# SIMULAÇÃO DIGITAL DE SISTEMAS CONTÍNUOS

POR

JÚLIO CÉSAR TIBÚRCIO

Dissertação submetida a Escola Federal de Engenharia de Itajubá como parte dos requisitos para a obtenção do grau de Mestre em Ciências em Engenharia Elétrica.

Itajubá, dezembro de 1972.

## AGRADECIMENTOS

O autor agradece ao Professor JOSÉ ABEL ROYO DOS SANTOS, seu orientador, pelo estímulo e orientação recebidos na elaboração deste trabalho.

# SUMÁRIO

# SUMÁRIO

O presente trabalho originou-se da necessidade de dispor a biblioteca de programas do Laboratório de Computação e Simulação da Escola Federal de Engenharia de Itajubá, de um programa para a simulação digital de sistemas contínuos.

A instalação digital do L.C.S. da Escola Federal de Engenharia de Itajubá está equipada com um computador TR - 86 TELEFUNKEN, sendo este o único computador para fins científicos de tal fabricante existente atualmente no país.

Impunha-se, portanto, para o atendimento à necessidade acima mencionada, a confecção (ou aquisição) de um programa para simulação de sistemas contínuos específico para o TR-86 ou então a adaptação para este computador de programa já existente, desenvolvido para outros computadores.

A primeira opção, mais desejável do ponto de vista de eficiência, seria contudo uma solução a longo prazo pois a confecção de um tal programa demanda tempo consideravelmente longo.

A segunda, com menor eficiência, possibilitaria, bem mais rapidamente, a disponibilidade, para simulação de sistemas contínuos, da instalação digital do L.C.S.

Escolheu-se, por isso, a segunda opção como a mais adequada, presentemente, às necessidades da E.F.E.I., indo constituir a primeira opção objeto de trabalhos posteriores.

Portanto, o trabalho que estamos propondo é, tão somente uma adaptação para as características específicas da instalação digital do L.C.S. da E.F.E.I., de um programa para si-

mulação de sistemas contínuos já existente, desenvolvido para outro tipo de computador.

O nosso mérito, neste trabalho, se algum existe, será apenas o de poder acrescentar à biblioteca de programas do L.C.S., uma ferramenta que, esperamos, possa ser útil a todos aqueles que alguma vez desejarem empregar o computador em trabalhos de análise de sistemas contínuos.

Inicialmente, nossa intenção era adaptar o programa denominado SIMIC.

Tal programa, apresentado por Yaohan Chu em sua obra: "Digital Simulation of Continuous systems"(1), é uma versão simplificada do processador MIMIC para a linguagem do mesmo nome, desenvolvido para a família de computadores IBM 7090 por Petersen e Sansom.

Tal adaptação não pôde ser realizada por razões que serão discutidas posteriormente.

Em vista disto, decidimos adaptar o programa denominado C.S.M.P. - "Continuous Systems Modeling Program" - desenvolvido para o computador IBM 1130.(7)

Apresentamos ainda, neste trabalho, a linguagem usada para se escrever um programa de simulação, utilizando o processador adaptado.

Finalizamos o trabalho com alguns exemplos de aplicação, onde procuramos, respeitando os critérios de simplicidade e clareza de entendimento, focalizar casos típicos de simulação de sistemas contínuos, esperando com isso tornar de fácil compreensão, a usuários não familiarizados com o processador,

a utilização do mesmo com real proveito.

Se este nosso objetivo puder ser alcançado, nos daremos por largamente recompensados.

# CAPÍTULO I

## PRELIMINARES

# PRELIMINARES

## 1 - INTRODUÇÃO

Na investigação científica de um grande número de sistemas físicos, não é prático, por uma variedade de razões, o emprego direto de processos matemáticos.

Realmente, não é tarefa das mais sedutoras resolver uma ou várias equações diferenciais, possìvelmente não lineares, um número suficiente de vezes para se obter uma compreensão razoável do sistema em investigação, com variações quase sempre mínimas de parâmetros ou de condições iniciais de solução para solução.

É altamente desejável, portanto, na investigação de sistemas físicos, contar-se com um método que permita o fácil e rápido manuseio das equações matemáticas que descrevem o sistema, de forma a ganhar uma considerável compreensão de seu comportamento, sem o inconveniente de manipulações matemáticas proibitivas.

A simulação em computadores, de uso já consagrado, constitui, sem dúvida, um método que atende às características acima mencionadas.

Na análise de sistemas contínuos os computadores analógicos, com sua característica de processamento contínuo e paralelo, aparecem como ferramenta, por assim dizer, natural para a simulação.

De fato, entre as vantagens do uso do computador analógico como simulador, conta-se a possibilidade de estabelecer uma estreita correspondência entre cada componente do sistema físico em estudo, e o elemento do computador que o está modelando, de tal maneira que se consegue uma visão, ao mesmo tem

po do sistema como um todo e também de cada uma de suas partes.

Com isso, é possível chegar-se a um entendimento grandemente satisfatório do comportamento das várias variáveis em investigação.

A simulação de sistemas contínuos através de computadores digitais, embora não pareça tão espontânea, nem por isso deixa de constituir um recurso de enorme utilidade.

Apesar da desvantagem de seu processamento sequencial obrigar à simulação com retardamento de tempo, a grande precisão alcançada e a larga capacidade de armazenamento de informações tornam o computador digital grandemente desejável como simulador.

Recomenda ainda o seu uso a extrema popularidade atingida pelos computadores digitais em nossos dias, o que o torna muito mais facilmente disponível que os computadores analógicos.

Além disso as instalações digitais fornecem uma documentação de programas e resultados muito mais completa para o usuário.

É óbvio que todas estas vantagens não foram alcançadas simultaneamente.

Elas são o resultado do desenvolvimento de linguagens especiais para simulação que tiveram de ser concebidas e aperfeiçoadas cada vez mais para que todo o potencial da simulação digital pudesse ser utilizado.

## 2 - LINGUAGENS DE SIMULAÇÃO

O pioneiro na utilização do computador digital como simulador de sistemas contínuos foi R.G.Selfridge.

Trabalhando em Yonkern, Califórnia na "U. S. Naval

Ordnance Test Station" êle publicou seu trabalho em 1955.

Em 1958, H. F. Lesh do Laboratório de propulsão a jato em Pasadena, Califórnia, publicou seu trabalho por êle chamado: Differential Equations Pseudo-Code Interpreter (DEPI).

Ambos estes estudos foram desenvolvidos em linguagem de máquina e utilizando aritmética de ponto fixo.

O simulador DEPI utilizava para a integração numérica um Runge-Kutta de 4ª. ordem.

Ainda em 1958 apareceu, novamente na Califórnia, um compilador denominado ASTRAL (Analog Schematic Translator to Algebraic Language).

Desenvolvido por M. L. Stein, J. Rose e D. B.Parker, o compilador aceitava declarações de entrada que modelavam um computador analógico e produzia um programa FORTRAN que era então processado normalmente.

A existência do programa FORTRAN como etapa intermediária permitia a modificação do programa neste nível, característica que se manteve depois, nos posteriores programas de simulação.

Depois destes passos iniciais, um grande número de simuladores digitais foram desenvolvidos.

Em 1959 foi publicado o programa DYANA (Dynamics Analyzer); em 1961 apareceram BLODI (Block Diagramed Compiler) e DYSAC (Digital Simulated Analog Computer); DYNASAR (Dynamic Systems Analyzer) e PARTNER (Proof of Analog Results Through Numerically Equivalent Routine) em 1962.

Menção especial merece o simulador DAS (Digital Analog Simulador) desenvolvido em 1963 por R. A. Gaskil, J. W. Harris e A. L. McKnight.

O DAS, embora tenha sido, êle próprio, um simulador de grande utilidade, tem grande importância porque foi a base para o desenvolvimento do simulador chamado MIDAS (Modified Integration Digital Analog Simulator).

MIDAS apareceu em 1963 como resultado do trabalho de R. T. Harnett, F. J. Sansom e H. E. Petersen e foi um marco no desenvolvimento das linguagens para simulação digital.

O MIDAS adotou a linguagem do DAS porém modificou a rotina de integração, por isso seu nome.

Além disso adicionou várias das características mais desejáveis de outras linguagens existentes como: ordenação automática da sequência de etapas de computação, meios para a solução de equações implícitas, etc.

Em 1964, R. D. Brennan desenvolveu a linguagem denominada PACTOLUS (nome derivado da mitologia grega, Pactolus foi o rio onde Midas lavou-se para livrar-se do dom a êle dado pelos deuses de transformar tudo que tocasse em ouro).

MIDAS, apesar de todas as suas vantagens, apresentava também algumas desvantagens que o simulador PACTOLUS procurava corrigir.

Realmente PACTOLUS provou ser um simulador de grande utilidade e serviu de base para o desenvolvimento de duas outras linguagens: o DSL-90 e o C.S.M.P.

O sucessor do MIDAS foi o simulador MIMIC, cuja primeira versão apareceu em 1965, desenvolvida pela mesma equipe que criou o MIDAS.

Em 1967 apareceu uma versão melhorada.

Atualmente procura-se padronizar as linguagens de simulação.

O comitê para software de simulação do Simulation

Councils, Inc. introduziu especificações para uma nova linguagem de simulação digital chamada C.S.S.L. (Continuous System Simulation Language)[6].

O C.S.S.L. já foi implementado em vários computadores existentes e será, sem dúvida, em muitos outros ainda.

É uma linguagem para a representação de sistemas contínuos dinâmicos que podem ser descritos por conjuntos de equações diferenciais ordinárias.

O C.S.S.L. é um adjunto do FORTRAN IV sendo que todas as características desta linguagem estão disponíveis no C.S.S.L.

Incorpora ainda um macro-processador que permite ao usuário grande flexibilidade de expansão da linguagem, procurando preencher todas as características de uma linguagem "ideal" que, segundo o mesmo comitê, são:

a) Áreas de aplicação:

- Simulação totalmente digital de sistemas essencialmente paralelos.

- Programação digital para problemas híbridos.

- Soluções de verificação para computadores analógicos e híbridos modernos.

b) Categorias de Usuários:

- Engenheiros e Cientistas com interêsse em problemas de simulação, porém não familiarizados com técnicas de computação digital.

- Analistas de Simulação.

- Programadores para aplicações digitais de alto nível.

c) Tipos de computadores digitais onde possa ser im-

plementada:

- Todos os tipos de computadores digitais científicos.

# CAPÍTULO II

## O PROCESSADOR SIMIC

# O P R O C E S S A D O R  S I M I C

## 1 - INTRODUÇÃO

Nossa primeira intenção, quando nos propusemos a adaptar uma linguagem de simulação digital para o sistema do TR-86, foi a de trabalhar com o processador denominado SIMIC, uma simplificação do MIMIC.

A escolha desse processador deveu-se à sua grande simplicidade, pois o mesmo foi desenvolvido com finalidades educacionais e apresentado por Yaohan Chu em seu compêndio "Digital Simulation of Continuous Systems" (1) com o fito principal de mostrar a lógica que deve ser seguida na confecção de um processador para simulação digital.

O programa consta de menos de 500 cartões compreendendo o programa executivo e cinco subrotinas.

Um programa SIMIC é compatível com o processador MIMIC, o que, sem dúvida lhe confere uma vantagem adicional.

Por outro lado, o número de funções aceitas pelo SIMIC é bem menor do que o número aceito pelo MIMIC e ainda, como não existe subrotina de ordenação no SIMIC, as declarações de um programa escrito nesta linguagem devem ser adequadamente ordenadas em sequência.

A ordem das declarações deve ser a seguinte:

a - Declarações de constantes (CON)

b - Declarações de parâmetros (PAR)

c - Declarações simulando o modelo, em ordem correta

d - Declarações de finalização de corrida (FIN)

e - Declarações de impressão de títulos (HDR)

f - Declarações de impressão de resultados (OUT)

g - Declaração de fim de compilação (END)

h - Cartões de dados que devem seguir a mesma ordem das declarações CON e PAR

São as seguintes as funções aceitas pelo SIMIC:

| Forma | Descrição |
|---|---|
| ABS (A) | R = \|A\| |
| ADD (A,B) | R = A + B |
| AND (A,B) | R = TRUE se A = B = TRUE |
| ATN (A,B) | R = ARCTG (A/B) (R em radianos) |
| CON (A,B,C,D,E,F) | Dá nome a constantes |
| COS (A) | R = COS(A). (A em radianos) |
| DIV (A,B) | R = A/B |
| END | FIM DO PROGRAMA |
| EQL (A) | R = A |
| EXP (A) | R = EXP(A) |
| FIN (A,B) | A corrida termina se A>B |
| FSW (A,B,C,D) | R = B se A<0 |
| | R = C se A=0 |
| | R = D se A>0 |
| HDR (A,B,C,D,E,F) | Imprime títulos quando T=0 |
| INT (A,B) | $R = B + \int_0^T A\,dt$ |
| IOR (A,B) | R = TRUE se A=TRUE ou B=TRUE |
| LOG (A) | R = L(A) (Logaritmo natural) |
| LSW (A,B,C) | R = B se A=TRUE |
| | R = C se A=FALSE |
| MPY (A,B) | R = A X B |
| NEG (A) | R = -A |

| Forma | Descrição |
|---|---|
| NOT(A) | R = TRUE se A = FALSE<br>R = FALSE se A = TRUE |
| OUT (A,B,C,D,E,F) | Imprime os valores de A,B,C,D,E e F cada DT unidades de Tempo |
| PAR (A,B,C,D,E,F) | Dá nome a parâmetros que devem ser modificados a cada corrida. |
| SIN (A) | R = SEN(A) (A em radianos) |
| SQR (A) | R = $\sqrt{A}$ |
| SUB (A,B) | R = A - B |

O processador SIMIC deve aceitar o programa SIMIC e gerar um programa em linguagem de máquina em duas etapas: a compilação e a montagem.

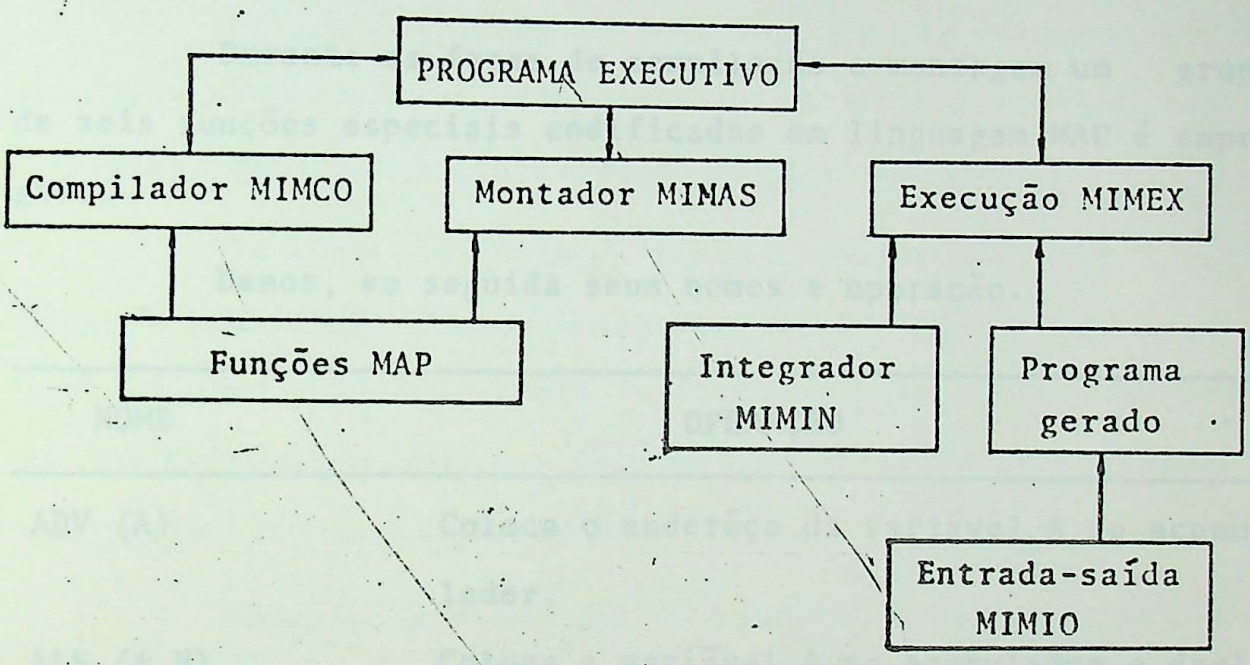
Uma terceira fase, a de execução, segue as duas anteriores, durante a qual o programa em linguagem de máquina é executado para produzir uma solução ao modelo codificado.

O programa executivo e as cinco subrotinas são programas em linguagem FORTRAN. Em edição, existe um grupo de funções, que são codificadas em MAP (Linguagem tipo ASSEMBLER da família de computadores 7090 da IBM), constituíndo o seguinte conjunto:

| PROGRAMA | LINGUAGEM | OPERAÇÃO |
|---|---|---|
| Programa Executivo | FORTRAN | Comando do programa |
| Subrotina MIMCO | FORTRAN | Compilação do programa SIMIC |
| Subrotina MIMAS | FORTRAN | Montagem do programa em L.M. |

| PROGRAMA | LINGUAGEM | OPERAÇÃO |
|---|---|---|
| Subrotina MIMEX | FORTRAN | Execução do programa em L.M. |
| Subrotina MIMIN | FORTRAN | Subrotina de Integração |
| Subrotina MIMIO | FORTRAN | Entrada e saída de dados |
| Funções MAP | MAP | Empregadas durante compilação e montagem. |

A organização do processador é a seguinte:




O programa executivo do processador SIMIC tem como função dar sequência às três fases de compilação, montagem e execução, chamando as subrotinas MIMCO, MIMAS e MIMEX, respectivamente.

O oompilador MIMCO lê as declarações de um programa SIMIC e as traduz em um programa em linguagem-função que fica armazenado em uma região da memória denominada BCD.

O montador MIMAS traduz o programa em linguagem-função aí armazenado, em um programa em linguagem de máquina que

é armazenado em outra região da memória denominada FF.

Este programa é então executado por meio da subrotina MIMEX.

Para a integração numérica a subrotina MIMIN é empregada.

O processo de integração utilizado é um preditor-corretor de segunda ordem de Adams-Bashforth.

Para a entrada e saída de dados utiliza-se a subrotina MIMIO.

Durante as fases de compilação e montagem um grupo de seis funções especiais codificadas em linguagem MAP é empregado.

Damos, em seguida seus nomes e operação.

| NOME | OPERAÇÃO |
|---|---|
| ADV (A) | Coloca o enderêço da variável A no acumulador. |
| ALS (A,N) | Coloca a variável A no acumulador e desloca o conteúdo do mesmo N bits para a esquerda. O resultado é deixado no próprio acumulador. |
| ARS (A,N) | Idem, exceto que o deslocamento é para a direita. |
| OR (A,B) | Realiza a operação lógica OR entre as variáveis A e B, deixando o resultado no acumulador. |
| F | Armazena a configuração do sistema e a transfere ao programa em linguagem de má- |

| NOME | OPERAÇÃO |
|---|---|
| FEXIT | quina localizado na região de memória FF. Retorna o controle ao programa que chamou. |

2 - SUBROTINAS ADAPTADAS

Não nos foi possível adaptar para o TR-86 o processador SIMIC devido à necessidade que tem esse processador de dispor das funções MAP, o que não acontece com o TR-86.

