70.       81.       103.

SHORTAGE        16.       13.       10.       7.        4.        1.
                          *** ANNUAL COSTS ***
SERVICE
LEVEL           0.84      0.87      0.90      0.93      0.96      0.99
-----------------------------------------------------------------------

CARRYING
 COST         405.00    435.00    472.00    525.00    607.00    772.00

LOSS OF
PROFIT        383.00    311.00    239.00    167.00     95.00     23.00

ORDERING
 COST         240.00    240.00    240.00    240.00    240.00    240.00

TOTAL
 COST        1028.00    986.00    951.00    932.00    942.00   1035.00
```

```
ANNUAL
TURNØVER   40320.00  41280.00  43200.00  44160.00  46080.00  47040.00

(THE CØSTS ARE IN DØLLARS)


DØ YØU WANT CURVES - ANSWER YES ØR NØ
 YES
CURVE P IS LØSS ØF PRØFIT.
CURVE C IS CARRYING CØST.
CURVE T IS TØTAL CØST.
THE T-CURVE'S MINIMUM DETERMINES THE ØPTIMUM SERVICE LEVEL
AND THE TØTAL CØST CØRRESPØNDING TØ THE GIVEN PARAMETERS.
-SERVICE
LEVEL  0.84  0.87  0.90  0.93  0.96  0.99

 ØLLARS
                                         T
1035.00
           T     T
 984.40
                       T           T
 933.80
                             T
 883.20

 832.60

 782.00
                                         C
 731.40

 680.80

 630.20
                                   C
 579.60

 529.00
                             C
 478.40
                 C     C
 427.80
           P
 377.20

 326.60
                 P
 276.00
                       P
 225.40

 174.80
                             P
 124.20
                                   P
  73.60

  23.00
                                         P
  23.00

DØ YØU WANT TØ INTRØDUCE NEW PARAMETERS - ANSWWER YES ØR NØ
=NØ

NØRMAL TERMINATIØN
```

This BASIC program performs a simple PERT network analysis. For each event in the network, the program will determine the following variables:

- TE, the earliest expected time of completion of the event within the network,
- V, the variance associated with the TE of the event within the network,
- TL, the latest allowable time for the completion of the event within the network without changing the TE of the final event, and
- SLACK, TL - TE

INSTRUCTIONS

Draw a PERT network following the convention that a circle corresponds to a "completion event" and an arrow corresponds to an "activity." In the network analysis two parameters are associated with each time consuming activity: a mean and a variance.

The above mentioned variable TE must not be confused with T(E). The value T(E) is the expected amount of time required for the completion of a single activity, independent of what has occurred before it. T(E) is defined as T(E) = (A + 4*M + B) /6, where

- A is the most optimistic time for completion of the activity.
- B is the most pessimistic time for completion of the activity.
- M is the most likely time for completion of the activity.

The data for T(E) of each event should be entered as described below. The same nomenclature is used with V and V(E).

Variance = V(E) = ((B-A)/6)2, where A and B are defined as above.

In this program no event can have more than two immediate predecessor events or more than two immediate successor events. To use this program with more complex networks containing more than two immediate predecessor or successor events, enter a numbered "dummy" event that has a T(E) = 0 and a V(E) = 0.

For example, if events 1, 2, and 3 precede event 5, enter a "dummy" event 4, of 0 time for completion and 0 variance, such that events 1 and 2, or 1 and 3, or 2 and 3 precede event 4. Then event 5 will have only two preceding events, as allowed by the program (i.e., event 5 will be preceded by 1 and 4, or 2 and 4, or 3 and 4).

Start entering the data. In line 3000, enter T, the total number of events in the network, including dummy events.

Then beginning with the final event as Number One (1), enter in line 3001 the following eight pieces of information about the event in the order indicated. Repeat this sequence, entering the data for each of the remaining events on a single line. Increment line numbers by 1 after line 3001.

1. The number of the event's first immediate predecessor event.
2. The T(E) associated with the activity bounded by this completion event and its first immediate predecessor event.
3. The variance V(E) associated with the activity bounded by this completion event and its first immediate predecessor event.
4. The number of the event's second immediate predecessor event. If none, enter 0.
5. The T(E) associated with the activity bounded by this completion event and its second immediate predecessor event. If none, enter 0.
6. The variance V(E) associated with the activity bounded by this completion event and its second immediate predecessor event. If unknown, enter 0.
7. The TE of the event, where known. If unknown, enter 0.
8. The variance V associated with the event, where known. If unknown, enter 0 here.

The last event (actually entered first, as above) must be labeled one. The other events need not have any order. Remember, when entering the data that you enter T (line 3000) the 8 pieces of data for event 1 (line 3001), then 8 for event 2 (line 3002), then 8 for event 3 (line 3003), etc., in strict sequential event number order, regardless of the physical layout order of the network events.

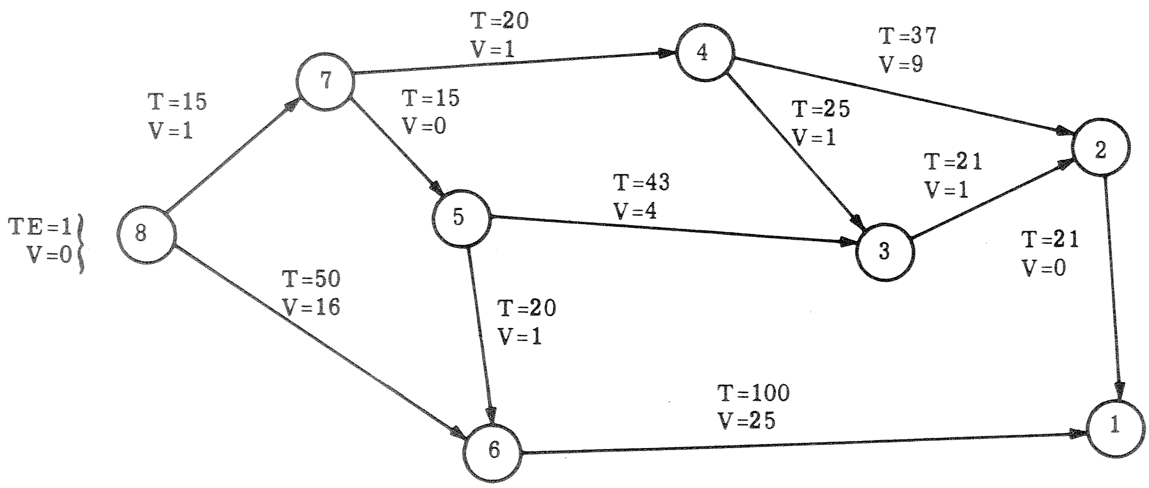
At least one event in the network must have its earliest completion time specified (i.e., at least one event's TE must be known).

If however, no event has a specified TE, let the TE of the initial event in the network be specified as one (1). Thus, the initial event in the network would have in its seventh data location a one, instead of a zero. Assuming event number 9 is this first event in the network and starts at time zero, its data form would look like XXXX DATA ------0-; however, if for the program to operate the event is assigned an arbitrary TE, say 1, its correct data statement would take the form XXXX DATA ------1-.

If there are more than ten events in the network enter a DIM statement in line 1600 such that E(T, 13), P(T) and S(T) are dimensioned, where T = number of events. For example, if your network has 18 events, the DIM statement would be DIM E(18, 13), P(18), S(18). After the data has been entered, type: RUN.

SAMPLE PROBLEM

Analyze the PERT network illustrated.



SAMPLE SOLUTION

```
*3000 DATA 8
*3001 DATA 2,21,0,6,100,25,0,0
*3002 DATA 4,37,9,3,21,1,0,0
*3003 DATA 4,25,1,5,43,4,0,0
*3004 DATA 7,20,1,0,0,0,0,0
*3005 DATA 7,15,0,0,0,0,0,0
*3006 DATA 5,20,1,8,50,16,0,0
*3007 DATA 8,15,1,0,0,0,0,0
*3008 DATA 0,0,0,0,0,0,1,0
*RUN
```

```
                    PERT ANALYSIS PROGRAM

EVENT NUMBER        TE          V          TL     TOTAL SLACK
----- ------        --          -          --     ----- -----

           8         1          0           1               0
           7        16          1          16               0
           6        51         16          51               0
           5        31          1          31               0
           4        36          2          84              48
           3        74          5         109              35
           2        95          6         130              35
           1       151         41         151               0

READY
*
```

This Fortran algorithm finds the shortest distance from the initial node of a network to all other nodes and indicates the paths used to achieve those distances. The minimal spanning tree for the network is also found.

INSTRUCTIONS

On execution the program will ask for the name of the data file. The first line of this file is used for problem identification. The data follows with one line containing the data for each arc in the following order:

line number, starting node, ending node, distance

The network initial node should be numbered 1. Due to dimension statements, no node can be numbered greater than 25.

SAMPLE PROBLEM

Analyze the network which consists of the following arcs:

Starting Node	Ending Node	Distance
1	2	5
1	3	2
1	4	12
1	5	15
1	7	2
2	3	4
2	4	2
2	5	2
2	7	9
3	6	23
4	7	8
5	6	3
5	7	16
6	7	10

SAMPLE SOLUTION

The data was stored in the file DATA. From the printout, it is seen, for example, that the shortest distance from node 1 to node 6 is 10 and the path with that distance is 1 to 2, 2 to 5, 5 to 6.

```
*LIST DATA

0010 TEST PRØBLEM FØR SHØRTEST PATH -- MIN SPANNING TREE
0020 1 2 5
0030 1 3 2
0040 1 4 12
0050 1 5 15
0060 1 7 2
0070 2 3 4
0080 2 4 2
0090 2 5 2
0100 2 7 9
0110 3 6 23
0120 4 7 8
0130 5 6 3
0140 5 7 16
0150 6 7 10
READY
*RUN SHØRTEST
1=TRACE ITER, 0=ANSWERS ØNLY, -1=MIN SPAN TREE ØNLY
= 0
DATA FILE NAME
= DATA
    TEST PRØBLEM FØR SHØRTEST PATH -- MIN SPANNING TR

SHØRTEST DISTANCE NØDE 1 TØ ALL ØTHERS, ITERATIØN  4

STARTING NØDE  ENDING NØDE  DISTANCE

              1             2       5.00

              1             3       2.00

              1             4       7.00

              1             5       7.00

              1             6      10.00

              1             7       2.00

THE SHØRTEST PATHS FRØM ARC ØNE ARE:
STARTING NØDE  ENDING NØDE  ARC LENGTH

              1             2       5.00

              1             3       2.00

              1             7       2.00

              2             4       2.00

              2             5       2.00


              5             6       3.00
THE MINIMAL SPANNING TREE CØNSISTS ØF THE FØLLØWING ARCS

STARTING NØDE  ENDING NØDE  DISTANCE

              1             3       2.00

              1             7       2.00

              2             3       4.00

              2             4       2.00

              2             5       2.00

              5             6       3.00

PRØGRAM STØP AT 2250
*
```

This Fortran subroutine calculates a smoothed series, given a time series and a smoothing constant, by using the triple smoothing technique. A quadratic approximation is also found that can be used to estimate the smoothed series for future time periods.

INSTRUCTIONS

The calling sequence is

CALL SMOOTH (VECTOR, NUMBER, SCONST, A, B, C, RESULT)

where

- VECTOR is the 1-dimensional array containing values for known data points
- NUMBER is the number of elements in VECTOR
- SCONST is the smoothing constant (0 < SCONST < 1) must be provided by user
- A, B, and C are coefficients of expression A+B*T+.5*C*T ↑ 2. This expression can be used to find estimates of the smoothed series for a given number of time periods (T) ahead. Estimates for these coefficients must be provided by the user. If the values provided are 0, the program will calculate first estimates.
- RESULT is the resultant vector containing smoothed values for the data in VECTOR.

SAMPLE PROBLEM

A mutual fund price history is given below. Perform triple smoothing on the data and estimate the fund price for the first three months of 1971.

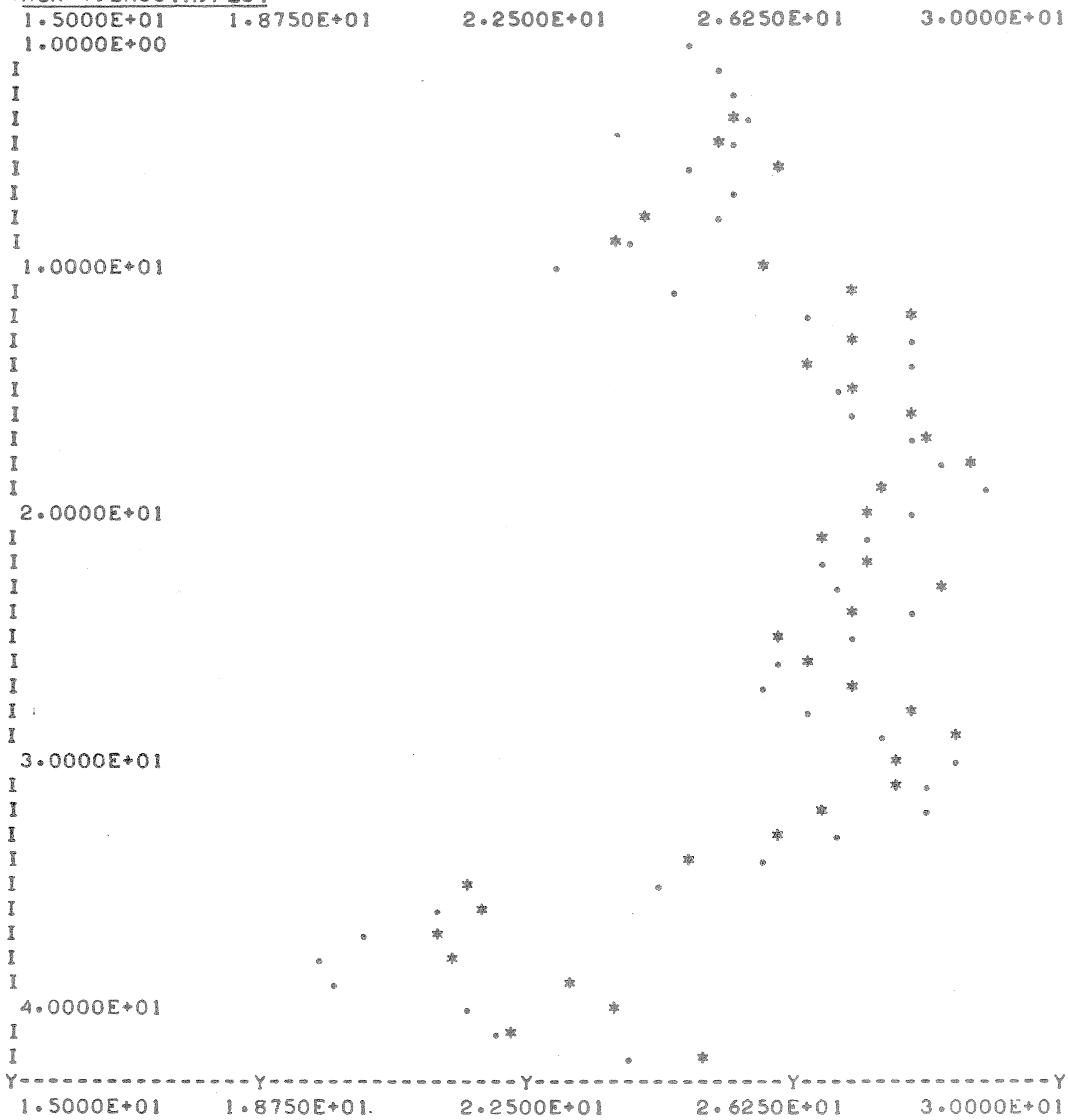
Fund Unit Prices

Month	1967	1968	1969	1970
January	------	25.805	27.797	28.086
February	------	24.520	27.690	27.007
March	------	23.959	27.050	26.304
April	------	26.226	27.684	25.047
May	------	27.393	28.717	21.950
June	------	28.285	27.369	22.022
July	25.170	27.534	26.380	21.475
August	25.555	26.739	26.734	21.547
September	25.779	27.494	27.326	23.276
October	25.795	28.262	28.244	23.958
November	25.406	28.511	28.853	23.852
December	26.392	29.063	28.022	25.358

SAMPLE SOLUTION

The data was entered in a file SMTHDATA. the SMOOTH routine was called with a smoothing constant of .3. The LIBRARY routine PLOT was used to plot the raw data (*) and the smoothed data (.).

```
*RUN *;SMOOTH;PLOT
```




```
FØRECAST FØR NEXT 3 MØNTHS
            1    2.5799242E+01
            2    2.6770192E+01
            3    2.7872444E+01

PRØGRAM STØP AT 170
*LIST

10 DIMENSIØN VECTØR(42),RESULT(42)
20 DIMENSIØN Y(2)
30 F(T)=A+B*T+C*T*T*.5
40 1 FØRMAT(V)
50 2 FØRMAT(2F10.3)
60 SCØNST=.3
70 CALL PLØT(X,Y,30.,15.,2,1,42)
80 READ("SMTHDATA",1)(I,VECTØR(J),J=1,42)
90 CALL SMØØTH(VECTØR,42,SCØNST,A,B,C,RESULT)
100 DØ 10 I=1,42
110 X=I
120 Y(1)=VECTØR(I);Y(2)=RESULT(I)
130 10 CALL PLØT(X,Y,30.,15.,2,0,42)
140 PRINT:"FØRECAST FØR NEXT 3 MØNTHS"
150 DØ 20 I=1,3
160 20 PRINT:I,F(FLØAT(I))
170 STØP
180 END
```

```
READY

*LIST SMTHDATA

010 25.170
020 25.555
030 25.779
040 25.795
050 25.406
060 26.392
070 25.805
080 24.520
090 23.959
100 26.226
110 27.393
120 28.285
130 27.534
140 26.739
150 27.494
160 28.262
170 28.511
180 29.063
190 27.797
200 27.690
210 27.050
220 27.684
230 28.717
240 27.369
250 26.380
260 26.734
270 27.326
280 28.244
290 28.853
300 28.022
310 28.086
320 27.007
330 26.304
340 25.047
350 21.950
360 22.022
370 21.475
380 21.547
390 23.276
400 23.958
410 23.852
420 25.358

READY

*
```

This Fortran object program provides solutions to a wide range of time series forecasting problems. The problem follows four fundamental steps to provide useful predictions:

1. Cyclic analysis of past data.
2. Trend analysis of past data.
3. An error analysis for comparing forecast with actual data.
4. A synthesis of analyses to form a forecast.

This program is coded in chain overlays. TCAST is the main driver routine. TCAST1 and TCAST2 are the two overlay files.

INSTRUCTIONS

Data Preparation

The data can be entered either from a data file or from the terminal. In either case the format is identical. The data file format is described. The file may be given any name. It can be built with or without line numbers. The first line of the file is alphanumeric title information. The second line contains the following parameters, separated by commas:

VARIABLE NAME	DESCRIPTION
L	LEAD TIME The number of time periods, for which the forecast is to be calculated and the forecasting parameters optimized. Lead time is usually specified as the minimum length of time desired to accurately forecast the future.
IH	FORECAST HORIZON The number of time periods for which the forecast is to be projected, regardless of lead time and accuracy.
I6	I6 = 0 - all three types of smoothing to be used by program. I6 = 1 - single smoothing to be used by program. I6 = 2 - double smoothing to be used by program. I6 = 3 - triple smoothing to be used by program.
Y-SMALL	Smallest ordinate for plot of forecast.
Y-LARGE	Largest ordinate for plot of forecast.

The historical data is entered next. This consists of a raw data point, and optional base series point, for each time period. A base series is a time series, of values which are used to adjust (transform) both the raw data and the forecast. Typically a base series may represent human judgment, cyclic variations, the results of a multiple regression correlation analysis, or known phenomena. The base series is usually used to transform the raw data into a new time series. This, in turn, is used for intermediate computations. The results are retransformed by the base series to form a forecast.

The raw data and base series data points (if there is a base series) are entered on the following lines, with one raw data point and one base series data point per line. Data points are entered in free-field format.

The termination of both the raw data and base series data points is specified with a final data point greater than or equal to 1E15. If there is no base series, then the first data line must have a value for the first base series data point greater than or equal to 1E15.

The program can handle up to 310 raw data points and 450 base series data points.

The final line in the data file contains up to eight trial smoothing constants, ALPHA, entered in free-field format. Additional ALPHA's may be specified during execution of the program. These constants must be between 0 and 1. The larger the constant, the more weight is given to the recent history in calculating the forecast.

Execution

To execute the program, access the programs TCAST1 and TCAST2 from the library. This can be done by typing:

GET LIBRARY/TCAST1, R; LIBRARY/TCAST2, R. Then run the program TCAST. The program will type:

ENTER INPUT AND OUTPUT FILENAMES (,DEFAULT) -

The input filename is the name of a previously prepared data file. If the data is to be entered from the terminal, enter blanks for the file name. There are seven groups of output data. For each group, there is the option of writing the output on the specified output file, or printing it at the terminal. The parameter DEFAULT can have the following meanings:

- If 0, give the output option on all seven groups of output.
- If 1, give the output option for only CYCLIC ERRORS, TREND ANALYSIS, and FORECAST DATA. All other output is to be written on the output file.
- If 2, print CYCLIC ERRORS and TREND ANALYSIS at the terminal, all other output is to be written on the output file.

This parameter is optional. If it is not entered, the value 0 is used. The output file can be listed at the terminal, printed at the central site (using BPRINT) or examined using EDIT.

A description of the output follows:

- INITIAL DATA

  The raw data points, base series, etc., as read from the data file. This output is useful to ascertain that the data was correctly entered.

- CYCLIC ERRORS

  The cyclic error ERR(K) is a relative measure of residual variance for a cycle of length K.

  The local minima of cyclic error indicate significant cyclic behavior corresponding to that cycle length. This type of analysis is useful to determine which cyclic intervals it would be meaningful to force, and other harmonics.

  The program prints the cycle length which minimizes the relative error. The user then selects the period to be used.

- CYCLIC VALUES

  The cyclic values give a quantitative description of the shape of the cycle.

- CYCLIC RESIDUES

  Next, the residues remaining after the raw data is corrected for both base series, and cyclic series are output.

  At this point the period can be changed and the cyclic values and cyclic residues recalculated.

- TREND ANALYSIS

  The mean absolute deviations (MAD), associated with each smoothing constant and type of smoothing, are printed.

  After the program has performed this analysis for each smoothing constant in the data file, additional constants can be entered from the terminal. A null response (carriage return) signifies there are no more constants to be entered. Specify the smoothing type and smoothing constant to be used for the forecast.

- FORECAST

  Specify the time period for which the forecast output is to begin (the output cannot begin before time period = lead time + 5). For each time period the output consists of:

  - Forecast of the residue, which is the forecast of (raw data point — base series data point — cyclic value).
  - Composite forecast, which equals forecast of (the residue + base series data point + cyclic value).
  - Raw data point.
  - Error in the forecast, which equals (raw data point — composite forecast).
  - A plot of the composite forecast (.) and raw data point (*) versus time for each time period. When the plot of the composite forecast and the raw data point occur in the same print position, an "=" is printed.

After the end of the historical data, there are no errors. Beyond this point, only the time period, forecast of the residue, and composite forecast are printed. It is these time periods which are of prime interest, since they constitute the forecast.

For this section of output, supply an additional output file for the plot. If this additional file is the same as the main output file then each data line will be followed by the plot line. Note that the plot can also be directed to the terminal by giving a null filename (blanks).

- STATISTICAL INFORMATION

  - S1, S2, S3, the exponentially smoothed variable for single, double, and triple smoothing respectively, followed by CED1, CED2, CED3, the current expected demand for single, double, and triple smoothing respectively.
  - C2, C3, the change per unit time in exponentially smoothed average for double and triple smoothing respectively, and finally RC3 the rate of change per unit time in the exponentially smoothed average for triple smoothing.
  - The time period after which the forecast is not within the mean absolute deviation (MAD) limits.
  - Linear least square curve fit.
  - Mean
  - Variance

  NOTE: The user can give a null response to any question, in which case the program chooses the best option or parameter.

REFERENCE

Series 6000/600/400/G-200 Time Series Forecasting Implementation Guide, Order No. BQ08).

SAMPLE PROBLEM

Perform a time series forecast on the following data with a lead time of 3 and a horizon of 6.

data point	1	2	3	4	5	6	1	2	3	2	3	4			
base point				3	3	3	0	0	0	1	1	1	3	3	3

SAMPLE SOLUTION

The data was entered from the terminal. However, the data could have been entered in a data file as below:

SAMPLE 1 – DATA FILE

```
       3, 6, 0, 0., 10.
       1
       2
       3
       4, 3
       5, 3
       6, 3
       1, 0
       2, 0
       3, 0
       2, 1
       3, 1
       4, 1
  1E15, 3
       3
       3
       1E15
       .1    .2    .3
```

During execution, part of the data was directed to the output file DUMP. This file is also listed.

```
*GET LIBRARY/TCAST1,R;LIBRARY/TCAST2,R;LIBRARY/TCAST,R
*RUN TCAST

TIME SERIES FØRECASTING PRØGRAM
ENTER INPUT AND ØUTPUT FILE NAMES(,DEFAULT)-
= ,DUMP

ENTER PRØBLEM TITLE-
= SAMPLE PRØBLEM - ENTERING DATA FRØM TERMINAL

ENTER LEAD TIME,HØRIZØN,SMØØTHING TYPE,YSMALL,YLARGE-
= 3,6,0,0,10

ENTER DATA,BASE PØINTS(ØNE PAIR/LINE)
= 1
= 2
= 3
= 4,3
= 5,3
= 6,3
= 1
= 2
= 3
= 2,1
= 3,1
= 4,1
= 1E15,3
= 3
= 3
= 1E15

DIRECT INITIAL DATA TØ FILE(Y ØR N)-
= Y

DIRECT CYCLIC ERRØR TØ FILE(Y ØR N)-
= N


CYCLIC ERRØR
 K              ERR(K)
  1             0.164797
  2             0.194950
  3             0.
  4             0.298828
  5             0.333333
  6             0.
```

```
PERIØD ØF MØST DØMINANT CYCLE=  3

PERFØRM ANALYSIS FØR PERIØD-
= 3

DIRECT CYCLIC VALUES TØ FILE-
= Y

DIRECT CYCLIC RESIDUES TØ FILE-
= Y

DØ YØU WANT TØ TRY A DIFFERENT PERIØD (Y ØR N)-
= N

DIRECT TREND ANALYSIS TØ FILE-
= N


TREND ANALYSIS


ENTER ALPHAS(MAX ØF 8)
= .1 .2 .3

ALPHA   TYP SM         ERRØR MAD
0.10000     1            0.00000
0.10000     2            0.00000
0.10000     3            0.00000
0.20000     1            0.00000
0.20000     2            0.00000
0.20000     3            0.00000
0.30000     1            0.00000
0.30000     2            0.00000
0.30000     3            0.00000

ADDITIØNAL ALPHAS-
= .05 .4
0.05000     1            0.00000
0.05000     2            0.00000
0.05000     3            0.00000
0.40000     1            0.
0.40000     2            0.
0.40000     3            0.00000

ADDITIØNAL ALPHAS-
=

ØPTIMUM SMØØTHING TYPE=2  ALPHA=0.40000000

WHAT SMØØTHING TYPE AND ALPHA-
= 1,.4

DIRECT FØRECAST DATA TØ FILE-
= Y

ENTER FILENAME FØR FØRECAST PLØT-
=


FØRECAST PLØT

BEGIN FØRECAST AT PERIØD-
= 0
```

```
TIME  O.                                                                      0.10000E+02

  8               *
  9                      *
 10               *
 11                      *
 12                             *
 13                             .
 14                                    .
 15                                          .
 16        .
 17               .
 18                      .

DIRECT STATISTICAL INFØRMATIØN TØ FILE-
= N

STATISTICAL INFØRMATIØN

S1=         1.00000
S2=         1.00000
S3=         1.00000

CED1=         1.00000
CED2=         1.00000
CED3=         1.00000

 C2=          0.
 C3=        -0.00000
RC3=          0.