Para a adaptação de tais funções nós necessitaríamos de conhecer em maiores detalhes a linguagem MAP do computador IBM 1130, bem como a linguagem de máquina do TR-86.

Durante essa fase de nosso trabalho nós procuramos obter a publicação "Macro Assembly Program (MAP) Language, version 13, IBM 7090/7094 IBSYS Operating System, Form C28-6392-0" da IBM Corporation que nos iria possibilitar um conhecimento mais detalhado da linguagem MAP.

Todavia, não nos foi possível obtê-la; em vista disso decidimos interromper a adaptação do SIMIC e optar por outro processador.

Já havíamos, no entretanto, àquela altura adaptado o programa executivo e 3 das 5 subrotinas (as que não utilizam funções MAP).

Em seguida, apresentamos as listagens da parte adaptada apenas como documentação de nosso trabalho.

```
TR 86 FORTRAN COMPILER MV 08 (70)   FUKA.

C         PROGRAMA EXECUTIVO DO COMPILADOR SIMIC
C         JULIO CESAR - EFEI - 1972
C
      REAL*8 SIMAGE(7),RIMAGE(7)
      COMMON NEQ,P(93),R(2500),S(2500),FF(4000),BCD(10,900)
      COMMON IOUT,IPAR,IHDR,IFIN,IEND,NPAR
      DATA SIMAGE(1),RIMAGE(1)/1HT,0.D0/
      DATA SIMAGE(2),RIMAGE(2)/1H ,0.D0/
      DATA SIMAGE(3),RIMAGE(3)/2HDT,.1D0/
      DATA SIMAGE(4),RIMAGE(4)/5HDTMAX,1000.D0/
      DATA SIMAGE(5),RIMAGE(5)/5HDTMIN,0.D0/
      DATA SIMAGE(6),RIMAGE(6)/4HTRUE,68719476735.D0/
      DATA SIMAGE(7),RIMAGE(7)/5HFALSE,0.D0/
C         INICIO DA EXECUCAO DO PROGRAMA.
      NPAR = 0
      NEQ = 0
      DO 10 K=1,7
      S(K) = SIMAGE(K)
   10 R(K) = RIMAGE(K)
C         COMPILACAO DO PROGRAMA
      CALL MIMCO(I)
C         IMPRESSAO DO PROGRAMA COMPILADO
      WRITE(1,30)
      WRITE(1,40)  (J,BCD(9,J),(BCD(K,J),K=1,8),J=1,I)
      WRITE(1,20)
C         MONTAGEM DO PROGRAMA
      CALL MIMAS
C         EXECUCAO DO PROGRAMA
      CALL MIMEX
      STOP
   20 FORMAT(1H1)
   30 FORMAT(1H1,24X,41H***PROGRAMA GERADO EM LINGUAGEM-FUNCAO***//4X,3H
     1IFN,3X,3HLCV,5X,9HRESULTADO,3X,3HFTN,5X,1HA,6X,1H8,6X,1HC,6X,1HD,
     16X,1HE,6X,1HF//)
   40 FORMAT((I6,3X,A6,3X,A6,3X,A3,3X,A6,5(1X,A6)))
      END
```

```
TR 86 FORTRAN COMPILER MV 08 (70)    FOKA.
C          PROGRAMA DE EXECUCAO
      SUBROUTINE MIMEX
      COMMON NEQ,P(93),R(2500),S(2500),FF(4000),BCD(10,900)
      COMMON IOUT,IPAR,IHDR,IFIN,IEND,NPAR
C          LEITURA DOS DADOS DE ENTRADA
   10 IPAR=1
      IEND=0
      IOUT=0
      IFIN=0
      IHDR=0
      R(1)=0
      CALL F
C          IMPRESSAO DE TITULOS E DE DADOS DE SAIDA. TESTE DE FIM DE
C          CORRIDA DE PROGRAMA
      IHDR=1
   20 IFIN=1
      IOUT=1
      CALL F
      IHDR=0
      IFIN=0
      IOUT=0
C          TESTE DE FIM DE CORRIDA
      IF(IEND.NE.0) GO TO 30
C          INTEGRACAO
      CALL MIMIN
      GO TO 20
C          TESTE PARA VERIFICAR SE HAVERA CORRIDAS POSTERIORES
   30 WRITE(1,40)
      IF(NPAR.NE.0) GO TO 10
      STOP
   40 FORMAT(1H1)
      RETURN
      END
```

```
IR 86 FORTRAN COMPILER MV 08 (70)   FOKA.

C         SUBROTINA DE INTEGRACAO
C         JULIO CESAR   EFEI     1972
      SUBROUTINE MIMIN
      DOUBLE PRECISION T
      LOGICAL DOUBLE
      DIMENSION Y(93),P(93),POLD(93)
      COMMON NEQ,PNEW(93),TNEW,BLANK,DT,DTMAX,DTMIN,TRUE,FALSE,YNEW(93)
C         INICIO
      TFINAL = TNEW + DT
   10 H = AMIN1 (DTMAX, TFINAL - TNEW)
      T = TNEW
      DO 20 I = 1,NEQ
      Y(I) = YNEW(I)
      P(I) = PNEW(I)
   20 POLD(I) = PNEW(I)
C         PREDICAO DO NOVO PONTO
   30 TNEW = T + H
      DO 40 I = 1,NEQ
   40 YNEW(I) = Y(I) + H*(3.*P(I) - POLD(I))/2.
C         CORRECAO DO NOVO PONTO E CALCULO DO ERRO RELATIVO MAXIMO.
      CALL F
      RMAX = 0.
      DO 50 I = 1,NEQ
      YNEW(I) = Y(I) + H*(P(I) + PNEW(I))/2.
      E = H*(PNEW(I) - 2.*P(I) + POLD(I))/12.
   50 RMAX = AMAX1 (RMAX,ABS(E/YNEW(I)))
C         TESTE DE REPARTICAO DE INTERVALO
      IF ((RMAX.GT.5.E-6).AND.(H.GT.DTMIN)) GO TO 70
C         TESTE DE TEMPO DE RETORNO.
      IF (TNEW.GE.TFINAL) RETURN
C         CALCULO DA DERIVADA CORRIGIDA.
      CALL F
C         CALCULO DA VARIAVEL LOGICA DOUBLE
      DOUBLE = (RMAX.LE..625E-6).AND.(2.*H.LE.DTMAX).AND.
     1           ((TNEW+2.*H).LE.TFINAL)
C         TESTE DE OVERSHOOT DE RETORNO.
      IF ((TNEW+H).GT.TFINAL) GO TO 10
C         AVANCO PARA CALCULAR NOVO PONTO.
      T= T + H
      IF (DOUBLE) H = 2.*H
      DO 60 I=1,NEQ
      Y(I) = YNEW(I)
      IF (.NOT.DOUBLE) POLD(I) = P(I)
   60 P(I) = PNEW(I)
      GO TO 30
C         DIVIDA O INTERVALO.
   70 DO 80 I=1,NEQ
   80 YNEW(I) = Y(I) - H*(3.*P(I) + POLD(I))/8.
      H = H/2.
      TNEW = T - H
      CALL F
      DO 90 I=1,NEQ
   90 POLD(I) = PNEW(I)
      GO TO 30
      END
```

```
TR 86, FORTRAN COMPILER MV 08 (70)    FOKA.

C          SUBROTINA PARA ENTRADA E SAIDA DE DADOS.
C          JULIO CESAR    EFEI    1972
      SUBROUTINE MIMIO(A,B,C,D,E,F,I)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),FMT(8)
      COMMON NEQ,P(99),R(2500),S(2500),FF(4000),BCD(10,900)
      COMMON IOUT,IPAR,IHOR,IFIN,IEND,NPAR
      DATA QF205/6H,F20.5/,QBL/1H /,Q14XA6/6H,14XA6/
      DATA FMT(1)/6H(1P    /,FMT(8)/6H)      /
      GO TO (2,4,6),I
    2 WRITE (6,20) A(2501),B(2501),C(2501),D(2501),E(2501),F(2501)
      RETURN
    4 READ (5,30) A,B,C,D,E,F
      WRITE (6,40)
      WRITE (6,20)  A(2501),B(2501),C(2501),D(2501),E(2501),F(2501)
    6 DO 10 I=2,7
   10 FMT(I) =.QE205
      R(2)=QBL
      IF (A.EQ.QBL) FMT(2)=Q14XA6
      IF (B.EQ.QBL) FMT(3)=Q14XA6
      IF (C.EQ.QBL) FMT(4)=Q14XA6
      IF (D.EQ.QBL) FMT(5)=Q14XA6
      IF (E.EQ.QBL) FMT(6)=Q14XA6
      IF (F.EQ.QBL) FMT(7)=Q14XA6
      WRITE(6,FMT) A,B,C,D,E,F
      RETURN
   20 FORMAT(6(11X,A6,3X))
   30 FORMAT(6E12.4)
   40 FORMAT(/)
      END
```

# C A P Í T U L O  I I I

## PROGRAMA SIMULADOR DE SISTEMAS CONTÍNUOS

# PROGRAMA SIMULADOR DE SISTEMAS CONTÍNUOS

## 1 - INTRODUÇÃO

Como não nos foi possível a adaptação do processador SIMIC, pelos motivos anteriormente expostos, decidimos adaptar então o programa chamado CSMP (Continuous System Modeling Program) desenvolvido, a partir do programa PACTOLUS, para os computadores 1130 da IBM.

O CSMP é um processador com linguagem orientada para diagramas de blocos que apresenta grande versatilidade de opções durante a sessão de simulação.

Esta versatilidade está baseada na utilização das chaves existentes no painel do computador IBM 1130 que, dependendo da sua posição, permitem ao programa executar diferentes trechos.

No computador TR-86 não existem tais chaves; razão pela qual nossa tarefa, ao adaptar o CSMP para o TR-86, era a de manter essa grande disponibilidade de opções através de alguma outra forma.

O recurso utilizado foi o de criar uma interação entre o usuário e o computador através de mensagens enviadas pelo teletipo do console pedindo ao usuário que faça suas opções e as forneça ao computador da mesma forma, via teletipo do console.

Com isto o programa tornou-se bastante conversacional.

A desvantagem deste método é que, devido à baixa velocidade de impressão do teletipo, o programa tornou-se mais

lento.

Ainda, no programa adaptado não aparecem as subrotinas para comandar a saída de resultados em forma de curvas uma vez que o nosso sistema não dispõe, presentemente, de traçador de gráficos.

Apesar dessas desvantagens acreditamos ser o programa um recurso bastante útil cuja extrema facilidade de utilização permite a qualquer usuário, mesmo sem experiência de simulação digital, tornar-se rapidamente familiarizado com o mesmo.

## 2 - DESCRIÇÃO GERAL

O PROGRAMA SIMULADOR DE SISTEMAS CONTÍNUOS(P.S.S.C.) é um simulador análogo digital no qual os blocos funcionais da linguagem de entrada representam os elementos e a organização de um computador analógico.

Nele estão compreendidos um grupo de elementos de simulação padrões e um grupo de elementos especiais que o usuário pode especificar para suas necessidades particulares.

Cada tipo de elemento tem um símbolo diagramático e um símbolo de linguagem.

O usuário começa por desenvolver um diagrama de blocos mostrando as várias interconexões dos elementos constituindo o modelo para a simulação.

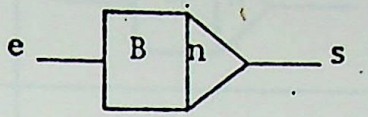
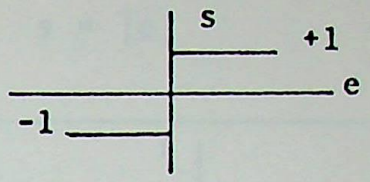
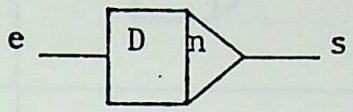
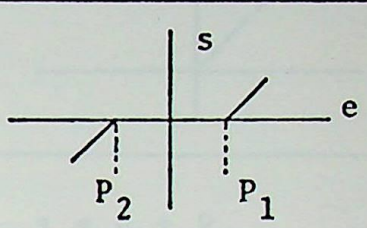
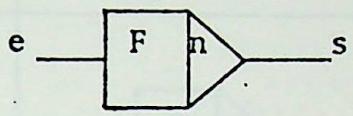
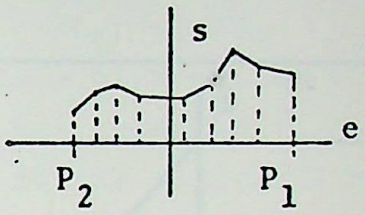
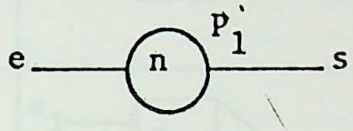
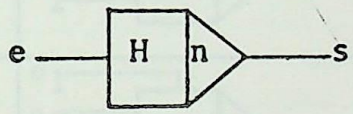
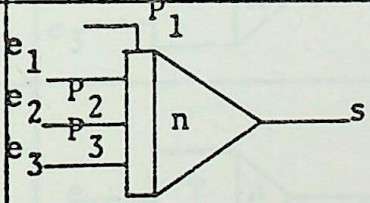
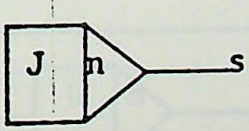
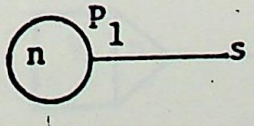
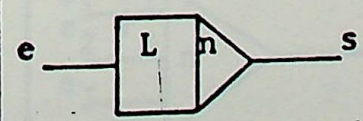
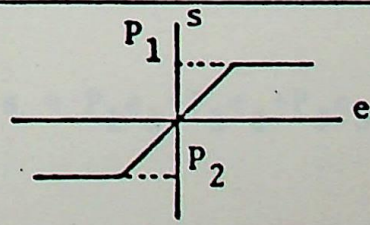
Em seguida esse diagrama é traduzido em um conjunto de declarações em linguagem PSSC que é então processado para produzir uma solução ao modelo utilizado.

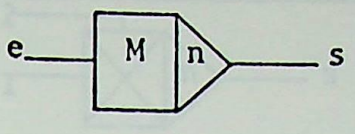
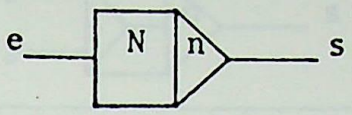
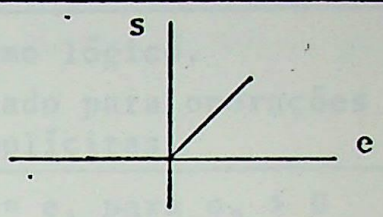
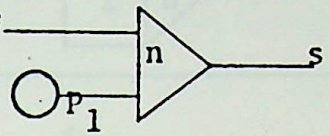
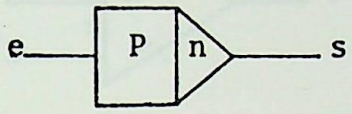
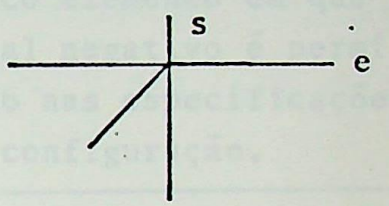
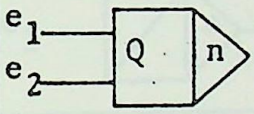
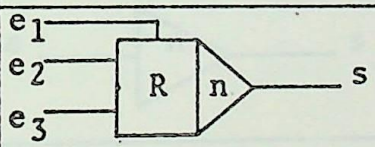
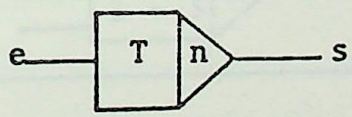
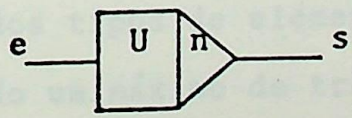
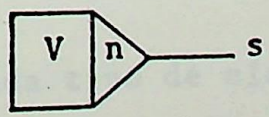
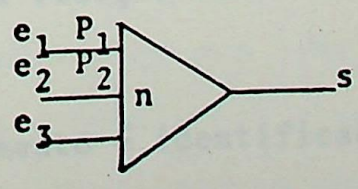
Uma característica bastante importante é a opção de entrar com essas declarações ou por meio de cartões perfurados ou diretamente pelo console.

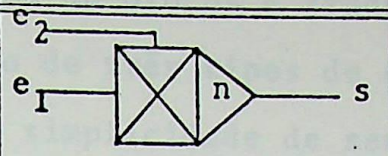
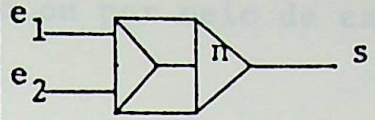
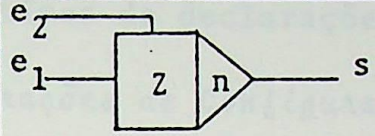
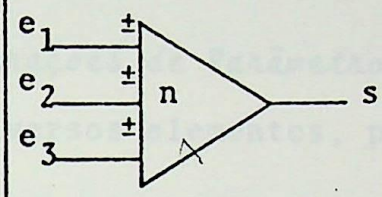
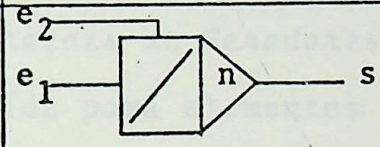
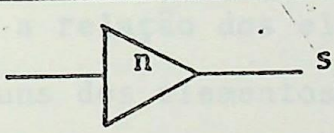
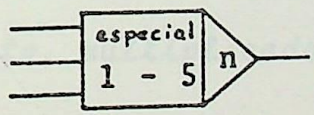
Durante a simulação o usuário recebe instruções do teletipo do console quanto à maneira de conduzir a simulação.

Ainda, durante uma simulação o usuário tem a possibilidade de interagir com o modelo como o faria em um computador analógico, modificando o modelo em si ou parâmetros do sistema em estudo.

O PSSC conta com 25 elementos funcionais padrão e 5 elementos especiais a serem especificados pelo usuário, compreendidos na relação a seguir:

| TIPO DE ELEMENTO | SIMBOLO de LINGUAGEM | SIMBOLO DE DIAGRAMA | DESCRIÇÃO |
|---|---|---|---|
| Bang-Bang | B |  |  |
| Espaço - Morto | D |  |  |
| Gerador de Funções | F |  |  |
| Multiplicador Constante | G |  | $s = P_1 e$ |
| Raiz - Quadrada | H |  | $s = \sqrt{e}$ |
| Integrador | I |  | $s = P_1 + \int (e_1 + P_2 e_2 + P_3 e_3)\, dt$ |
| Gerador de números Aleatórios | J |  | Gerador de números aleatórios entre ±1 |
| Gerador de Constante | K |  | $s = P_1$ |
| Limitador | L |  |  |

| TIPO DE ELEMENTO | SIMBOLO LINGUAGEM | SIMBOLO DE DIAGRAMA | DESCRIÇÃO |
|---|---|---|---|
| Valor Absoluto | M |   | $s = \|e\|$ |
| "Clipper" Negativo | N |   |   |
| Somador de Constante | O |   | $s = e + P_1$ |
| "Clipper" Positivo | P |   |   |
| Elemento Finalizador | Q |   | A Simulação termina se $e_1 > e_2$ |
| Relé | R |   | $s = \begin{cases} e_2 & \geq 0 \\ e_3 & < 0 \end{cases}$ para $e_1$ |
| Gerador de Pulsos | T |   | Gera um trem de pulsos com período igual a $P_1$ começa quando $e \geq 0$ |
| Retardo Unitário | U |   | $s = e(t - \Delta t/2)$ |
| Vácuo | V |   | $s \neq f(e)$ usado em conjunto com o elemento Y |
| Somador Ponderado | W |   | $s = P_1e_1 + P_2e_2 + P_3e_3$ |

| TIPO DE ELEMENTO | SÍMBOLO LINGUAGEM | SÍMBOLO DE DIAGRAMA | DESCRIÇÃO |
|---|---|---|---|
| Multiplicador | X |   | $s = e_1 \times e_2$ |
| Elemento Y | Y |   | Ramo lógico. Usado para operações implícitas. |
| Sustentador de ordem zero | Z |   | $s = e_1$ para $e_2 > 0$ s não muda para $e_2 \leq 0$ |
| Somador | + |   | $s = \pm e_1 \pm e_2 \pm e_3$ Único elemento em que o sinal negativo é permitido nas especificações de configuração. |
| Divisor | / |   | $s = e_1 / e_2$ |
| Inversor | - |   | $s = -e$ |
| Elementos Especiais | 1-5 |   | Subrotinas especificadas pelo usuário. |

OBS.: n representa o número do bloco.

Cada um dos tipos de elemento especifica uma relação funcional envolvendo um máximo de três variáveis de entrada e três parâmetros.

A saída de cada tipo de elemento é uma quantidade escalar definida pela relação funcional particular desse tipo de elemento.

Cada elemento é identificado por um símbolo diagramático e por um símbolo de linguagem.

O diagrama de blocos é traduzido em um programa de computador por meio de três tipos de declarações em linguagem PSSC que, devido à simplicidade de seu formato, podem ser fornecidas ao programa ou por meio de cartões perfurados ou através do console.

Os três tipos de declarações são:

1) *Declarações de Configuração*, que definem a interconexão dos blocos e especificam a operação funcional desejada.

2) *Declarações de Parâmetros*, que associam constantes numéricas a diversos elementos, particularizando sua operação funcional.

3) *Declarações de Geradores de função*, que definem a relação entrada/saída para elementos geradores de função.

Observando a relação dos elementos disponíveis no PSSC nota-se que alguns dos elementos tem uma ou mais entradas, porém nenhum parâmetro associado, como por exemplo os elementos: *valor absoluto*, *multiplicador* e *divisor*.

Cada vez que um desses elementos é usado em uma simulação êle requer uma declaração de configuração separada.

Os elementos que têm parâmetros associados com sua descrição requerem declarações de configuração e ainda declarações de parâmetros a cada vez que são usados.

É o caso, por exemplo do multiplicador constante que requer uma declaração de configuração que irá estabelecer sua identidade e interconexão em uma simulação e ainda uma declaração de parâmetro que especificará o valor constante pelo qual este elemento está multiplicando a variável de entrada.

O elemento gerador de função, por sua vez requer todos os três tipos de declaração:

A declaração de configuração para especificar sua interconexão no diagrama de blocos, a declaração de parâmetros que determina os valores máximo e mínimo da variável de entrada e a declaração de gerador de função que especifica o valor da variável de saída em cada um dos pontos de interseção.

Todos os três tipos de declaração utilizam formato fixo, o que simplifica o trabalho de preparação dos cartões perfurados.

O programa é não procedural quanto às declarações de configuração o que quer dizer que elas podem dar entrada no programa em qualquer ordem, sendo a ordenação para a sequenciação correta dos cálculos realizada automaticamente.

Todas as declarações para a simulação podem ser preparadas antes da sessão, usando cartões perfurados ou podem dar entrada no programa diretamente, através do teletipo do console.

Da mesma forma, adições ou modificações podem ser feitas nas declarações durante a simulação, enquanto o usuário testa o modelo.

O PSSC pode ser classificado como um tipo especial de programa para resolver equações diferenciais no qual o conjunto de equações diferenciais é especificado por uma linguagem especial orientada para o diagrama de blocos.

Para realizar esta tarefa o PSSC dispõe de um algoritmo de ordenação que determina a sequência de computação e ainda de uma fórmula de integração numérica.

Diferentemente da maioria dos programas digitais nos quais a ordem da codificação é importante, o PSSC é organizado como uma linguagem de processamento paralelo, isto é, a ordem na qual as declarações de configuração dão entrada no programa não afeta a subsequente solução.

O algoritmo de ordenação é usado depois de cada mudança na configuração de modelo para determinar a ordem correta de cálculo dos elementos funcionais.

Nenhum elemento pode ser processado até que valores atualizados de suas variáveis de entrada estejam disponíveis.

Supõe-se que a cada intervalo de tempo, as constantes e os valores na saída de elementos com memória estejam disponíveis.

Usando-se estas constantes e elementos, um ou mais blocos podem ser processados e seus valores de saída então ficam disponíveis como entradas de outros blocos.

Se a configuração do modelo for consistente, todos os blocos poderão ser processados de acordo com o algoritmo de ordenação.

Esta operação é realizada automaticamente e requer apenas alguns segundos.

Se o algoritmo de ordenação indica uma configuração inconsistente, o programa envia uma mensagem de erro.

A causa mais comum de erro na ordenação é a existência de um "loop" algébrico, isto é, uma malha fechada no diagrama que não inclui nenhum elemento com memória, como um integrador.

Caso a simulação requeira uma operação implícita, os

elementos vácuo e Y devem ser usados para desfazer tais "loops" porque êles contêm o mecanismo de iteração necessário.

A operação de integração é aproximada pelo uso de um método numérico Runge-Kutta de segunda ordem.

A simplicidade deste método é particularmente importante ao usuário que deseja desenvolver elementos especiais que envolvem memória.

O ponto essencial do método é que a cada vez que um bloco é avaliado, o valor da variável independente, tempo, é aumentado da metade do valor do intervalo de integração.

O programa, automàticamente, envia uma mensagem solicitando ao usuário entrar com o valor do intervalo de integração através do teletipo do console.

Da mesma forma o usuário é solicitado a entrar com o valor do tempo total, ou seja,duração desejada da corrida.

É importante notar que a unidade de tempo está implícita na definição do modelo.

A unidade utilizada deve ser coerente em toda a simulação.

## 3 - TÉCNICA DE MODELAGEM

O modelo para a simulação é obtido interligando-se apropriadamente os elementos funcionais de forma a corresponder ao sistema em estudo.

A técnica de diagramas de blocos é a mais indicada para a realização dessa tarefa.

Cada bloco no diagrama de simulação é identificado por um número que pode ser arbitrariamente designado entre 1 e

75. Geralmente, no diagrama de blocos, este número é indicado dentro do símbolo do elemento.

Para um grande número de elementos a ordem na qual as entradas vêm especificadas é importante.

Os elementos que admitem apenas uma entrada devem ter a mesma especificada como entrada 1.

Os elementos *Finalizador*, *Relê*, *Y*, *Sustentador de ordem zero* e *Divisor* exigem que as entradas sejam especificadas na ordem indicada na relação anterior.

Por exemplo, se a ordem das entradas do elemento *divisor* for invertida obter-se-á o inverso do quociente desejado.

De uma maneira geral, a entrada 3 só é usada se as entradas 1 e 2 já o foram e a entrada 2 só é usada se a entrada 1 já o foi.

As exceções a esta regra são os elementos *integrador*, *somador* e *somador ponderado* nos quais qualquer uma ou todas as três entradas podem ser usadas.

O somador é o único elemento ao qual um sinal negativo pode ser associado a qualquer uma de suas três entradas caso seja desejado.

Por exemplo a seguinte declaração:

| BLOCO | TIPO | ENTRADA 1 | ENTRADA 2 | ENTRADA 3 |
|---|---|---|---|---|
| 38 | + | | -27 | 8 |

faz gerar a diferença entre as saídas dos blocos 8 e 27.

O número máximo de parâmetros que pode ser associado a cada bloco é três.

Alguns tipos de elementos utilizam os três parâme-

tros, enquanto que outros não utilizam nenhum.

Por exemplo, os integradores utilizam o primeiro parâmetro como condição inicial e os dois restantes como ganhos para as entradas 2 e 3 respectivamente, enquanto que os divisores não utilizam parâmetros.

Para todos os elementos que têm *memória* ( integradores, retardos unitários, vácuo, sustentador de ordem zero), o primeiro parâmetro é sempre usado para especificar a condição inicial, ou valor da saída no tempo t = 0.

Uma condição inicial ou um parâmetro que não é especificado explicitamente é, automaticamente, feito igual a zero.

Dessa forma, uma condição inicial ou um parâmetro só necessitam ser declarados caso sejam diferentes de zero.

Caso, em alguma porção do modelo, disponha-se de uma relação Y = f(X) em forma tabular ou gráfica, a mesma pode ser declarada ao P.S.S.C. utilizando-se o elemento *gerador de funções*.

Tal relação deve poder ser aproximada aproximadamente por dez segmentos de reta igualmente espaçados no eixo dos X, dentro do intervalo de interêsse.

Os valores máximo e mínimo da variável de entrada X dentro do intervalo de interesse devem ser especificados como parâmetros do gerador de função.

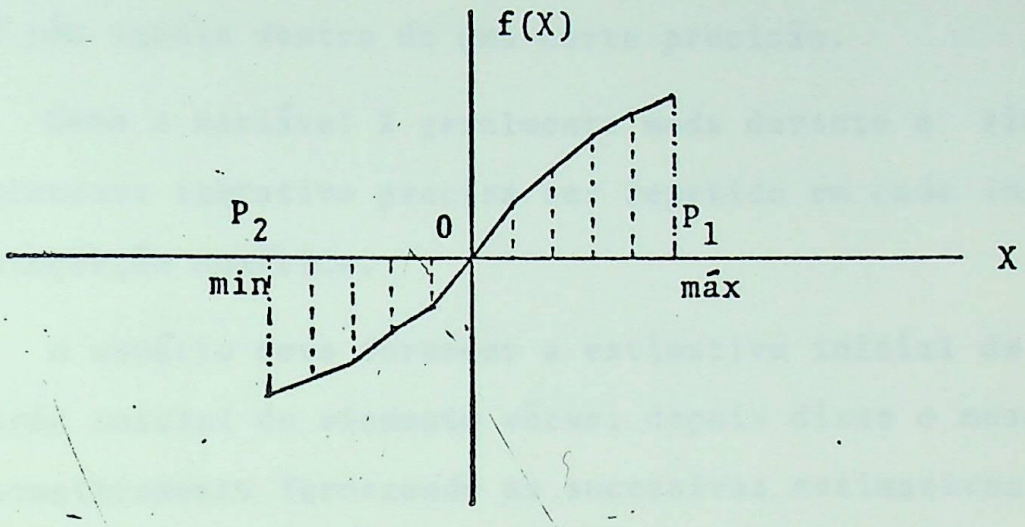
Os valores da função nos pontos de interseção dos segmentos são declarados por meio de cartões perfurados ou pelo teletipo do console.

Os valores de saída dos geradores de função são cal-

culados por meio de interpolação linear entre os pontos de interseção.

Como exemplo, a resposta de um dispositivo qualquer de um sistema poderia ser simulada conforme a figura 3.1.

As especificações associadas com o gerador de função utilizado poderiam entrar através de cartões perfurados ou pelo console.




| BLOCO | TIPO | ENTRADA 1 | ENTRADA 2 | ENTRADA 3 |
|---|---|---|---|---|
| (52) | (F) | (28) | ( ) | ( ) |

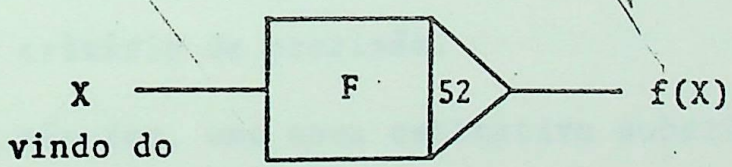



fig. 3.1. *Especificação de gerador de função*

O número máximo de geradores de função para uma configuração qualquer utilizando os elementos padrões é três.

Caso seja necessária a utilização de um número maior dos mesmos pode-se fazê-lo através do uso dos elementos *especiais*.

Se a simulação envolve relações implícitas da forma

Y = f(Y,X), isto pode ser modelado através do uso combinado do ramo lógico chamado *elemento Y* e do elemento *vácuo*.

Em computação digital, tais relações são calculadas através de processos iterativos. Dada uma estimativa inicial do valor da variável Y, o segundo membro da equação pode ser calculado para produzir uma estimativa melhor para o valor de Y. Este processo é repetido até que duas sucessivas estimativas de Y são iguais dentro de uma certa precisão.

Como a variável X geralmente muda durante a simulação, o processo iterativo precisa ser repetido em cada intervalo da integração numérica.

O usuário deve fornecer a estimativa inicial de Y como condição inicial do elemento *vácuo*; depois disso o mesmo vai, automaticamente fornecendo as sucessivas estimativas de Y.

O *elemento Y*, usado em associação com o elemento *vácuo*, determina se a iteração na variável Y é suficiente para satisfazer o critério de precisão.

Se não for, uma nova estimativa substitue o valor presente na saída do elemento *vácuo* e uma outra iteração é iniciada.

Quando a convergência for atingida a variável Y é gerada como saída do *elemento Y* e a computação prossegue de acordo com o restante da configuração do modelo.

A maneira como estes elementos devem ser usados está indicada na figura 3.2.

A entrada 1 do elemento Y deve ser a variável f(Y,X), calculada usando a estimativa do valor de Y vinda do elemento

*vácuo*.

A entrada 2 do elemento Y deve ser a saída do elemento vácuo.

O parâmetro 1 do elemento Y é usado para especificar o critério de precisão para o teste de convergência do resultado.

Por exemplo se a especificação for: $P_1 = 0,001$ para o elemento Y, isto significa que o teste de convergência será satisfeito quando a diferença entre duas sucessivas estimativas de sua saída for menor do que 0,001.

O parâmetro 2 é um fator de aceleração para o processo iterativo cujo valor deve ser especificado entre zero e a unidade.

Se a operação implícita é de convergência rápida $P_2$ pode receber o valor zero; se a iteração tende a oscilar $P_2$ pode ser feito igual a 0,5 por exemplo.

Com respeito à simulação de funções implícitas, dois aspectos devem ser ressaltados:

a) Antes de especificar o valor de $P_2$ para o elemento Y deve-se experimentar com alguns valores de forma a garantir rápida convergência.

b) O número designado para um bloco que foi definido como *vácuo* deve ser o maior número da configuração de forma a aumentar a eficiência do processo iterativo.

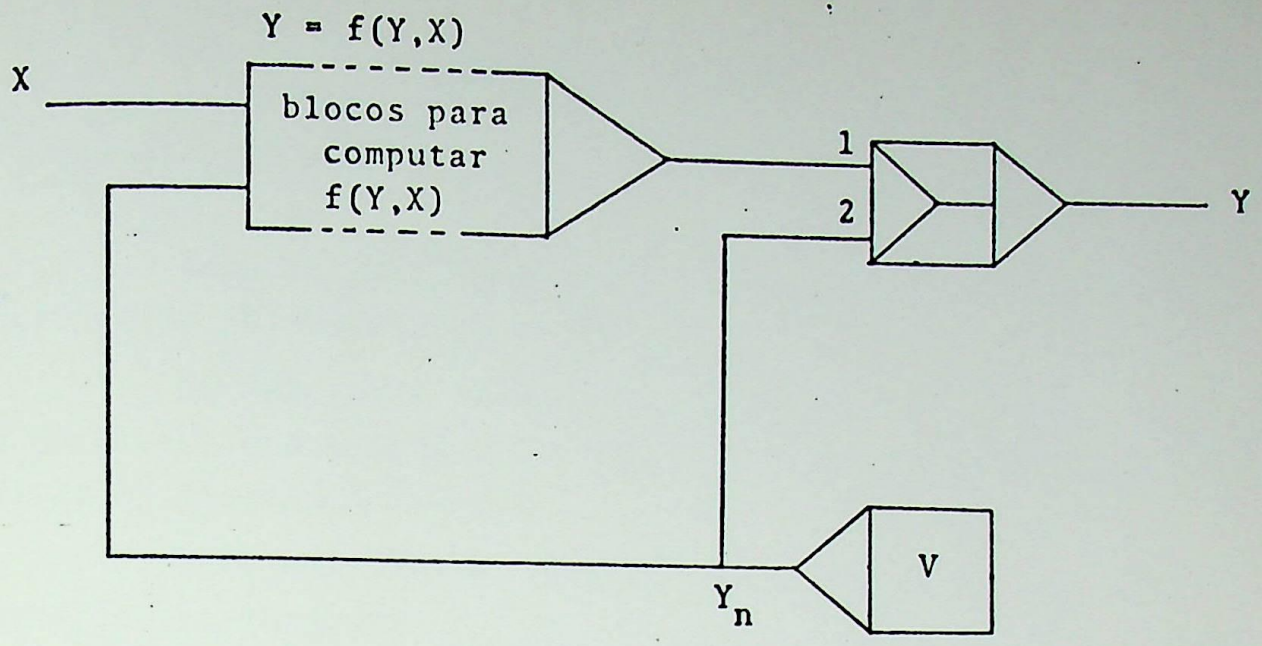



*Fig. 3.2 - Uso dos elementos Vácuo e Y para operações implícitas.*

# CAPÍTULO IV

## PROGRAMA SIMULADOR DE SISTEMAS CONTÍNUOS

## (MANUAL DO PROGRAMA)

# MANUAL DO PROGRAMA

## 1 - INTRODUÇÃO

Logo em seguida à fase de modelagem vem a fase de solução em um estudo de simulação.

No PSSC esta fase se inicia com a tradução do diagrama de blocos que modela o sistema em estudo, em declarações escritas em linguagem P.S.S.C.

Existem 3 possibilidades de se entrar com estas declarações no programa:

a) O usuário pode preparar de antemão um deck de cartões perfurados contendo as especificações completas de configuração, parâmetros e geradores de função.

b) Ele pode preparar um deck de cartões perfurados contendo parte das especificações de configuração e entrar com as demais declarações por meio do teletipo do console.

c) O usuário pode entrar com todas as declarações pelo teletipo do console.

A maneira mais efetiva de usar o programa é preparar as especificações completas em cartões perfurados e utilizar o console apenas para modificações de configuração e de parâmetros durante a sessão de simulação.

## 2 - PREPARAÇÃO DO DECK DE DADOS

O primeiro cartão do deck é um cartão "STARTE" cuja forma é:

```
cc  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
-   & S T , F O R T R A  ,  A  B  =  F  O  R  T  R  A  +  2

cc  23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
-   ,  '  '  S  T  A  R  T  E  :  '  '  ,  '  '  0  1  0  2

cc  42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
-   0  2  5  5  0  3  0  4  0  4  5  6  0  0  '  '  ;
```

Tal cartão endereça os seguintes números lógicos a periféricos do sistema:

1 ao teletipo do console (entrada e saida)

2 à leitora de cartões perfurados (entrada)

3 à impressora rápida (saida)

4 à perfuradora de cartões (saida)

e comanda o início da execução do programa objeto gerado no computador.

Em seguida devem vir os cartões com as especificações de configuração, caso forem usados; seguidos de um cartão em branco.

Logo a seguir devem vir os cartões com as especificações de parâmetros, se forem usados, também seguidos de um cartão em branco.