CAUTIØN,FØRECAST NØT WITHIN MAD LIMITS AFTER TIME     15

LEAST SQUARES CURVE FIT

Y=          2.636+          0.056*X

MEAN=              3.000
VARIANCE=              2.167

PRØGRAM STØP AT 120
*LIST DUMP


 PRØBLEM NAME:
 SAMPLE PRØBLEM - ENTERING DATA FRØM TERMINAL

 INITIAL DATA

 NUMBER ØF RAW DATA PØINTS-- 12
 NUMBER ØF BASE DATA PØINTS-- 15
 FØRECAST HØRIZØN--  6
 LEAD TIME--  3

 TIME          RAW DATA
   1            1.00000
   2            2.00000
   3            3.00000
   4            4.00000
   5            5.00000
   6            6.00000
   7            1.00000
   8            2.00000
   9            3.00000
  10            2.00000
  11            3.00000
  12            4.00000
```

TIME	BASE SERIES	RESIDUE
1	0.	1.00000
2	0.	2.00000
3	0.	3.00000
4	3.00000	1.00000
5	3.00000	2.00000
6	3.00000	3.00000
7	0.	1.00000
8	0.	2.00000
9	0.	3.00000
10	1.00000	1.00000
11	1.00000	2.00000
12	1.00000	3.00000
13	3.00000	-3.00000
14	3.00000	-3.00000
15	3.00000	-3.00000

CYCLIC VALUES

PERIØD= 3

T	C(T)
1	1.000000
2	1.000000
3	-2.000000

CYCLIC RESIDUES

TIME	RESIDUE
1	1.00000
2	1.00000
3	1.00000
4	1.00000
5	1.00000
6	1.00000
7	1.00000
8	1.00000
9	1.00000
10	1.00000
11	1.00000
12	1.00000

FØRECAST DATA

USED ALPHA TYP SM
0.40000 1

TIME	RESIDUE	CØMPØSITE (.)	ACTUAL (*)	ERRØR
8	1.0000	2.0000	2.0000	0.
9	1.0000	3.0000	3.0000	0.
10	1.0000	2.0000	2.0000	0.
11	1.0000	3.0000	3.0000	0.
12	1.0000	4.0000	4.0000	0.
13	1.0000	4.0000		
14	1.0000	5.0000		
15	1.0000	6.0000		
16	1.0000	1.0000		
17	1.0000	2.0000		
18	1.0000	3.0000		

READY

*

This BASIC program is based on an algorithm to solve "The Transportation Problem." For example, if M factories supply N warehouses with a product, factory I (I = 1 to M) produces A(I) units and warehouse J (J = 1 to N) requires B(J) units, what shipping pattern minimizes total transportation costs? Many other problems fit the same model and can be solved with this program.

INSTRUCTIONS

List the program for instructions.

SAMPLE PROBLEMS

Sample #1 shows the conventional transportation problem. Data is entered in lines 10000 - 10050:

10000	3 Factories, 5 Warehouses
10010	Supply of each of 3 factories
10020	Demand of each of 5 warehouses
10030	Unit costs of Factory #1 to ship to each warehouse
10040	The same information for Factory #2
10050	The same information for Factory #3

Sample #2 shows the transportation problem with certain restricted or prohibited routes, where it is unfeasible, impossible or impracticable to ship on certain routes and a high figure (99) has been entered to eliminate them from allocation.

Sample #3 illustrates the assignment problem, one of the most widely illustrated linear programming problems that can be solved by the transportation method. In general, the assignment problem involves the assignment of resources to jobs. The restrictions are such that each resource can be assigned to only one job and, conversely, each job can have only one resource. When solved by the transportation method, a dummy resource or job is added if the number of resources does not equal the number of jobs. In addition, this problem is looking for the maximum solution. Therefore, a rating matrix of some kind is generally used (note — sign and zeros). The optimum assignment of the resources is then determined by the transportation method. Obtaining the data for the rating matrix is probably the most difficult part of the problem formulation.

This assignment problem has three resources and four jobs. A dummy resource is employed to balance the supply and demand totals. Rating coefficients are used instead of cost amounts. As expected, only three of the four jobs are satisfied. The other job has the dummy resource assigned to it.

Sample #4 represents a typical transportation model for a production or resource scheduling problem. Other problems such as the "Caterer Problem" are similar in construction. Frequently, several time periods are considered for these types of problems.

This model represents a problem with a certain type of resource available at three dispatching points. Demands for these resources are at three receiving points. The demands are known for three periods ahead. The maximum number of resource units that can be had at each of the dispatching points for the three time periods are also known.

Since the resources are not readily available and since there is a time lag in getting the resources to the demand points, resources may have to be held over at a dispatch point for one or two periods to satisfy the demands. The present problem contains a holdover charge of 5 cost units per unit of resource per time period.

This transportation model also contains several routes which are meaningless or prohibited. For example, resources available at any of the dispatch points during period 2 cannot satisfy any demands for period 1. Also, receiving point 3 is too distant from dispatching points 1 and 2 to draw on the available supplies at these points during the same time period.

A slack variable is added as a dummy destination because of the holdover charges at the dispatching points. Typical of this situation would be the distribution of railroad cars.

REFERENCE

Reinfeld and Vogel, Mathematical Programming; Prentice-Hall, Englewood Cliffs, N.J., 1958

SAMPLE SOLUTION 1

```
10000 DATA 3,5
*10010 DATA 1000,800,600
*10020 DATA 400,700,300,500,500
*10030 DATA 4,5,7,4,6
*10040 DATA 7,5,8,5,8
*10050 DATA 6,4,6,7,5
*RUN


HAVE YØU ENTERED DATA BEGINNING IN LINE 10000?
IF NØT, LIST PRØGRAM FØR INSTRUCTIØNS

THE SØLUTIØN MATRIX =

  400     0     0   400   200

    0   700     0   100     0

    0     0   300     0   300


THE TØTAL MINIMUM CØST ØF THE SØLUTIØN =          11700
```

SAMPLE SOLUTION 2

```
READY
*10000 DATA 3,5
*10010 DATA 400,700 DEL
10010 DATA 900,800,700
*10020 DATA 400,700,300,500,500
*10030 DATA 4,6,7,99,99
*10040 DATA 7,5,8,5,8
*10050 DATA 99,99,6,7,5
*RUN


HAVE YØU ENTERED DATA BEGINNING IN LINE 10000?
IF NØT, LIST PRØGRAM FØR INSTRUCTIØNS

THE SØLUTIØN MATRIX =

  400   400   100     0     0

    0   300     0   500     0

    0     0   200     0   500


THE TØTAL MINIMUM CØST ØF THE SØLUTIØN =          12400
```

SAMPLE SOLUTION 3

```
READY
*10000 DATA 4,4
*10010 DATA 1,1,1,1
*10020 DATA 1,1,1,1
*10030 DATA -90,-95,-94,-93
*10040 DATA -92,-98,-93,-94
*10050 DATA -91,-97,-96,-92
*10060 DATA 0,0,0,0
*RUN


HAVE YØU ENTERED DATA BEGINNING IN LINE 10000?
IF NØT, LIST PRØGRAM FØR INSTRUCTIØNS

THE SØLUTIØN MATRIX =

     0       0       0       1

     0       1       0       0

     0       0       1       0

     1       0       0       0


THE TØTAL MINIMUM CØST ØF THE SØLUTIØN =            -287
```

SAMPLE SOLUTION 4

```
READY
*10000 DATA 9,10
*10010 DATA 13,20,15,8,8,8,6,6,6
*10020 DATA 4,11,6,8,10,9,12,7,4,19
*10030 DATA 2,6,99,7,11,99,12,16,19,0
*10040 DATA 3,7,99,8,12,13,13,17,18,0
*10050 DATA 99,99,5,99,99,10,99,11,15,0
*10060 DATA 99,99,99,2,6,99,7,11,99,0
*10070 DATA 99,99,99,3,7,99,8,12,13,0
*10080 DATA 99,99,99,99,99,5,99,99,8,0
*10090 DATA 99,99,99,99,99,99,2,6,99,0
*10100 DATA 99,99,99,99,99,99,99,995,0
*RUN


HAVE YOU ENTERED DATA BEGINNING IN LINE 10000?
IF NOT, LIST PROGRAM FOR INSTRUCTIONS


OUT OF DATA IN  720


READY
*10100 DATA 99,99,99,99,99,99,3,7,99,0
*10110 DATA 99,99,99,99,99,99,99,99,5,0
*RUN


HAVE YOU ENTERED DATA BEGINNING IN LINE 10000?
IF NOT, LIST PROGRAM FOR INSTRUCTIONS

THE SOLUTION MATRIX =

     4    9    0    0    0    0    0    0    0    0

     0    2    0    0    2    0    0    0    0    15

     0    0    6    0    0    1    0    7    0    1

     0    0    0    8    0    0    0    0    0    0

     0    0    0    0    8    0    0    0    0    0

     0    0    0    0    0    8    0    0    0    0

     0    0    0    0    0    0    6    0    0    0

     0    0    0    0    0    0    6    0    0    0

     0    0    0    0    0    0    0    0    4    2


THE TOTAL MINIMUM COST OF THE SOLUTION =          379


READY
*
```

This Fortran subroutine finds a non-negative or a positive solution and the corresponding tableau for the underdetermined system of equations ($N \leq M$):

$$\sum_{j=1}^{M} A_{ij} X_j = C_i \quad , \quad i = 1, \ldots, N$$

INSTRUCTIONS

The arguments are passed to the routines by blank common. The calling sequence is:

```
COMMON  M, N, X (50), A (50,50), INBASE (50),
                KEY, NZSW, EPSLON, C (50)
     .
     .
     .
CALL  UNDEQ
```

where

- M is the number of variables
- N is the number of equations
- X is used to return the solution
- A on entry is the array of coefficients and on exit is the tableaux corresponding to the solution
- INBASE is a vector indicating which variables are in the basis at the solution
- KEY on exit, indicates the status of return
  - If -1, no non-negative solution exists
  - If 0, a solution was found
  - If 1, a solution was found but there are redundant equations
- NZSW, on entry, indicates the solution option:
  - If 0, try to find a non-negative solution
  - If 1, try to find a positive solution
- EPSLON is the tolerance for zero
- C on entry, is the vector of right hand-side constants

The tableaux A and vector INBASE may be used to generate additional solutions to the system. For example, let X be any solution, R be any real number, and L be the index of any variable not in the basis. Then the vector Y defined by:

$$Y(INBASE(I)) = X(I) - R*A(I,L) \quad , \quad I = 1, \ldots, N$$

$$Y(L) = R$$

$$Y(J) = X(J), \quad \text{otherwise}$$

is also a solution (which may not be non-negative). By taking all possible combinations as above, all possible solutions may be generated.

RESTRICTIONS

A Linear Programming Phase I algorithm is used, hence the standard LP non-degeneracy condition is assumed. Also $N \leq M \leq 50$.

SAMPLE PROBLEM

Find the non-negative solution to the following system:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 1 & 2/3 & 2/3 & -1 & 0 & 1 \\ 0 & 3 & 3 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 & 1 \end{bmatrix} X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Also find the form of the general solution.

SAMPLE SOLUTION

From the sample printout below, it can be seen that the general solution is given by:

$X_4 = 1$

$X_1 = .7$

$X_6 = .1$

$X_3 = .3 - R$

$X_5 = .6 + R$

$X_2 = R$

```
100*....SAMPLE DRIVER PRØGRAM FØR UNDEQ....
110 CØMMØN M,N,X(50),A(50,50),INBASE(50),KEY1,KEY2,EPS,C(50)
120 DØ 5 I=1,5
130 C(I)=0.
140 DØ 5 J=1,6
150 5 A(I,J)=0.
160 A(1,1)=1.; A(1,2)=1.; A(1,3)=1.
170 A(2,1)=1.; A(2,2)=1.; A(2,3)=1.
180 A(3,1)=1.; A(5,3)=1.; A(5,5)=1.
190 A(3,6)=1.; A(4,6)=1.; A(5,6)=1.
200 A(3,2)=.66666667; A(3,3)=.66666667
210 A(4,2)=3.; A(4,3)=3.
220 A(2,4)=-1.; A(3,4)=-1.; A(4,4)=-1.; A(5,4)=-1.
230 C(1)=1.
240 M=6
250 N=5
260 EPS=1E-8
270 KEY2=0
280 CALL UNDEQ
290 PRINT 10,KEY1,(X(I),I=1,M)
300 10 FØRMAT("OFEASIBLE SØLUTIØN (KEY=",I1,")"/6F7.3)
310 PRINT 20, (INBASE(I),I=1,N)
320 20 FØRMAT("OBASIC VARIABLES ",6I3)
330 PRINT 30
340 30 FØRMAT("OTABLEAUX")
350 PRINT 40, ((A(I,J),J=1,6),I=1,5)
360 40 FØRMAT(6F7.3)
370 STØP
380 END



READY

*RUN *;UNDEQ
<W>7 MEMØRY EXPANDED. USE $LIMITS ØR CØRE= ØPTIØN FØR NEXT RUN


FEASIBLE SØLUTIØN (KEY=0)
 0.700  0.     0.300  1.000  0.600  0.100

BASIC VARIABLES   4  1  6  3  5

TABLEAUX
 0.     0.     0.     1.000  0.     0.
 1.000  0.     0.     0.     0.     0.
 0.     0.     0.     0.     0.     1.000
 0.     1.000  1.000  0.     0.     0.
 0.    -1.000  0.     0.     1.000  0.

*
```

ENGINEERING

This Fortran program calculates gain and phase of any linear circuit composed of R, L, C, dependent and independent sources. Input is via a description of the circuit schematic. The program handles circuits of up to 30 nodes, 50 branches, 70 components, 20 dependent current sources.

METHOD

The program forms the node-voltage equations, and solves them for the value of the node voltage at the highest-numbered node. This is done at each frequency point, using gauss elimination with row interchange.

If the results of the program are obviously not correct, and further errors are not found after rigorous rechecking, renumber the circuit nodes, keeping the last node as the output node. Then run the new data statements. Although the method of solution protects against problems in this area, some combination of values and interconnections can cause imperfect results. By merely shuffling the node numbers, a satisfactory solution should be obtained.

INSTRUCTIONS

Data Preparation

Draw the circuit schematic into its equivalent circuit form. Replace transistors, transformers, F.E.T's, etc., with the proper model. Label the schematic as follows:

1. Number the nodes or interconnection points of the circuit from 0 to N where the N-th or last node is the desired output node and the 0-th node is the ground or reference node.
2. Number the branches consecutively, starting with one. Each branch may include all or part of the generalized branch shown in Figure 1.

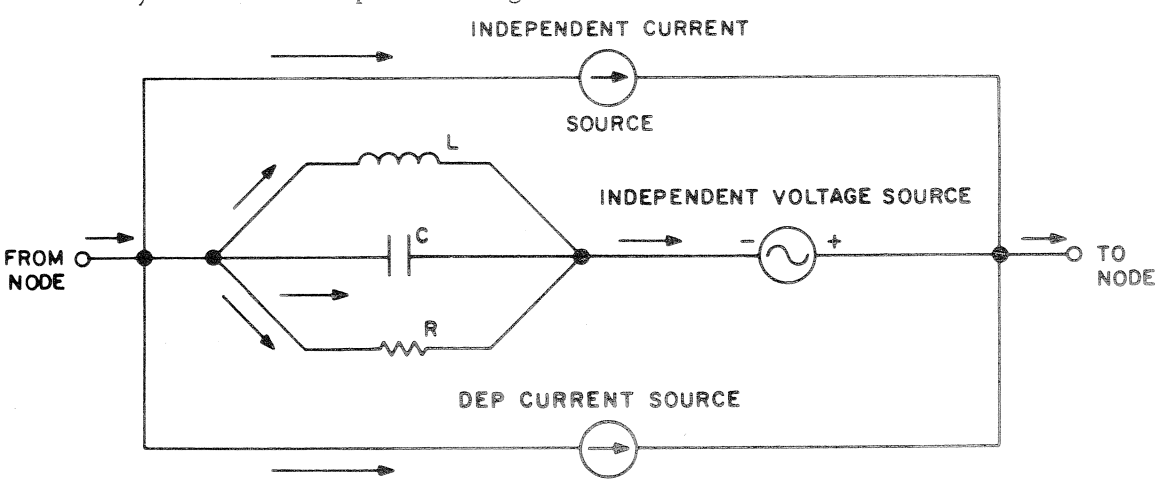


Figure 1. Generalized Circuit Branch

3. Number the parts, with all independent sources numbered first. Dependent sources must numerically follow their controlling part.

   When the program encounters a dependent source, the source is assumed to be controlled by a resistor branch already described. Later data statements may add additional parallel branches. This also means that a dependent source must be described after its controlling branch.

   An independent voltage source must include a resistor in series with it, within that total branch.

4. Sketch in an arbitrary current flow through the entire branch. Note that independent currents flow opposite to the general branch flow. The "FROM" - and "TO" - node convention is used to indicate that current flows "FROM" some node, through the general branch, and "TO" some other node.

Create a data file containing the circuit description. Select any file name up to eight characters in length. The first line of the data file must be:

Line number P, B, N

where

- P is the problem name (a short string)
- B is the number of branches in the circuit
- N is the number of nodes (does not include the datum node)

Each succeeding line must now describe a circuit element as follows:

P BRANCH, NAME, FROM-NODE, TO-NODE, VALUE, TOLERANCE, (OPT) CONTROL BR.

where

- P is a line number which may be used to indicate the number of the part.
- BRANCH is the branch number of the element being described.
- NAME is a 1 to 8 character name. The first character of the name describes the component as follows:
  - R, Resistance
  - C, Capacitance
  - L, Inductance
  - B, Current-controlled current source
  - G, Voltage-controlled current source
  - E or V, Independent current source
  - I, Independent current course
- VALUE is the size or magnitude of the element in ohms, henries, farads, volts, etc.

- TOLERANCE is in percent for all parts.
- CONTROL BRANCH is the number of the branch which controls the dependent source. This entry is only used with dependent sources.

Execution

Enter the code number of the command to be executed. These commands are:

COMMAND	CODE	RESULT
SWEEP	1	Output vs. frequency. Calculates highest number node voltage. DB is in 20* log to the base 10, relative to 1.0 volt at 0 degrees. The program will ask for the initial frequency, terminal frequency, and step size (DELTA). If DELTA = 0 the sweep will logarithmically increment (1, 2, 4, 8...).
PART EFFECTS AND WORST CASE	2	Calculates the nominal case at frequency of interest. Partials for each element are based upon a 1% increment and a value of 100% indicates a one-for-one correspondence. Worst case assumes that the signs of the partials do not change from the nominal case.
STEP A PART	3	On-line change of a part in 10 steps at a given frequency.
CHANGE A PART	4	This routine permanently changes the component or value for the run. The program will ask for the component number, code and value. Supply them, followed by a carriage return. Independent sources cannot be changed to any other element. Resistors, capacitors and inductors may be interchanged, but they may not be changed to independent or dependent sources. The values of all components, except independent sources, may be changed. When changing the value of a dependent source and its control branch, the value of the control branch must be changed first.
EFFECT OF ONE PART ONLY	5	See Command 2.
WORST CASE ONLY	6	See Command 2.

COMMAND	CODE	RESULT
CHANGE OUTPUT NODE	7	On-line renumbering of the circuit graph so as to calculate the voltage at any node. Node numbers are traded permanently for that run.
NEW FREQUENCY	8	Reset the frequency of interest to a new value.
MONTE CARLO	9	The program will ask for the number of trials over which the analysis is to be carried out. The routine randomly generates sets of parts (assuming uniform distribution) within the specified tolerance. The output for each set of parts is then used to calculate the circuit's statistical variation.
STOP	10	STOP

REFERENCE

Grout, J.S., ACNET-2 On-Line Branch-Input Network Frequency Response by Computer, General Electric Company T.I.S. #68 APD-2 (1968). GE Technics Information Exchange, P.O. Box 43, Bldg. 5, Schenectady, N.Y.

SAMPLE PROBLEM

Perform an analysis of the transistor amplifier circuit in Figure 2.

Q_1: B = 100
R_{in} = 1K

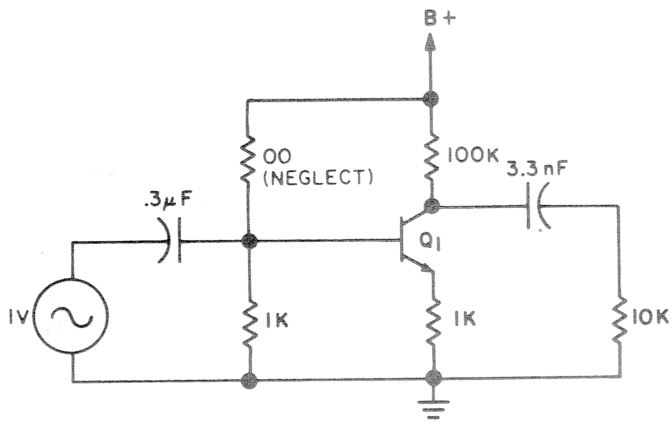


Figure 2

Figure 3 shows one way of redrawing the schematic for generating the data input.

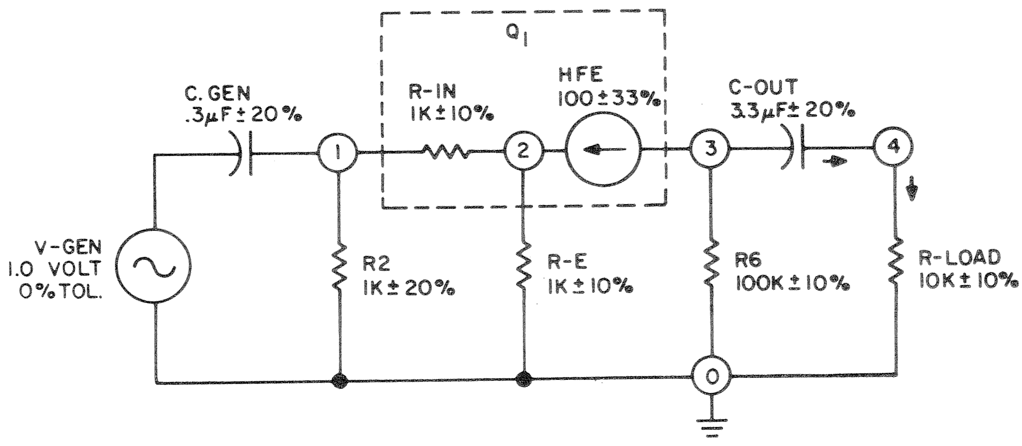


Figure 3

SAMPLE SOLUTION

The data was stored in the file ACDATA which is listed below.

```
*RUN
FILENAME =
= ACDATA


700 SAMPLE PRØBLEM FØR ACNET
 8 BRANCHES
 4 NØDES
LINE BRANCH    NAME        VALUE
 705     1    V-GEN        1.000E+00
 710     1    C-GEN        3.000E-07
 715     2    R2           1.000E+03
 720     3    R-IN         1.000E+03
 725     4    R-E          1.000E+03
 730     5    B-HFE        1.000E+02
 735     6    R6           1.000E+05
 740     7    C-ØUT        3.300E-09
 745     8    R-LØAD       1.000E+04
```

```
COMMANDS:
 1=FREQUENCY SWEEP
 2=PART EFFECTS AND WORSE CASE
 3=STEP A PART
 4=CHANGE A PART
 5=EFFECT OF ONE PART ONLY
 6=WORST CASE ONLY
 7=CHANGE OUTPUT NODE
 8=SET OR RESET FREQUENCY
 9=MONTE CARLO
 10=STOP


NOMINAL OUTPUT AT      1000. HZ.

GAIN =     7.195E+00,    17.14 DB, -128.15 DEGREES

COMMAND NO.=
= 1

SWEEP:  START, STOP, AND DELTA FREQ. ARE
= 10,10000,0

LIST OR PLOT
= PLOT
MAX AND DELTA INC.  (RANGE IS 60 STEPS) =
= 70,2
        -50.00                                                    70.0
  FREQ-HZ +----+----+----+----+----+----+----+----+----+----+----+----+
       10.+Q
       20.+    I Q
       40.+    I    +  Q
       80.+    I    +    I   Q
      100.+    I    +    I    +Q
      200.+    I    +    I    +    IQ
      400.+    I    +    I    +    I    +Q
      800.+    I    +    I    +    I    +  Q
     1000.+    I    +    I    +    I    +   Q
     2000.+    I    +    I    +    I    +   Q
     4000.+    I    +    I    +    I    +   Q
     8000.+    I    +    I    +    I    +   Q
    10000.+    I    +    I    +    I    +   Q
    20000.+    I    +    I    +    I    +   Q

COMMAND NO.=
= 1

SWEEP:  START, STOP, AND DELTA FREQ. ARE
= 10,10000,0

LIST OR PLOT
= LIST
```

FREQ-HZ	OUTPUT	DB-OUTPUT	PHASE-DEG
10.	3.793E-03	-48.42	-2.38
20.	1.515E-02	-36.39	-4.75
40.	6.029E-02	-24.39	-9.48
80.	2.363E-01	-12.53	-18.83
100.	3.637E-01	-8.79	-23.42
200.	1.294E+00	2.24	-44.99
400.	3.594E+00	11.11	-79.12
800.	6.494E+00	16.25	-117.47
1000.	7.195E+00	17.14	-128.15
2000.	8.409E+00	18.50	-152.64
4000.	8.781E+00	18.87	-166.12
8000.	8.879E+00	18.97	-173.03
10000.	8.891E+00	18.98	-174.42
20000.	8.907E+00	18.99	-177.21

```
COMMAND NO.=
= 2

AT     20000. HZ

                   EFFECT ON        TOLERANCE
PART  NAME         OUTPUT-PCT       EFFECT-PCT
  1   V-GEN         100.00            10.00
  2   C-GEN           0.10             0.
  3   R2              0.10             0.
  4   R-IN           -0.90             0.
  5   R-E           -98.90            -9.80
  6   B-HFE           1.90             0.60
  7   R6              9.10             0.90
  8   C-OUT           0.               0.
  9   R-LOAD         91.70             9.20

WORST CASE:   MINIMUM      NOMINAL      MAXIMUM    RANGE-PCT
OUTPUT        6.491E+00    8.907E+00    1.204E+01   62.30

COMMAND NO.=
= 8

FREQUENCY IN HZ =
= 1000

NOMINAL OUTPUT AT      1000. HZ.

GAIN =     7.195E+00,    17.14 DB, -128.15 DEGREES

COMMAND NO.=
= 2

AT      1000. HZ

                   EFFECT ON        TOLERANCE
PART  NAME         OUTPUT-PCT       EFFECT-PCT
  1   V-GEN         100.00            10.00
  2   C-GEN          22.00             4.40
  3   R2             22.00             4.40
  4   R-IN           -0.90             0.
  5   R-E           -98.70            -9.80
  6   B-HFE           2.10             0.70
  7   R6             23.70             2.40
  8   C-OUT          15.90             3.20
  9   R-LOAD         93.20             9.30

WORST CASE:   MINIMUM      NOMINAL      MAXIMUM    RANGE-PCT
OUTPUT        4.256E+00    7.195E+00    1.071E+01   89.70

COMMAND NO.
= 3
AT      1000. HZ.
INCREASE THE VALUE OF A PART IN 10 STEPS.
PART NO., INITIAL VALUE, STEP SIZE=
= 5,500,100
```

```
  R-E           ØUTPUT
  5.0000E+02  1.4220E+01
  6.0000E+02  1.1897E+01
  7.0000E+02  1.0226E+01
  8.0000E+02  8.9671E+00
  9.0000E+02  7.9840E+00
  1.0000E+03  7.1951E+00
  1.1000E+03  6.5481E+00
  1.2000E+03  6.0079E+00
  1.3000E+03  5.5500E+00
  1.4000E+03  5.1570E+00

CØMMAND NØ.=
= 4

CHANGE A PART
PART NØ.,NAME,NØM. VALUE, AND TØL(PCT)=
= 5,R4NEW,680,10

NØMINAL ØUTPUT AT     1000. HZ.