Finalmente vem os cartões com especificações de geradores de função, se usados, seguidos de vários cartões em branco.

*2.1 - Especificações de configuração*

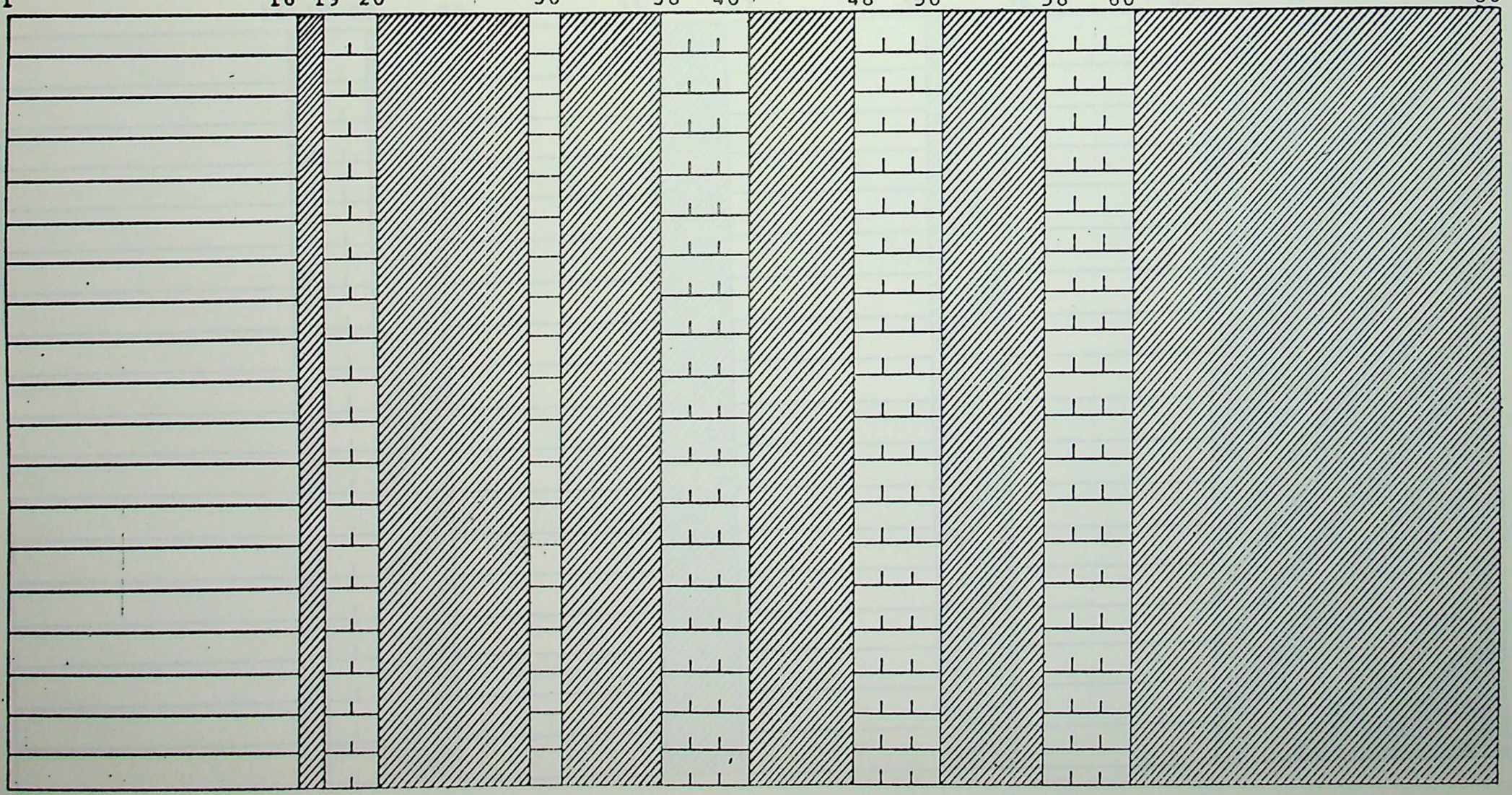
Nesta denominação estão compreendidas as informações de formato e conteúdo das declarações de configuração para o P.S.S.C.

Cada declaração de configuração contém as seguintes informações:

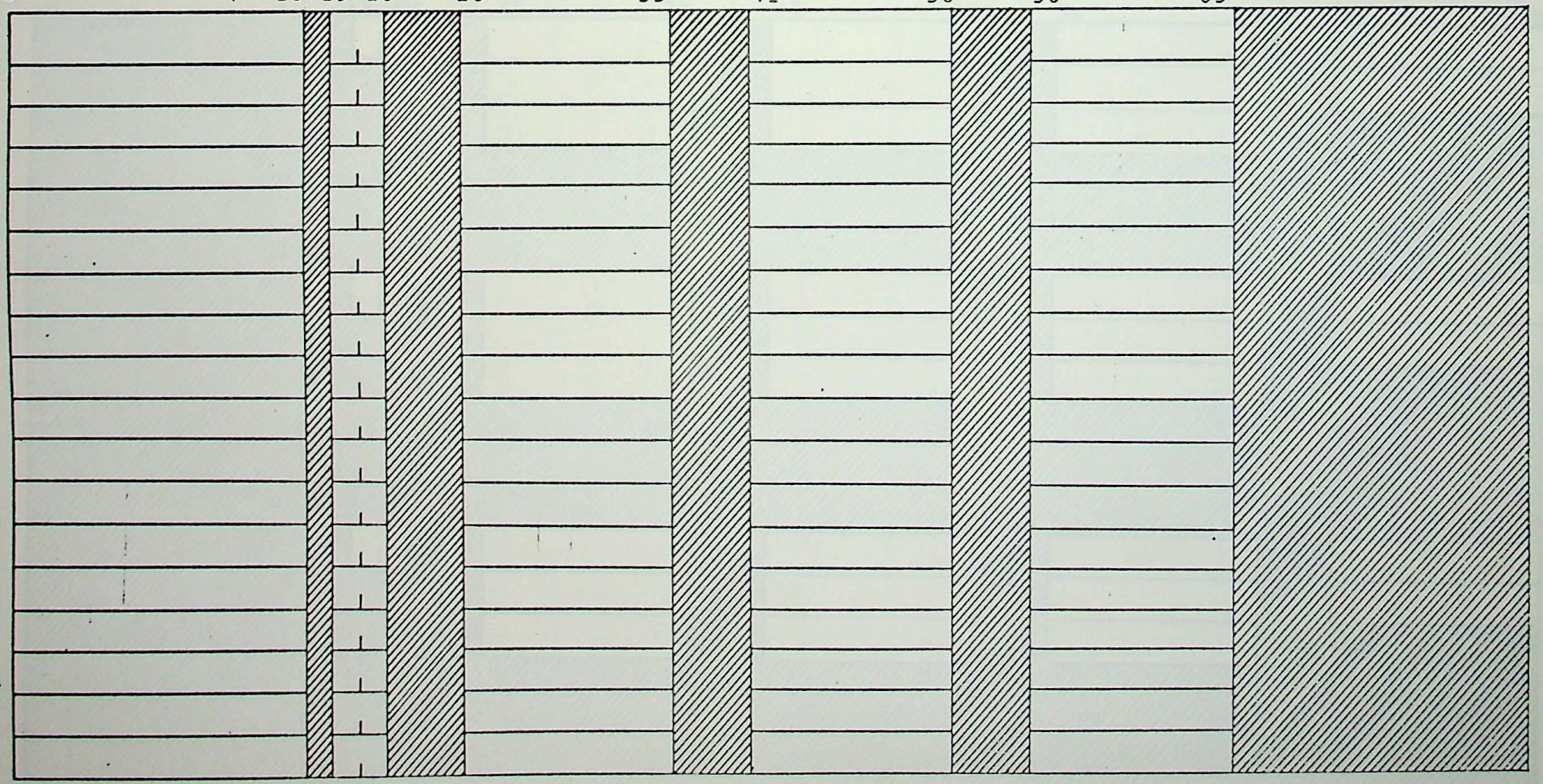
- Nome da variável de saida (opcional)
- Número do bloco
- Símbolo de linguagem do elemento
- Números dos blocos cujas saidas fornecem as variáveis de entrada para o bloco que está sendo declarado.

O nome da variável de saida é escrito nas colunas de 1 a 16 do cartão com a declaração de configuração e permite ao usuário associar um nome simbólico ou um comentário a cada declaração de configuração, facilitando a identificação da va-

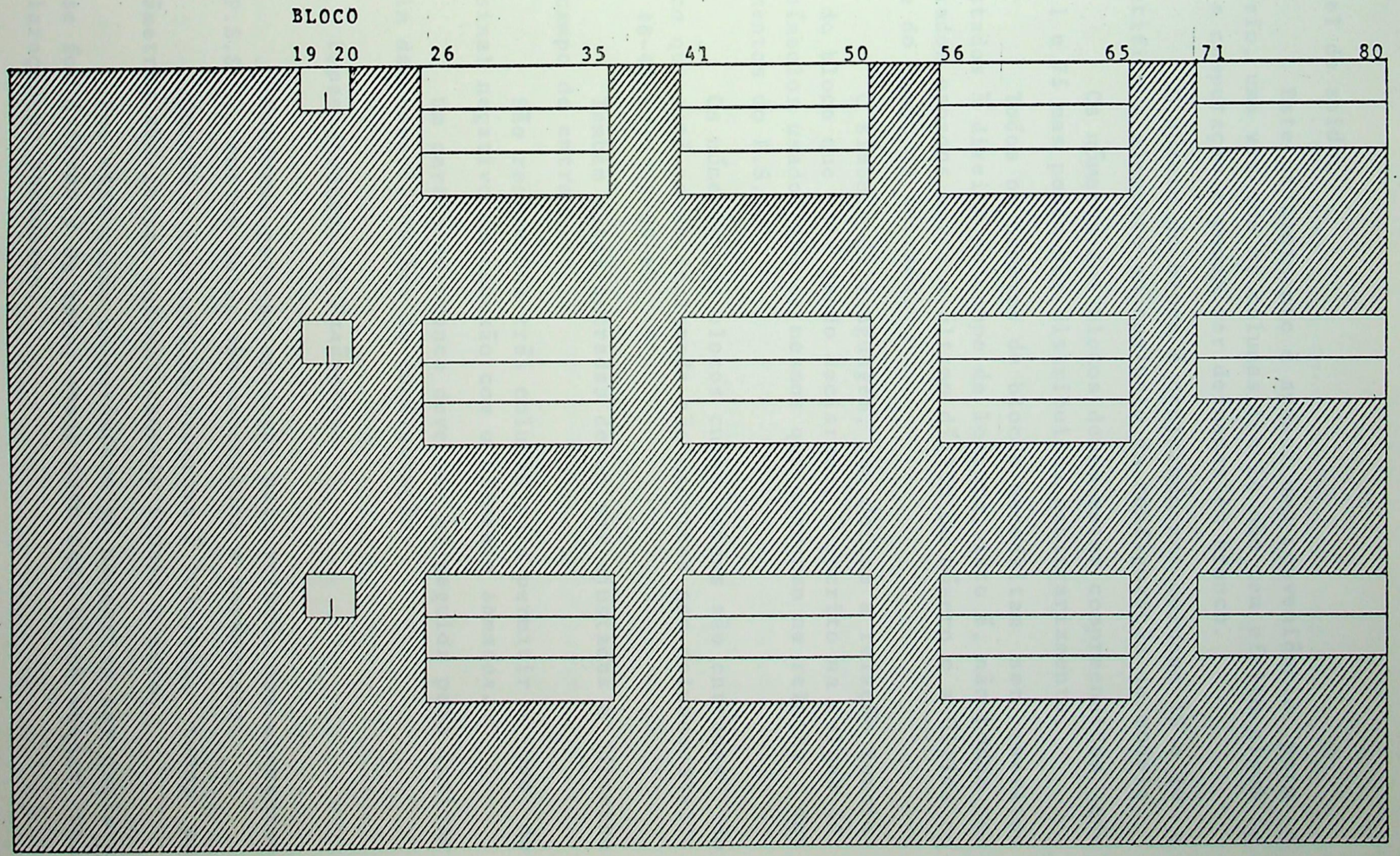
# ESPECIFICAÇÕES DE CONFIGURAÇÃO



## CONDIÇÕES INICIAIS E PARÂMETROS



GERADORES DE FUNÇÃO



riável de saida.

Este procedimento é apenas uma conveniência para o usuário, uma vez que as colunas 1 a 16 não tem efeito algum sobre a computação e podem ser deixadas em branco.

O número do bloco é declarado nas colunas 19 e 20 e identifica o elemento no conjunto da configuração do modelo.

Os números dos blocos devem estar compreendidos entre 1 e 75 mas podem ser distribuidos arbitrariamente.

Todos os números de blocos necessitam ser escritos ajustados à direita no campo de leitura, isto é, não podem ser deixados espaços entre o último dígito do número e a última coluna do campo de entrada.

O símbolo de linguagem, que define a relação funcional do bloco que está sendo declarado, é escrito na coluna 30. Os símbolos usados são os mesmos que aparecem na relação dos elementos do P.S.S.C.

Os números dos blocos cujas saidas são entradas do bloco que está sendo declarado são escritos nas colunas 38-39-40, 48-49-50 e 58-59-60.

Também estas entradas devem ser ajustadas à direita no campo de entrada.

São reservadas três colunas para permitir o uso de um sinal negativo em conexão com o elemento *somador*.

Um cartão em branco deve vir em seguida para indicar o fim das especificações de configuração.

## *2.2 - Especificações de Parâmetros*

Conforme pode-se observar pela relação dos elementos do P.S.S.C., muitos deles são associados a parâmetros.

Quando a execução do programa é iniciada, todos os parâmetros são feitos automaticamente iguais a zero.

Por essa razão um parâmetro só necessita ser declarado se for diferente de zero; nesse caso cada bloco requer uma declaração de parâmetro.

Para blocos cuja operação envolve memória, como por exemplo os *Integradores, Retardos unitários, Vãcuos e Sustentadores de ordem zero*, o primeiro parâmetro é usado para especificar a condição inicial.

O parâmetro $P_1$, portanto, para esses blocos, indica o valor da saida dos mesmos no tempo zero.

As colunas de 1 a 16 do cartão perfurado com as declarações de parâmetros, podem ser usadas para associar um nome ao conjunto de parâmetros do bloco.

Estas colunas não tem efeito na computação e podem ser deixadas em branco.

O número do bloco é perfurado nas colunas 19 e 20.

Os parâmetros $P_1$, $P_2$ e $P_3$ vem nas colunas 26-35, 41-50 e 56-65, respectivamente.

Os parâmetros são escritos como constantes reais FØR TRAN podendo ter de 1 até 7 dígitos decimais, com ponto decimal.

O conjunto dos cartões com declarações de parâmetros também pode ser arranjado em qualquer ordem.

Um cartão em branco deve vir imediatamente após o último cartão de parâmetros para indicar o fim das declarações de parâmetros.

## 2.3 - *Especificações de geradores de função*

Além dos cartões contendo as especificações de configuração e de parâmetros do modelo, são ainda necessários cartões com especificações de geradores de função, caso se esteja usando este tipo de elemento na simulação.

São necessários três cartões perfurados para cada gerador de função existente no diagrama.

A função a ser gerada é aproximada, como já dissemos por dez segmentos de reta igualmente espaçados com respeito à variável independente.

Portanto existem onze pontos de interceção dos seg -

mentos; a função é especificada através dos valores que assume em cada um dos onze pontos de interceção.

As colunas 19 e 20 do primeiro cartão de cada conjunto de três devem conter o número do bloco designado para o gerador de função.

As colunas 26 a 35 são usadas para o valor da função no ponto $X = P_2$, isto é: no início do campo de estudo da função.

As colunas 41 a 50, 56 a 65 e 71 a 80 são usadas para os valores da função nos próximos três pontos de interceção a partir de $P_2$.

No segundo cartão os valores da função nos pontos de interceção quinto, sexto, sétimo e oitavo são declarados nas colunas 26 a 35, 41 a 50, 56 a 65 e 71 a 80, respectivamente.

O nono, décimo e décimo primeiro valores de interceção são declarados nas colunas 26 a 35, 41 a 50 e 56 a 65 do terceiro cartão do conjunto, sendo que o último valor corresponde ao ponto $X = P_1$, término do campo de estudo da função.

Da mesma maneira que para as especificações anteriores, um cartão em branco deve vir em seguida às especificações de geradores de função, para indicar o seu término.

Um deck de cartões de dados como o descrito acima é suficiente para a primeira corrida.

Em caso de corridas subsequentes o programa sempre procura obter dados de cartões antes de permitir a entrada pelo console.

Caso o usuário não saiba previamente quais as modificações que possam ser necessárias, mas deseja experimentar interativamente com o modelo, ele deve adicionar ao deck inicial uma quantidade de cartões em branco suficiente para que o programa, ao procurar dados em cartões encontre a indicação de fim de especificações e permita a entrada pelo console.

A medida que se ganha experiência com o P.S.S.C. pode desejar-se preparar previamente cartões para modificações em especificações de configuração parâmetros e geradores de

função de maneira não ordenada.

Nesse caso a colocação adequada de cartões em branco causará a correta entrada dos dados.

Por exemplo, se para uma mesma configuração, deseja-se três corridas com parâmetros diferentes em cada uma delas, o deck deverá ser:

a) especificações de configuração
b) 1 cartão em branco
c) primeiro conjunto das especificações de parâmetros
d) 1 cartão em branco
e) especificações de geradores de função (se houver)
f) 2 cartões em branco
g) segundo conjunto das especificações de parâmetros
h) 3 cartões em branco
i) terceiro conjunto das especificações de parâmetros
j) 1 cartão em branco

Entretanto é preferível, no caso de usuário com pouca experiência, entrar com todas as modificações através do teletipo do console para evitar enganos que poderiam levar a resultados errôneos.

## 3 - OPERAÇÃO DO PROGRAMA

Tendo sido organizado o deck de cartões com as declarações que modelam o sistema, a sessão de simulador pode ser iniciada.

Se o programa processador P.S.S.C. estiver residindo no disco, o mesmo deverá inicialmente ser carregado na memória do computador.

Se o processador não estiver residindo no disco, deverá ser lido e compilado antes.

A operação do programa pode ser dividido em seis fases:

- fase de iniciação
- fase de especificações de configuração
- fase das especificações de parâmetros
- fase das especificações de geradores de função

o valor 1 ou 2, dependendo de sua opção.

Se o valor fornecido a KEY11 for 1 o programa lerá os cartões com as especificações de configuração e não as imprimirá no protocolo da impressora.

Se o valor fornecido for 2, o programa a medida que for lendo os cartões de configuração, irá imprimindo as especificações no protocolo da impressora de forma a iniciar uma documentação mais completa da simulação.

Quando o último cartão com as especificações de configuração for lido a seguinte mensagem será enviada pelo teletipo:

ENTRE COM O VALOR DE KEY1 IGUAL A 1 SE DESEJAR ADICIONAR OU MODIFICAR ESPECIFICAÇÕES DE CONFIGURAÇÃO VIA CONSOLE, CASO CONTRÁRIO ENTRE COM KEY1 IGUAL A 2.

KEY1 = ( )

Se o usuário não preparou um deck de cartões com especificações de configuração anteriormente ou preparou apenas parte das especificações ou ainda deseja modificar as especificações lidas em cartões deve entrar com o valor 1 no espaço entre parênteses.

**Nesse caso, o teletipo imprimirá uma sucessão de parênteses delimitadores desta forma:**

( ) ( ) ( ) ( )

e o usuário deve entrar com os valores respectivamente do número do bloco, do símbolo de linguagem do bloco, e dos números dos blocos cujas saídas chegam às três entradas do bloco que está sendo declarado.

Para a entrada via teletipo do console valem as mesmas observações feitas para a entrada por cartões.

Declarado o primeiro bloco o teletipo imprime uma segunda sucessão de parênteses para o segundo bloco, que deverá

ser declarado da mesma maneira que o primeiro.

Este procedimento continua até que seja declarado o último bloco.

Após declarado o último bloco o programa imprime nova sucessão de parênteses que agora não receberão nenhum valor ou receberão o valor zero. Note-se que mesmo que nenhum número seja escrito entre os parênteses, *é necessário que o carro do teletipo percorra toda a extensão do campo de leitura.*

Com isso o programa reconhece que terminaram as especificações de configuração e passa para a fase seguinte.

Se o usuário não deseja modificar as especificações entradas por cartões, basta entrar com o valor de KEY1 igual a 2, o que fará com que o programa passe diretamente à fase seguinte.

*3.3 - Fase das especificações de parâmetros*

Esta fase é iniciada pela impressão do seguinte título através da impressora rápida:

— CONDIÇÕES INICIAIS E PARÂMETROS

Em seguida a seguinte mensagem é enviada pelo teletipo:

**ENTRE COM O VALOR DE KEY11 IGUAL A 1 SE DESEJAR SUPRIMIR A IMPRESSÃO DOS DADOS LIDOS NOS CARTÕES. CASO CONTRÁRIO, ENTRE COM KEY11 IGUAL A 2.**

KEY11 = ( )

Se o valor entrado entre parênteses for 1 os dados de condições iniciais e parâmetros lidos não serão impressos.

Se o usuário entrar com o valor 2 no espaço entre parênteses o programa irá imprimindo no protocolo da impressora, as especificações de condições iniciais e parâmetros, à medida que as lê dos cartões.

Ao ser lido o último cartão do conjunto o teletipo envia a seguinte mensagem:

ENTRE COM O VALOR DE KEY2 IGUAL A 1 SE DESEJAR ADICIONAR OU MODIFICAR CONDIÇÕES INICIAIS OU PARÂMETROS VIA CONSOLE, CASO CONTRÁRIO ENTRE COM KEY2 IGUAL A 2.

KEY2 = ( )

Se o usuário não deseja entrar ou modificar nenhuma especificação de parâmetros, deve entrar com o valor 2 no espaço entre parênteses. Dessa forma o programa irá passar diretamente à fase seguinte.

Se desejar alterar ou adicionar alguma especificação basta entrar com o valor 1.

Nesse caso o programa imprime pelo teletipo do console a seguinte sucessão de parênteses:

( ) ( ) ( ) ( )

e o usuário deverá fornecer, nos espaços entre parênteses, respectivamente, o número do bloco e os parâmetros a ele associados.

**Note-se que se um dos parâmetros for uma condição inicial, deverá ser escrito como o primeiro dos três.**

**Recebida a primeira especificação, nova sucessão de parênteses será impressa para a segunda especificação, e assim sucessivamente até a última.**

Após dar entrada a última especificação de parâmetros, nova sucessão de parênteses será impressa sendo que agora deverão ser escritos zeros ou espaços em branco como valores de entrada.

Com isso o programa reconhece o término das especificações via console e passa à fase seguinte de sua operação.

## 3.4 - Fase das especificações de geradores de função

Esta fase só existirá se nas especificações de configuração constarem um ou mais (até três) blocos geradores de função.

Se, no diagrama de blocos, não figurar este elemento, esta fase não será processada.

Ao iniciá-la o programa imprime, através da impressora rápida, o título

ESPECIFICAÇÕES DOS GERADORES DE FUNÇÃO

Logo a seguir, a seguinte mensagem é enviada pelo teletipo:

ENTRE COM O VALOR DE KEY11 IGUAL A 1 SE DESEJAR SUPRIMIR A IMPRESSÃO DOS DADOS DE GERADORES DE FUNÇÃO LIDOS NOS CARTÕES. CASO CONTRÁRIO ENTRE COM KEY11 IGUAL A 2.

KEY11 = ( )

Também neste caso, se o usuário deseja a documentação, no protocolo da impressora dos dados de geradores de função lidos nos cartões, deve entrar com o valor 2 no espaço entre parênteses.

Se o valor entrado for 1 será suprimida a impressão.

Uma vez terminada a leitura dos dados dos cartões, o teletipo envia a mensagem:

ENTRE COM O VALOR DE KEY3 IGUAL A 1 SE DESEJAR ADICIONAR OU MODIFICAR ESPECIFICAÇÕES DE GERADORES DE FUNÇÃO VIA CONSOLE. CASO CONTRÁRIO ENTRE COM KEY3 IGUAL A 2.

KEY3 = ( )

Se não há nenhuma modificação a ser feita nas especi-

ficações, o valor a ser entregue é 2.

Caso o usuário deseje alterar especificações deve entrar com o valor 1.

Nesse caso o teletipo imprime a seguinte sucessão de parênteses:

( ) ( ) ( ) ( ) ( )

Neles deverão ser fornecidos, respectivamente, o número de bloco especificado como gerador de função e os quatro primeiros valores da função nos quatro primeiros pontos de interceção.

A seguir é impressa nova sucessão de parênteses, da seguinte forma:

( ) ( ) ( ) ( )

Onde deverão ser fornecidos os próximos quatro valores da função nos próximos pontos de interceção.

Nova sucessão de parênteses é impressa a seguir:

( ) ( ) ( )

Onde são fornecidos os três últimos valores da função nos três últimos pontos de interceção.

A seguir o teletipo envia a seguinte instrução:

ESPECIFIQUE OS LIMITES SUPERIOR E INFERIOR PARA
O G.F. DO BLOCO nn

( ) PAR 1 ( ) PAR 2

O usuário deve então especificar os valores máximo e mínimo da variável de entrada do gerador de função, definindo, portanto, o intervalo de interêsse da função.

Com isso fica definida a especificação para o primeiro gerador de função.

Se houver outros (até o máximo de mais 2), o procedimento para declará-los é idêntico.

Se não houver, quando o programa imprimir a próxima sucessão de parênteses (a que contém espaço para o número do bloco), o usuário deve entrar com zeros ou espaços no campo de leitura.

Com isso o programa reconhece o término das especificações de geradores de função e passa à próxima fase.

### 3.5 - *Fase de controle do processamento*

Nesta fase, o programa, interativamente com o usuário, estabelece o controle para o processamento das especificações até então fornecidas.

Como primeira ação dentro desta fase, é oferecida ao usuário a possibilidade de obter um deck de cartões perfurados com as especificações de configuração, parâmetros e geradores de função que irão ser processados.

A seguinte mensagem é enviada pelo teletipo:

ENTRE COM O VALOR DE KEY12 IGUAL A 1, CASO DESEJE QUE SEJA PERFURADO UM DECK DE CARTÕES (COM ESPECIFICAÇÕES DE CONFIGURAÇÃO, CONDIÇÕES INICIAIS, PARÂMETROS E GERADORES DE FUNÇÕES) ATUALIZADO. SE NÃO DESEJAR, ENTRE COM KEY12 IGUAL A 2.

KEY12 = ( )

Se o valor fornecido for 1 a seguinte mensagem é enviada:

CARREGUE CARTÕES VIRGENS NA PERFURADORA E APERTE O BOTÃO DE PARTIDA.

Em seguida o programa faz uma pausa para que o usuário possa preparar os cartões para a impressão.

Quando o usuário acionar novamente o programa, será

impresso um deck de dados completo e atualizado da simulação.

Após a perfuração do deck, ou no caso do usuário não ter optado pela perfuração o programa pede que seja fornecido o valor do intervalo de integração, imprimindo a seguinte mensagem:

( ) INTERVALO DE INTEGRAÇÃO

O usuário deve fornecer, no espaço entre parênteses, o valor do intervalo de integração desejado.

O melhor valor para o intervalo de integração é aquele que permite a maior rapidez de cálculo possível sem prejudicar a precisão dos resultados.

Uma estimativa razoável para o intervalo de integração é um décimo do valor do período correspondente à maior frequência esperada.

A seguir o programa solicita ao usuário que especifique o valor do tempo total para a simulação, através da mensagem:

( ) TEMPO TOTAL

O valor do tempo total desejado para a simulação deve ser fornecido.

Nova mensagem é então impressa:

( ) INTERVALO DE IMPRESSÃO

O usuário deve agora entrar com o valor desejado do intervalo de tempo entre duas impressões sucessivas de resultados.

Especificado este valor o programa envia a seguinte mensagem:

TEMPO SAIDA ( ) SAIDA( ) SAIDA( ) SAIDA( ) SAIDA( )

O usuário deve agora especificar quais os blocos cu

jas saidas ele deseja como resultado. Até 5 saidas podem ser consultadas sendo que a variável independente é sempre o tempo.

Escolhidas as variáveis desejadas como resultado a seguinte mensagem é enviada:

ENTRE COM O VALOR DE KEY16 IGUAL A 1, CASO DESEJE INTERROMPER A CORRIDA PARA A INTRODUÇÃO DE NOVAS ESPECIFICAÇÕES. CASO DESEJE CONTINUAR ENTRE COM O VALOR DE KEY16 IGUAL A 2.

KEY16 = ( )

Se alguma especificação ainda necessita ser modificada o usuário poderá fazê-lo antes que o programa entre na fase de cálculos.

Se for esta a opção escolhida o programa retornará à fase das especificações de configuração para refazer a montagem do diagrama de blocos.

Caso não haja nenhuma modificação a fazer, o programa passa diretamente à próxima fase.

*3.6 - Fase de cálculos e saida de resultados*

Nesta fase toda a configuração do diagrama é processada sequencialmente a cada intervalo de integração produzindo então a solução para o modelo apresentado.

O valor das variáveis especificadas como saida é apresentado como resultado a cada intervalo de impressão.

O programa continuará nesta fase até que seja atingido o tempo total ou que a corrida seja terminada pelo elemento finalizador.

Terminada a corrida o teletipo envia a seguinte mensagem:

ENTRE COM KEY1 IGUAL A 1 SE DESEJAR UMA NOVA CORRIDA . CASO DESEJAR ENCERRAR A SIMULAÇÃO ENTRE COM KEY1 IGUAL A 2.

KEY1 = ( )

Se o valor fornecido for 2 o programa é encerrado.

Se for 1, o programa retorna à fase das especificações de configuração e repete todo o procedimento descrito.

## 4 - MENSAGENS DE ERRO DO PROGRAMA

Cada vez que o usuário comete um erro em entradas de dados, o programa o deteta e envia, pelo teletipo do console, uma mensagem apontando o erro e permitindo que o mesmo seja corrigido.

As mensagens de erro podem ser separadas de acordo com as fases do programa:

### *4.1 - Fase das especificações de configuração*

Nesta fase temos as seguintes mensagens:

A ESPECIFICAÇÃO PARA O BLOCO nn ESTÁ ERRADA

Esta mensagem será enviada toda vez que o número do bloco apontado, ou o número de um dos blocos cujas saidas são entradas do bloco apontado, não estiver compreendido entre 1 e 75.

**Em seguida a ela o programa imprime os cinco pares de parênteses descritos em 3.2 para permitir ao usuário entrar, pelo teletipo, com as especificação correta para aquele bloco. Note-se que esta mensagem é enviada quer o erro tenha ocorrido numa entrada por cartão, quer numa entrada por teletipo.**

SÍMBOLO NÃO PERMITIDO ESPECIFICADO PARA O BLOCO nn

Se o símbolo de linguagem especificado para o bloco nn não for um dos que constam na relação anterior, esta mensagem será enviada.

Os mesmos cinco pares de parênteses serão impressos em seguida, para permitir a correção do erro.

ESPECIFICAÇÃO PRÉVIA DESTRUIDA

Esta mensagem é enviada toda vez que a especificação de configuração para um bloco previamente definido for modificada, ou simplesmente repetida.

O BLOCO nn, UMA DAS ENTRADAS DO BLOCO mm, NÃO FOI ESPECIFICADO

Esta mensagem indica que, nas especificações de configuração, foi definido como entrada para o bloco mm, um bloco nn que não consta do diagrama que modela o sistema.

Após esta mensagem o programa volta ao ponto das especificações de configuração para que a especificação do bloco mm possa ser corrigida ou a especificação do bloco nn possa ser adicionada à configuração.

O PSSC REQUER, PELO MENOS, 1 INTEGRADOR

Para que o programa possa ser processado pelo um dos blocos da configuração deve ser um integrador.

Se nenhum dos blocos foi definido como integrador, esta mensagem é impressa e o programa retorna ao ponto das especificações de configuração para que pelo menos 1 integrador seja incluido.

O MÁXIMO DE 25 RETARDOS UNITÁRIOS FOI ULTRAPASSADO

O número máximo de retardos unitários permitido em uma configuração é 25.

A mensagem é enviada sempre que este máximo for ultrapassado, Em seguida o programa retorna às especificações de configuração para que o erro possa ser corrigido.

O MÁXIMO DE 3 GERADORES DE FUNÇÃO FOI ULTRAPASSADO

Foram especificados mais de três geradores de função o que não é permitido.

Após a mensagem o programa volta às especificações de configuração para o erro ser corrigido.

O MÁXIMO DE 25 INTEGRADORES FOI ULTRAPASSADO

Também para os integradores, o número máximo permitido em uma configuração é 25.

A mensagem é enviada sempre que este máximo for ultrapassado.

Em seguida o programa volta às especificações de configuração para permitir a correção do erro.

ERRO NA ORDENAÇÃO - BLOCO nn

A mensagem indica que o algoritmo de ordenação não conseguiu encontrar uma sequência de computação que forneça novos valores para todas as entradas do bloco nn, resultando um erro de ordenação.

O programa retorna ao ponto das especificações de configuração para que o erro possa ser corrigido.

Conforme já foi apontado, a causa mais comum para este erro é a existência de um "loop" algébrico, isto é uma malha do diagrama de blocos onde não existe nenhum elemento com memória, tal como um integrador.

**4.2 - Fase das especificações de parâmetros**

**Nesta fase podem ser enviadas as seguintes mensagens:**

A ESPECIFICAÇÃO ACIMA CONTÉM UM NÚMERO DE BLOCO INVÁLIDO

Caso uma especificação de condição inicial ou de parâmetros tenha sido feita para um bloco cujo número não está compreendido entre 1 e 75, o programa imprime a especificação errada e, logo em seguida, esta mensagem.

A seguir são impressos os quatro pares de parênteses descritos em 3.3 para que a especificação correta possa ser en

trada pelo console.

Esta mensagem é enviada tanto no caso da especificação incorreta ter ocorrido numa entrada por cartões como numa entrada pelo teletipo.

ESPECIFICAÇÃO DE PARÂMETRO IMPRÓPRIA PARA O ELEMENTO

Esta mensagem ocorre toda vez que uma especificação de parâmetro for incompatível com o tipo do elemento previamente especificado para o bloco em questão.

Por exemplo, seria incompatível especificar parâmetros para o bloco 36 se este bloco tivesse previamente sido definido como um *multiplicador*.

Da mesma forma, seria incompatível especificar o parâmetro $P_2$ ou $P_3$ (ou os dois) para o bloco 27, se este bloco, anteriormente, tivesse sido definido como *gerador de constante*.

Em seguida à mensagem o programa imprime os quatro pares de parênteses descritos em 3.3 para permitir a correção da especificação.

NÃO HÁ ESPECIFICAÇÃO DE CONFIGURAÇÃO CORRESPONDENTE

Esta mensagem é impressa toda vez que uma especificação de parâmetros for feita para um bloco que não foi declarado nas especificações de configuração.

Aqui também, após a mensagem, o programa permite a correção da especificação, via console.

*4.3 - Fase das especificações de geradores de função*

Nesta fase apenas uma mensagem de erro é enviada:

O BLOCO nn NÃO FOI ESPECIFICADO COMO GERADOR DE FUNÇÃO

Caso o número do bloco especificado não conste das declarações de configuração como gerador de função, esta mensagem é impressa e seguida pela primeira linha da especificação

errada (a não ser que esta linha tenha acabado de ser impressa)

O conjunto de parênteses descritos em 3.4 é impresso a seguir para que a especificação errada possa ser corrigida.

*4.4 - Fase de controle*

Na fase de controle podem ocorrer as seguintes mensagens:

O INTERVALO DE INTEGRAÇÃO DEVE SER MAIOR QUE ZERO

( ) INTERVALO DE INTEGRAÇÃO

O intervalo de integração especificado anteriormente não era maior que zero.

Um novo valor correto deve ser fornecido no espaço entre parênteses.

O TEMPO TOTAL DEVE SER MAIOR QUE O INTERVALO DE INTEGRAÇÃO

( ) TEMPO TOTAL

O tempo total especificado para a corrida era menor que o intervalo de integração.

O novo valor corrigido deve ser entrado no espaço entre parênteses.

O INTERVALO DE IMPRESSÃO NÃO PODE SER MENOR QUE O INTERVALO DE INTEGRAÇÃO

( ) INTERVALO DE IMPRESSÃO

Foi especificado um intervalo de impressão menor que o intervalo de integração.

O novo valor corrigido deve ser entrado no espaço entre parênteses.

4.5 - Fase de cálculos

Nesta fase apenas uma mensagem pode ser enviada.

ERRO NO PROCESSAMENTO

Esta mensagem é enviada toda vez que uma corrida for terminada por motivo de ter sido detetado um erro computacional durante a mesma.

As causas mais prováveis são ou erro na especificação dos limites superior e inferior para um bloco gerador de função ou a entrada 2 de um bloco divisor tornou-se igual a zero.

No caso de ocorrer esta mensagem o programa procura iniciar outra corrida e o usuário deverá corrigir o erro.

## 5 - ELEMENTOS ESPECIAIS

Os elementos "especiais" são um grupo de cinco elementos ( com simbolo de linguagem 1, 2, 3, 4 e 5, respectivamente) cuja operação funcional não é definida no programa processador.

Estes elementos podem ser definidos e redefinidos de acordo com a necessidade de cada usuário.

Um conhecimento razoável de programação FORTRAN é necessário para desenvolver elementos especiais.

Se o usuário deseja desenvolver, por exemplo, um elemento para simular um tipo especial de atrito, ele deve programar a subrotina que realiza a simulação desejada e colocá-la em lugar de uma das subrotinas para elementos especiais existentes no processador, a qual não poderá mais ser usada para aquela simulação.

O nome para as subrotinas especiais deve ser um dos seguintes: SUB1, SUB2, SUB3, SUB4, SUB5.

Todas as variáveis são transmitidas a essas subrotinas através de declarações COMMON; não ocorrem argumentos de qualquer tipo nas declarações CALL de chamada dessas subrotinas.

Todas as variáveis necessárias ao PSSC são armazenadas em duas regiões em COMMON na memória.

Uma dessas regiões, chamada INTS, é usada para o armazenamento de variáveis *inteiras*, tais como os números de blocos.

A outra, chamada ROYAL, é usada para o armazenamento de variáveis *reais*, tais como parâmetros de elementos.