GAIN =     1.052E+01,   20.44 DB, -128.04 DEGREES

CØMMAND NØ.=
= 5

EFFECT ØF ØNE PART AT     1000. HZ, PART NØ. =
= 5

                 EFFECT ØN      TØLERANCE
PART  NAME       ØUTPUT-PCT     EFFECT-PCT
  5   R4NEW       -98.20          -9.70

CØMMAND NØ.=
= 6

WØRST CASE:   MINIMUM     NØMINAL     MAXIMUM   RANGE-PCT
ØUTPUT        6.204E+00   1.052E+01   1.569E+01   90.10

CØMMAND NØ.=
= 9

AT      1000. HZ

MØNTE CARLØ ANALYSIS
  (UNIFØRM DIST. PARTS)
NUMBER ØF TRIALS =
= 50

NØMINAL ØUTPUT =  1.0522E+01
AVERAGE ØUTPUT =  1.0568E+01
        SIGMA =  1.1712E+00

3-SIGMA LIMITS =  7.0546E+00 TØ  1.4082E+01

CØMMAND NØ.=
= 7

NEW ØUTPUT NØDE NUMBER =
= 3

NØMINAL ØUTPUT AT     1000. HZ.

GAIN =     5.183E+01,   34.29 DB,  153.67 DEGREES
```

```
COMMAND NO.=
= 2

AT      1000. HZ

                   EFFECT ON       TOLERANCE
PART  NAME         OUTPUT-PCT      EFFECT-PCT
  1   V-GEN         100.00          10.00
  2   C-GEN          22.20           4.40
  3   R2             22.10           4.40
  4   R-IN           -1.40           0.
  5   R4NEW         -98.20          -9.70
  6   B-HFE           2.70           0.90
  7   R6             23.70           2.40
  8   C-OUT         -79.00         -15.70
  9   R-LOAD         -3.40          -0.30

WORST CASE:    MINIMUM       NOMINAL       MAXIMUM     RANGE-PCT
OUTPUT         3.063E+01     5.183E+01     8.308E+01    101.20

COMMAND NO.=
= 10

PROGRAM STOP AT 12600
*LIST ACDATA

700 SAMPLE PROBLEM FOR ACNET, 8, 4
705 1, V-GEN, 0, 1, 1.0, 10
710 1, C-GEN, 0, 1, 0.3E-6, 20
715 2, R2, 1, 0, 1E3, 20
720 3, R-IN, 1, 2, 1E3, 10
725 4, R-E, 2, 0, 1E3, 10
730 5, B-HFE, 3, 2, 100, 33, 3
735 6, R6, 3, 0, 100E3, 10
740 7, C-OUT, 3, 4, 3.3E-9, 20
745 8, R-LOAD, 4, 0, 10E3, 10

READY

*
```

This BASIC program evaluates and recommends the correct steel beam to be used in a number of load and support applications. Available options include:

Loading (L)

- Uniformly distributed load.
- Concentrated midpoint load.
- Uniformly distributed load and concentrated midpoint load.
- Two symmetric concentrated loads.

Supports (B)

- Simple supports at both ends.
- Simple support at one end; other end fixed.
- Both ends fixed.
- One end fixed; other end unsupported (Cantilever).

INSTRUCTIONS

To use this program type RUN. The program will respond with:

DO YOU WANT INSTRUCTIONS (1 = YES, 0 = NO) ?

If you are not familiar with exact input requirements, type '1' and the program will provide the information necessary to run a case, including the definition of the six input items which will be requested (see sample solution). If you are familar with the program requirements, type 0 and the program will respond:

WHAT ARE L, B, S, W, P, A?

where

L is 1 for uniformly distributed load,
2 for single midpoint load,
3 for uniform load + single midpoint load, or
4 for two equal symmetric loads

B is 1 for beam supported at both ends
2 for one end fixed, other end supported,
3 for beam fixed at both ends,
4 for one end fixed (Cantilever)

S is the length of the span in feet

W is the distributed load in pounds per foot
(set = 0 if not applicable)

P is each concentrated load in pounds
(set = 0 if not applicable)

A is the location of load(s) in feet from end
(set = 0 if not applicable)

Enter values for the six items, and the program will select the lightest available standard sized beam meeting the load requirements. The weight of the beam itself is taken into account.

SAMPLE PROBLEM

Determine the type of beam required for each of the 6 cases listed in the following table.

Case No.	L	B	S	W	P	A
I	1	1	23	1000	0	0
II	1	2	23	100	0	0
III	1	3	23	1000	0	0
IV	2	4	23	0	25,000	0
V	3	4	23	500	10,000	0
VI	4	1	23	0	10,000	4' 9'

SAMPLE SOLUTION

Solutions for sample problem cases provide at least one example of each unique data situation.

```
WHAT ARE L,B,S,W,P,A ?1,1,23,1000,0,0
   RECØMMENDED BEAM IS A      14 WF    30
MØRE DATA (1=YES,0=NØ) ?1
WHAT ARE L,B,S,W,P,A ?1,2,23,100,0,0
   RECØMMENDED BEAM IS A       8 JR   6.5
MØRE DATA (1=YES,0=NØ) ?1
WHAT ARE L,B,S,W,P,A ?1,3,23,1000,0,0
   RECØMMENDED BEAM IS A      12 WF    27
MØRE DATA (1=YES,0=NØ) ?1
WHAT ARE L,B,S,W,P,A ?2,4,23,0,25000,0
   RECØMMENDED BEAM IS A      24 WF    84
MØRE DATA (1=YES,0=NØ) ?1
WHAT ARE L,B,S,W,P,A ?3,4,23,500,10000,0
   RECØMMENDED BEAM IS A      24 WF    76
MØRE DATA (1=YES,0=NØ) ?1
WHAT ARE L,B,S,W,P,A ?4,1,23,0,10000,4
   RECØMMENDED BEAM IS A      12 B    22
MØRE DATA (1=YES,0=NØ) ?1
WHAT ARE L,B,S,W,P,A ?4,1,23,0,10000,9
   RECØMMENDED BEAM IS A      16 WF    36
MØRE DATA (1=YES,0=NØ) ?0

READY
*
```

This BASIC program calculates gas and vapor (but not steam) control valve coefficients and the required valve rangeability.

INSTRUCTIONS

Enter input data using the following format:

20 DATA minimum temperature in degrees Farenheit
maximum temperature in degrees Farenheit, molecular weight.

25 DATA enter DATA in groups of 3, as many groups as you need,
in this order......PPH flow, inlet PSIG, outlet PSIG,
2nd flow, 2nd press., etc...

Then type RUN.

NOTE: When pressure drop is critical, the program assume outlet pressure is half the absolute inlet pressure. The program also converts pressure and temperature to absolute.

Reference is equation 4 of the voluntary standard agreed to in November, 1961 by the Fluid Controls Institute, Inc.

For additional instructions, list the program.

SAMPLE PROBLEM

Determine the control valve coefficients which satisfy the data used in the sample solution.

SAMPLE SOLUTION

```
*20 DATA 470,870,112.5
*25 DATA 1350,145,80,8000,145,80,1350,195,105,8000,195,105
*RUN


RATIØ ØF HIGH/LØW CØEFF.=REQUIRED VALVE RANGEABILITY.


WHEN FLØW IN PPH =        1350   FLØW IN SCFH =    4550.531

        TEMP     INLET PRESS   ØUTLET PRESS     PRESS DRØP   VALVE CØEFF.
       DEG F            PSIG           PSIG            PSI

         470             145             80             65       2.207188
         870             145             80             65       2.639512

WHEN FLØW IN PPH =        8000   FLØW IN SCFH =    26966.11

        TEMP     INLET PRESS   ØUTLET PRESS     PRESS DRØP   VALVE CØEFF.
       DEG F            PSIG           PSIG            PSI

         470             145             80             65       13.07963
         870             145             80             65       15.64155

WHEN FLØW IN PPH =        1350   FLØW IN SCFH =    4550.531

        TEMP     INLET PRESS   ØUTLET PRESS     PRESS DRØP   VALVE CØEFF.
       DEG F            PSIG           PSIG            PSI

         470             195            105             90       1.648433
         870             195            105             90       1.971314

WHEN FLØW IN PPH =        8000   FLØW IN SCFH =    26966.11

        TEMP     INLET PRESS   ØUTLET PRESS     PRESS DRØP   VALVE CØEFF.
       DEG F            PSIG           PSIG            PSI

         470             195            105             90       9.768495
         870             195            105             90       11.68186

READY
*
```

This BASIC program calculates liquid valve coefficients and the required valve rangeability.

INSTRUCTIONS

Enter input data using the following format:

20 DATA minimum pressure drop, maximum pressure drop, minimum flow and maximum flow.

25 DATA one or more values of fluid specific gravity, for example, expected high and low.

Then type RUN.

NOTE: To size more valves, enter new data in lines 20 and 25, then type RUN.

Reference is the voluntary standard (November, 1961) of the Fluid Controls Institute, Inc. No viscosity corrections are available yet. Consult valve manufacturers about viscous or non-Newtonian fluids.

For additional instructions, list the program.

SAMPLE PROBLEM

Determine the valve coefficients which satisfy the data used in the sample solution.

SAMPLE SOLUTION

```
*20 DATA 30,70,3.5,12.5
*25 DATA .9,1,1.1
*RUN


***THE IDENTITY OF THE VALVE IS

WHEN SPECIFIC GRAVITY =  0.9

        FLOW,GPM   PRESS.DROP,PSI   VALVE COEFF.
          3.5              30       0.6062178
         12.5              30        2.165064
          3.5              70       0.3968627
         12.5              70        1.417367
RATIO OF HIGH/LOW COEFFICIENTS=VALVE RANGEABILITY

WHEN SPECIFIC GRAVITY =     1

        FLOW,GPM   PRESS.DROP,PSI   VALVE COEFF.
          3.5              30       0.6390096
         12.5              30       2.282177
          3.5              70         0.41833
         12.5              70        1.494036
RATIO OF HIGH/LOW COEFFICIENTS=VALVE RANGEABILITY

WHEN SPECIFIC GRAVITY =  1.1

        FLOW,GPM   PRESS.DROP,PSI   VALVE COEFF.
          3.5              30        0.670199
         12.5              30        2.393568
          3.5              70       0.4387482
         12.5              70        1.566958
RATIO OF HIGH/LOW COEFFICIENTS=VALVE RANGEABILITY

READY
*
```

This BASIC program assists the electronic engineer in the design of low pass RC active filters. The following filter types may be designed:

1. Butterworth
2. Bessel
3. Chebyshev

When designing Chebyshev filters, the maximum output ripple voltage may be specified as one of the following:

1. 1/2 db (decibels)
2. 1 db
3. 2 db
4. 3 db

Filters of up to tenth order may be synthesized. Either of two construction options may be selected, common form or Rauch form.

The file LFLTIN is an instruction file and the file LFLDAT is a data file used by the program.

The Butterworth Filter is often referred to as a maximally flat amplitude filter because the response at a frequency f, less than the break frequency f_0, is flat. At $f = f_0$, the response is -3db and for $f > f_0$, the response rolls off at 20n decibels (db) per decade, where n is the order of the filter.

Figures LFILTR-1 and LFILTR-2 show the amplitude and phase characteristics of this filter as a function of the normalized frequency, $f_0 = 1$.

The Bessel Filter (see Figures LFILTR-3 and LFILTR-4) has a similar amplitude response to that of the Butterworth filter. At frequencies between DC and f_0, the Bessel filter amplitude decreases with increasing frequency at a very low rate, whereas the Butterworth is flat until f nears f_0. The Bessel filter does not achieve -3db response at $f = f_0$ for all filter orders; a high gain level continues past f_0 for some orders. Roll off past $f = f_0$ is not as fast as that of the Butterworth. The phase response of the Bessel filter is more linear than that of the Butterworth, particularly for high orders. This characteristic makes the Bessel filter suitable to the synthesis of pure delay elements.

The Chebyshev filter (Figures FILTR-5 through FILTR-12) is often referred to as the equal ripple filter because the peak amplitude response deviation of each ripple is the same for all ripples. The number of ripples in the response is equal to (n-2) where n is the filter order. The roll off for $f > f_0$ is greater than -20dc/decade initially, but asymptotically approaches -20db/decade as the frequency increases. Note that the response above f_0 is attentuated much more quickly than the preceding two cases.

Selection of filter type must be determined by the particular application.

The two construction types, Rauch or conventional are shown for various filter orders in the diagrams of Figures LFILTR-13 through LFILTR-15.

The Rauch filter form requires one more resistor than the conventional and must have an operational amplifier as the active device. For 2^{nd}, 3^{rd}, 6^{th}, 7^{th}, 10^{th}, 11^{th}, ... $(2 \cdot (N+1))^{th}$, $(2 \cdot (N+1) +1)^{th}$ orders, the output signal is inverted. When all resistors in a filter are equal, all odd order filters have a 6db attenuation.

The common form requires only a unity gain amplifier as an active device. For low cost, this can be an emitter follower circuit. The filter gain at DC is unity.

INSTRUCTIONS

LFILTR is an interactive program; i.e., the data is input on-line. The following codes are used:

Type	Code
Butterworth	1
Bessel	11
Chebyshev - 1/2 db	21
Chebyshev - 1 db	31
Chebyshev - 2 db	41
Chebyshev - 3 db	51

Form	Code
Common	1
Rauch	2

The data line is as follows:

"T, N, C, F, R"

where

T = Type Code

N = Filter order, first through tenth

C = Construction Form Code

F = f_0 the break frequency in hertz

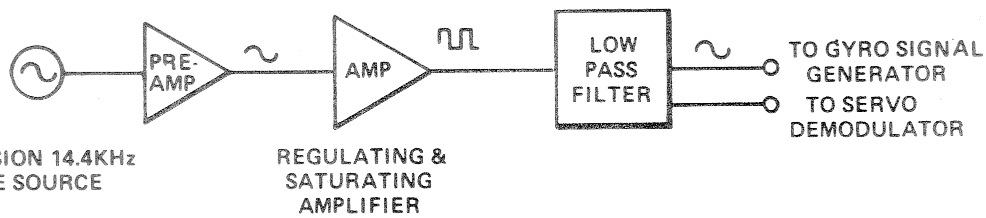
R = Resistance of filter resistor in ohms (all resistors must be equal)

All data is delineated by commas and exponential notation, i.e. 100 = $1.0E+_2$, is permitted.

The output is a labeled listing of capacitor values corresponding to the circuit configurations in the diagrams of Figures LFILTR-13 through LFILTR-15.

SAMPLE PROBLEM

An aircraft navigation system requires a regulated 14.4KH_z sine wave to power the gyro signal generators. The low pass filter must cut-off the harmonics of 14.4KHz present in the square wave. This is achieved by the following circuitry:



The design specification calls for at least 60 db of attenuation at the third harmonic. Since the asympototic response of a 6^{th} order Butterworth filter is -57.5 db at the 3^{rd} harmonic and the actual response is beneath the asymptope, the designer may choose such a filter. To be sure of meeting the requirement, suppose a 7^{th} order filter is selected since it only requires 2 more components. To minimize the insertion loss, the designer selects the common construction form with unity gain amplifiers and resistors of 25KΩ.

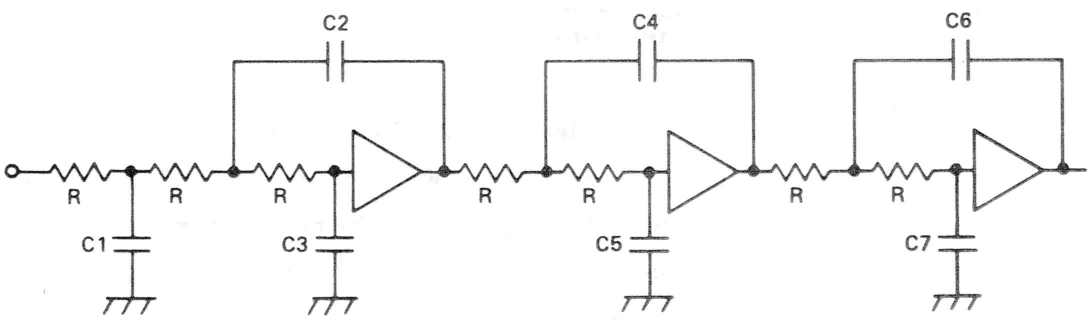
The data line appears as:

1, 7, 1, 15.8E+3, 25E+3

The cut-off frequency of 15.8KH$_z$ was selected to minimize the loss of 14.4KH$_z$ and maintain 60 db attenuation at the 3rd Harmonic (43.2KH$_z$).

```
*GET LIBRARY/LFLDAT,R
*LIB LFILTR
READY
*RUN


REV 0

COPYRIGHT HONEYWELL INC. 1970
FOR INSTRUCTION LIST LFLTIN.
INPUT:
FILTER TYPE,ORDER,CONSTRUCTION,FREQUENCY(HZ.),RESISTANCE
?1,7,1,15.8E+3,25E+3
C(   1)   5.57272E-10
C(   2)   1.01456E-09
C(   3)   1.16167E-10
C(   4)   4.45730E-10
C(   5)   3.62660E-10
C(   6)   1.80790E-09
C(   7)   8.98230E-11
DO YOU WANT TO RUN PROGRAM AGAIN ? 1 = YES : 2 = NO
?2