A região INTS compreende 587 posições de memória, enquanto que a região ROYAL compreende 395 posições.

No caso das subrotinas para elementos especiais os valores das variáveis que poderão ser usados são especificados conforme as instruções abaixo:

| *Variável* | *Instruções* |
|---|---|
| I | Identifica o número do bloco<br>I deve estar em equivalência com INTS(376) |
| J | Identifica o número do bloco cuja saida é a entrada 1 do bloco I<br>J é obtido pela declaração J = MTRX2(I)<br>MTRX2(1) deve estar em equivalência com INTS(76)<br>O conjunto MTRX2 deve ser dimensionado como MTRX2(75) |
| K | Identifica o número do bloco cuja saida é a entrada 2 do bloco I<br>K é obtido pela declaração: K = MTRX3(I)<br>MTRX3(1) deve estar em equivalência com INTS(151)<br>O conjunto MTRX3 deve ser dimensionado como MTRX3(75) |
| L | Identifica o número do bloco cuja saida é a entrada 3 do bloco I<br>L é obtido pela declaração: L = MTRX4(I)<br>MTRX4(1) deve estar em equivalência com INTS(226)<br>O conjunto MTRX4 deve ser dimensionado como MTRX4(75) |

| *Variável* | *Instruções* |
|---|---|
| C(J) | Contém o valor corrente da entrada 1 do bloco I |
| C(K) | Contém o valor corrente da entrada 2 do bloco I |
| C(L) | Contém o valor corrente da entrada 3 do bloco I<br>C(1) deve estar em equivalência com ROYAL(2)<br>A dimensão do conjunto C é C(76) |
| PAR1(I) | Contém o parâmetro $P_1$ do bloco I<br>PAR1(1) deve estar em equivalência com ROYAL(81)<br>A dimensão do conjunto PAR1 é PAR1(75) |
| PAR2(I) | Contém o parâmetro $P_2$ do bloco I<br>PAR2(1) deve estar em equivalência com ROYAL(156)<br>A dimensão do conjunto PAR2 é PAR2(75) |
| PAR3(I) | Contém o parâmetro $P_3$ do bloco I<br>PAR3(1) deve estar em equivalência com ROYAL(231)<br>A dimensão do conjunto PAR3 é PAR3(75) |
| C(I) | Local onde é armazenada a saida do bloco I |
| T | Variável independente tempo<br>T deve estar em equivalência com ROYAL(77) |
| DT | Intervalo de integração<br>DT deve estar em equivalência com ROYAL(78) |
| TTOT | Tempo total da corrida<br>TTOT deve estar em equivalência com ROYAL(79) |

Um conjunto de cartões DIMENSION, COMMON e EQUIVALENCE completo seria:

```
      REAL    ROYAL(395)
      INTEGER*2    INTS(587)
      DIMENSION    C(76), MTRX2(75), MTRX3(75), MTRX4(75)
      DIMENSION    PAR1(75), PAR2(75), PAR3(75)
      COMMON    ROYAL, INTS
      EQUIVALENCE    (ROYAL(2), C(1) )
      EQUIVALENCE    (ROYAL(81), PAR1(1) )
```

```
      EQUIVALENCE   (ROYAL(156), PAR2(1) )
      EQUIVALENCE   (ROYAL(231), PAR3(1) )
      EQUIVALENCE   (INTS(76), MTRX2(1) )
      EQUIVALENCE   (INTS(151), MTRX3(1) )
      EQUIVALENCE   (INTS(226), MTRX4(1))
      EQUIVALENCE   (INTS(376),   I )
```

O usuário deve tomar cuidado especial quando for desenvolver subrotinas para elementos especiais que envolvam "memória" ou armazenagem de dados gerados.

Em geral as regiões da memória necessárias para a armazenagem devem ser definidas *na subrotina* desenvolvida, uma vez que não existe nenhuma região em COMMON prevista para este uso.

Um elemento como este pode ser usado *somente uma vez* em cada configuração.

Se o elemento fosse usado em vários blocos, os dados armazenados pelo processamento de um dos blocos seriam destruidos durante o processamento dos seguintes.

Caso vários blocos deste tipo sejam necessários, o usuário deve definir vários elementos especiais usando a mesma subrotina, porém para cada elemento o nome da subrotina deve ser diferente.

Como exceção a esta regra, caso o elemento requeira no máximo três locais de armazenamento, pode-se usar os parâmetros $P_1$, $P_2$ e $P_3$ para o armazenamento de dados pertinentes a um bloco.

Isto é possível porque um conjunto de três localizações é reservado na cadeia de parâmetros para cada bloco.

Assim o mesmo elemento especial pode ser usado para vários blocos na configuração.

Os elementos especiais atualmente disponíveis no processador são:

| Número | Operação | Subrotina |
|---|---|---|
| 1 | função SENO | SUB1 |
| 2 | função COSSENO | SUB2 |
| 3 | função EXPONENCIAL | SUB3 |
| 4 | função ARCOTANGENTE | SUB4 |
| 5 | função LOGARITMICA | SUB5 |

# C A P Í T U L O  V

## O PROCESSADOR PSSC

# O PROCESSADOR PSSC

Neste capítulo apresentamos o processador para o PSSC, tal como foi adaptado ao computador TELEFUNKEN TR-86.

Do diagrama original foram adaptados, além do programa principal, treze subrotinas.

Não foram adaptadas três subrotinas, as de nome CSM9, CSM9A e CSM9B, por se tratarem de subrotinas para o comando e a execução da saida de resultados em forma gráfica, sendo que a nossa instalação não dispõe de PLOTTER.

Na forma em que está apresentado, é um programa de características grandemente conversacionais que permitem a quaisquer usuários, mesmo não familiarizados com técnicas de simulação, dele se utilizarem com grande facilidade.

Toda a interação programa-usuário é feita pelo teletipo do console, enquanto que os resultados são apresentados pela impressora, o que confere ao programa uma maior rapidez de processamento.

O requerimento, em termos de memória, é de cerca de 17 K palavras.

```
TR 86 FORTRAN COMPILER NV 08 (70)   FOKA.
C         PROGRAMA SIMULADOR DE SISTEMAS CONTINUOS
C         JULIO CESAR TIBURCIO   EFEI   1972
C         PROGRAMA PRINCIPAL PARA O PSSC
C
C
      REAL   ROYAL(395)
      INTEGER*2   INTS(587), TEST1, TEST3, TEST4, TEST7, KEY1,
     1KEY2, KEY3, KEY4, KEY12, KEY13, KEY14, KEY15, KEY16, IARG
      DIMENSION  C(76)
      DIMENSION  TOMP(76)
      COMMON  ROYAL, INTS
      EQUIVALENCE   ( INTS(380),  KEY1  )  ,  ( ROYAL(  2),  C(1)  )
      EQUIVALENCE   ( INTS(381),  KEY2   )
      EQUIVALENCE   ( INTS(382),  KEY3   )
      EQUIVALENCE   ( INTS(383),  KEY4   )
      EQUIVALENCE   ( INTS(391),  KEY12  )
      EQUIVALENCE   ( INTS(392),  KEY13  )
      EQUIVALENCE   ( INTS(393),  KEY14  )
      EQUIVALENCE   ( INTS(394),  KEY15  )
      EQUIVALENCE   ( INTS(395),  KEY16  )
      EQUIVALENCE   ( INTS(525),  TEST1  )
      EQUIVALENCE   ( INTS(527),  TEST3  )
      EQUIVALENCE   ( INTS(528),  TEST4  )
      EQUIVALENCE   ( INTS(531),  TEST7  )
      EQUIVALENCE   ( INTS(587),  IARG   )
C        SUBROTINA DE INICIALIZACAO
      CALL  CSM0
C        FASE DAS ESPECIFICACOES DE CONFIGURACAO.
C        O PROGRAMA NAO PROCESSARA ESTA FASE ATE QUE O TESTE DE
C        ORDENACAO SEJA BEM SUCEDIDO, QUANDO ENTAO A CHAVE TEST1 SERA
C      - FEITA IGUAL A 2.
C
   10 GO TO (12, 11), TEST1
   11 GO TO  (12,100), KEY1
C        O PROGRAMA AGORA LE AS ESPECIFICACOES DE CONFIGURACAO.
   12 CALL CSM1
C        PREPARACAO PARA A ORDENACAO
      CALL CSM2
C        TEST1 = 1 SE A PRE-ORDENACAO INDICA ERRO.
C        TEST1 = 2 SE A PRE-ORDENACAO NAO INDICA ERRO.
      GO TO (12, 13), TEST1
C        ORDENACAO DAS ESPECIFICACOES.
   13 CALL CSM3
C        TESTE PARA VERIFICAR SE A ORDENACAO FOI BEM SUCEDIDA.
C        TEST1 = 1 SE A ORDENACAO NAO FOI BEM SUCEDIDA.
C        TEST1 = 2 SE A ORDENACAO FOI BEM SUCEDIDA.
      GO TO (12, 100), TEST1
  100 CONTINUE
C        FASE DE MONTAGEM DO SISTEMA.
C
C        ENTRADA DE PARAMETROS E CONDICOES INICIAIS.
      GO TO (110, 109), TEST3
  109 GO TO (110, 115), KEY2
  110 CALL CSM4
  115 CONTINUE
C        MONTAGEM DOS GERADORES DE FUNCOES.
      GO TO (121, 118), TEST4
  118 GO TO (120, 119), TEST3
  119 GO TO (120,121), KEY3
  120 CALL CSM5
```

SEITE 2

```
  121 CONTINUE
C           A CHAVE TEST3 E FEITA IGUAL A DOIS PARA INDICAR FIM DAS
C           ESPECIFICACOES INICIAIS DE CONFIGURACAO , PARAMETROS E
C           GERADORES DE FUNCAO.
        TEST3 = 2
C           PONTO DE INTERRUPCAO.
        WRITE (1,122)
  122 FORMAT (/3X,'ENTRE COM O VALOR DE KEY16 IGUAL A 1, CASO DESEJE INT
     1ERROMPER'/3X,'A CORRIDA PARA A INTRODUCAO DE NOVAS ESPECIFICACOES.
     2 CASO'/3X,'DESEJE CONTINUAR, ENTRE COM O VALOR DE KEY16 IGUAL A 2.
     3'//3X,'KEY16 = ( )')
        READ (1, 123) KEY16
  123 FORMAT (11X,I1)
        GO TO (225, 125), KEY16
  125 CONTINUE
C           CONSULTA SE O USUARIO DESEJA A PERFURACAO DE DECK ATUALIZADO
C           DE CARTOES.
        GO TO (127, 126), KEY12
  126 WRITE (1,300)
  300 FORMAT (/3X,'ENTRE COM O VALOR DE KEY12 IGUAL A 1, CASO DESEJE QUE
     1SEJA'/3X,'PERFURADO UM DECK DE CARTOES (COM ESPECIFICACOES DE CONF
     2IGU-'/3X,'RACAO, CONDICOES INICIAIS, PARAMETROS E GERADORES DE FUN
     3COES)'/3X,'ATUALIZADO. SE NAO DESEJAR, ENTRE COM KEY12 IGUAL A 2.'
     4//3X,'KEY12 = ( )')
        READ (1,123) KEY12
        GO TO (127, 128), KEY12
  127 CALL CSM6
C           PONTO DE INTERRUPCAO.
        WRITE (1,122)
        READ (1,123) KEY16
        GO TO (225, 128), KEY16
  128 CONTINUE
C           INFORMACOES SOBRE A VARIAVEL TEMPO.
        GO TO (130, 129), TEST7
C           TEST1 = 1 ATE QUE CSM7 SEJA CHAMADA PELA PRIMEIRA VEZ.
C           TEST7 = 2 DEPOIS QUE CSM7 FOI CHAMADA PELA PRIMEIRA VEZ.
  129 GO TO (130, 135), KEY4
  130 CALL CSM7
C           PONTO DE INTERRUPCAO.
        WRITE (1,122)
        READ (1,123) KEY16
        GO TO (225, 135), KEY16
  135 CONTINUE
C           ESPECIFICACOES DE SAIDA.
        IARG = 1
  170 CALL CSM8
C           PONTO DE INTERRUPCAO.
        WRITE (1, 122)
        READ (1, 123) KEY16
        GO TO (225, 200), KEY16
  200 CONTINUE
C
        GO TO (210, 220), KEY15
  210 CONTINUE
C           READ (1'1)  (C(N), N=1,76 )
        DO 251 N = 1, 76
  251 C(N) = TOMP(N)
C           FASE DE CALCULOS.
  220 CALL CSM10
C           CHAMADA DE SUBROTINA DE INTERRUPCAO PARA NOVA DISPOSICAO DAS
C           CHAVES.
```

SEITE 9

```
  225 CALL CSM12
      GO TO (230, 240), KEY13
  230 CALL CSM13
      GO TO 225
  240 CONTINUE
      GO TO (250, 10 ,9999), KEY14
C        9999 E UM ENDERECO FICTICIO QUE NUNCA SERA ATINGIDO.
C
C
  250 CONTINUE
      DO 211 N = 1, 76
  211 TOMC(N) = C(N)
C         WRITE (1,1)  (C(N), N=1, 76)
      GO TO 10
 9999 CONTINUE
      STOP
      END
```

```
C                    SEITE   4
C
C
C       SUBROTINA  CSMO
C       A FUNCAO DESTA SUBROTINA E PREPARAR OS DADOS E VARIAVEIS DE
C       CONTROLE PARA O INICIO DA SIMULACAO, BEM COMO FORNECER MENSA-
C       GENS DE INSTRUCAO AOS USUARIOS NAO FAMILIARIZADOS COM ESTE PRO-
C       GRAMA.
C
C
      SUBROUTINE  CSMO
C
      REAL    ROYAL(395)
      INTEGER*2    INTS(587), TYPE(40), TEST1, TEST2, TEST3, TEST4,
     1TEST5, TEST6, TEST7, TEST8, TEST9, KEY1, KEY2, KEY3, KEY4, KEY5,
     2KEY6, KEY7, KEY8, KEY9, KEY10, KEY11, KEY12, KEY13, KEY14, KEY15,
     3KEY16
      COMMON  ROYAL, INTS
C
      EQUIVALENCE   (INTS(380), KEY1    )
      EQUIVALENCE   (INTS(381), KEY2    )
      EQUIVALENCE   (INTS(382), KEY3    )
      EQUIVALENCE   (INTS(383), KEY4    )
      EQUIVALENCE   (INTS(384), KEY5    )
      EQUIVALENCE   (INTS(385), KEY6    )
      EQUIVALENCE   (INTS(386), KEY7    )
      EQUIVALENCE   (INTS(387), KEY8    )
      EQUIVALENCE   (INTS(388), KEY9    )
      EQUIVALENCE   (INTS(389), KEY10   )
      EQUIVALENCE   (INTS(390), KEY11   )
      EQUIVALENCE   (INTS(391), KEY12   )
      EQUIVALENCE   (INTS(392), KEY13   )
      EQUIVALENCE   (INTS(393), KEY14   )
      EQUIVALENCE   (INTS(394), KEY15   )
      EQUIVALENCE   (INTS(525), TEST1   )
      EQUIVALENCE   (INTS(526), TEST2   )
      EQUIVALENCE   (INTS(527), TEST3   )
      EQUIVALENCE   (INTS(528), TEST4   )
      EQUIVALENCE   (INTS(529), TEST5   )
      EQUIVALENCE   (INTS(530), TEST6   )
      EQUIVALENCE   (INTS(531), TEST7   )
      EQUIVALENCE   (INTS(532), TEST8   )
      EQUIVALENCE   (INTS(533), TEST9   )
      EQUIVALENCE   (INTS(546), TYPE(1))
C
      WRITE (1,22)
   22 FORMAT (1H1,10X,'PROGRAMA SIMULADOR DE SISTEMAS CONTINUOS',///3X,
     1'UM PROGRAMA SIMULADOR ANALOGO DIGITAL ADAPTADO PARA O TR 86'//)
      DO 4 N=1,587
    4 INTS(N) = 0
      DO 5 N=1,395
    5 ROYAL(N) = 0.0
    6 KEY1 = 2
      KEY2 = 2
      KEY3 = 2
      KEY4 = 2
      KEY5 = 2
      KEY6 = 2
      KEY7 = 2
      KEY8 = 2
```

SEITE 5

```
      KEY9 = 2
      KEY10 = 2
      KEY11 = 2
      KEY12 = 2
      KEY13 = 2
      KEY14 = 2
      KEY15 = 2
      KEY16 = 2
    7 TEST1 = 1
      TEST2 = 1
      TEST3 = 1
      TEST4 = 1
      TEST5 = 1
      TEST6 = 1
      TEST7 = 1
      TEST8 = 1
      TEST9 = 1
C        MONTAGEM DO CONJUNTO TYPE (REFERE-SE AOS TIPOS DOS BLOCOS).
      TYPE(02) = 45098
      TYPE(04) = 53290
      TYPE(06) = 61482
      TYPE(07) = 65578
      TYPE(08) = 69674
      TYPE(09) = 73770
      TYPE(10) = 77866
      TYPE(11) = 81962
      TYPE(12) = 86058
      TYPE(13) = 90154
      TYPE(14) = 94250
      TYPE(15) = 98346
      TYPE(16) = 102442
      TYPE(17) = 106538
      TYPE(18) = 110634
      TYPE(20) = 118826
      TYPE(21) = 122922
      TYPE(22) = 127018
      TYPE(23) = 131114
      TYPE(24) = 135210
      TYPE(25) = 139306
      TYPE(26) = 143402
      TYPE(27) = 245802
      TYPE(28) = 180266
      TYPE(29) = 208938
      TYPE(30) = 217130
      TYPE(31) = 4138
      TYPE(32) = 8234
      TYPE(33) = 12330
      TYPE(34) = 16426
      TYPE(35) = 20522
      TYPE(40) = 172074
C
   50 CONTINUE
      RETURN
      END
```

SEITE 6

```
C       SUBROTINA CSM1.
C       ESPECIFICACOES DE CONFIGURACAO.
C
      SUBROUTINE CSM1
C
C
      REAL  ROYAL(395)
      INTEGER*2   INTS(587), TYPE(40), TEST1, TEST2, MTRX(75,5), SY(4),
     1I, J, K, L, KEY1, KEY11, ITYPE
      COMMON   ROYAL,  INTS
      EQUIVALENCE   ( INTS(  1),  MTRX(1,1)  )
      EQUIVALENCE   ( INTS(376),  I          )
      EQUIVALENCE   ( INTS(377),   J         )
      EQUIVALENCE   ( INTS(378),   K         )
      EQUIVALENCE   ( INTS(379),   L         )
      EQUIVALENCE   ( INTS(380),   KEY1      )
      EQUIVALENCE   ( INTS(390),   KEY11     )
      EQUIVALENCE   ( INTS(525),   TEST1     )
      EQUIVALENCE   ( INTS(526),   TEST2     )
      EQUIVALENCE   ( INTS(546),   TYPE(1)   )
C
      GO TO (5, 70), TEST2
C       TEST2 = 1 PARA ENTRADA POR CARTAO.
C       TEST2 = 2 PARA ENTRADA PELO CONSOLE.
    5 WRITE(1,201)
C
C       ENTRADA POR CARTOES.
C
      WRITE (1,300)
  300 FORMAT (/3X,'ENTRE COM O VALOR DE KEY11 IGUAL A 1 SE DESEJAR SUPRI
     1MIR A'/3X,'IMPRESSAO DOS DADOS LIDOS NOS CARTOES, CASO CONTRARIO,
     2ENTRE'/3X,'COM KEY11 IGUAL A 2.'//3X,'KEY11 = ( )')
      READ (1,301) KEY11
  301 FORMAT (11X,I1)
      GO TO (10,15), KEY11
   15 WRITE (1,209)
   10 READ (2,202) SY(1), SY(2), SY(3), SY(4), I, ITYPE, J, K, L
      IF (I)  150, 50, 100
   20 IF(KEY11.EQ.1) GO TO 35
C
C       IMPRESSAO DE CARTOES.
C
      WRITE (1,208)  (SY(N), N=1,4), I, ITYPE, J, K, L
   35 CONTINUE
      GO TO 10
C
C       SECAO DE TESTE DE ENTRADA PELO CONSOLE.
C
   50 CONTINUE
      TEST2 = 2
C
C
      WRITE (1,310)
  310 FORMAT (/3X,'ENTRE COM O VALOR DE KEY1 IGUAL A 1 SE DESEJAR ADICIO
     1NAR OU'/3X,'MODIFICAR ESPECIFICACOES DE CONFIGURACAO VIA CONSOLE,
     2CASO'/3X,'CONTRARIOENTRE COM KEY1 IGUAL A 2.'//3X,'KEY1 = ( )')
      READ (1,311) KEY1
```