READY
*
```

The circuit then is:



ILLUSTRATIONS

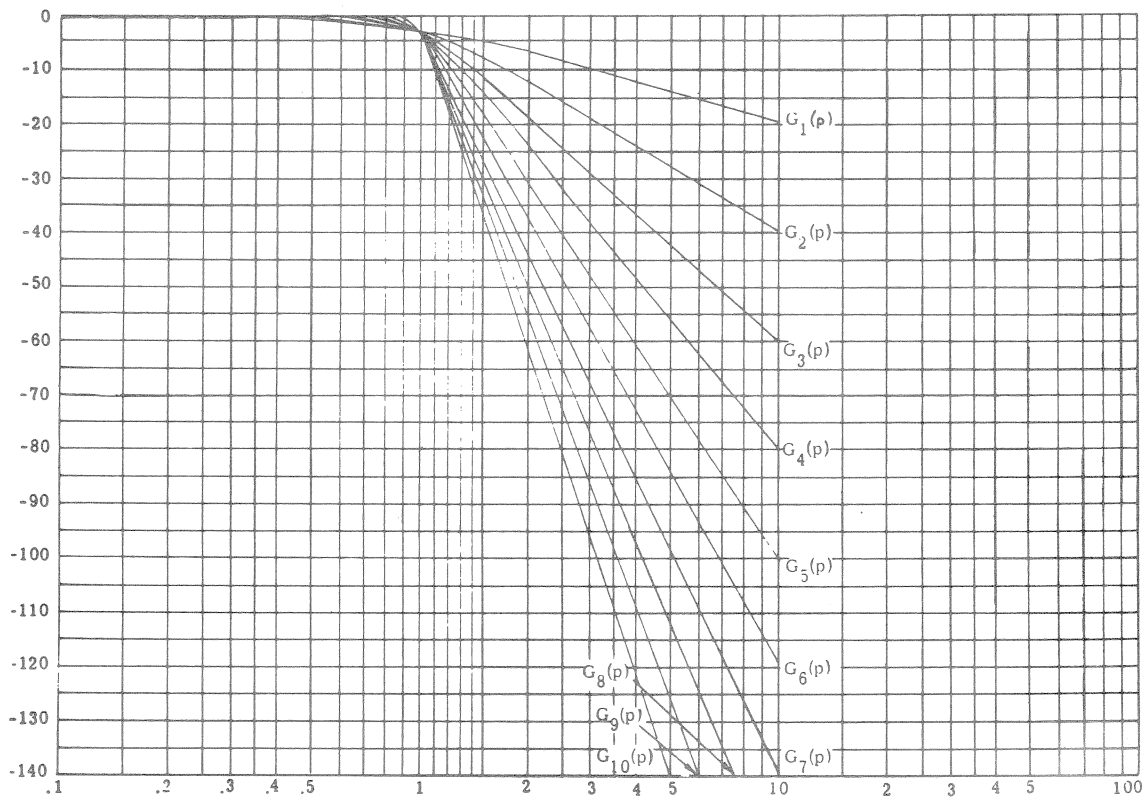


Figure LFILTR-1. Gain (db) vs Frequency Normalized - Butterworth

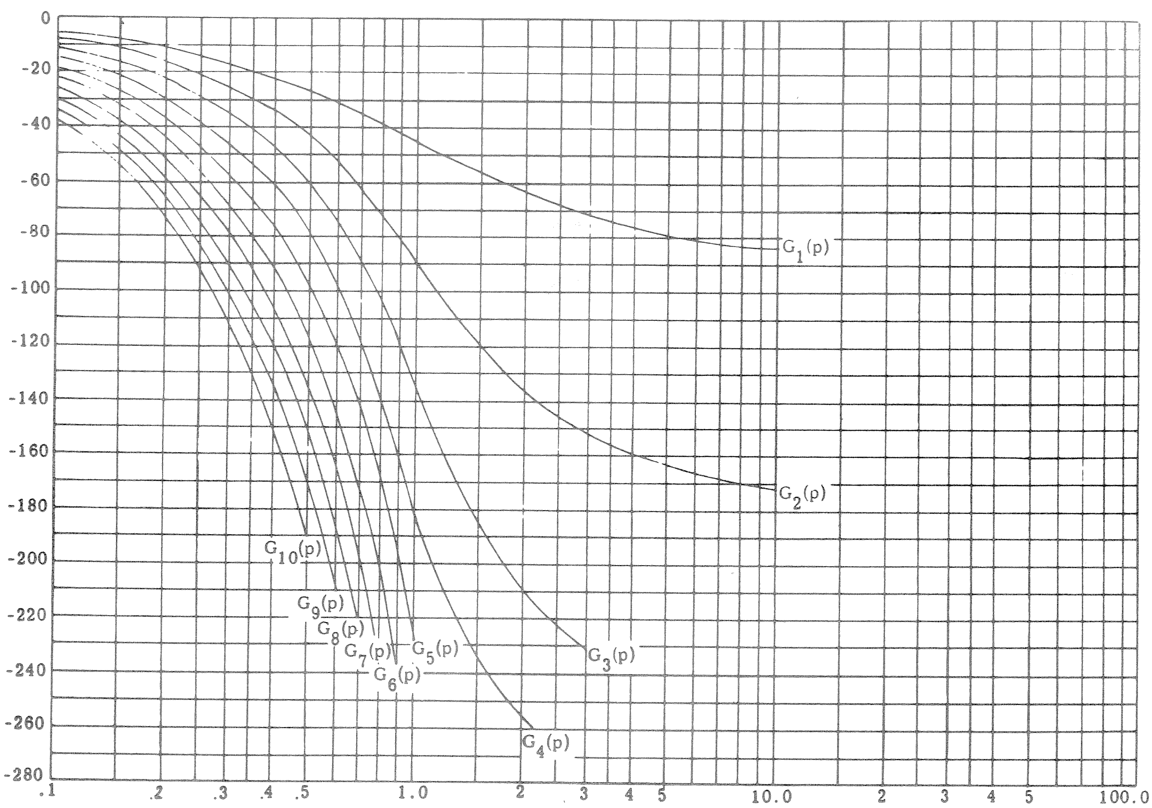


Figure LFILTR-2. Phase (Deg) vs Frequency Normalized - Butterworth

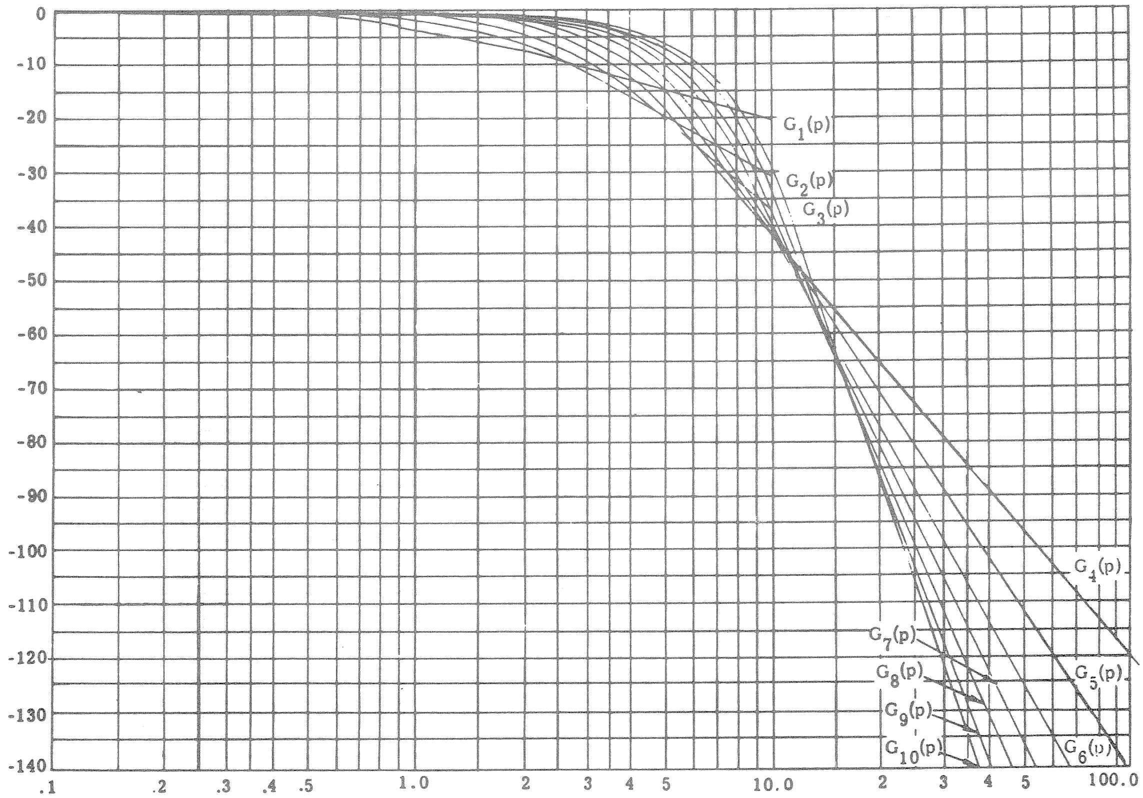


Figure LFILTR-3. Gain (db) vs Frequency Normalized Bessel

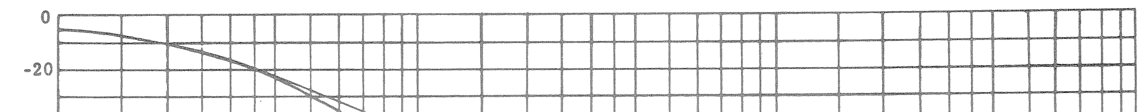


Figure LFILTR-4. Phase (deg) vs Frequency Normalized Bessel

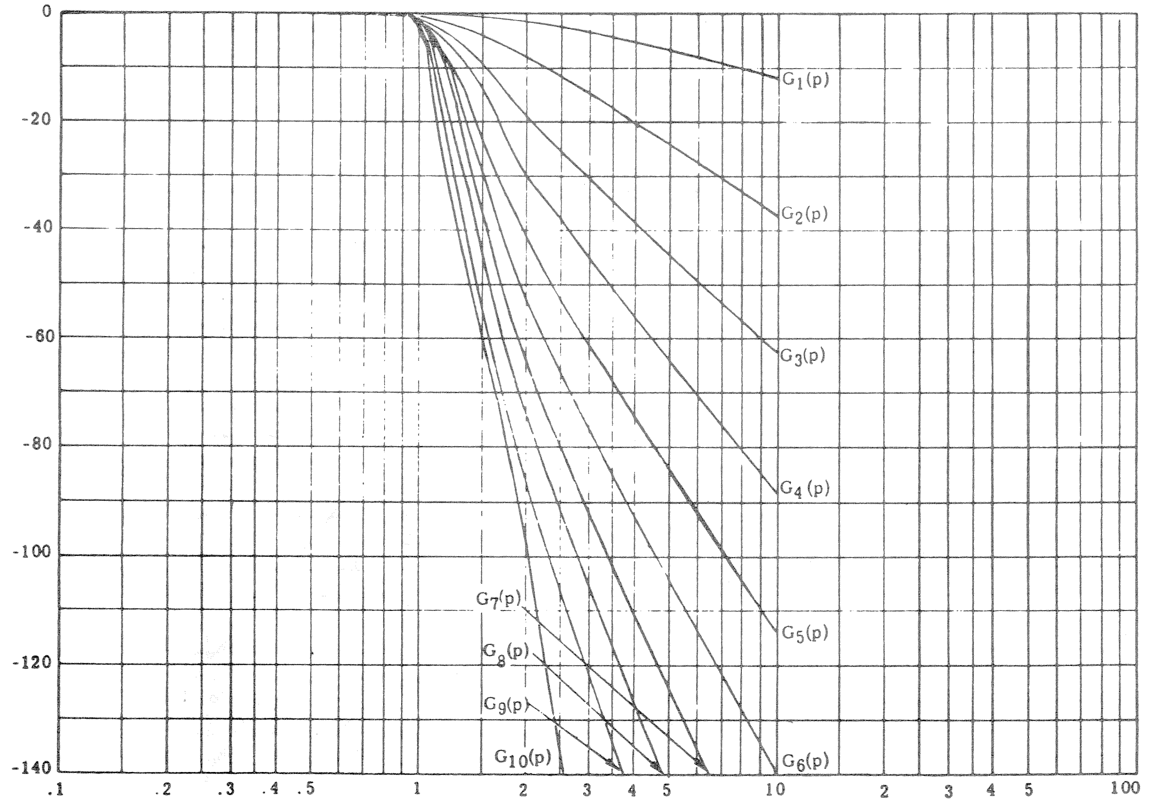


Figure LFILTR-5. Gain (db) vs Frequency Normalized - Chebyshev 1/2 db Ripple

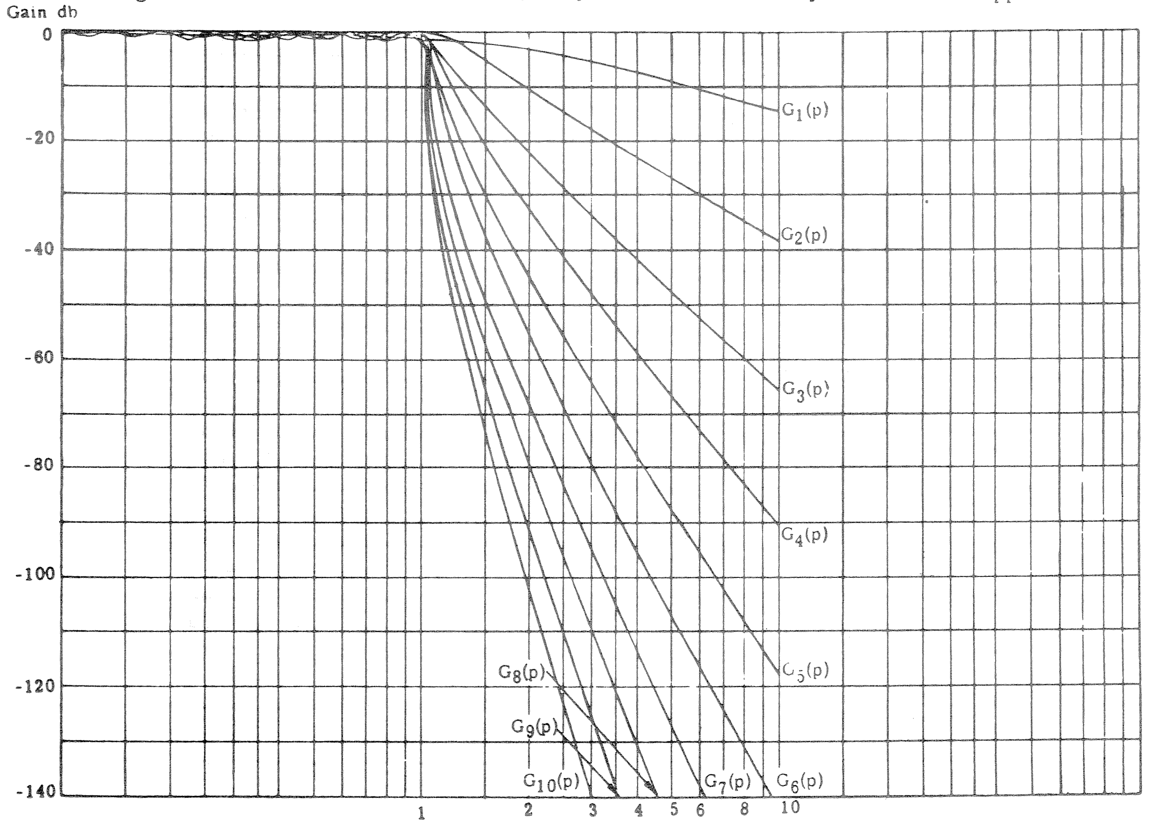


Figure LFILTR-6. Gain (db) vs Frequency Normalized - Chebyshev 1 db Ripple

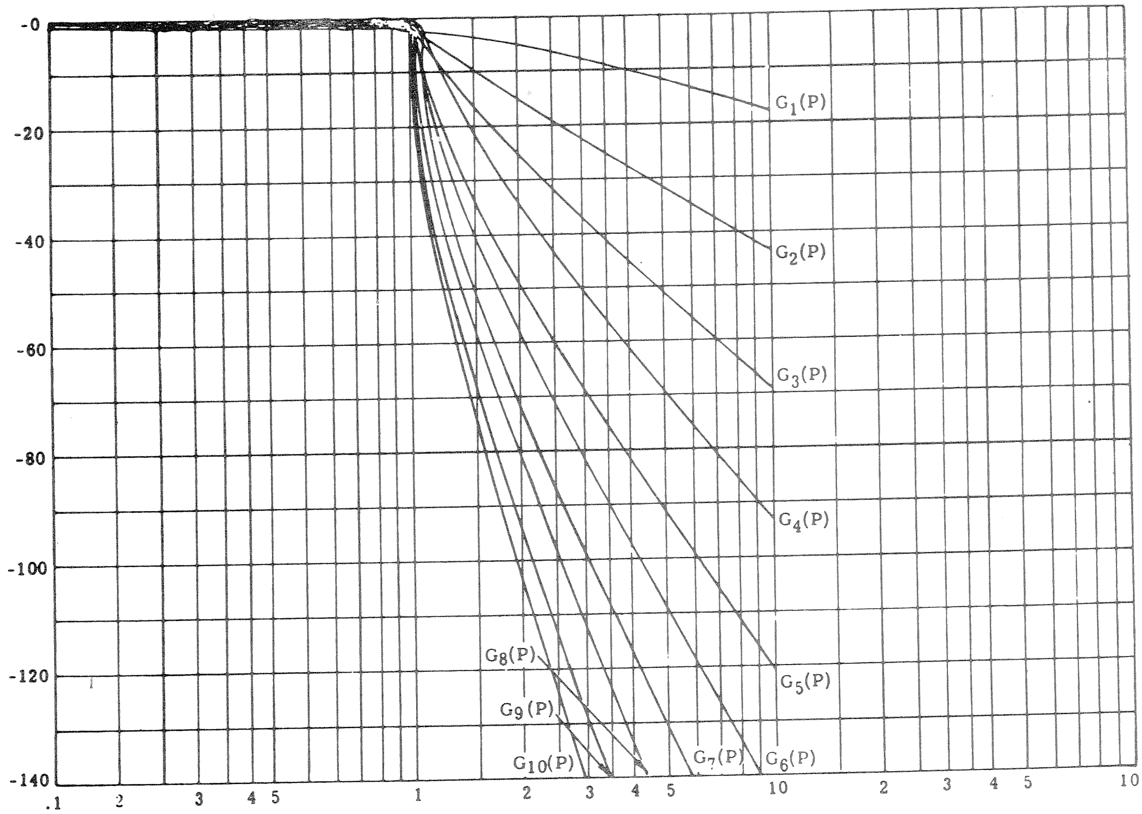


Figure LFILTR-7. Gain (db) vs Frequency Normalized Chebyshev 2 db

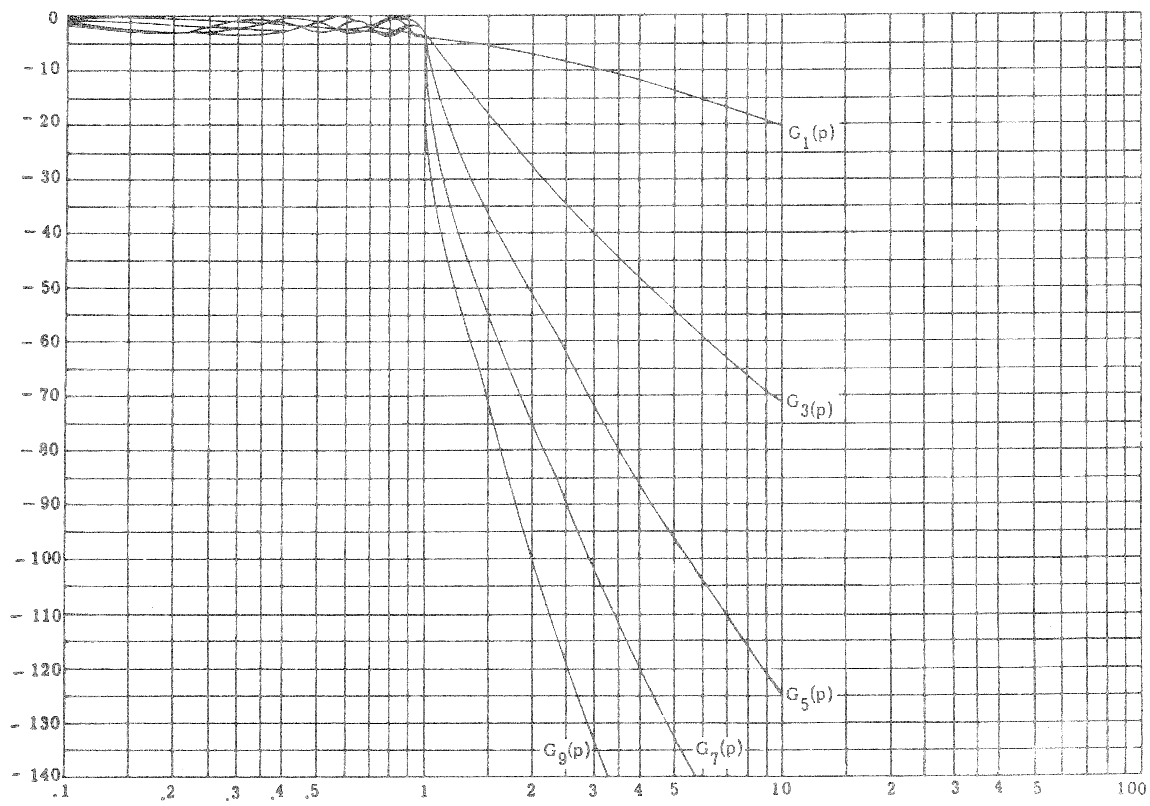


Figure LFILTR-8-a. Gain (db) vs Frequency Normalized Chebyshev (3 db Ripple)

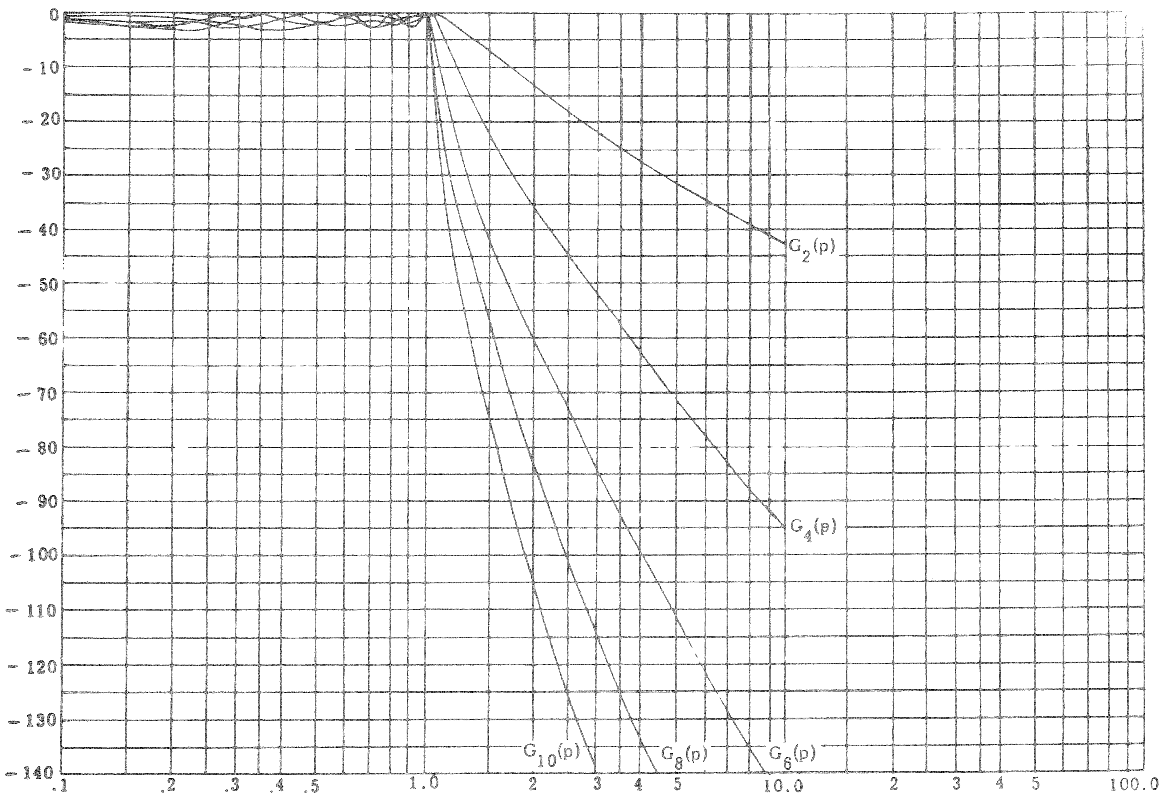


Figure LFILTR-8-b. Gain (db) vs Frequency Normalized Chebyshev (3 db Ripple)

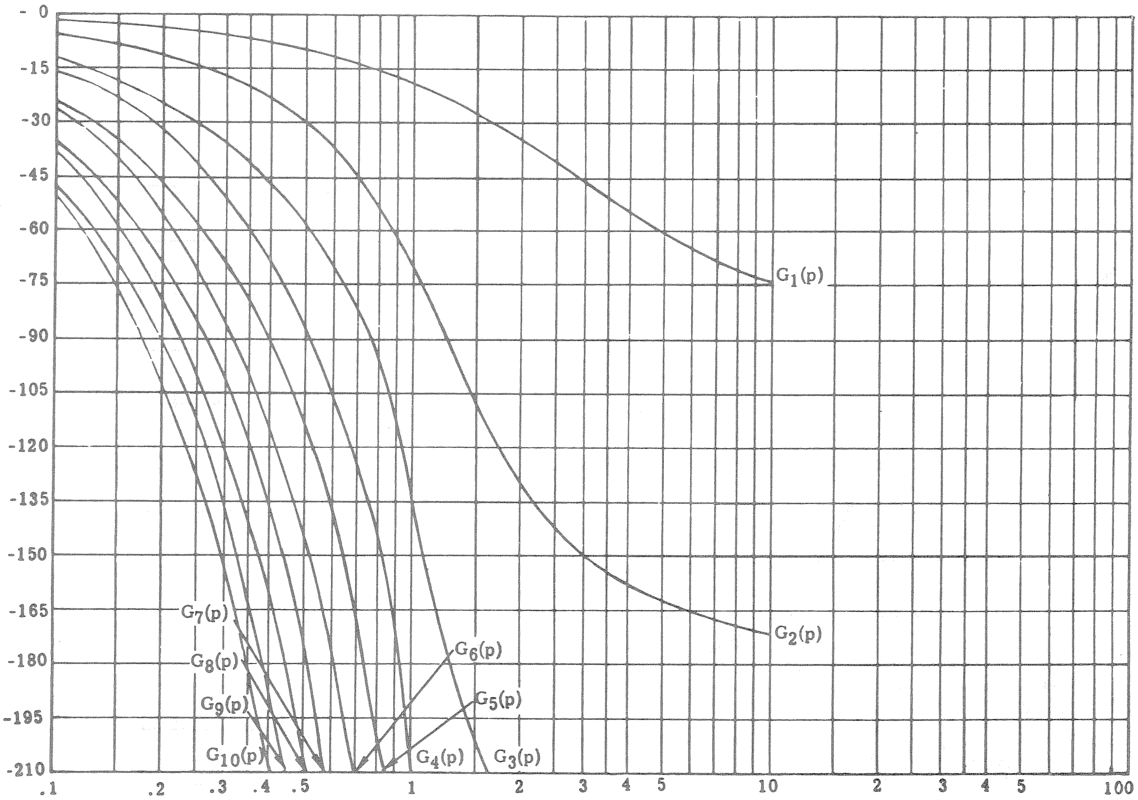


Figure LFILTR-9. Phase (Deg) vs Frequency Normalized Chebyshev 1/2 db Ripple

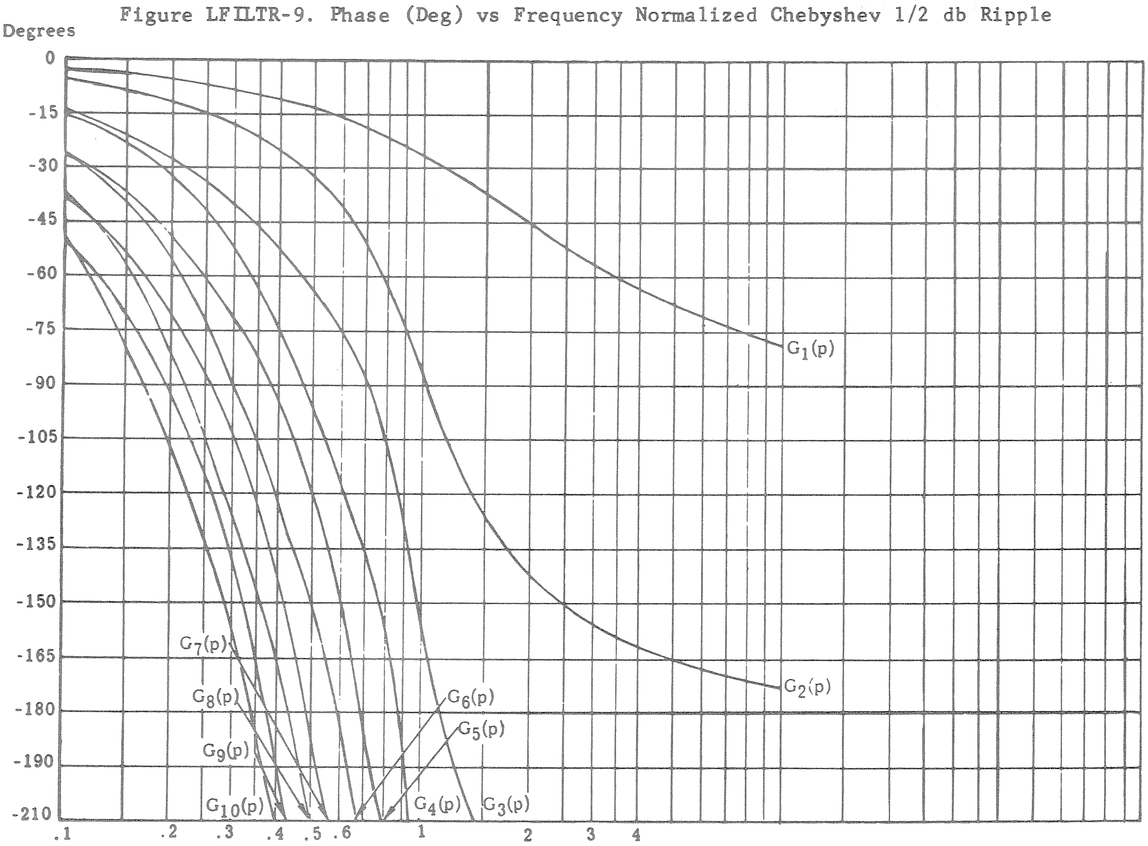


Figure LFILTR-10. Phase (Deg) vs Frequency Normalized - Chebyshev 1 db Ripple

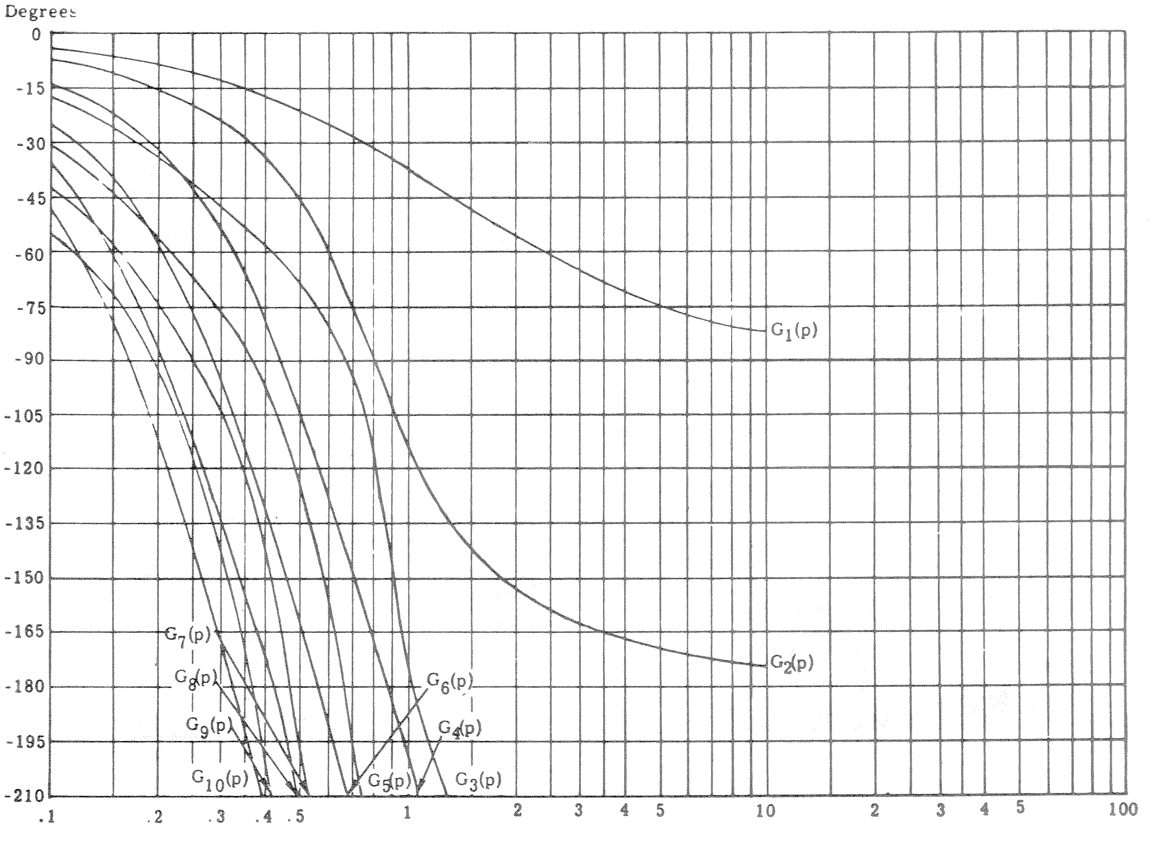


Figure LFILTR-11. Phase (Deg) vs Frequency Normalized Chebyshev 2 db Ripple

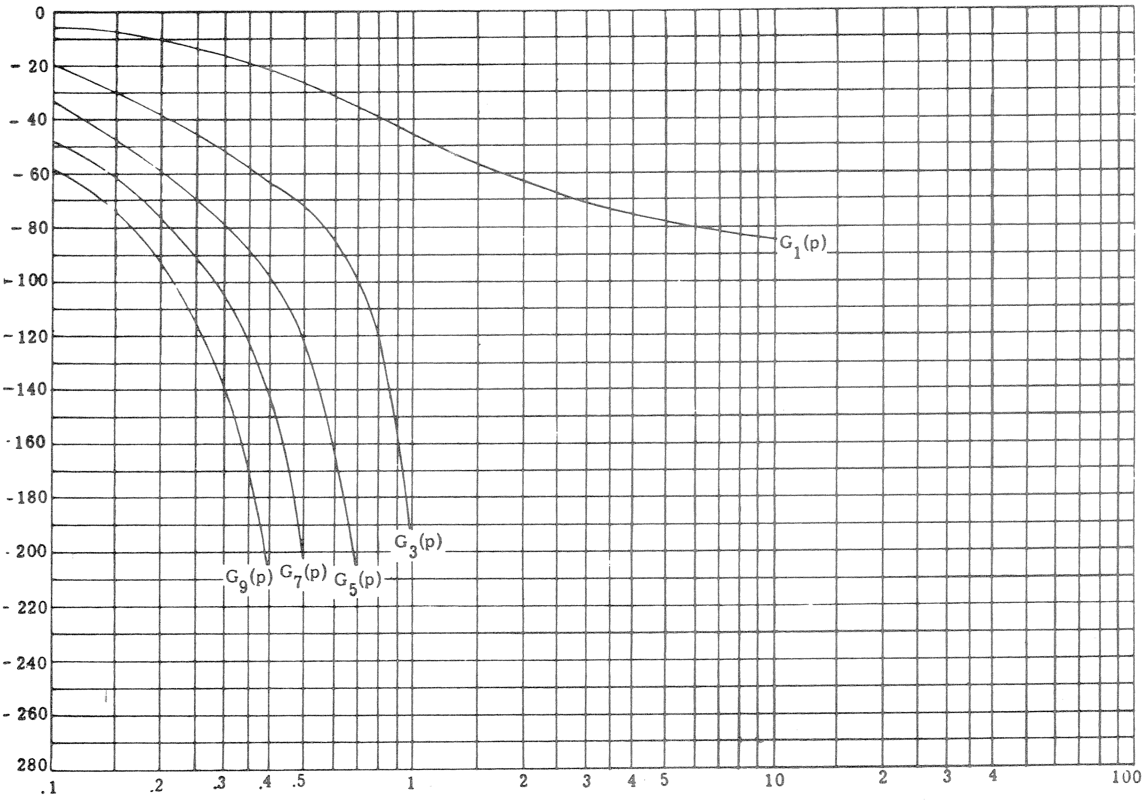


Figure LFILTR-12-a. Phase (Deg) vs Frequency Normalized Chebyshev (3 db Ripple)

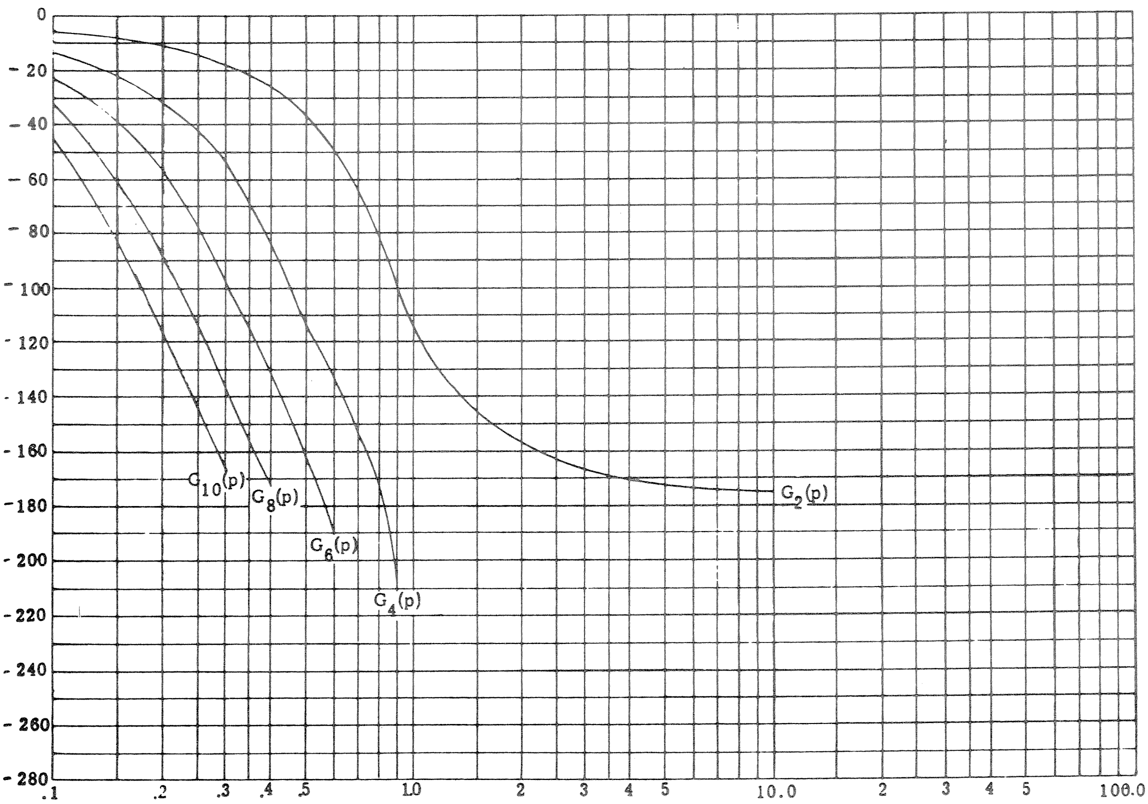


Figure LFILTR-12-b. Phase (Deg) vs Frequency Normalized Chebyshev (3db Ripple)

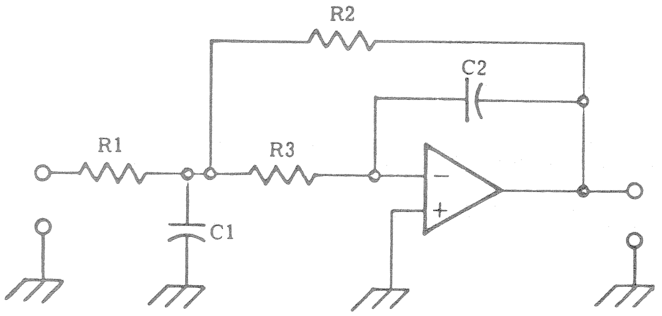


Figure LFILTR-13-a. Second Order Rauch Filter

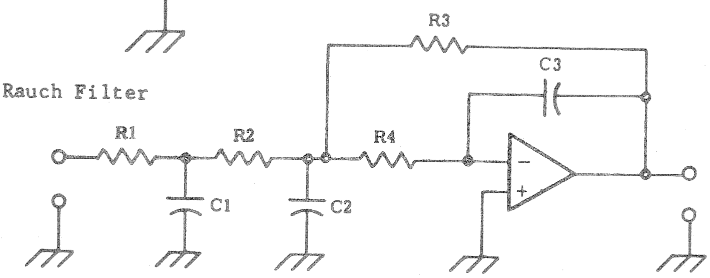


Figure LFILTR-13-b. Third Order Rauch Filter

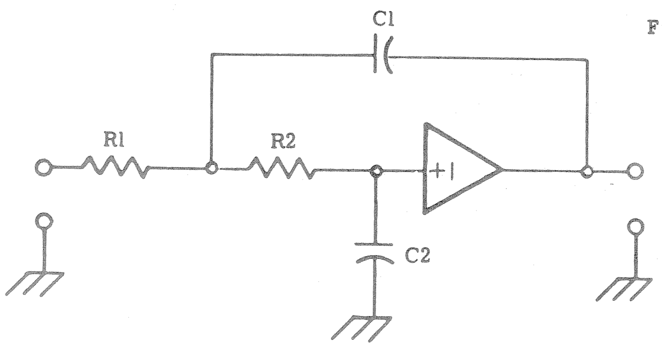


Figure LFILTR-14-a. Second Order Conventional Filter

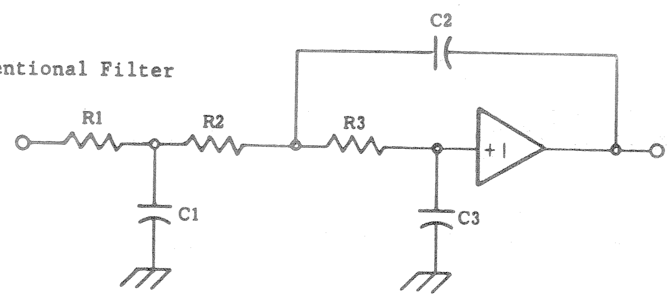


Figure LFILTR-14-b. Third Order Conventional Filter

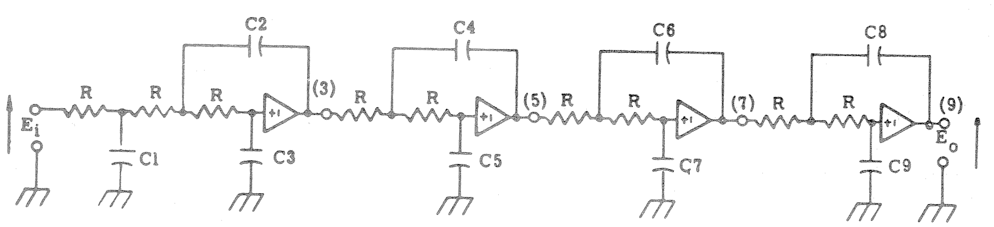


Figure LFILTR-15-a. Ninth-Order Filter

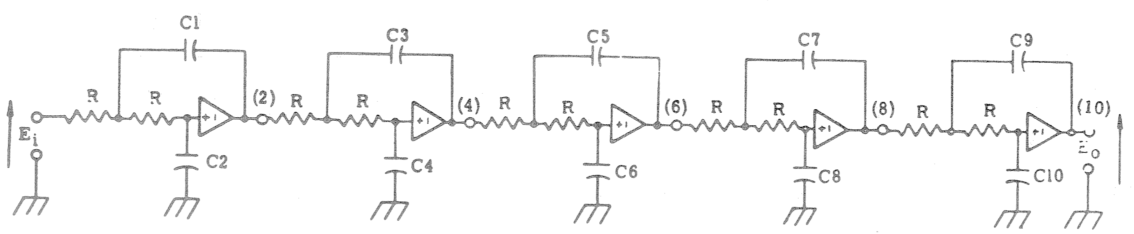


Figure LFILTR-15-b. Tenth-Order Filter

This BASIC program designs low pass filters using constant K prototype T section and M-derived [M=0.6] termination L sections. Up to nine additional M-derived T sections may be included to give high attenuation at specified frequencies in the stop band.

INSTRUCTIONS

To use this program, enter data into lines 10 - 15 as:

10 DATA R, C, N, F(1), F(2), . . . , F(N)

where

- R is the desired characteristic impedance in ohms
- C is the desired cutoff frequency in cycles/second
- N is the number of attenuators desired in stop band
- F(I) is the frequency for attenuator I

Then type RUN.

For additional instructions, list the program.

SAMPLE PROBLEM

Design a low pass filter with a 50 ohm impedance, 20 KH_z cut-off frequency, 2 attenuators in stop band, 455KH_z and 91 KH_z attenuators in the filter.

SAMPLE SOLUTION