SEITE 7

```
  311 FORMAT (10X,I1)
      GO TO (70,200), KEY1
C
C        SECAO DE ENTRADA PELO CONSOLE
C
   60 CONTINUE
   70 WRITE (1,203)
      READ (1,204) I, ITYPE, J, K, L
      IF (I)   150, 75, 100
   75 GO TO ( 10,60), TEST2
C
C        DECODIFICACAO E TESTE DAS ESPECIFICACOES DE CONFIGURACAO.
C
  100 CONTINUE
      IF  ( I - 75 )  111,111, 150
      J1 = J
  111 IF  (IABS(J1) - 76 )  112, 112, 150
      K1 = K
  112 IF  ( IABS(K1) - 76 )  113, 113, 150
      L1 = L
  113 IF  ( IABS(L1) - 76 )  120, 120, 150
C
C        O PROGRAMA CONTA COM 39 TIPOS DE ELEMENTOS.
C        O ELEMENTO TIPO 40 CORRESPONDE A ESPACO.
  120 DO 125  ITEST =1, 40
      IF (ITYPE.EQ.TYPE(ITEST) ) GO TO 130
  125 CONTINUE
      GO TO 160
C
C        SE O TIPO E ESPACO, IGNORE A DECLARACAO PREVIA PARA O BLOCO.
  130 IF  ( ITEST - 40 ) 140, 135, 135
  135 MTRX (I,1) = 0
      MTRX (I,5) = 0
      WRITE (1,207)
      GO TO 149
C
C        INTERPRETACAO DO SIMBOLO & COMO SIMBOLO DE SOMA.
  140 IF ( ITEST-30) 142, 141, 142
  141   ITEST = 27
        ITYPE = TYPE(27)
  142 CONTINUE
C
C        NUMEROS DE BLOCOS NEGATIVOS PERMITIDOS PARA SOMADORES SOMENTE.
      IF ( ITEST - 27 )  143, 146, 143
  143 IF ( J )  150, 144, 144
  144 IF ( K )  150, 145, 145
  145 IF ( L )  150, 146, 146
C
C        ARMAZENAGEM DA CONFIGURACAO.
  146 IF ( MTRX(I,1) )  148, 148, 147
  147 WRITE (1,207)
  148 MTRX(I,1) = ITEST
      MTRX(I,2) = J
      MTRX(I,3) = K
      MTRX(I,4) = L
  149 GO TO (20,60) , TEST2
C
C        SECAO DE DETECAO DE ERROS.
  150 WRITE (1,205) I
```

SEITE 8

```
  151 GO TO (155,70), TEST2
  155 GO TO (70, 156), KEY11
  156 WRITE (1,202)   (SY(N),N=1,4),I, ITYPE, J, K, L
      GO TO 70
C
C         MENSAGEM DE ERRO SE O TIPO DO ELEMENTO NAO FOR ENCONTRADO.
C
  160 WRITE (1,206) I
      GO TO 151
C
C         SAIDA DESTA SUBROTINA.
  200 CONTINUE
      TEST1 = 1
      TEST2 = 1
      RETURN
C
  201 FORMAT (//17X,'ESPECIFICACOES DE CONFIGURACAO.')
  202 FORMAT (4A4,2X,I2,9X,A1,3(7X,I3))
  203 FORMAT (18X,4H(   ),4X,3H( ),5X,5H(    ),2(7X,5H(    )))
  204 FORMAT (18X,I2,6X,A1,7X,I3,2(9X,I3))
  205 FORMAT (/3X,'A ESPECIFICACAO PARA O BLOCO',I2,'ESTA ERRADA')
  206 FORMAT (/3X,'SIMBOLO NAO PERMITIDO ESPECIFICADO PARA O BLOCO',I3)
  207 FORMAT (/3X,'ESPECIFICACAO PREVIA DESTRUIDA')
  208 FORMAT (4A4,2X,I2,6X,A1,7X,I3,9X,I3,9X,I3)
  209 FORMAT (//2X,'DENOMINACAO   BLOCO   TIPO   ENTRADA 1   ENTRADA 2
     1 ENTRADA 3')
      END
```

SEITE 9-

```
C          SUBROTINA CSM2
C          A FUNCAO DESTA SUBROTINA E PREPARAR PARA A OPERACAO DE ORDENA-
C          CAO.
      SUBROUTINE CSM2
      REAL    ROYAL(395)
      INTEGER*2   INTS(587), TEST1, TEST2, TEST3, TEST4, TEST5, TEST6,
     1TEST7, TEST8, TEST9, DELAY(25), ORDER(76), MTRX(75,5), INTG(25),
     2NCON, NOD, NEQ, IERR, IFG
      COMMON  ROYAL ,   INTS
      EQUIVALENCE   ( INTS(  1),  MTRX(1,1)   )
      EQUIVALENCE   ( INTS(396),  INTG(1)     )
      EQUIVALENCE   ( INTS(424), DELAY(1)     )
      EQUIVALENCE   ( INTS(449), ORDER(1)     )
      EQUIVALENCE   (INTS(525),   TEST1       )
      EQUIVALENCE   (INTS(526),   TEST2       )
      EQUIVALENCE   (INTS(527),   TEST3       )
      EQUIVALENCE   (INTS(528),   TEST4       )
      EQUIVALENCE   (INTS(529),   TEST5       )
      EQUIVALENCE   (INTS(530),   TEST6       )
      EQUIVALENCE   (INTS(531),   TEST7       )
      EQUIVALENCE   (INTS(532),   TEST8       )
      EQUIVALENCE   (INTS(533),   TEST9       )
      EQUIVALENCE   ( INTS(540),  NCON        )
      EQUIVALENCE   ( INTS(541),  NOD         )
      EQUIVALENCE   ( INTS(542),  NEQ         )
      EQUIVALENCE   ( INTS(543),  NFG         )
C
C          REAJUSTAGEM DO INDICADOR DE ERRO E DOS CONTADORES.
           IERR = 2
           NOD  = 0
           NEQ = 0
           IFG = 0
           NCON = 2
        ORDER(1) = 76
C
C          TESTE PARA ELEMENTOS ESPECIAIS.
      DO    85    I=1,75
      IMTRX = MTRX(I,1)
            ITYPE = IABS( IMTRX )
      IF ( ITYPE ) 85,85,1
    1    MTRX(I,1) = ITYPE
C
C          TESTE SE O ELEMENTO E UM RETARDO UNITARIO.
      IF ( ITYPE - 21)  10, 5, 10
    5         NOD  = NOD + 1
       DELAY(NOD) = I
      GO TO 45
   10 CONTINUE
C
C          TESTE SE O ELEMENTO E UM INTEGRADOR.
      IF ( ITYPE - 9 )  20, 15, 20
   15             NEQ =NEQ + 1
          INTG(NEQ) = I
          MTRX(I,5) = NEQ
      GO TO 45
```

SEITE 10

```
   20 CONTINUE
C
C        TESTE SE O ELEMENTO E UM GERADOR DE CONSTANTE.
      IF ( ITYPE - 11)  30, 25, 30
   25  ORDER( NCON ) = I
               NCON = NCON + 1
      GO TO 50
   30 CONTINUE
C
C        TESTE SE O ELEMENTO E UM GERADOR DE FUNCAO.
      IF ( ITYPE - 6) 40, 35, 40
   35            IFG = IFG + 1
   40 CONTINUE
C
C        O NUMERO INDICADOR DO ELEMENTO E FEITO NEGATIVO ATE DEPOIS DA
C        ORDENACAO.
   45    MTRX(I,1)  = -ITYPE
   50 CONTINUE
      DO    75      M = 2,4
      IMTRX = MTRX (I,M)
            LTEST = IABS ( IMTRX )
      IF (LTEST)    75, 75, 55
   55 IF (LTEST - 76)    60, 75, 75
   60 IF ( MTRX(LTEST,1) )    75, 65, 75
C
C        IMPRESSAO DA MENSAGEM DE ERRO E AJUSTE DO INDICADOR DE ERRO.
   65 WRITE (1,203) LTEST,I
               IERR= 1
   75 CONTINUE
   85 CONTINUE
C
C        TESTE SE O NUMERO DE INTEGRADORES, RETARDOS UNITARIOS E GERA-
C        DORES DE FUNCAO NAO E EXCESSIVO.
      IF ( NEQ      ) 150, 150, 100
  100 IF (NEQ - 25)   110, 110, 155
  110 IF (NOD - 25)   120, 120, 160
  120 IF (IFG -  3)   130, 130, 170
  130       TEST4 = 2
      IF ( IFG )    140, 140, 141
  140       TEST4 = 1
                NFG = 0
      GO TO 300
  141 IF ( IFG - NFG )    143, 143, 142
  142       TEST3 = 1
C
C        TEST3 E FEITO IGUAL A 1 PARA INDICAR A ADICAO DE BLOCO GERADOR
C        DE FUNCAO.
  143           NFG= IFG
      GO TO 300
  150 WRITE (1,200)
      GO TO 190
  155 WRITE (1,204)
      GO TO 190
  160 WRITE (1,201)
      GO TO 190
  170 WRITE (1,202)
  190           IERR= 1
  300 GO TO (310,320) , IERR
C        SAIDA DEVIDA A ERRO NA PRE-ORDENACAO.
  310       TEST1 = 1
            TEST2 = 2
```

SEITE 11

```
      GO TO 350
C        PRE-ORDENACAO BEM SUCEDIDA.
  320     TEST1 = 2
  350 RETURN
  200 FORMAT (/3X,'O PSSC REQUER, PELO MENOS, 1 INTEGRADOR.')
  201 FORMAT(/3X,' O MAXIMO DE 25 RETARDOS UNITARIOS FOI ULTRAPASSADO.')
  202 FORMAT (/3X,'O MAXIMO DE 3 GERADORES DE FUNCAO FOI ULTRAPASSADO.')
  203 FORMAT (/3X,'BLOCO',I3,'UMA DAS ENTRADAS DO BLOCO',I3,'NAO FOI ESP
     1ECIFICADA')
  204 FORMAT (/3X,'O MAXIMO DE 25 INTEGRADORES FOI ULTRAPASSADO.')
      END
```

SEITE 12

```
C       SUBROTINA CSM3.
CCCC
C       SUBROTINA DE ORDENACAO.
      SUBROUTINE CSM3
      REAL ROYAL(395)
      INTEGER*2  INTS(587), TEST1, TEST2, DELAY(25), ORDER(76), INTG(25)
     1, MTRX(75,5), NLIST, NCON, NOD, NEQ, IERR, N, M
      DIMENSION   IMTRX(75,5)
      COMMON  ROYAL  ,  INTS
C
      EQUIVALENCE  ( INTS(  1),  MTRX(1,1)  )
      EQUIVALENCE  ( INTS(396),  INTG(1)    )
      EQUIVALENCE  ( INTS(424), DELAY(1)    )
      EQUIVALENCE  ( INTS(449), ORDER(1)    )
      EQUIVALENCE  ( INTS(525),  TEST1      )
      EQUIVALENCE  ( INTS(526),  TEST2      )
      EQUIVALENCE  ( INTS(534), NLIST       )
      EQUIVALENCE  ( INTS(540),  NCON       )
      EQUIVALENCE  ( INTS(541),   NOD       )
      EQUIVALENCE  ( INTS(542),   NEQ       )
C
C       REAJUSTAGEM DO INDICADOR DE ERRO.
C
            IERR = 2
      DO 10       N = NCON ,76
   10       ORDER(N) = 0
C
C       OPERACAO DE ORDENACAO.
C
            NLIST = NCON - 1
   20 DO 150     I = 1,75
      IF (MTRX(I,1) )   30, 150, 150
   30 CONTINUE
      DO  100     M = 2, 4
      IMTRX (I,M) = MTRX (I,M)
      LTEST = IABS (IMTRX(I,M) )
      IF(LTEST)  40, 100, 40
   40 CONTINUE
      IF (NOD)    70, 70, 50
   50 DO 60  N = 1, NOD
      IF (LTEST - DELAY(N))  60, 100, 60
   60 CONTINUE
   70 CONTINUE
      DO 80    N = 1,NEQ
      IF (LTEST - INTG(N)) 80, 100, 80
   80 CONTINUE
      DO 90    N = 1,NLIST
      IF (LTEST - ORDER(N) )   90, 100, 90
   90 CONTINUE
      GO TO 150
  100 CONTINUE
C
  110       NLIST = NLIST + 1
      ORDER(NLIST) = I
```

SEITE 13

```
          MTRX(I,1) = -MTRX(I,1)
      GO TO 20
  150 CONTINUE
C
C        TESTE DA ORDENACAO.
      DO 180 I= 1,75
      IF(MTRX(I,1)) 160, 180, 180
C
CC       AJUSTAGEM DO INDICADOR DE ERRO, IMPRESSAO DA MENSAGEM DE ERRO
CC       EM TIPO, EM SEGUIDA COLOCACAO DO BLOCO NA LISTA DE ORDENACAO
CC       PARA DETERMINAR SE O RESTANTE DA CONFIGURACAO ESTA CORRETA.
C
  160        IERR = 1
          MTRX(I,1) = -MTRX(I,1)
      WRITE(3,199) I
          NLIST = NLIST + 1
      ORDER(NLIST) = I
      GO TO 20
  180 CONTINUE
      GO TO (190,200), IERR
C
C        ORDENACAO MAL SUCEDIDA.
  190          TEST1 = 1
               TEST2 = 2
      GO TO 210
C
CC       ORDENACAO BEM SUCEDIDA.
C
  200          TEST1 = 2
  210 RETURN
C
  199 FORMAT (3X,'ERRO NA ORDENACAO - BLOCO',I4)
      END
```

SEITE 14

```
C
C       SUBROTINA CSM4.
C
C       ESPECIFICACOES DE CONDICOES INICIAIS E PARAMETROS.
C
C
      SUBROUTINE CSM4
C
      REAL  ROYAL(395)
      INTEGER*2  INTS(587), TEST2, MTRX1(75), I, KEY2, KEY11, ITYPE
      DIMENSION  PAR1(75), PAR2(75), PAR3(75)
      COMMON  ROYAL, INTS
      EQUIVALENCE  ( INTS(  1),  MTRX1(1)   )
      EQUIVALENCE  ( INTS(376),  I          )
      EQUIVALENCE  ( INTS(381),  KEY2       )  ,  ( ROYAL(81), PAR1(1)  )
      EQUIVALENCE  ( INTS(390),  KEY11      )  ,  ( ROYAL(156), PAR2(1) )
      EQUIVALENCE  ( INTS(526),  TEST2      )  ,  ( ROYAL(231),PAR3(1)  )
      TEST2 = 1
      WRITE (1,201)
C
C       SECAO DE ENTRADA POR CARTAO.
C
      WRITE (1,300)
  300 FORMAT (/3X,'ENTRE COM O VALOR DE KEY11 IGUAL A 1 SE DESEJAR SUPRI
     1MIR A'/3X,'IMPRESSAO DOS DADOS LIDOS NOS CARTOES. CASO CONTRARIO,
     2ENTRE'/3X,'COM KEY11 IGUAL A 2.'//3X,'KEY11 = ( )')
      READ (1,301) KEY11
  301 FORMAT (11X,I1)
      GO TO (10,15), KEY11
   15 WRITE (1,209)
   10 READ (2,206)  SY1, SY2, SY3, SY4, I, P1, P2, P3
      IF ( I)  150, 50, 100
   20 IF (KEY11.EQ.1) GO TO 35
C
C       CARTOES PERFURADOS.
C
   30 WRITE (1,202) SY1, SY2, SY3, SY4, I, P1, P2, P3
   35 CONTINUE
      GO TO 10
C
C       SECAO DE TESTE DO CONSOLE.
C
   50 CONTINUE
      TEST2 = 2
C
C       TESTE DE ENTRADA PELO CONSOLE.
C
   60 WRITE (1,310)
  310 FORMAT (/3X,'ENTRE COM O VALOR DE KEY2 IGUAL A 1 SE DESEJAR ADICIO
     1NAR OU'/3X,'MODIFICAR CONDICOES INICIAIS OU PARAMETROS VIA CONSOLE
     2. CASO'/3X,'CONTRARIO ENTRE COM KEY2 IGUAL A 2.'//3X,'KEY2 = ( )')
      READ (1,311) KEY2
  311 FORMAT (10X,I1)
      GO TO ( 70, 200), KEY2
C
C       SECAO DE ENTRADA PELO CONSOLE.
```

SEITE 15

```
C
   70 WRITE (1,203)
      READ (1,205) I, P1, P2, P3
      IF ( I ) 150, 60, 100
C
C        TESTE E ARMAZENAGEM DE PARAMETROS.
  100 IF( I - 75 )  101, 101, 150
  101       ITYPE = MTRX1(I)
      IF ( ITYPE )  111, 102, 103
  102 WRITE (1,207)
      GO TO 111
  103 GO TO (111,104,111,108,111,108,106,104,111,104,106,108,104,104,
     1    106,104,104,104,111,106,106,106,111,104,108,106,104,104,
     2     104,104,111,111,111,111,111)   , ITYPE
  104 IF ( P1 )  110, 106, 110
  106 IF ( P2 )  110, 108, 110
  108 IF ( P3 )  110, 111, 110
  110 WRITE (1,208)
  111       PAR1 (I) = P1
            PAR2(I) = P2
            PAR3(I) = P3
  145 GO TO ( 20, 60), TEST2
C
C        SECAO DE ERRO.
C
  150 GO TO (160, 170), TEST2
  160 WRITE (3,202) SY1, SY2, SY3, SY4, I, P1, P2, P3
  170 WRITE (1,204)
      GO TO 70
C
  201 FORMAT ('////15X,'CONDICOES INICIAIS E PARAMETROS')
  202 FORMAT (4A4,4X,I2,3X,F10.4,2(4X,F10.4),/)
  203 FORMAT (19X,4H(   ),1X,12H(          ),2(2X,12H(          )))
  204 FORMAT (3X,'A ESPECIFICACAO ACIMA CONTEM UM NUMERO DE BLOCO INVALI
     1DO')
  205 FORMAT (19X,I2,3X,F10.4,2(4X,F10.4))
  206 FORMAT (4A4,2X,I2,3F15.4)
  207 FORMAT (3X,'NAO HA ESPECIFICACAO DE CONFIGURACAO CORRESPONDENTE')
  208 FORMAT(3X,'ESPECIFICACAO DE PARAMETRO IMPROPRIA PARA O ELEMENTO')
  209 FORMAT (//2X,'CI/NOME PAR.   BLOCO     CI/PAR1          PAR2
     1     PAR3')
C
  200   TEST2 = 1
      RETURN
      END
```