```
 10 DATA 50,2E4,2,455000,91E3
*RUN

LPFILT



DESIGN FØR DESIRED LØW PASS FILTER:

0<--------          50          ØHM LINE             -------->0
I                                                              I
+--------   0.4244132 MH +         0.095493   MFD ----------+
I                                                              I
>                   0.6366198   MH                             I
I                                                              I
+-------------   0.3183099     MFD     ---------------------+
I                                                              I
>                   0.7953901   MH                             I
I                                                              I
+--------   0.0003848 MH +       0.3180022   MFD ----------+
I                                                              I
>                   0.7856615   MH                             I
I                                                              I
+--------   0.0098505 MH +         0.310527   MFD ----------+
I                                                              I
>                   0.6268912   MH                             I
I                                                              I
+--------   0.4244132 MH +         0.095493   MFD ----------+
I                                                              I
0<--------          50          ØHM LINE             -------->0

TERMINATING SECTIØNS GIVE MAXIMUM ATTENUATIØN AT    25000
CPS IN ADDITIØN TØ THE SPECIFIED ATTENUATØR FREQUENCIES.
```

This Fortran program performs a general steady-state circuit analysis. Up to 30 nodes and 60 parts may be accommodated. Input is via a description of the circuit schematic. Networks to be analyzed may consist of R, C, L, β, α, V, I, diode, thermal resistance, and ambient temperature elements. All elements are individually named and toleranced.

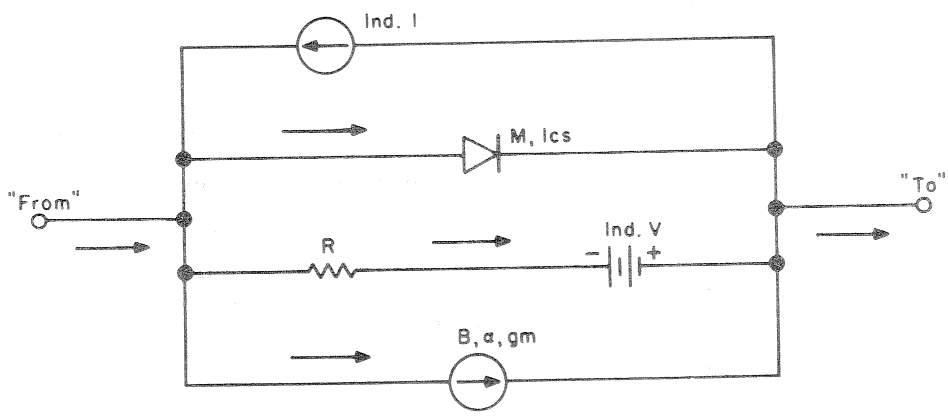
INSTRUCTIONS

Data Preparation

Draw the circuit schematic into its equivalent circuit form. Replace transistors, transformers, F.E.T.'s, etc., with the proper model. Label the schematic according to the following:

1. Number of the nodes or interconnection points of the circuit from 0 to N where the N-th or last node is the desired output node and the 0-th node is the ground or reference node.

2. Number the branches consecutively, starting with one. Each branch may include all or part of the generalized branch shown below.



3. Number the parts with all independent sources numbered first. Dependent sources must numerically follow their controlling part and thermal resistances must precede the diodes they reference.

   When the program encounters a dependent source, the source is assumed to be controlled by a resistor or diode branch already described. Later data statements may add additional parallel branches. This also means that a dependent source must be described after its controlling branch.

   An independent voltage source must include a resistor in series with it within that total branch.

4. Sketch in an arbitrary current flow through the entire branch. Note that independent currents flow opposite to the general branch flow. The "FROM" - and "TO" - node convention is used to indicate that current flows "FROM" some node, through the general branch, and "TO" some other node.

Create a data file containing the circuit description. Select any file name up to eight characters in length. The first line of the data file must be:

Line number P, B, N

where

- P is the problem name (a short string)
- B is the number of branches in the circuit
- N is the number of nodes (does not include the datum node)

Each succeeding line must now describe a circuit element as follows:

P BRANCH NAME, FROM-NODE, TO-NODE, VALUE, TOLERANCE, (OPT) CONTROL BR.

where

- P is a line number which may be used to indicate the number of the part.
- BRANCH is the branch number of the element being described.
- NAME is a 1- to 8-character part name. The first letter of the name describes the part:
  - R, Resistor
  - D, Diode (see above for an example of data for this element).
  - "OH", Thermal resistance in degrees Celsius per watt
  - B, Current-controlled dependent current source
  - G, Voltage-controlled dependent current source
  - E or V, Independent voltage source
  - I, Independent current source
  - T, Ambient temperature
- VALUE is the size or magnitude of the element in ohms, henries, farads, volts, etc.
- TOLERANCE Percent for all parts.
- CONTROL BRANCH is used with dependent sources and thermal resistances to indicate controlling branch number. Include two branch numbers with thermal resistance.

Example of typical lines:

A. Diode, called DBE3, located in branch 5 between nodes 7 and 8, with an "M" of 1.5±5% and I-ECS of 1.3 picoamps.

    17 5, DBE3, 7,8,1.5,5,1.3E-12

B. Thermal Resistances of the same diode called "O-JAS", with a value of 25±33% degrees/watt and controlled by dissipation in itself (5) and branches 8 and 9

    16 5, O-JAS, 7,8,25,33,8,9

C. Alpha Generator located in branch 6 with an equivalent Beta of 150±33%, connected between nodes 8 and 9 and controlled by the current through branch 5 (DBE3). (Note the negative value of Beta which used to signal an Alpha.)

19 6, BALPHA, 8,9,-150, 33,5

D. Ambient Temperature of 25±33% degrees.

37 0, T-A,0,0,25,33

(Use 0's to skip undesired data.)

EXECUTION

Enter the code number of the command to be executed. These commands are:

COMMAND	CODE	RESULT
ALL VOLTAGES AND CURRENTS	1	All branch voltages, currents, dissipations, and node voltages are found.
PART EFFECTS AND WORST CASE	2	The normalized partial derivatives of the output, expressed as a percent for each part, are calculated together with the part's sensitivity (partial times tolerance). This is done by varying the part 1%, and comparing the new and nominal outputs. The Worst Case assumes that the signs of the partials do not change from the nominal case.
STEP A PART	3	On-line change of a part in 10 steps.
CHANGE A PART	4	Nominal values, tolerances, and names changed for the remainder of the run.
EFFECT OF ONE PART ONLY	5	See Command = 2
WORST CASE ONLY	6	See Command = 2
MONTE CARLO	7	Assuming uniformly distributed parts, sets of parts are appropriately generated, and output calculated and statistically accumulated.
STOP	8	Terminate run.

NOTE: Error messages generally indicate faulty data. The data should then be rechecked, corrected, and rerun.

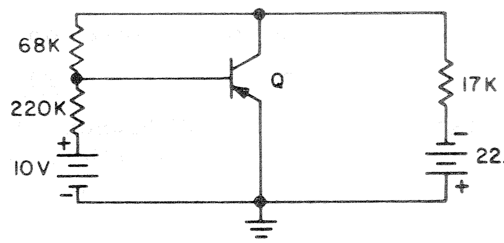
The program handles circuits of up to 30 nodes, 50 branches, 70 components, 20 dependent current sources.

REFERENCE

Grout, J.S., NLNET, Branch-Input, Non-Linear Steady State Response by Computer., General Electric Company, T.I.S. 69 APD-2, 1969, GE Technical Information Exchange, P.O. Box 43, Bldg. 5, Schenectady, N.Y. 12301

SAMPLE PROBLEM

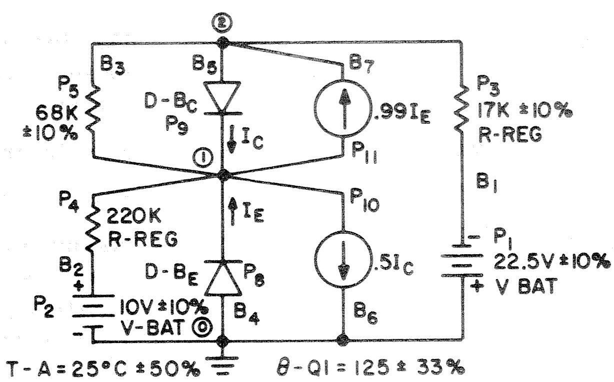
Analyze the circuit illustrated.



Q-data
$\beta_N = 99$
$\beta_I = 1$
$I_{ES} = I_{CS} = 1$ na
$M_E = M_C = .973607$

SAMPLE SOLUTION

Redraw it into its equivalent circuit. Number all nodes, branches, and parts. Sketch in an assumed current direction. Label all parts with a name, value, and tolerance.



The data was entered into the file NLDATA. NLNET was run illustrating all eight commands.

```
*RUN
FILENAME=
= NLDATA


1 IEEE TEST PRØBLEM
 7 BRANCHES
 2 NØDES
LINE BRANCH    NAME       VALUE
  10     1     VBAT        1.000E+01
  20     2     V-BAT       1.000E+01
  30     1     R-REG       1.000E+01
  40     2     R-RG        1.000E+01
  50     3     R68K        1.000E+01
  60     4     Ø-Q1        3.300E+01
  70     5     Ø-Q1        3.300E+01
  80     4     D-BE        5.000E+00
  90     5     D-BC        5.000E+00
 100     6     BETAR       0.
 110     7     BETAF       0.
 120     0     T-A         5.000E+01

CØMMANDS:
 1=ALL VØLTAGES AND CURRENTS
 2=PART EFFECTS AND WØRSE CASE
 3=STEP A PART
 4=CHANGE A PART
 5=EFFECT ØF ØNE PART ØNLY
 6=WØRST CASE ØNLY
 7=MØNTE CARLØ
 8=STØP


NØMINAL ØUTPUT =  -4.2403E+00 AT NØDE   2


CØMMAND NØ. =
= 1

          BRANCH          BRANCH          BRANCH          NØDE
NØ.       VØLTS           MA.             MW.             VØLTS
 1     -4.2403E+00     1.0741E+00     -4.5545E+00     -3.4457E-01
 2      3.4457E-01     4.7021E-02      1.6202E-02     -4.2403E+00
 3      3.8958E+00     5.7291E-02      2.2319E-01
 4      3.4457E-01     1.0271E+00      3.5390E-01
 5     -3.8958E+00    -1.0881E-06      4.2391E-06
 6     -3.4457E-01    -5.4406E-07      1.8747E-07
 7      3.8958E+00     1.0168E+00      3.9612E+00


CØMMAND NØ. =
= 2

                       EFFECT ØN     TØLERANCE
PART NAME              ØUTPUT-PCT    EFFECT-PCT
 1   VBAT                 21.3          2.1
 2   V-BAT                69.2          6.9
 3   R-REG               -17.3         -1.7
 4   R-RG                -71.5         -7.2
 5   R68K                 89.1          8.9
 6   Ø-Q1                 -0.0         -0.0
 7   Ø-Q1                 -0.0         -0.0
 8   D-BE                 10.2          0.5
 9   D-BC                  0.0          0.0
10   BETAR                 0.0          0.
11   BETAF             -1626.4          0.
12   T-A                  -2.0         -1.0


WØRST CASE    MINIMUM      NØMINAL      MAXIMUM     RANGE-PCT
ØUTPUT      -3.1754E+00 -4.2403E+00 -5.6021E+00    57.2
```

```
COMMAND NO. =
= 3

INCREASE THE VALUE OF A PART IN 10 STEPS.
 PART NO., INITIAL VALUE, STEP SIZE=
= 12,0,5

   T-A              OUTPUT
    0.             -4.3269E+00
    5.0000E+00     -4.3095E+00
    1.0000E+01     -4.2923E+00
    1.5000E+01     -4.2750E+00
    2.0000E+01     -4.2576E+00
    2.5000E+01     -4.2403E+00
    3.0000E+01     -4.2230E+00
    3.5000E+01     -4.2056E+00
    4.0000E+01     -4.1882E+00
    4.5000E+01     -4.1705E+00
    5.0000E+01     -4.1526E+00


COMMAND NO. =
= 4

CHANGE A PART
 PART NO., NAME, NOM. VALUE, AND TOLERANCE(PCT)=
= 12,HIGH-T-A,65,20

NOMINAL OUTPUT =  -4.0932E+00 AT NODE   2


COMMAND NO. =
= 5

 EFFECT OF ONE PART.  PART NO.=
= 12

                     EFFECT ON     TOLERANCE
PART NAME            OUTPUT-PCT    EFFECT-PCT
12  HIGH-T-A           -7.2          -1.4


COMMAND NO. =
= 6


WORST CASE   MINIMUM     NOMINAL     MAXIMUM   RANGE-PCT
OUTPUT      -3.0103E+00 -4.0932E+00 -5.4513E+00  59.6


COMMAND NO. =
= 7


MONTE CARLO ANALYSIS
  (UNIFORM DIST. PARTS)
 NUMBER OF TRIALS =
= 25


NOMINAL OUTPUT = -4.0932E+00
AVERAGE OUTPUT = -4.1401E+00
          SIGMA =  3.0411E-01

3-SIGMA LIMITS = -5.0525E+00 TO -3.2278E+00
```

```
CØMMAND NØ. =
= 1

        BRANCH          BRANCH          BRANCH          NØDE
NØ.     VØLTS           MA.             MW.             VØLTS
 1   -4.0932E+00     1.0828E+00     -4.4319E+00     -2.3445E-01
 2    2.3445E-01     4.6520E-02      1.0907E-02     -4.0932E+00
 3    3.8588E+00     5.6746E-02      2.1897E-01
 4    2.3445E-01     1.0361E+00      2.4291E-01
 5   -3.8588E+00    -2.6932E-04      1.0392E-03
 6   -2.3445E-01    -1.3466E-04      3.1571E-05
 7    3.8588E+00     1.0257E+00      3.9581E+00

CØMMAND NØ. =
= 8

PRØGRAM STØP AT 17600
*LIST NLDATA

1 IEEE TEST PRØBLEM,7,2
10 1,VBAT,2,0,22.5,10
20 2,V-BAT,0,1,10,10
30 1,R-REG,2,0,17E3,10
40 2,R-RG,0,1,220E3,10
50 3,R68K,1,2,68E3,10
60 4,Ø-Q1,0,1,125,33,5,7
70 5,Ø-Q1,2,1,125,33,4,7
80 4,D-BE,0,1,.973607,5,1E-9
90 5,D-BC,2,1,.973607,5,1E-9
100 6,BETAR,1,0,.5,0,5
110 7,BETAF,1,2,.99,0,4
120 0,T-A,0,0,25,50

READY

*
```

This BASIC program calculates various quantities for an Otto cycle-engine. This is a program which will calculate the following items for the Otto cycle using the CFR engine.

1. Air flow into engine
2. Fuel flow into engine
3. Air fuel ratio
4. Brake horsepower
5. Friction horsepower
6. Indicated horsepower
7. Brake thermal efficiency
8. Indicated thermal efficiency
9. Brake specific fuel consumption
10. Indicated specific fuel consumption
11. Ideal thermal efficiency
12. Relative efficiency
13. Volumetric efficiency

INSTRUCTIONS

The above values are calculated from data which is requested by the computer as it is required.

SAMPLE PROBLEM

The user solves the problem by answering the questions asked in the program.

SAMPLE SOLUTION

```
BRAKE SPECIFIC FUEL CØNSUMPTIØN =    6.676572 LBS/HP-HR

INDICATED SPECIFIC FUEL CØNSUMPTIØN =    3.338286 LBS/HP-HR

CØMPRESSIØN RATIØ = ?10.

IDEAL THERMAL EFFICIENCY =    .6018928

RELATIVE EFFICIENCY =    .0603198


PRESSURE DRØP ACRØSS LAMINAR METER (IN. ØF WATER) = ?.7
BARØMETRIC PRESSURE (INCHES ØF HG) = ?30.75
RØØM TEMPERATURE (FAHRENHEIT) = ?80.

AIR FLØW INTØ ENGINE =    8.704195 CUBIC FEET PER MINUTE

AIR FLØW INTØ ENGINE =    .6568853 PØUNDS PER MINUTE

TIME REQUIRED FØR 21.5 CC ØF FUEL (MINUTES) = ?.21
SPECIFIC GRAVITY ØF FUEL = ?.8

FUEL FLØW INTØ ENGINE =    .1804882 PØUNDS PER MINUTE

AIR FUEL RATIØ =    3.639491

BRAKE WATTMETER READING IN KILØWATTS = ?1.21

BRAKE HØRSEPØWER =    1.621984

FRICTIØN WATTMETER READING IN KILØWATTS = ?1.21

FRICTIØN HØRSEPØWER =    1.621984


INDICATED HØRSEPØWER =    3.243968

HIGHER HEATING VALUE ØF FUEL = ?21000

BRAKE THERMAL EFFICIENCY = .018153

INDICATED THERMAL EFFICIENCY =    .0363061

BRAKE SPECIFIC FUEL CØNSUMPTIØN =    6.676572 LBS/HP-HR

INDICATED SPECIFIC FUEL CØNSUMPTIØN =    3.338286 LBS/HP-HR

CØMPRESSIØN RATIØ = ?10.

IDEAL THERMAL EFFICIENCY =    .6018928

RELATIVE EFFICIENCY =    .0603198

CYLINDER BØRE (ASSUMED) = 3.25 INCHES
PISTØN STRØKE (ASSUMED) = 4.5 INCHES
ENGINE SPEED (900<=RPM<=1000) = ?1000

VØLUMETRIC EFFICIENCY =    .4029059

READY
*
```

This BASIC program calculates the cost and the number of tons of paving material that will be needed to pave a stretch of road. The program contains a ratio of .055 for tons per square yard for 1 inch of covering. The user can change this ratio; it is stored at line 110.

Other information required to run the program follows:

1. Price/ton of mix.
   a. 1/2 in. stone mix
   b. 3/4 in. stone mix
2. Number of segments to be paved
3. Width of each segment in feet
4. Length of each segment in feet
5. Thickness of each segment in inches

Use decimal notation for fractions of feet and inches.

INSTRUCTIONS

Type RUN and the program will ask for the needed information. After the data is entered, the output is printed.

SAMPLE PROBLEM

The user calculates the cost and number of tons of paving material needed by answering questions interactively.

```
# RUN

   DO YOU HAVE PRICES AVAILABLE FOR MIX ?YES

   ENTER PRICE/TON OF 1/2 INCH STONE MIX ?12.50

   ENTER PRICE/TON OF 3/4 INCH STONE MIX ?10.50

   ENTER THE TOTAL NUMBER OF SEGMENTS
 TO BE PAVED ?3


 SEGMENT NO. 1

   ENTER WIDTH OF SEGMENT NO.     1  ?29.8

   ENTER THE LENGTH OF SEGMENT NO.     1  ?489

   ENTER COVERING THICKNESS IN INCHES THAT IS DESIRED
 FOR SEGMENT NO.     1 .USE DECIMAL NOTATION FOR
 FRACTIONS OF AN INCH    ?2

 FOR SEGMENT NO.     1 , SPECIFY STONE MIX BY TYPING
  '1' FOR 1/2 IN. MIX, OR '2' FOR 3/4 IN. MIX.   ?2

 FOR SEGMENTS 2 -     3 ENTER FOUR PIECES OF DATA.  THE WIDTH
 THE LENGTH, THE THICKNESS, AND THE STONE MIX.

 SEGMENT NO.     2  ?33,3000,3,2
 SEGMENT NO.     3  ?61.2,5280,4,1


                  TONS               TONS
 SEGMENT          1/2 INCH           3/4 INCH          SQ.YDS . TO
 NUMBER           STONE MIX          STONE MIX         PAVE
 =========================================================================

      1                              178.105            1619.133

      2                             1815.000           11000.000

      3           7898.880                             35904.000

                  ---------          ---------         ---------
 TOTALS           7898.880          1993.105           48523.133
 TOTAL COSTS--    $119663.60
```

```
  DØ YØU WISH TØ MAKE ANY CHANGES IN
CØVERING THICKNESS ØR STØNE SIZE MIX
FØR ANY SEGMENT ?YES

  IN HØW MANY SEGMENTS DØ YØU WISH TØ MAKE CHANGES ?1

  FØR EACH CHANGE, ENTER SEGMENT NUMBER TØ BE
CHANGED, THE NEW THICKNESS IN INCHES (DECIMAL
FØR FRACTIØNS ØF AN INCH), AND THE NEW STØNE
SIZE MIX (1 FØR 1/2 IN.,2 FØR 3/4 IN.). IF YØU
WISH TØ CHANGE ØNLY ØNE ØF THESE, TYPE '0' FØR
VARIABLE NØT TØ BE CHANGED


 FØR CHANGE NØ.       1  ?3,4.8,2


 RESULTS FØR ALTERATIØN NØ.      1



                  TØNS              TØNS
SEGMENT           1/2 INCH          3/4 INCH          SQ.YDS. TØ
NUMBER            STØNE MIX         STØNE MIX         PAVE
==================================================================

      1                              178.105           1619.133

      2                             1815.000          11000.000

      3                             9478.656          35904.000

                  ---------         ---------         ---------
TØTALS                 .000        11471.760          48523.133
TØTAL CØSTS--     $120453.48


  DØ YØU WISH TØ MAKE FURTHER CHANGES ?NØ

READY
*
```

This Fortran program, given a temperature and pressure, uses Newton's method to compute the molar volume of a gas whose pressure-temperature behavior is described by the Beattie-Bridgeman equation of state.

INSTRUCTIONS

Coefficients for a number of compounds are contained within the program (see the sample solution listing). The program will request the chemical formula for the gas. If the coefficients for that gas are not in the program, the user must supply them. Either the pressure or the temperature may be varied. A plot may also be requested.

SAMPLE PROBLEM

Find the molar volumes for CO_2 at 1 atmosphere pressure between 40°C and 100°C.

SAMPLE SOLUTION

```
*RUN



PVT


DØ YØU WANT INSTRUCTIØNS?
= YES

GIVEN A TEMPERATURE AND PRESSURE , THIS PRØGRAM
USES NEWTØNS METHØD TØ CØMPUTE THE MØLAR VØLUME ØF A
GAS WHØSE PRESSURE-TEMPERATURE BEHAVIØR IS DESCRIBED BY THE
BEATTIE-BRIDGEMAN EQUATIØN ØF STATE.  INPUT DATA IS IN DEGREES C
AND ATMØSPHERES.  EITHER PARAMETER MAY BE VARIED.CØEFFICIENTS
ARE ØN FILE FØR THE FØLLØWING CØMPØUNDS-REF 'THERMØDYNAMICS'
J.H.KEENAN, WILEY-1941,PP.356-357.
```

FØRMULA	AØ	A	BØ	B	C*10↑(-4)
HE	0.02160	0.05984	0.01400	0.	0.00400
NE	0.21250	0.02196	0.02060	0.	0.10100
AR	1.29070	0.02328	0.03931	0.	5.99000
H2	0.19750	-0.00506	0.02096	-0.04359	0.05040
N2	1.34450	0.02617	0.05046	-0.00691	4.20000
Ø2	1.49110	0.02562	0.04624	0.00421	4.80000
AIR	1.30120	0.01931	0.04611	-0.00110	4.34000
CØ2	5.00650	0.07132	0.10476	0.07235	66.00000
(C2H5)2Ø	31.27800	0.12426	0.45446	0.11954	33.33000
C2H4	6.15200	0.04964	0.12156	0.03597	22.68000
NH3	2.39300	0.17031	0.03415	0.19112	476.87000
CØ	1.34450	0.02617	0.05046	-0.00691	4.20000
N2Ø	5.00650	0.07132	0.10476	0.07235	66.00000
CH4	2.27690	0.01855	0.05587	-0.01587	12.83000
C2H6	5.88000	0.05861	0.09400	0.01915	90.00000
C3H8	11.92000	0.07321	0.18100	0.04293	120.00000
C4H10	17.79400	0.12161	0.24620	0.09423	350.00000
C7H16	54.52000	0.20066	0.70816	0.19179	400.00000

CHEMICAL FØRMULA FØR GAS?
= CØ2

IS THE VARIABLE PARAMETER TEMPERATURE ØR PRESSURE?
= TEMPERATURE

INITIAL AND FINAL TEMPERATURES AND PRESSURE (C AND ATM)?
= 40,100,1

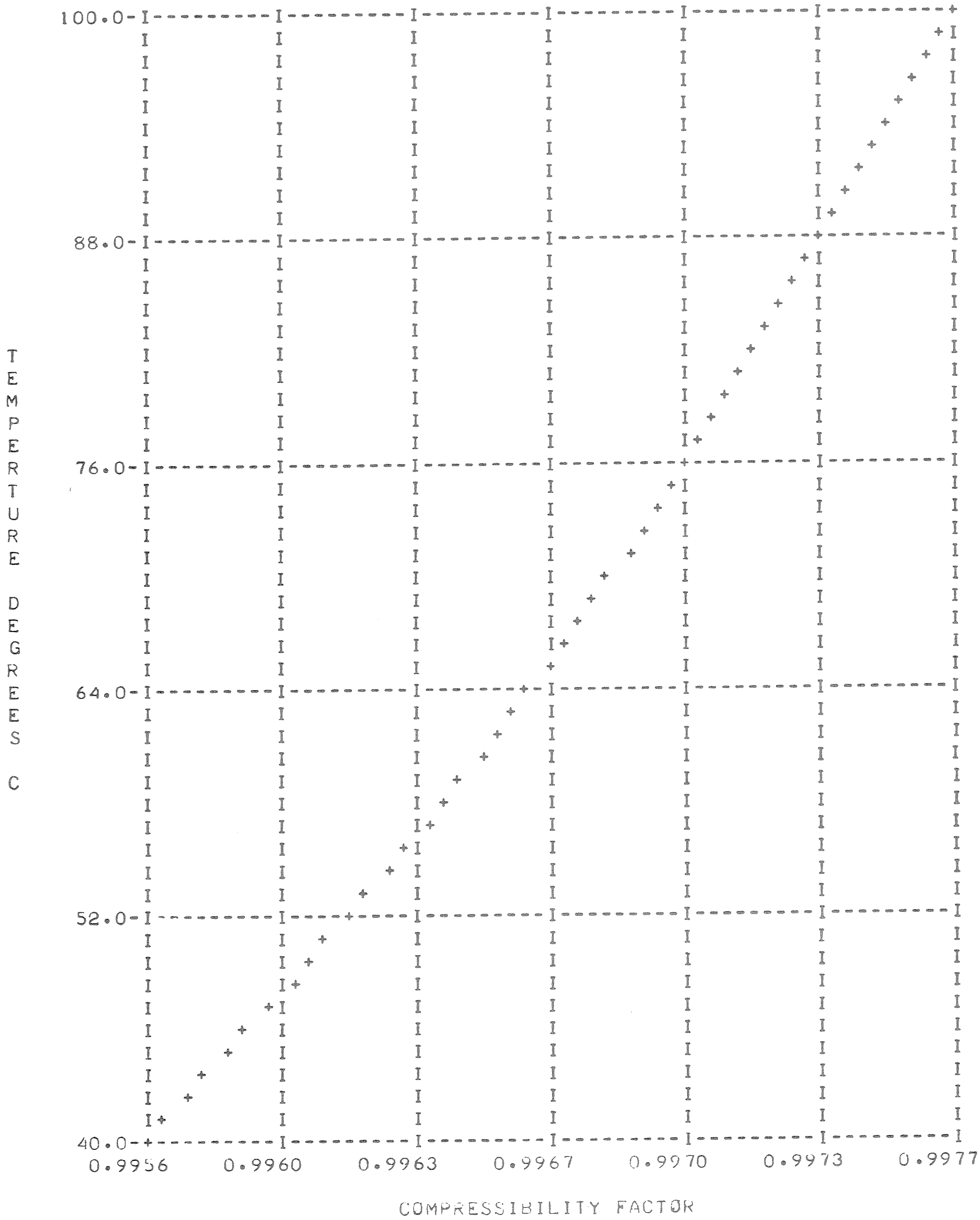
NUMBER ØF TEMPERATURE PØINTS TO TAKE?
= 11

TEMPERATURE DEGREES C	CØMPRESSIBILITY FACTØR	PRESSURE ATM	ITERATIØNS
40.0	0.995644	1.000000	3
46.0	0.995913	1.000000	3
52.0	0.996164	1.000000	3
58.0	0.996397	1.000000	3
64.0	0.996615	1.000000	3
70.0	0.996819	1.000000	3
76.0	0.997010	1.000000	3
82.0	0.997189	1.000000	3
88.0	0.997357	1.000000	3
94.0	0.997515	1.000000	3
100.0	0.997663	1.000000	3

PLØT RESULTS?
= YES

PLØT OF EQUATIØN ØF STATE FØR CØ2

PRESSURE CØNSTANT AT 1.00 ATMØSPHERES



PRØGRAM STØP AT 2710
*

This BASIC program calculates steam control valve coefficients and the required valve rangeability.

INSTRUCTIONS

Enter input data using the following format:

In line 20, enter degrees Farenheit of superheat in the steam or enter 0 if it is 0.

Then in line 25, enter data in groups of three, as many groups as you wish or need, in this order... flow in PPH, inlet PSIG, outlet PSIG; then second flow, second inlet pressure, second outlet press, etc.

Then type RUN.

NOTES:

1. When pressure drop is critical, the program assumes outlet pressure is half the absolute inlet pressure. The program also converts pressures to absolute.
2. Reference is equation 8 of the November, 1961 Fluid Controls Institute, Inc. voluntary standard.
3. To size more valves, enter new data in 20 and 25 and type RUN. If for saturated steam, zero in 20 need be entered only once. It will stay in.

For additional instructions, list the program.

SAMPLE PROBLEM

Determine the valve coefficients, if the superheat in the steam is 463 degrees Farenheit. Other input includes:

	Example 1	Example 2	Example 3	Example 4
Flow (in pph)	1000	10,000	1000	100,000
Inlet	106	106	66	66
Outlet	104	104	64	64

SAMPLE SOLUTION

```
*20 DATA 463
*25 DATA 1000,106,104,10000,106,104,1000,66,64,100000,66,64
*RUN


RATIØ ØF HIGH/LØW CØEFF.=REQUIRED VALVE RANGEABILITY

*** THE  IDENTITY ØF THIS VALVE IS

WHEN SUPERHEAT IN DEGREES F =  463

    INLET PSIG     ØUTLET PSIG     PR.DRØP,PSI        FLØW,PPH    VALVE CØEFF.
        106             104              2             1000         28.81538
        106             104              2            10000         288.1538
         66              64              2             1000         35.31363
         66              64              2           100000         3531.363

READY
*
```

This BASIC program determines for steel sections the resisting moment capacity, axial load capacity, and combined direct stress and bending for any yield point stress.

This program is written to determine, for any WF or I section of any yield point stress:

1. Resisting moment capacity for any unsupported
2. Axial load capacity for main and secondary members, and percentage of overstress and understress
3. Solve formula 6 or formula 7a and 7b for members subject to combined stress and bending

For beams it can be used for sections and yield point stresses not included in charts and tables in the AISC Handbook. The effect of moment at points of lateral support as reflected in coefficient C_b can be determined where formula 4 governs. For axially loaded columns, it can be used for sections not included in column tables and where Kx1X/rx is maximum. For combined direct stress and bending, it determines the adequacy of any section and yeild point stress.

INSTRUCTIONS

See Sample Solution

SAMPLE PROBLEM

See Sample Solution

SAMPLE SOLUTION

```
*1400 DATA 40000,0,20000,36000,.6
*1410 DATA 6.5,.24,.4,11.96,34.1,144
*1420 DATA 7.97,144,144,5.06,1.44,1,1,1
*RUN

SECAP


IF YØU WISH METHØD FØR ENTERING DATA TYPE 1,
IF NØT TYPE 2.
?1
THIS PRØGRAM CØMPUTES FØR WF AND I SECTIØNS RES. MØM.CAP.
AXIAL L'D CAP., AND SØLVES FØRMULAS 6,7A/7B FØR CØMBINED
DIRECT STRESS AND BENDING FØR ANY YP STRESS.
ENTER DATA AS FØLLØWS:
1400 DATA M (AB'T X AXIS), M(AB'T Y AXIS), AXIAL L'D, YP
IN PSI, C SUB M. IF M(X)>0.
1410 DATA B(FL'GE), T(WEB),T(FL'GE),D,S(X),L IN INCHES.
IF P>0, 1420 DATA A,L(X),L(Y),R(X),R(Y),K(X),K(Y),Q(0 FØR
SECØNDARY MEMBER, 1 FØR MAIN MEMBER)
ENTER MØM. IN PF AND AXIAL L'D IN PDS.
PRØGRAM CURRENTLY NØT SET TØ CHECK FØR BENDING AB'T
Y AXIS. SET M(AB'T Y AXIS)=0.
C SUB C =   126.1
L(C) IN INCHES= 82.2
F(B) IN PSI=   22
L(U) IN INCHES=  118
R=  1.8
VALUE ØF C(B) IN FØRMULA 4 IS ?1.8
F(B) FRØM FØRMULA 4 IN KSI= 19.2
K1*L1/R1= 28.4585
K2*L2/R2=  100
F.S.=   1.905043
F(A)=   12955.14
SMALL FA/CAP FA=   .1936999
FØRMULA 7A=   .6396655
FØRMULA 7B=   .8472019

READY
*
```

GEOMETRIC AND PLOTTING

This BASIC program divides circles into any number of equal parts, giving the angles in decimal degrees and degrees, minutes, and seconds, and calculating the horizontal and vertical distances from the center to the point on the circumference.

INSTRUCTIONS

To use this program enter the input data in the following format:

10 DATA N, r, ...

where

- N is the number of parts
- R is the radius of circle

As many cases as desired may be entered in the same way by continuing the data list.

After all the data is entered, type RUN.

Additional instructions can be found in the listing.

SAMPLE PROBLEM

Divide three sheet metal disks of 18.351 inches radius into equal segments of 3, 7, and 15 parts.

SAMPLE SOLUTION

```
*10 DATA 3,18.351,7,18.351,15,18.351
*RUN
```

D I V I S I Ø N   Ø F   C I R C L E

CASE NUMBER: 1 NUMBER ØF PARTS: 3 RADIUS: 18.351

INDEX	..........ANGLE.......... DECIMAL	DEG	MIN	SEC	.....CØ-ØRDINATES..... HØRIZ	VERT
1	120	120	0	0	-9.175	15.892
2	240	240	0	0	-9.176	-15.892
3	360	360	0	0	18.351	0

CASE NUMBER: 2 NUMBER ØF PARTS: 7 RADIUS: 18.351

INDEX	..........ANGLE.......... * DECIMAL	DEG	MIN	SEC	.....CØ-ØRDINATES..... HØRIZ	VERT
1	51.43	51	25	42.9	11.442	14.347
2	102.86	102	51	25.7	-4.083	17.891
3	154.29	154	17	8.6	-16.534	7.962
4	205.71	205	42	51.4	-16.534	-7.962
5	257.14	257	8	34.3	-4.083	-17.891
6	308.57	308	34	17.1	11.442	-14.347
7	360	359	59	60	18.351	0

CASE NUMBER: 3 NUMBER ØF PARTS: 15 RADIUS: 18.351

INDEX	..........ANGLE.......... DECIMAL	DEG	MIN	SEC	.....CØ-ØRDINATES..... HØRIZ	VERT
1	24	24	0	0	16.764	7.464
2	48	48	0	0	12.279	13.637
3	72	72	0	0	5.671	17.453
4	96	96	0	0	-1.918	18.25
5	120	120	0	0	-9.175	15.892
6	144	144	0	0	-14.846	10.786
7	168	168	0	0	-17.95	3.815
8	192	192	0	0	-17.95	-3.815
9	216	216	0	0	-14.846	-10.786
10	240	240	0	0	-9.176	-15.892
11	264	264	0	0	-1.918	-18.25
12	288	288	0	0	5.671	-17.453
13	312	312	0	0	12.279	-13.637
14	336	336	0	0	16.764	-7.464
15	360	360	0	0	18.351	0

```
READY
*
```

This Fortran subroutine plots a maximum of nine curves simultaneously.

INSTRUCTIONS

The calling sequence is:

CALL PLOT (X, Y, YMAX, YMIN, N, IND, NTOT)

where

- X is the value of the independent variable to be plotted.
- Y is the array which contains the dependent values of the curves to be plotted.
- YMAX is the maximum calculated value of all of the curves.
- YMIN is the minimum calculated value of all of the curves
- N is the number of curves to be plotted maximum of 9.
- IND is the indicator which must be 1 the first time the routine is called and 0 for each subsequent time.
- NTOT is the total number of X points to be plotted.

1. Values of Y(J) greater than YMAX or less than YMIN are plotted as YMAX or YMIN respectively.
2. The subroutine must be initialized (IND = 1) and called for every point (X) to be plotted (IND = 0).
3. Labeling of the plot:
   a. Labeling of the X-axis is always along the left side of the plot.
   b. Labeling of the Y-axis is determined by the input values of YMAX and YMIN and consists of 5 values of Y incrementing from YMIN to YMAX.
   c. The location of the labeling for the Y-axis is across the top of the plot.
   d. If the series of curves does not plot Y(J) for an exact X = 0, the Y-axis definition is not at the 0 point, but is placed at the bottom of the plot. In this case the Y-axis labeling is repeated along the bottom of the graph.
4. The maximum value of N is 9. No more than nine curves may be plotted at one time.
5. The scale factor used by the routine for plotting the curves is computed as (YMAX - YMIN)/70.

SAMPLE PROBLEM

Problem — Plot the following four curves:

$$Y1 = 3e^{-X/4}$$

$$Y2 = SIN(\pi X/2)$$

$$Y3 = 3e^{-X/4} SIN(\pi X/2)$$

$$Y4 = 3e^{-X/4}$$

where

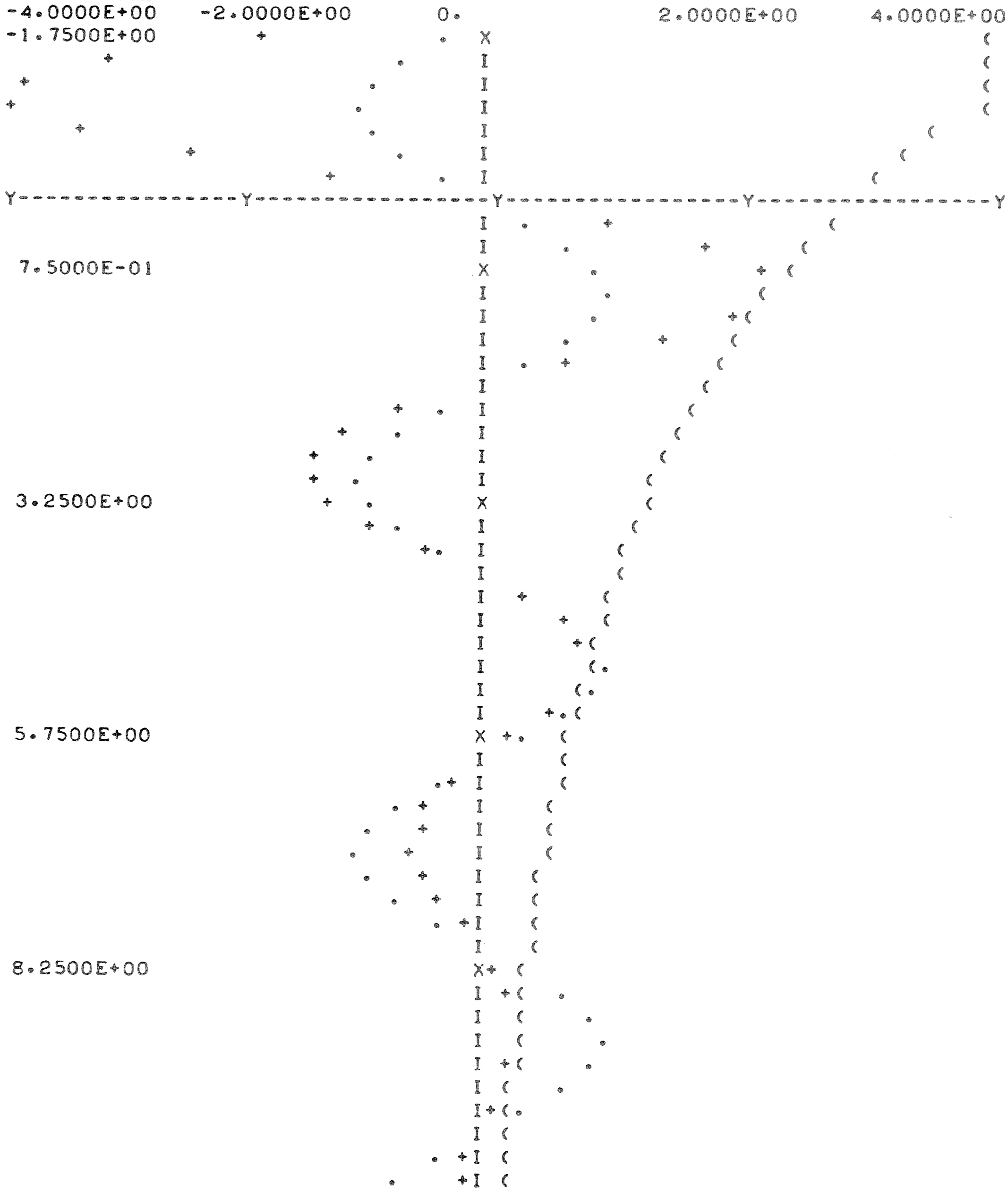
YMAX = 4.0, YMIN = -4.0,

and the number of curves, N = 4

SAMPLE SOLUTION

```
*LIST

100 DIMENSIØN Y(4)
110 N=4
120 YMAX=4.
130 YMIN=-4.
140 DX=.25
150 CALL PLØT(X,Y,YMAX,YMIN,N,1,50)
160 DØ 100 J=1,50
170 X=J*DX-2.
180 Y(1)=3.*EXP(-.25*X)
190 Y(2)=SIN(1.5705*X)
200 Y(3)=Y(1)*Y(2)
210 Y(4)=Y(1)
220 CALL PLØT(X,Y,YMAX,YMIN,N,0,50)
230 100 CØNTKNUE
240 STØP
250 END
```

```
READY

*RUN *;PLØT
```




```
PROGRAM STØP AT 240
*
```

This BASIC program plots one to six functions of X simultaneously. All functions have the same upper and lower limits for the plot. The functions are called A, B, C, D, E, F and are plotted in that order of priority. Where plots would overlap, the lower priority functions are suppressed. Values exceeding the selected bounds are disregarded.

INSTRUCTIONS

To use this program, enter information in the following format:

```
100  DATA NUM, XMIN, XMAX, DELX, HMIN, HMAX
200  Let A = Any 'BASIC' function of X
210  Let B = Any 'BASIC' function of X and/or A
220  Let C = Ditto for X and/or A and/or B
230  [Similarly for D]
240  [Similarly for E]
250  [Similarly for F]
```

where

NUM is the number of functions given [1-6]. XMIN and XMAX are the lower and upper limits for X, DELX is the Increment for X. HMIN and HMAX are the lower and upper limits for the values of the function.

NOTE: The horizontal increment is always HMAX-MIN /40.

Then type RUN.

For additional instructions, list the prograr

SAMPLE PROBLEM

Plot the same six curves twice where the second plot will be an enlarged version of a section of the first graph.

The functions plotted are illustrated in statements 200 - 250 in the sample solutions. The option of plotting all six functions is used. Since functions D and E were out of the specified range, they were not shown on the second plot.

SAMPLE SOLUTION

```
*100 DATA 6,0,1,.05,0,1
*200 LET A=1-EXP(-X)
*210 LET B=X↑2
*220 LET C=X↑3
*230 LET D=X*C
*240 LET E=EXP(-X/2)
*250 LET F=EXP(-X)
*RUN
```

```
               MULTIPLE PLØT ØF THE FUNCTIØNS A, B, C, D, E, F
                        HØRIZ INCREMENT =    0.025
                0                                       1
                :....:....:....:....:....:....:....:....:
X-VALUE

          0     A                                       E
       0.05     B A                                   FE
        0.1     B   A                               F E
       0.15     CB    A                           F  E
        0.2     C B    A                         F  E
       0.25     DCB      A                     F   E
        0.3     DC  B     A                   F   E
       0.35      DC  B      A               F     E
        0.4      D C  B      A             F     E
       0.45       D C   B     A           F     E
        0.5       D  C    B     A       F      E
       0.55         D  C    B    A     F      E
        0.6          D   C    B   A   F       E
       0.65            D   C     B A F       E
        0.7               D   C     A       E
       0.75                  D   C F AB    E
        0.8                     D F C A   BE
       0.85                      F   D A CE  B
        0.9                     F       A D  C  B
       0.95                    F         A       DC B
  0.9999999                    F        EA              B

                :....:....:....:....:....:....:....:....:
                0                                       1
```

TYPE 'S' TØ STØP, ØR GIVE NEW VALUES ØF
NUM,XMIN,XMAX,DELX,HMIN,HMAX ?6,0,.5,.025,0,.1

```
               MULTIPLE PLØT ØF THE FUNCTIØNS A, B, C, D, E, F
                        HØRIZ INCREMENT =   0.0025
                0                                     0.1
                :....:....:....:....:....:....:....:....:
X-VALUE

          0     A
      0.025     B         A
       0.05     CB                  A
      0.075     C B                          A
        0.1     C   B                                 A
      0.125     DC    B
       0.15     DC       B
      0.175     D C         B
        0.2      D C            B
      0.225      D   C              B
       0.25       D   C                  B
      0.275       D     C                     B
        0.3        D       C                        B
      0.325         D         C
       0.35           D          C
      0.375             D            C
        0.4               D               C
      0.425                  D                 C
       0.45                     D                   C
      0.475                         D
        0.5                              D

                :....:....:....:....:....:....:....:....:
                0                                     0.1
```

TYPE 'S' TØ STØP, ØR GIVE NEW VALUES ØF
NUM,XMIN,XMAX,DELX,HMIN,HMAX ?S

READY
*

This Fortran program plots equations in polar coordinates using either R or THETA as the independent variable and plots equations stated in parametric form.

INSTRUCTIONS

To use this program, enter the equations into lines 700 - 710 in standard polar form (R = F(THETA)) and then type RUN.

The program will then ask the user if he desires instructions. An answer of NO will result in the program requesting values for P and N

where

- P may have any positive value.
- N may not exceed a maximum value of 500.

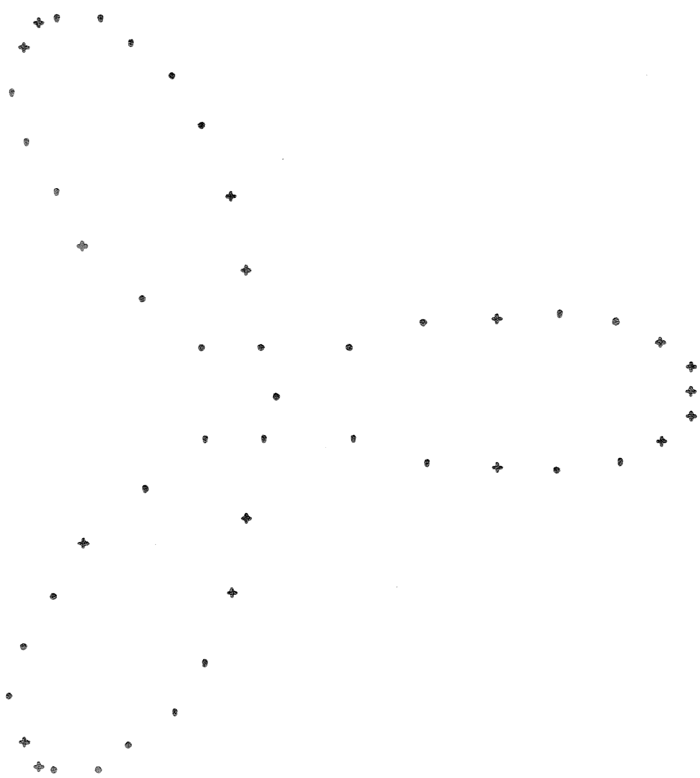
Additional instructions can be obtained by replying YES to the question, DO YOU WANT INSTRUCTIONS?

SAMPLE PROBLEM

Plot the curve for equation R = COS(3. * THETA) with values of P=2, and N=50.

SAMPLE SOLUTION

```
 700 R=CØS(3*THETA)
*RUN
DØ YØU WANT INSTRUCTIØNS?
ANSWER WITH YES ØR NØ. A 'NØ' ASSUMES THE EQUATIØN
IS ALREADY ENTERED
= NØ



PØLPLØ



P AND N
= 2,50
```



```
PRØGRAM STØP AT 845
*
```

This BASIC program solves spherical triangles having the apex at the North Pole and the other two corners defined by their respective latitudes and longitudes.

INSTRUCTIONS

Enter data for as many cases as desired successively. Data must be entered in data statements 10 - 99 in the following format:

10 DATA LTD, LTM, LGD, LGM, RLTD, RLTM, RLGD, RLGM, ALD, ALM

where

each pair of numbers psecifies a location in the form degrees, minutes as follows:

- LTD, LTM is local latitude
- LDG, LGM is local longitude
- RLTD, RLTM is remote latitude
- RLGD, RLGM is remote longitude
- ALD, ALM is observed altitude [if any]

For South latitudes and East longitudes, enter the degree values as negative numbers. If there is no observed altitude, set ALD and ALM equal to zero.

For additional instructions, list the program.

SAMPLE PROBLEM

Solve the spherical triangle problem using the following data:

Local Latitude	-	40 deg.	50 min.	North Lat.
Local Longitude	-	73 deg.	30 min.	West Long.
Remote Latitude	-	23 deg.	26 min.	North Lat.
Remote Longitude	-	133 deg.	30 min.	West Long.
Observed Altitude	-	37 deg.	20 min.	

SAMPLE SOLUTION

```
*10 DATA 40, 50, 73, 30, 23, 26, 133, 30, 37, 20
*RUN
```

```
   S P H E R I C A L   T R I A N G L E   S Ø L U T I Ø N -

CASE NUMBER     1

LØCAL PØSITIØN:

    40 DEG        50 MIN   NØRTH LATITUDE
    73 DEG        30 MIN   WEST LØNGITUDE

REMØTE PØSITIØN:

    23 DEG        26 MIN   NØRTH LATITUDE
   133 DEG        30 MIN   WEST LØNGITUDE

LØCAL HØUR ANGLE (AT NØRTH PØLE):

    60 DEG
    59 DEG        60 MIN
     3 HRS        59 MIN        60 SEC

ZENITH (GREAT CIRCLE) DISTANCES:

  52.6 DEG
    52 DEG        37 MIN
  3157 NAUTICAL MILES
   3635.5  STATUTE MILES

TRUE BEARINGS (GREAT CIRCLE CØURSES):

  REMØTE PØSITIØN FRØM LØCAL PØSITIØN:
    270.1 DEG
   270 DEG         4 MIN

  LØCAL PØSITIØN FRØM REMØTE PØSITIØN:
  55.6 DEG
    55 DEG        33 MIN

ALTITUDE (REMØTE CELESTIAL PØSITIØN
   ABØVE LØCAL PØSITIØN HØRIZØN):

  37.4 DEG
    37 DEG        23 MIN

ØBSERVED ALTITUDE:

    37 DEG        20 MIN
    37.33 DEG

LINE ØF PØSITIØN:

     3 MILES AWAY ØN LINE BEARING   90.1 DEGREES TRUE

READY
*
```

This is a BASIC program to find the unknown features of any triangle; given one side and any two other parts.

INSTRUCTIONS

To use this program supply values as required by the selected option. Specify angles as 'DEG, MIN, DEC'. For additional instructions, list the program.

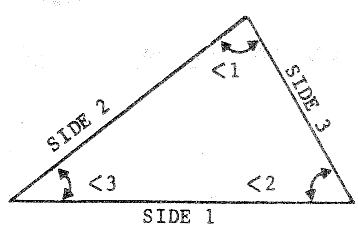
SAMPLE PROBLEM

Calculate the value of the three enclosed angles of a triangle whose sides are 17, 7.8, and 13.9.

Calculate the value of three enclosed angles.

This program has a specific way for naming the sides and angles of the triangle. Care must be exercised to avoid erroneous answers.

Angle N is the angle opposite the side you choose to call side N, as shown in the following diagram.



All angles are entered as two distinct values. The first is the angle degrees and the second is minutes (which can have a fractional part). If one or more of your angles is in the whole degrees only, you must supply 0 for the minutes portion.

Use the following table to select the proper option:

If you have	Use option
Side 1, Side 2, Side 3	1
Side 1, Angle 3, Side 2	2
Side 1, Side 3, Angle 1	3
Angle 1, Side 3, Angle 2	4
Angle 3, Angle 1, Side 3	5

The values calculated by option 1 are used to demonstrate the other options.

SAMPLE SOLUTION

```
*RUN

TRIANG

THIS PRØGRAM WILL FIND THE UNKNØWN FEATURES ØF ANY
TRIANGLE, GIVEN ØNE SIDE AND ANY TWØ ØTHER PARTS.

WHAT WILL BE GIVEN (1=SSS, 2=SAS, 3=SSA, 4=ASA, 5=AAS) ?1

WHAT ARE SIDE1, SIDE2, SIDE3 ?17, 7.8, 13.9

                            1              2              3
                         -------        -------        -------

SIDE                         17            7.8           13.9

ANGLE (RAD)            1.732681      0.4699307      0.9389807

ANGLE (DEG)                  99             26             53
      (MIN)                  16             55             47
      (SEC)               31.16          30.17          58.67

ALT TØ SIDE            6.294261       13.71826       7.698017

RADIUS ØF CIRCUMSCR CIRCLE =     8.612608
RADIUS ØF INSCRIBED CIRCLE =     2.764921
          AREA ØF TRIANGLE =     53.50121

ANØTHER CASE (1=YES) ?1

WHAT WILL BE GIVEN (1=SSS, 2=SAS, 3=SSA, 4=ASA, 5=AAS) ?2

NØTE: SPECIFY ANGLES AS 'DEGREES, MINUTES, SECØNDS' ØR
      'DEGREES, MINUTES.DECIMAL, 0' (I.E., SECØNDS=0)

WHAT ARE SIDE1, ANGLE3, SIDE2 ?17, 53, 47, 58.67, 7.8

                            1              2              3
                         -------        -------        -------

SIDE                         17            7.8           13.9

ANGLE (RAD)            1.732681      0.4699307      0.9389807

ANGLE (DEG)                  99             26             53
      (MIN)                  16             55             47

      (SEC)               31.16          30.17          58.67

ALT TØ SIDE            6.294261       13.71826       7.698017

RADIUS ØF CIRCUMSCR CIRCLE =     8.612608
RADIUS ØF INSCRIBED CIRCLE =     2.764921
          AREA ØF TRIANGLE =     53.50121

ANØTHER CASE (1=YES) ?0

READY
*
```

This BASIC program plots simultaneously two functions of a single variable, X.

INSTRUCTIONS

To use this program type:

```
10 LET Y =  [the first function of X]
20 LET Z =  [the second function of X and/or Y]
```

Then type RUN.

The functions Y and Z may be any legitimate BASIC expressions. Intermediate variables may be defined using intermediate lines, if the functions are too complicated to fit on one line.

For additional instructions, list the program.

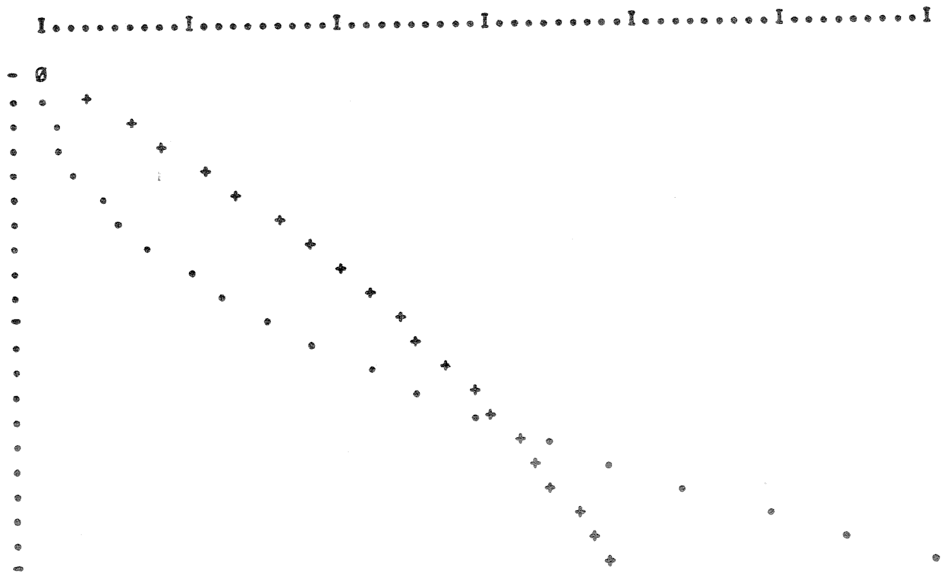
SAMPLE PROBLEM

Plot the first two curves that were used to illustrate the program PLOTTO.

NOTE: It is advisable to use TWOPLO rather than PLOTTO when only two functions are required because of the shorter running time.

SAMPLE SOLUTION

```
*10 LET Y=1-EXP(-X)
*20 LET Z=X↑2
*RUN



WHAT ARE  FMIN,FMAX,XMIN,XMAX,DELX ?0,1,0,1,.05

(NØTE: Y IS PLØTTED '+', Z IS '.', AND '0' IS CØMMØN PØINT)

FØR X:        TØP =     0  BØTTØM =     1  INCREMENT =  0.05
FØR FCTS:    LEFT =     0   RIGHT =     1  INCREMENT =   0.0166667
```



```
TYPE 'S' TØ STØP NØW, ØR ELSE SPECIFY NEW VALUES
FØR  FMIN,FMAX,XMIN,XMAX,DELX ?S

READY
*
```

This BASIC program plots single valued functions of X, with X on the vertical axis.

INSTRUCTIONS

To use this program type:

```
10  LET Y =  [ the function to be plotted ]
```

then type RUN.

During running, the program will ask for YMIN and YMAX [the limits on the horizontal Y-axis] , for XMIN and XMAX [the limits on the vertical X-axis] , and for DELX, the increment to be used along the X-axis.

Lines 11-99 of the program may be used as desired to express complicated functions. For additional instructions, list the program.

SAMPLE PROBLEM

For illustration purposes, we will use this program to plot a normal curve. In its simplest form, the normal can be represented as:

$$Y = e^{-x^2}$$

From looking at the equation we can see that the Y limits are $e^{-\infty} = 0$ and $e^{0} = 1$. The X limits are $\pm\infty$, but Y remains very close to zero if X is less than -2 or greater than +2. We will use these as our limits and split the X axis into 40 segments ($\Delta X = .1$).

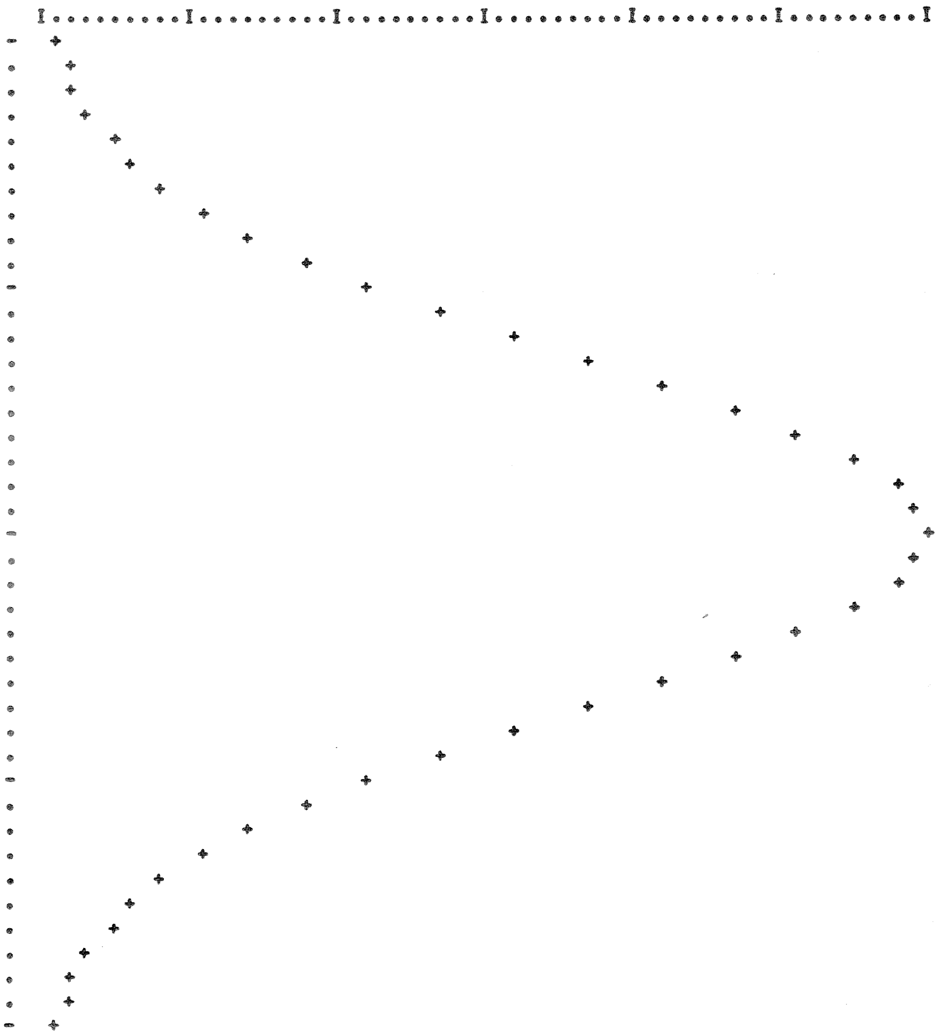
NOTES

1. You control both the X and Y axis limits by answering the YMIN, YMAX, XMIN, XMAX, DELX question. If incorrect values were used for YMIN and/or YMAX, and the plot went off-axis at certain X points, the program would have typed "off-scale" and the actual Y value at those points. If a clean plot is required you must run the program again and modify YMIN and/or YMAX. By knowing YMIN and YMAX from analyzing the equation, it is possible to get a plot which fills all the available space.

2. You can use this program to get semi-log or log-log plots by using appropriate transforms at lines 10 through 99. If this is done, though, the plot axis will still be linear and you must properly interpret the result. (When this feature is used, the transforms must have variable names — separate and distinct from those variable names used by the program. The variable names used by the program are: Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, X, and Y.)

SAMPLE SOLUTION

```
*10 LET Y=EXP(-(X↑2))
*RUN



WHAT ARE  YMIN,YMAX,XMIN,XMAX,DELX ?0,1,-2,2,.1

FØR X:    TØP =     -2  BØTTØM =      2  INCREMENT =   0.1
FØR Y:   LEFT =      0   RIGHT =      1  INCREMENT =   0.0166667
```



```
TYPE '0' TØ STØP ØR '1' TØ CHANGE LIMITS.  WHICH ?0

READY
*
```

EDUCATIONAL AND TUTORIAL

This Fortran object program is the driver program used, in conjunction with the preprocessor PREPRS, to run a translated lesson in the EXPER language (see Time-Sharing EXPER Language, Order No. BS05). It will administer the lesson and write the students' responses on a specified file. EXPER is a CAI (Computer Assisted Instruction) language.

INSTRUCTIONS

To use the DRIVES subroutine, write a 3-line program as follows:

```
10$USE LIBRARY/DRIVES, R
20 CALL DRIVER(nHXXXXXX, mHYYYYYY)
30 END
```

where

- n is the number of characters in the name of the file containing the translated lesson (must be a number from 1 to 6).
- XXXXXX is the name of the file containing the translated lesson.
- m is the number of characters in the name of the file which will receive the student responses (must be a number from 1 to 6).
- YYYYYY is the name of the student response file.
- R and H must appear as illustrated here.

SAMPLE PROBLEM

A sample lesson was previously translated and stored in the file LESSON (see PREPRS sample problem).

SAMPLE SOLUTION

A student response file RESPF, was created. The 3-line driver program was written and executed. Then the response file was listed.

```
*NEW
READY
*SAVE RESPF
DATA SAVED--RESPF
*NEW
READY
*10$USE LIBRARY/DRIVES,R
*20 CALL DRIVER(6HLESSØN,5HRESPF)
*30 END
*RUN
WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
OF EXPER?
        A.   A LANGUAGE DESIGNED FØR SCIENTIFIC CALCULATIONS.
        B.   A LANGUAGE DESIGNED FØR BUSINESS APPLICATIONS.
        C.   A LANGUAGE DESIGNED FØR USE BY INSTRUCTORS
             WITH A NEED TO WRITE LESSØNS TØ BE ADMINISTERED
             BY A COMPUTER.
        D.   ALL ØF THE ABOVE.
= B
WRØNG. EXPER IS A C.A.I.(COMPUTER ASSISTED INSTRUCTION)
AUTHØR LANGUAGE. WITH THIS INFØRMATION TRY THE QUESTION
AGAIN.

WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
ØF EXPER?
        A.   A LANGUAGE DESIGNED FØR SCIENTIFIC CALCULATIONS.
        B.   A LANGUAGE DESIGNED FØR BUSINESS APPLICATIONS.
        C.   A LANGUAGE DESIGNED FØR USE BY INSTRUCTØRS
             WITH A NEED TØ WRITE LESSONS TØ BE ADMINISTERED
             BY A CØMPUTER.
        D.   ALL OF THE ABOVE.
= C
THATS RIGHT. EXPER IS DESIGNED FØR EASY LESSØN WRITING
WITH CAPABILITIES FØR MATCHING RESPØNSES AND BRANCHING
ØN THE VALUE ØF SCØRE COUNTERS.

PRØGRAM STØP AT 0
*LIST RESPF

A       B
A       C

READY

*
```

These five programs (EXPER1 through EXPER5) are source language files written in the EXPER language. They teach the writing of EXPER programs. EXPER is a CAI (Computer Assisted Instruction) language.

INSTRUCTIONS

Create a file for the translated lesson and a file for the response file. Access the lesson desired from the LIBRARY. Run the preprocessor program PREPRS and then the driver program DRIVES.

SAMPLE SOLUTION

The file L1 is created for the translated lesson, and the file RESPON is created for the response file. The first lesson in the series, EXPER1, is accessed. The preprocessor and driver programs are run. The program was terminated after the first student response.

```
*NEW
READY
*SAVE L1;RESPON
DATA SAVED--L1
DATA SAVED--RESPON
*10$USE LIBRARY/PREPRS,R
*20 CALL PREPRO(6HEXPER1,2HL1)
*30 END
*RUN
 125   113-NAME-
 188   145-1
 230   166-1A
 238   170-1B
 246   174-22
 276   189-2
 296   199-2A
 308   205-2B
 314   208-2C
 368   235-3
 400   251-3A
 410   256-3B
 420   261-3C
 460   281-4
 482   292-4A
 492   297-4B
 516   309-5
 526   314-OTHER
 554   328-5A
 564   333-5B
 576   339-5C
 586   344-5D
 596   349-5E
 610   356-5F
 668   385-OOPS
 674   388-END
END PASS 1

PROGRAM STOP AT 0
*10$USE LIBRARY/DRIVES,R
*20 CALL DRIVER(2HL1,6HRESPON)
*RUN
```

HELLØ AND WELCØME TØ SERIES 600/6000 TIME SHARING. THIS SEQUENCE ØF PROGRAMS IS DESIGNED TØ HELP YØU LEARN TØ WRITE INSTRUCTIØN-AL MATERIALS IN THE EXPER CØMPUTER LANGUAGE.

FIRST, LETS ESTABLISH THE METHØD BY WHICH WE WILL CØMMUNICATE. WHEN I WISH TØ HAVE YØU RESPØND TØ A QUESTIØN ØR STATEMENT, I WILL TYPE AN EQUAL SIGN (=) AND THEN STØP, WAITING FØR YØUR RESPØNSE.

LETS TRY IT. WHEN I TYPE AN EQUAL SIGN AND STØP, YØU TYPE YØUR NAME AND THEN PRESS THE RETURN KEY TØ LET ME KNØW THAT YØU ARE FINISHED. (THE RETURN KEY IS LØCATED AT THE FAR RIGHT HAND SIDE ØF THE KEYBØARD.)
= <u>HØNEYWELL</u>

GØØD. THAT'S HØW WE WILL CØMMUNICATE.

This Fortran object program is the preprocessor used, in conjunction with DRIVES, to process and run the EXPER language (see Time-Sharing EXPER Language, Order No. BS05). This subroutine will translate an EXPER source program and save the translated lesson on a specified file. EXPER is a CAI (Computer Assisted Instruction) language.

INSTRUCTIONS

To use this subroutine, write a 3-line program as follows:

```
10$USE LIBRARY/PREPRS, R
20 CALL PREPRO(nHXXXXXX, mHYYYYYY)
30 END
```

where

- n is the number of characters in the name of the file in which the EXPER source program is saved (must be a number from 1 to 6).
- XXXXXX is the name of the EXPER source file.
- m is the number of characters in the name of the file in which the translated lesson will be saved (must be a number from 1 to 6).
- YYYYYY is the name of the file which will receive the translated lesson.
- R and H must appear as illustrated.

SAMPLE PROBLEM

A sample lesson was written and saved in the file SOURCE. Translate this lesson prior to execution of the driver (see DRIVES sample problem).

SAMPLE SOLUTION

The translated lesson is saved in the file LESSON.

```
*LIST SOURCE

100-A
110 WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
120 ØF EXPER?
130        A.   A LANGUAGE DESIGNED FØR SCIENTIFIC CALCULATIØNS.
140        B.   A LANGUAGE DESIGNED FØR BUSINESS APPLICATIØNS.
150        C.   A LANGUAGE DESIGNED FØR USE BY INSTRUCTØRS
160             WITH A NEED TØ WRITE LESSØNS TØ BE ADMINISTERED
170             BY A CØMPUTER.
180        D.   ALL ØF THE ABØVE.
190:INP
200:M21:C:
210:JK2::ØN:
220 WRØNG. EXPER IS A C.A.I.(CØMPUTER ASSISTED INSTRUCTIØN)
230 AUTHØR LANGUAGE. WITH THIS INFORMATIØN TRY THE QUESTIØN
240 AGAIN.
250
260:JK2:A:
270-ØN
280 THATS RIGHT. EXPER IS DESIGNED FØR EASY LESSON WRITING
290 WITH CAPABILITIES FØR MATCHING RESPØNSES AND BRANCHING
300 ØN THE VALUE ØF SCØRE CØUNTERS.
310:*ND

READY

*NEW
READY
*10$USE LIBRARY/PREPRS,R
*20 CALL PREPRØ(6HSØURCE,6HLESSØN)
*30 END
*RUN
 100  100-A
 270  117-ØN
END PASS 1

PRØGRAM STØP AT 0
*LIST LESSØN

100- 0 A
101 17 WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
102  4 ØF EXPER?
103 21        A.   A LANGUAGE DESIGNED FØR SCIENTIFIC CALCULATIØNS.
104 20        B.   A LANGUAGE DESIGNED FØR BUSINESS APPLICATIØNS.
105 19        C.   A LANGUAGE DESIGNED FØR USE BY INSTRUCTØRS
106 20             WITH A NEED TØ WRITE LESSØNS TØ BE ADMINISTERED
107  9             BY A CØMPUTER.
108 10        D.   ALL ØF THE ABØVE.
109: 7 INPA
110: 3 M21:C:
111: 2 JK2    0      117
112 20 WRØNG. EXPER IS A C.A.I.(CØMPUTER ASSISTED INSTRUCTIØN)
113 20 AUTHØR LANGUAGE. WITH THIS INFORMATIØN TRY THE QUESTIØN
114  3 AGAIN.
115  2
116: 1 JK2    100
117- 0 ØN
118 19 THATS RIGHT. EXPER IS DESIGNED FØR EASY LESSØN WRITING
119 19 WITH CAPABILITIES FØR MATCHING RESPØNSES AND BRANCHING
120 12 ØN THE VALUE ØF SCØRE CØUNTERS.
121: 3 *ND
 END ØF FILE  0

READY

*
```

DEMONSTRATION

This BASIC program can construct a maze of any dimensions to a maximum of 23 by 25.

Each maze is unique and is guaranteed to have one and only one solution.

INSTRUCTIONS

To use this program, type RUN and follow instructions.

```
RUN

THIS PROGRAM WILL PRINT OUT A DIFFERENT MAZE EVERY TIME IT
IS RUN AND GUARANTEES ONLY ONE PATH THROUGH.  YOU CAN CHOOSE
THE DIMENSIONS OF THE MAZE. I.E. THE NUMBER OF SQUARES WIDE
AND THE NUMBER OF SQUARES LONG.  A 23 BY 25 MAZE IS THE
MAXIMUM, BUT ANY DIMENSIONS UP TO THESE LIMITS ARE OK.

WHAT ARE YOUR WIDTH AND LENGTH ?16,10


:--:--:--:--:--:--:--:--:--:--:--:--:  :--:--:--:
I              I        I           I     I     I
:--:--:--:  :  :  :  :  :  :  :  :  :--:  :  :  :
I           I     I  I  I  I  I  I        I  I  I
:  :--:--:--:--:--:  :  :  :  :  :--:--:--:  :  :
I           I        I  I  I  I  I           I  I
:  :--:--:  :  :--:--:  :  :  :  :  :--:--:--:  :
I     I     I        I  I  I  I        I     I  I
:--:--:  :--:--:--:  :  :  :--:  :--:--:  :  :--:
I        I     I     I     I     I        I     I
:  :  :--:  :  :  :--:--:--:  :--:  :--:--:--:  :
I  I     I  I     I     I        I  I           I
:--:--:  :  :--:--:--:  :  :--:  :  :  :--:--:--:
I     I  I  I           I     I     I  I        I
:  :  :  :--:  :  :--:--:--:  :--:--:--:  :--:  :
I  I  I        I  I        I  I           I  I  I
:--:  :--:  :--:  :  :--:  :  :  :--:--:--:  :  :
I     I     I  I     I     I  I        I     I  I
:  :--:  :  :  :--:  :  :--:  :--:--:  :  :--:  :
I        I  I        I     I           I        I
:--:--:--:--:--:--:--:--:--:  :--:--:--:--:--:--:

READY
*
```

This BASIC program is a simulated card game of Las Vegas-type blackjack.

INSTRUCTIONS

For instructions run the program.

SAMPLE SOLUTION

This is an actual demonstration game conducted briefly to show some of the points of the game.

```
* RUN
BLKJAK


THIS DEMØNSTRATIØN SHØWS THE VERSATILITY ØF GE TIME-
SHARING BY SIMULATING A GAME ØF BLACKJACK.  DØ YØU NEED
INSTRUCTIØNS (1=YES,0=NØ) ?1


HERE ARE THE LAS VEGAS RULES FØR PLAYING BLACKJACK:

>WAGER: THE HØUSE LIMIT IS $500, SØ TYPE IN A NUMBER
  FRØM 0 TØ 500. TØ TERMINATE GAME, ENTER ZERØ. -

>THE DEAL: I DEAL MYSELF 2 CARDS AND SHØW YØU ØNE. THEN I
  DEAL YØU TWØ CARDS, AND ASK IF YØU WANT A HIT (ANØTHER
  CARD). YØU HAVE SEVERAL ØPTIØNS DEPENDING ØN THE CARDS
  YØU HØLD AND MY UP CARD:
    * STAND - BY TYPING A ZERØ
    * TAKE A HIT - BY TYPING A ØNE
    * GØ DØWN FØR DØUBLES - BY TYPING A TWØ
    * SPLIT A PAIR - BY TYPING A THREE

>INSURANCE: IF MY UP CARD IS AN ACE, I WILL ASK IF YØU
  WANT INSURANCE. IF YØU DØ TYPE A ØNE, BETTING ØNE-HALF
  ØF YØUR WAGER THAT I DØ HAVE BLACKJACK. IF I DØ, I PAY
 2-TØ-1 ØN YØUR INSURANCE BET. YØU LØSE YØUR ØRIGINAL WAGER
  SINCE I HAVE BLACKJACK, SØ WE ARE EVEN FØR THE HAND.
  IF I DØN'T HAVE BLACKJACK, YØU LØSE YØUR INSURANCE BET
  AND THE GAME CØNTINUES.

  IF YØU REFUSE INSURANCE (BY TYPING A ZERØ) THE GAME
  CØNTINUES AS NØRMAL.

>THE PLAY: WHEN YØU FINALLY STAND (BY TYPING A ZERØ)
  I WILL DRAW CARDS UNTIL:
    *I HAVE AT LEAST A HARD 17 (HARD MEANS THE TØTAL
     DØES NØT INCLUDE AN ACE BEING CØUNTED AS 11)
    *I HAVE A SØFT 18 (SØFT MEANS THE TØTAL INCLUDES AN
     ACE CØUNTED AS 11)
    *I REACH A TØTAL ØF 21
    *I EXCEED 21 AND BUST

>ITEMS:
    *I PAY 1.5-TØ-1 ØN BLACKJACK
    *I DØN'T RECØGNIZE 5-CARDS-AND-UNDER
    *YØU MAY DØUBLE DØWN ØN A SPLIT HAND
    *YØU DØN'T LØSE ØN A TIE HAND...WE PUSH

              <<<GØØD LUCK>>>
```

```
THE 600 IS THE DEALER AND GETS A BREAK AT 1945 HØURS.  WHAT
TIME IS IT NØW ?300

WAGER ?50

I SHØW                              ACE ØF DIAMØNDS
FIRST CARD IS    JACK ØF HEARTS
NEXT  CARD IS       2 ØF DIAMØNDS
INSURANCE ANYØNE  (TYPE 1 ØR 0) ?1
YØU WIN $    50  ØN YØUR INSURANCE BET**I HAVE BLACKJACK** -
MY HØLE CARD IS                     JACK ØF CLUBS
YØU'RE EVEN

WAGER ?50

I SHØW                                 8 ØF HEARTS
FIRST CARD IS    KING ØF HEARTS
NEXT  CARD IS       9 ØF DIAMØNDS
HIT ?0
YØUR TØTAL IS     19
MY HØLE CARD IS                        6 ØF CLUBS
I DRAW                                 6 ØF SPADES
MY TØTAL IS     20
YØU'RE BEHIND $    50

WAGER ?50

I SHØW                                 6 ØF HEARTS
FIRST CARD IS       3 ØF HEARTS
NEXT  CARD IS       6 ØF DIAMØNDS
HIT ?1
NEXT  CARD IS       3 ØF CLUBS
HIT ?1
NEXT  CARD IS       5 ØF DIAMØNDS
HIT ?0
YØUR TØTAL IS     17
MY HØLE CARD IS                     QUEEN ØF CLUBS
I DRAW                                 7 ØF HEARTS
I BUSTED***MY TØTAL IS     23
YØU'RE EVEN

WAGER ?50

I SHØW                                 5 ØF CLUBS
FIRST CARD IS      10 ØF HEARTS
NEXT  CARD IS    ACE ØF HEARTS
***BLACKJACK***
MY HØLE CARD WAS                       8 ØF SPADES
YØU'RE AHEAD $    75

WAGER ?0

READY
*
```

This BASIC program computes and prints annual projections of an area's population. Any requested interval of years may be requested to a maximum of 99. The generated data is not exact, since it is based on the compound interest formula and the assumption of a steady increase each year. However, it is useful in showing how an area may be expected to grow.

INSTRUCTIONS

To use the program, type RUN and follow instructions: For example:

```
RUN


THIS PROGRAM WILL PROJECT POPULATION GROWTH FOR ANY NUMBER OF
YEARS USING THE COMPOUND INTEREST FORMULA.

WHAT IS THE NAME OF THE AREA WE ARE STUDYING
?METRO PHOENIX

PRINT THE ANNUAL PERCENT OF GROWTH FOR YOUR POPULATION ?5.5

FOR HOW MANY YEARS DO YOU WISH TO HAVE DATA COMPUTED ?10
LIST THE FIRST 9 YEARS TO BE COMPUTED, SEPARATE YOUR NUMBERS
WITH COMMAS.
?1972,1973,1974,1975,1976,1977,1978,1979,1980

LIST THE LAST      1 YEARS AND     8 ZEROS
?1981,0,0,0,0,0,0,0,0

WHAT IS THE YEAR FOR YOUR BASIC DATA ?1972

WHAT IS THE POPULATION FOR THE BASE YEAR(NO COMMAS PLEASE)
?1200000
POPULATION PROJECTION IS AS FOLLOWS

METRO PHOENIX

        DATE        POPULATION

         1972         1200000
         1973         1265999
         1974         1335629
         1975         1409089
         1976         1486589
         1977         1568351
         1978         1654611
         1979         1745614
         1980         1841623
         1981         1942912

DO YOU SUPPORT PLANNED PARENTHOOD???
READY
*
```

This BASIC program finds the prime factorization of a number.

INSTRUCTIONS

To use the program, type RUN and follow instructions. For example:

```
READY
*RUN


THIS PROGRAM FINDS THE PRIME FACTORIZATION OF A NUMBER.
IF YOU ASK IT TO FACTOR 0, IT WILL STOP.

WHAT NUMBER IS TO BE FACTORED ?1872

THE PRIME FACTORS OF 1872 ARE:

          PRIME        MULTIPLICITY
          -----        ------------

            2               4
            3               2
           13               1

WHAT NUMBER IS TO BE FACTORED ?134217728
SORRY!  THIS PROGRAM IS ONLY DESIGNED TO FACTOR NUMBERS
OF 8 DIGITS OR LESS!  YOU MAY TRY AGAIN--

WHAT NUMBER IS TO BE FACTORED ?1342177

THE NUMBER 1342177 IS PRIME.

WHAT NUMBER IS TO BE FACTORED ?0


READY
*
```

This BASIC program prints "THE TWELVE DAYS OF CHRISTMAN" adorned with appropriate holiday symbols. It is suitable for an unusual Christmas card or as guidance for a sing-along.

INSTRUCTIONS

Type RUN. For example, this is the last of 12 verses.

```
RUN

    E V E R Y B O D Y     S I N G

ON THE    12 TH DAY OF CHRISTMAS
MY TRUE LOVE SENT TO ME
TWELVE LORDS A-LEAPING,
ELEVEN LADIES DANCING,
TEN PIPERS PIPING,
NINE DRUMMERS DRUMMING,
EIGHT MAIDS A-MILKING,
SEVEN SWANS A-SWIMMING,
SIX GEESE A-LAYING,
FIVE GO-OLD RINGS,
FOUR CALLING BIRDS,
THREE FRENCH HENS,
TWO TURTLEDOVES AND
A PARTRIDGE IN A PEAR TREE.

          0
          *
         ***
        *****
          I
```

UTILITY AND MISCELLANEOUS

This Fortran Y program calculates and prints the day of the week of any date, the calendar dates of the beginning and ending of a continuing project, and the Julian dates from the calendar dates and total elapsed days. The subprograms which do the actual Julian calendar date conversions can easily be extracted by the user for inclusion in his own programs.

INSTRUCTIONS

For details of the program type LIST. To use the program, type RUN and enter information as requested.

The program is presently set for a 5-day work week. The user may reset for a 6- or 7-day week by typing DATA DWT/O, the number of days in work week, asterisk, 1 in line 60 where 1 = a work day and 0 = a nonwork day.

Note that the holiday entries (or no holidays) are terminated by three 0's. The program is terminated by four 0's.

The Julian data calculation is based on the assumption that day 1 = January 1, 1900 (1, 1, 1900).

SAMPLE PROBLEM

A project of 119 working days, holidays and a 5-day work week. (SUN, SAT- NON-WORK)

SAMPLE SOLUTION

```
RUN
ENTER HØLIDAY CALENDAR DATE AS: MØNTH,DAY,YEAR
=5,29,1972
=7,3,1972
=7,4,1972
=11,23,1972
=11,24,1972
=
=0,0,0
ENTER CURRENT DATE AND NUMBER ØF WØRKING DAYS
=6,26,1972,119

     6/26/1972     MON     JUNE 26,1972     26475

    12/13/1972     WED     DEC  13,1972     26645     171
=0,0,0,0
NØRMAL TERMINATIØN
```

Same project on a 6-day work week (SUN - NON-WORK)

```
*60 DATA DWT/0,6*1/
*RUN
ENTER HØLIDAY CALENDAR DATE AS: MØNTH,DAY,YEAR
=7,3,1972
=7,4,1972
=11,23,1972
=11,24,1972
=0,0,0
ENTER CURRENT DATE AND NUMBER ØF WØRKING DAYS
=6,26,1972,119

    6/26/1972     MØN    JUNE 26,1972      26475

   11/13/1972     MØN    NØV  13,1972      26615        141

=0,0,0,0

NØRMAL TERMINATIØN

*@
```

Same project on a 7-day work week.

```
*160 DATA DWT/7*1/
*RUN
ENTER HOLIDAY CALENDAR DATE AS:  MONTH,DAY,YEAR
=7,3,1972
=7,4,1972
=11,23,1972
=11,24,1972
=0,0,0
ENTER CURRENT DATE AND NUMBER OF WORKING DAYS
=6,26,1972,119

      6/26/1972       MON       JUNE 26,1972       26475

     10/24/1972       TUES       OCT 24,1972       26595       121
```

To determine the day of one's birthday, enter birthdate and 0 days:

```
*RUN
ENTER HOLIDAY CALENDAR DATA AS:  MONTH,DAY,YEAR
=0,0,0
ENTER CURRENT DATA AND NUMBER OF WORKING DAYS
=2,21,1910,0

      2/21/1910       MON       FEB 21,1910       3704
```

This EDIT built file is a catalog of the programs available in the time-sharing library, as documented in this manual. A listing of this file is contained in the front of this manual.

INSTRUCTIONS

The file can be listed by either typing

LIB CATALOG
LIST

or by using the BPRINT command in CARDIN. The file may be searched for a given program name or key word, using the EDIT system.

The entries in the catalog are divided into the same major catagories as this manual. Many of the catagories are futher subdivided for ease in finding programs to perform a given function.

All program names begin in column one. Columns 11-14 contain an indicator of the file format (language, subroutine, etc.). Columns 17-72 contain a brief program description.

This BASIC program determines the strongest possible conclusion in specified variables which follows as a logical consequence from a given set of statements of propositional logic and prints its truth table.

INSTRUCTIONS

To use this program type RUN. When the program prints PREMISE? , enter a statement or type DONE to indicate that all premises have been entered. After DONE is typed, the program will ask for the variables from which to draw a conclusion. Enter the names of these variables, as requested, or type BEST to have the program find the strongest possible conclusions in the fewest possible variables.

A maximum of 12 variables chosen from the letters A, B, ..., T may be chosen. Statements of propositional logic are built using the connectives — (NOT), (AND), V (OR), = > (IF...THEN), < = > (IF AND ONLY IF), and / (NOT BOTH). Formulas may employ the propositional variables A, B, ..., T.

SAMPLE PROBLEM

Determine the strongest possible conclusion where the premises are:

(A & B) = > C

(A/D) < = > C

SAMPLE SOLUTION

```
*RUN


 LIST FØR INSTRUCTIØNS-

PREMISE ?(A&B)=>C
PREMISE ?(-A/D)<=>C
PREMISE ?DØNE

VARIABLE ?BEST
NØ CØNCLUSIØNS CAN BE MADE BASED ØN ØNLY    1 VARIABLE

CØNCLUSIØNS IN    2 VARIABLES:


A     C
T     T     T
T     F     F

C     D
T     F     T
F     F     F
```

```
CØNCLUSIØNS IN    3 VARIABLES:

A    B    C
T    T    T    T
T    T    F    F
T    F    T    T
T    F    F    F

A    C    D
T    T    T    T
T    T    F    T
T    F    T    F
T    F    F    F
F    T    T    F
F    T    F    T
F    F    T    T
F    F    F    F

B    C    D
T    T    F    T
T    F    F    F
F    T    F    T
F    F    F    F

CØNCLUSIØNS IN    4 VARIABLES:

A    B    C    D
T    T    T    T    T
T    T    T    F    T
T    T    F    T    F
T    T    F    F    F
T    F    T    T    T
T    F    T    F    T
T    F    F    T    F
T    F    F    F    F
F    T    T    T    F
F    T    T    F    T
F    T    F    T    T
F    T    F    F    F
F    F    T    T    F
F    F    T    F    T
F    F    F    T    T
F    F    F    F    F

DØ YØU WISH TØ DRAW A CØNCLUSIØN IN ØTHER VARIABLES ?NØ

READY
*
```

This BASIC program converts measurements from one scale to another. For temperature, it converts degrees Celsius, Farenheit, and Kelvin. For length, it converts millimeters, inches, feet, kilometers, and miles. For area, it converts acres, hectares, square miles, and square kilometers. For pattern density, it converts measurements per square mile, square kilometer, acre, and hectare. For weight, it converts tons, pounds, ounces, and kilograms. For liquid measure, it converts gallons, quarts, ounces, and liters. For volume, it converts cubic inches, cubic feet, cubic yards, and cubic meters.

INSTRUCTIONS

Enter input data as requested by the program.

SAMPLE PROBLEM

Convert 125, 220, and 20 pounds to other units of weight.

SAMPLE SOLUTION

```
*RUN

THIS PRØGRAM WILL CØNVERT VARIØUS MEASUREMENTS FRØM ØNE SCALE
TØ ANØTHER.  DØ YØU WISH TØ DEAL WITH MEASURES ØF:

1   TEMPERATURE
2   LENGTH
3   AREA
4   DENSITY ØF PATERN
5   WEIGHT
6   LIQUID MEASURE
7   VØLUME

ANSWER WITH THE NUMBER ØF YØUR CHØICE.   WHICH ?5

ENTER THE NUMBER INDICATING FØRM ØF INPUT DATA:

1 FØR TØNS
2 FØR PØUNDS
3 FØR ØUNCES
4 FØR KILØGRAMS
?2

PLEASE INPUT THE FIGURES YØU WISH TØ HAVE CØNVERTED ØNE AT
A TIME.  SIGNAL THE END ØF VALUES BY ENTERING 999999.
?125
?180
?220
?20
?999999

          TØNS          PØUNDS          ØUNCES          KILØS

         .0625           125             2000          56.69963
          .09            180             2880          81.64746
          .11            220             3520          99.79134
          .01             20              320           9.07194

READY
*
```

This Fortran subroutine sorts a real array in ascending or descending order. In addition, the elements of a corresponding real array are rearranged so that they remain parallel with the sorted array. This program uses a bubble sort. The elements of each array are physically moved.

INSTRUCTIONS

The calling sequence for DBLSORT is:

```
CALL DBLSORT (KODE, SEEDS, FOLLO, JAX, LAX)
```

where

- KODE indicates the type of a sort:

  if KODE = 1, ascending sort
  if KODE = 2, descending sort

- SEEDS indicates the array to be sorted.
- FOLLO indicates the corresponding array.
- JAX indicates the subscript of the first element of the array to be included in the sort.
- LAX indicates the subscript of the last element of the array to be included in the sort.

NOTE: If the array to be sorted is 2-dimensional, JAX and LAX must be translated into linear subscripts. For example:

To sort elements A (1, 1) through A (6, 50)
then JAX = 1
LAX = 300

To sort elements A (3, 3) through A (4, 4)
then JAX = 9
LAX = 16

SAMPLE PROBLEM

Sort the array:

5, 9, 2, 4, 8

keeping

4, 7, 2, 1, 9

parallel to it.

SAMPLE SOLUTION

```
10       DIMENSIØN  SEEDS(50),FØLLØ(50)
20       PRINT:"HØW MANY ITEMS TØ BE SØRTED?"
30       READ:LAX
40       PRINT:"ENTER ARRAY TØ BE SØRTED"
50       READ:(SEEDS(I),I=1,LAX)
60       PRINT:"ENTER THE FØLLØWING ARRAY"
70       READ:(FØLLØ(I),I=1,LAX)
80       CALL DBLSØRT(1,SEEDS,FØLLØ,1,LAX)
90       PRINT:"THE SØRTED ARRAYS"
100       PRINT 1, (SEEDS(I),FØLLØ(I),I=1,LAX)
110    1 FØRMAT(/1P2E16.8)
120       STØP;END

READY

*RUN *;DBLSØRT
HØW MANY ITEMS TØ BE SØRTED?
= 5
ENTER ARRAY TØ BE SØRTED
= 5,9,2,4,8
ENTER THE FØLLØWING ARRAY
= 4,7,2,1,9
THE SØRTED ARRAYS

  2.00000000E+00  2.00000000E+00

  4.00000000E+00  1.00000000E+00

  5.00000000E+00  4.00000000E+00

  8.00000000E+00  9.00000000E+00

  9.00000000E+00  7.00000000E+00

PRØGRAM STØP AT 120
*
```

This Fortran program strips line numbers from a file.

INSTRUCTIONS

The first line of the program contains the required RUN list. The program will request the names of the input and output files. A question mark (?) in response to any question will cause an explanation to be printed. For example:

```
*LIST IN

10 LINE NUMBER 1
20&LINE NUMBER 2
30      LINE NUMBER 3
40#5 LINE NUMBER 4
500 LINE NUMBER 5
6000 LINE NUMBER 6
70000 LINE NUMBER 7


READY

*RUN DESEQ
FILES
=IN;ØT

*LIST ØT

 LINE NUMBER 1
&LINE NUMBER 2
      LINE NUMBER 3
5 LINE NUMBER 4
 LINE NUMBER 5
 LINE NUMBER 6
 LINE NUMBER 7


READY

*
```

This Fortran program converts a Fortran source file from NFORM to FORM format. The input file can be in NFORM, NLNO or in NFORM, LNO formats. It can be in any system standard media code (BCD, ASCII). The output file is in FORM, NLNO or in FORM, LNO. It is time-sharing ASCII line images (media = 5).

The following characteristics apply to an output file in the FORM, NLNO format.

1. Comment lines are flagged by a C or an asterisk (*) in the first character position following the line number (character position 1).
2. A continuation line is flagged by an ampersand (&) in character position 6.
3. Statement numbers appear in character positions 1 through 5.
4. Statement text appears in character positions 6 through 72.

A file in FORM, LNO is not acceptable to the Fortran compiler; however, a file in this format as produced by this program is converted to FORM, NLNO format by CARDIN with NORMAL, MOVE, or STRIP specified. The following characteristics apply to an output file in the FORM, LNO format.

1. A line number field begins in character position 1. The line number will begin with 100 and increment by 10 to 99999990.
2. Comment lines are flagged by a C or an asterisk (*) in character position 1.
3. A continuation line is flagged by an ampersand (&) in character position 6.
4. Statement text appears in character positions 7 through 72.
5. Statement numbers containing less than five digits appear in character positions 2 through 5. If a statement number contains five digits, the first position following the line number contains the pound symbol (#). The statement number appears in character positions 2 through 6. The statement text appears in character positions 8 through 73.

The following comments apply to either form of output.

1. Significant trailing blanks on an input line such as in an nH field are not recognized by the program.
2. If a single input line must be broken and continued on a second line for the output file, no attempt is made to break the line at a logical division point such as a comma. This should cause no problem unless the break occurs within blanks of a character literal field.
3. Control cards ($ in first position) are treated like comments and copied to the output file asis.

INSTRUCTIONS

The fist line of the program contains the required RUN list. The program will request the names of the input and output files and the LNO, NLNO option for each. A question mark (?) given in response to any question will cause an explanation to be printed. For example:

```
*LIST IN

10 PRINT,"THIS IS A SAMPLE FILE TØ ILLUSTRATE THE REFØRMATTING CAPABILIT
20&ES ØF REFØRM"
30* NØTE HOW THE ABØVE CONTINUATIØN LINES TURN ØUT
40 1 CØNTINUE
50 12 CØNTINUE
60 123 CØNTINUE
70 1234 CØNTINUE
80 12345 CØNTINUE
90 STØP
100           END


READY

*RUN REFØRM
FILES
=IN;ØT
FØR INPUT--LNØ,NLNO
=LNØ
FØR ØUTPUT--LNØ,NLNØ
=LNØ


*LIST ØT

100       PRINT,"THIS IS A SAMPLE FILE TØ ILLUSTRATE THE REFØRMATTING CAP
110      &ABILITI
120      &ES ØF REFØRM"
130* NØTE HØW THE ABØVE CØNTINUATIØN LINES TURN ØUT
140      1 CØNTINUE
150     12 CØNTINUE
160    123 CØNTINUE
170   1234 CØNTINUE
180#12345 CØNTINUE
190       STØP
200       END


READY

*
```

This Fortran subroutine reads a line from a data file, deletes the line number if present, counts the number of entries on the line, and saves the input line in a character array. The data in the line may then be assigned to other variables using DECODE.

INSTRUCTIONS

The calling sequence is:

CALL RLINE (IFC, IENT, LSW, IB, *, *, *)

where

- IFC is the integer file code of input file.
- IENT on output is the number of entries found on the line.
- LSW indicates whether line numbers are present on the file (LSW=2) or not present on the file (LSW=3). If LSW=1 on input, the subroutine resets LSW to 2 if the first character of the next line is numeric or to 3 if it is not.
- IB on output, is the line image with the line number stripped. A character IB*72 statement must occur in the calling routine.

The asterisks indicate statement numbers for alternate returns.

First alternate return — EOF on input file.

Second alternate return — line number too big (maximum of seven digits).

Third alternate return — a line number was expected but the first character is nonnumeric.

RESTRICTIONS

1. The input line cannot exceed 72 characters.
2. If blanks follow an exponent indicator E, D or G, the value returned in IENT will be incorrect.
3. If quoted character fields contain imbedded blanks or commas, the value returned in IENT will be incorrect.
4. The routine must be compiled in ASCII.

For example:

```
*LIST

100    CHARACTER IB*72,K(30)
110    LSW = 1
120 10 CALL RLINE(5,IENT,LSW,IB,$30,$30,$30)
130    IF(IENT .EQ. 0) STØP
140    DECØDE(IB,20) (K(I),I=1,IENT)
150 20 FØRMAT(V)
160    PRINT, (K(I),I=1,IENT)
170    GØ TØ 10
180 30 PRINT, "ERRØR RETURN"
190    GØ TØ 10
200    END



READY

*RUN *;RLINE=(CØRE=19)
=FIRST LINE ØF INPUT
FIRST   LINE    ØF      INPUT
=NEXT LINE
NEXT    LINE
=THIRD
THIRD
=LINE AFTER THIS WILL BE LAST -- CR ØNLY
LINE    AFTER   THIS    WILL    BE      LAST    --      CR      ØNLY

=

*
```

This Fortran subroutine sorts any array of real numbers in either ascending or descending order.

INSTRUCTIONS

The calling sequence for SGLSORT is:

CALL SGLSORT (KODE, SEEDS, JAX, LAX)

where

- KODE indicates the type of sort:

  if KODE = 1, ascending sort
  if KODE = 2, descending sort

- SEEDS is the array to be sorted.
- JAX indicates the subscript of the first array element to be included in the sort.
- LAX indicates the subscript of the last array element to be included in the sort.

NOTE: The array to be sorted must be dimensioned in the main program. If it is a 2-dimensional array, LAX and JAX must be translated into linear subscripts. For example:

To sort A (1, 1) through A (3, 3)
then JAX = 1
LAX = 9

SAMPLE PROBLEM

Sort the array:

2, 8, 6, 9, 5, 4.5, 1, 3, 2

SAMPLE SOLUTION

```
10      DIMENSIØN SEEDS(100)
20      PRINT:"NUMBER ØF ITEMS TØ BE SØRTED"
30      READ:LAX
40      PRINT:"ENTER THE ARRAY"
50      READ:(SEEDS(I),I=1,LAX)
60      CALL SGLSØRT(1,SEEDS,1,LAX)
70      PRINT:"THE SØRTED ARRAY"
80      PRINT:(SEEDS(I),I=1,LAX)
90      STØP;END

READY

*RUN *;SGLSØRT
NUMBER ØF ITEMS TØ BE SØRTED
= 9
ENTER THE ARRAY
= 2,8,6,9,5,4.5,1,3,2
THE SØRTED ARRAY
    1.0000000E+00    2.0000000E+00    2.0000000E+00    3.0000000E+00
    4.5000000E+00    5.0000000E+00    6.0000000E+00    8.0000000E+00
    9.0000000E+00

PRØGRAM STØP AT 90
*
```

This Fortran subroutine is intended to locate the position of a specified number in an array of numbers.

INSTRUCTIONS

The calling sequence for TLU1 is:

CALL TLU1(ARG, NTAB, TAB, J, IERR)

where

- ARG is the specified number.
- NTAB is the number of elements in the table.
- TAB is the name of the table.
- J and IERR are outputs as follows:

	J	IERR
ARG < TAB(1)	1	-1
ARG=TAB(K)	K	0
TAB(K) < ARG < TAB(K+1)	K	0
ARG > TAB(NTAB)	NTAB	1

RESTRICTION

The elements of the table must be in monotonic ascending order.

SAMPLE PROBLEM

Locate the position of the number 5.3 in the table, TAB = 1.0, 2.0, 4.0, 7.0, 10.

SAMPLE SOLUTION

```
10        DIMENSIØN TAB(5)
20        J=0
30        IERR=0
40        TAB(1)=1.0
50        TAB(2)=2.0
60        TAB(3)=4.0
70        TAB(4)=7.0
80        TAB(5)=10.0
90        CALL TLU1(5.3,5,TAB,J,IERR)
100        IF (IERR) 5,10,7
110     5 PRINT 6
120     6 FØRMAT(/38H ARGUMENT LESS THAN FIRST ENTRY IN TAB)
130        GØ TØ 20
140     7 PRINT 8
150     8 FØRMAT(/40H ARGUMENT GREATER THAN LAST ENTRY IN TAB)
160        GØ TØ 20
170    10 K=J+1
180       PRINT 11,J,J,K
190    11 FØRMAT(/13H ARGUMENT=TAB,I2,15H ØR BETWEEN TAB,I2,8H AND TAB,I2)
200    20 STØP
210       END

READY

*RUN *;TLU1

ARGUMENT=TAB 3 ØR BETWEEN TAB 3 AND TAB 4

PRØGRAM STØP AT 200
*
```

This Fortran subroutine sorts a real array in ascending or descending order. In addition, the elements of two corresponding real arrays are rearranged so that they remain parallel with the sorted array. This program uses a bubble sort. The elements of each of the three arrrays are physically moved.

INSTRUCTIONS

The calling sequence for TPLSORT is:

CALL TPLSORT (KODE, SEEDS, FOLLO, TAKE, JAX, LAX)

where

- KODE indicates the type of sort:

  if KODE = 1, ascending sort
  if KODE = 2, descending sort

- SEEDS indicates the array to be sorted.
- FOLLO indicates one corresponding array.
- TAKE indicates a second corresponding array.
- JAX indicates the subscript of the first element of the array, SEEDS to be included in the sort.
- LAX indicates the subscript of the last element of SEEDS to be included in the sort.

NOTE: If the array to be sorted is 2-dimensional, JAX and LAX must be translated into linear subscripts. For example:

To sort elements A (1, 1) through A (6, 50)
then JAX = 1
LAX = 300

To sort elements A (3, 3) through A(4, 4)
then JAX = 9
LAX = 16

SAMPLE PROBLEM

Sort the array:

9, 5, 7, 1, 4

Keeping:

4, 6, 5, 3, 9

And:

1, 9, 5, 7, 3

Parallel to it.

SAMPLE SOLUTION

```
10      DIMENSIØN SEEDS(100),FØLLØ(100),TAKE(100)
20      PRINT:"NUMBER ØF ITEMS IN ARRAY"
30      READ:LAX
40      PRINT:"ENTER THE ARRAY"
50      READ:(SEEDS(I),I=1,LAX)
60      PRINT:"ENTER THE FIRST FØLLØWING ARRAY"
70      READ:(FØLLØ(I),I=1,LAX)
80      PRINT:"ENTER THE SECØND FØLLØWING ARRAY"
90      READ:(TAKE(I),I=1,LLAX)
100      CALL TPLSØRT(1,SEEDS,FØLLØ,TAKE,1,LAX)
110      PRINT:"THE SØRTED ARRAYS"
120      PRINT 1,(SEEDS(I),FØLLØ(I),TAKE(I),I=1,LAX)
130    1 FØRMAT(/1P3E16.8)
140      STØP;END

READY

*RUN *;TPLSØRT
NUMBER ØF ITEMS IN ARRAY
= 5
ENTER THE ARRAY
= 9,5,7,1,4
ENTER THE FIRST FØLLØWING ARRAY
= 4,6,5,3,9
ENTER THE SECØND FØLLØWING ARRAY
= 1,9,5,7,3
THE SØRTED ARRAYS

  1.00000000E+00  3.00000000E+00  7.00000000E+00

  4.00000000E+00  9.00000000E+00  3.00000000E+00

  5.00000000E+00  6.00000000E+00  9.00000000E+00

  7.00000000E+00  5.00000000E+00  5.00000000E+00

  9.00000000E+00  4.00000000E+00  1.00000000E+00

PRØGRAM STØP AT 140
*
```

Honeywell Bull

HONEYWELL INFORMATION SYSTEMS