SEITE 16

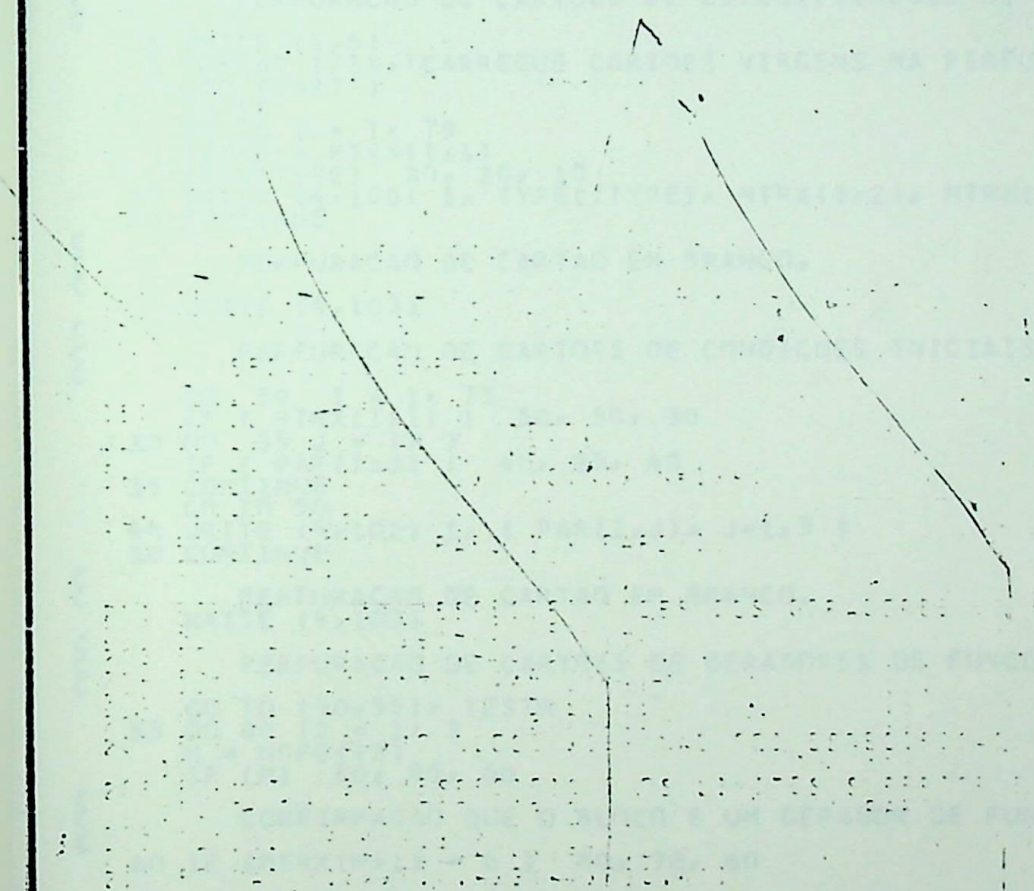
```
C         SUBROTINA CSM5.
CC        ESPECIFICACOES DE GERADORES DE FUNCOES.
C
      SUBROUTINE CSM5
C
      REAL  ROYAL(395)
      INTEGER*2  INTS(587), TEST2, MTRX(75,5), NOFG(3), KEY3, KEY11, I
      DIMENSION  F(3,11), C(76), PAR1(75), PAR2(75)
      COMMON  ROYAL, INTS
C
      EQUIVALENCE  ( INTS(  1),  MTRX(1,1)  ) ,
      EQUIVALENCE  ( INTS(382),  KEY3       ) , ( ROYAL(  2), C(1)      )
      EQUIVALENCE  ( INTS(390),  KEY11      ) , ( ROYAL( 81), PAR1(1)   )
      EQUIVALENCE  ( INTS(421),  NOFG(1)    ) , ( ROYAL(156),  PAR2(1)  )
      EQUIVALENCE  ( INTS(526),  TEST2      ) , ( ROYAL(306),  F(1,1)   )
C
      TEST2 = 1
      WRITE (1,201)
CC        SECAO DE ENTRADA POR CARTAO.
C
      WRITE (1,300)
  300 FORMAT (/3X,'ENTRE COM O VALOR DE KEY11 IGUAL A 1 SE DESEJAR SUPRI
     1MIR A'/3X,'IMPRESSAO DOS DADOS DE GERADORES DE FUNCAO LIDOS NOS CA
     2RTOES.'/3X,'CASO CONTRARIO ENTRE COM KEY11 IGUAL A 2.'//3X,'KEY11
     3= ( )')
      READ (1,301) KEY11
  301 FORMAT (11X,I1)
   10 READ (2,202) I,(C(N),N=1,4)
      IF ( I )  15, 50, 15
   15 READ (2,203) (C(N), N=5,11)
      IF( I )  150, 50, 100
CC        IMPRESSAO DOS DADOS LIDOS EM CARTOES.
C
   20 IF (KEY11.EQ.1) GO TO 35
   30 WRITE (1,210)  I, (C(N), N=1,4)
      WRITE (1,211) ( C(N), N=5,11)
   35 CONTINUE
      GO TO 10
CC        SECAO DE TESTE DO CONSOLE.
C
   50 CONTINUE
      TEST2 = 2
CC        TESTE DE ENTRADA PELO CONSOLE
   60 WRITE (1,310)
  310 FORMAT (/3X,'ENTRE COM O VALOR DE KEY3 IGUAL A 1 SE DESEJAR ADICIO
     1NAR OU'/3X,'MODIFICAR ESPECIFICACOES DE GERADORES DE FUNCAO VIA CO
     2NSOLE.'/3X,'CASO CONTRARIO ENTRE COM KEY3 IGUAL A 2.'//3X,'KEY3 =
     3( )')
```

SEITE 17

```
      READ (1,311) KEY3
  311 FORMAT (10X,I1)
      GO TO (70,200), KEY3
C
C        SECAO DE ENTRADA PELO CONSOLE.
C
   70 WRITE (1,204)
      READ (1,210) I, (C(N), N=1,4)
      IF ( I )  150, 60, 80
   80 CONTINUE
      WRITE (1,205)
      READ(1,211)  (C(N), N=5,8)
      WRITE (1,206)
      READ (1,211)  (C(N), N=9,11)
      GO TO 100
C
C        TESTE E ARMAZENAGEM DOS GERADORES DE FUNCAO.
C
  100 CONTINUE
      IF (I-75) 110, 110, 150
  110 IF (MTRX(I,1) - 6)  150, 115, 150
  115 DO 120 I2=1,3
      M = NOFG(I2)
      IF ( M )  119, 125, 119
  119 IF (MTRX(M,1) - 6)  125, 120, 125
  120 CONTINUE
      GO TO 150
  125 MTRX(I,5) = I2
      NOFG(I2) = I
  130 DO 140  M = 1, 11
  140 F(I2,M) = C(M)
C
C        TESTE DAS VARIAVEIS PAR1 E PAR2
C
  141 IF ( PAR1(I) - PAR2(I) )  190, 190, 145
  145 GO TO ( 20, 60 ),  TEST2
C
C        SECAO DE ERROS.
C
  150 WRITE (1,207)  I
      GO TO (170, 70),  TEST2
  170 GO TO (180, 70),  KEY11
  180 WRITE (1,202)  I,  (C(N), N=1,4)
      GO TO 70
  190 WRITE (1,208)  I
      READ (1,209)  PAR1(I), PAR2(I)
      GO TO 141
C
  201 FORMAT (/15X,'ESPECIFICACOES DOS GERADORES DE FUNCAO.')
  202 FORMAT (1HX,I2,4(5X,F10.4))
  203 FORMAT (25X,F10.4,3(5X,F10.4))
  204 FORMAT (/3X,4H(   ),4(2X,12H(           )))
  205 FORMAT (7X,          4(2X,12H(           )))
  206 FORMAT (7X,          3(2X,12H(           )))
  207 FORMAT (/3X,'O BLOCO',I3,'NAO FOI ESPECIFICADO COMO GERADOR DE FUN
     1CAO!')
  208 FORMAT (/3X,'ESPECIFIQUE OS LIMITES SUPERIOR E INFERIOR PARA O G.
     1F. DO'/3X,'BLOCO',I3/3X,'(          ) PAR1   (          ) PAR2')
  209 FORMAT (3X,F10.0,10X,F9.0)
  210 FORMAT (//4X,I2,4(4X,F10.4))
  211 FORMAT (10X,F10.4,3(4X,F10.4))
```

SEITE 18

```
C
  200 TEST2 = 1
      RETURN
      END
```



SEITE 19

```
C          SUBROTINA CSM6.
C
C          OPCAO PARA A PERFURACAO DE DECK DE CARTOES DE DADOS ATUALIZADOS.
C
C
      SUBROUTINE CSM6
C
      REAL    ROYAL(395)
      INTEGER*2    INTS(587), TEST4, TYPE(40), MTRX(75,5), NOFG(3), I,
     1ITYPE, N, I2, K
      DIMENSION  F(3,11), PAR(75,3)
      COMMON  ROYAL, INTS
      EQUIVALENCE  ( INTS(528),  TEST4       ), ( ROYAL( 81),  PAR(1,1))
      EQUIVALENCE  ( INTS(  1),  MTRX(1,1)   ), ( ROYAL(306),  F(1,1)   )
      EQUIVALENCE  ( INTS(421),  NOFG(1)     )
      EQUIVALENCE  ( INTS(546),  TYPE(1)     )
C
C          PERFURACAO DE CARTOES DE ESPECIFICACOES DE CONFIGURACAO.
C
    5 WRITE (1,6)
    6 FORMAT (/3X,'CARREGUE CARTOES VIRGENS NA PERFURADORA E APERTE O BO
     1TAO START')
      PAUSE
      DO 20 I = 1, 75
      ITYPE = MTRX(I,1)
      IF (ITYPE)  20, 20, 10
   10 WRITE (4,100) I, TYPE(ITYPE), MTRX(I,2), MTRX(I,3), MTRX(I,4)
   20 CONTINUE
C
C          PERFURACAO DE CARTAO EM BRANCO.
C
      WRITE (4,103)
C
C          PERFURACAO DE CARTOES DE CONDICOES INICIAIS E PARAMETROS.
C
      DO  50  I = 1, 75
      IF ( MTRX(I,1) )  50, 50, 30
   30 DO  35 J = 1, 3
      IF ( PAR(I,J) )  40, 35, 40
   35 CONTINUE
      GO TO 50
   40 WRITE (4,102) I, ( PAR(I,J), J=1,3 )
   50 CONTINUE
C
C          PERFURACAO DE CARTAO EM BRANCO.
      WRITE (4,103)
C
C          PERFURACAO DE CARTOES CE GERADORES DE FUNCOES.
C
      GO TO (90,55), TEST4
   55 DO 80 I2 = 1, 3
      N = NOFG(I2)
      IF (N)  80, 80, 60
C
C          CONFIRMACAO QUE O BLOCO E UM GERADOR DE FUNCAO.
C
   60 IF (MTRX(N,1) - 6 )  80, 70, 80
```

SEITE 20

```
   70 CONTINUE
      WRITE (4,104) N, (F(I2,K), K=1,4 )
      WRITE (4,105)  ( F(I2,K), K=5,11 )
   80 CONTINUE
C
C        PERFURACAO DE CARTAO EM BRANCO.
C
      WRITE (4,103)
C
   90 RETURN
  100 FORMAT (18X, I2, 9X, A1, 3(7X,I3) )
  102 FORMAT (18X, I2, 3F15.5 )
  103 FORMAT (72X)
  104 FORMAT (18X, I2, 4(5X,F10.5) )
  105 FORMAT (25X, F10.5, 5X, F10.5, 5X, F10.5, 5X, F10.5 )
C
      END
```

```
C
C        SUBROTINA CSM7
C
C        PEDIDO DE INFORMACOES SOBRE A VARIAVEL TEMPO.
C
C
      SUBROUTINE  CSM7
      REAL    ROYAL(395)
      INTEGER*2    INTS(587) , TEST7
      COMMON  ROYAL , INTS
      EQUIVALENCE   ( INTS(531),  TEST7 ) , ( ROYAL( 78) , DT     )
      EQUIVALENCE                           ( ROYAL( 79) , DTS2   )
      EQUIVALENCE                           ( ROYAL( 80) , TTOT   )
C
      GO TO (1,10) , TEST7
C
C        TEST7 = 1 ATE QUE CSM7 SEJA CHAMADA A PRIMEIRA VEZ.
C        TEST7 = 2 DEPOIS QUE CSM7 FOI CHAMADA A PRIMEIRA VEZ.
C
    1 TEST7 = 2
   10 WRITE (1,100)
      READ (1,101)  DT
      IF ( DT )  50, 50, 20
   20 WRITE (1,102)
      READ (1,101)  TTOT
      IF ( TTOT - DT )  60,60,30
C
  100 FORMAT (//3X,'(          ) INTERVALO DE INTEGRACAO.')
  101 FORMAT (3X,F10.0)
  102 FORMAT (//3X,'(          ) TEMPO TOTAL.')
  103 FORMAT (//3X,'O INTERVALO DE INTEGRACAO DEVE SER MAIOR QUE ZERO.')
  104 FORMAT (//3X,'O TEMPO TOTAL DEVE SER MAIOR QUE O INTERVALO DE INTE
     1GRACAO.')
  105 FORMAT (///5X,'INTERVALO DE INTEGRACAO =',F8.4)
  106 FORMAT (///5X,'TEMPO TOTAL =',F13.4)
C
C        MONTAGEM DAS EQUACOES DE INTEGRACAO E TEMPO.
C
   30 CONTINUE
      DTS2 = DT*0.5
      WRITE  (3,105)  DT
      WRITE (3,106) TTOT
      RETURN
C
C        ERRO EM DT.
C
   50 WRITE (1,103)
      GO TO 10
C
C        ERRO EM TTOT.
C
   60 WRITE (1,104)
      GO TO 20
      END
```

SEITE 22

```
C       SUBROTINA CSM10.
C
C       CONTROLA OS CALCULOS E A SAIDA.
C
      SUBROUTINE CSM10
C
      REAL    ROYAL(395)
      INTEGER*2   INTS(587), TEST5, ORDER(76), INTG(25), KEY15, KEY16,
     1NLIST, NEQ, IARG, INTNO, N, NN, IX, M, NEXT
      DIMENSION  C(76), PAR(75, 3), Y(25), DYDT(25), YK(25)
      COMMON  ROYAL , INTS
C
      EQUIVALENCE ( INTS(394), KEY15    ) , ( ROYAL(  2), C(1)      )
      EQUIVALENCE ( INTS(395), KEY16    ) , ( ROYAL( 77), T         )
      EQUIVALENCE                         , ( ROYAL( 78), DT        )
      EQUIVALENCE ( INTS(396), INTG(1)  ) , ( ROYAL( 79), DTS2      )
      EQUIVALENCE ( INTS(449), ORDER(1) ) , ( ROYAL( 80), TTOT      )
      EQUIVALENCE ( INTS(529), TEST5    ) , ( ROYAL( 81), PAR(1,1). )
      EQUIVALENCE ( INTS(534), NLIST    ) , ( ROYAL(341), Y(1)      )
      EQUIVALENCE                         , ( ROYAL(366), DYDT(1)   )
      EQUIVALENCE ( INTS(542), NEQ      ) , ( ROYAL(391), TSAMP     )
      EQUIVALENCE ( INTS(587), IARG     )
C
      GO TO (60, 1) , KEY15
C
C       MONTAGEM NORMAL.
C
    1 CONTINUE
      DO 10 NEXT = 2, NLIST
      I = ORDER(NEXT)
   10 C(I) = PAR(I, 1)
      DO 20 INTNO = 1,NEQ
      I = INTG(INTNO)
   20 Y(INTNO) = PAR(I,1)
      T = 0.0
      TZERO = 0.0
      GO TO 50
C
C       MONTAGEM DE REINICIO.
C
   60 CONTINUE
      DO 70 INTNO = 1,NEQ
      N= INTG(INTNO)
   70 Y(INTNO) = C(N)
      TZERO = C(76)
C
   50 CONTINUE
      IR = 1099511627776
      EPSLN = DTS2 / (TSAMP*2.0)
      TEST5 = 1
      M = 1
      NN = 1
      CALL CSM11
      IARG = 2
      CALL CSM8
  100 CONTINUE
C
C       INICIO DA EXECUCAO.
```

SEITE 23

```
C
C
      TEST5 = 2
      DO 110 IX = 1,NEQ
      YK(IX) = Y(IX)
  110 Y(IX) = Y(IX) + DTS2 * DYDT(IX)
      T = N * DTS2 + TZERO
      N = N+1
      CALL CSM11
C
      TEST5 = 3
      DO 120 IX = 1,NEQ
  120 Y(IX) = YK(IX) + DT * DYDT(IX)
      T = N * DTS2 + TZERO
      N = N+1
      CALL CSM11
C
  130 GO TO (140, 140, 140, 150, 150, 150) , TEST5
  140 CONTINUE
C
C
      M = T/TSAMP + EPSLN
      IF ( M - NN ) 160, 150, 150
  150 CONTINUE
C         IMPRESSAO.
      IARG = 2
      CALL CSM8
      NN = M+1
C         A CORRIDA JA TERMINOU?
  160 CONTINUE
      GO TO (170, 170, 170, 199, 195, 197) , TEST5
  170 CONTINUE
  190 IF ( T - TTOT + EPSLN ) 100, 210, 210
  195 WRITE (1,196)
  196 FORMAT (/3X,'SIMULACAO TERMINADA PELA VARIAVEL KEY16')
      GO TO 210
  197 WRITE (1,198)
  198 FORMAT (/3X,'SIMULACAO TERMINADA PELO ELEMENTO FINALIZADOR')
      GO TO 210
  199 WRITE (1,200)
  200 FORMAT (/3X,'ERRO NO PROCESSAMENTO')
  210 CONTINUE
      RETURN
      END
```

SEITE 24

```
C          SUBROTINA CSM11.
C
C          FAZ A COMPUTACAO PARA A METADE DE DT.
C
C
      SUBROUTINE CSM11
C
      REAL    ROYAL(395)
      INTEGER*2    INTS(587), TEST5, ORDER(76), INTG(25), MTRX1, MTRX2,
     1MTRX3, MTRX4, MTRX5, I, J, K, L, KEY16, NLIST, NCON, NEQ, INTNO,
     2NEXT, ITYPE, NF, N
      DIMENSION    C(76), F(3,11), Y(25), DYDT(25)
      DIMENSION    MTRX1(75), MTRX2(75), MTRX3(75), MTRX4(75), MTRX5(75)
      DIMENSION    PAR1(75), PAR2(75), PAR3(75)
      COMMON    ROYAL , INTS
      EQUIVALENCE ( INTS(  1), MTRX1(1)   ) , ( ROYAL(  2), C(1)     )
      EQUIVALENCE ( INTS( 76), MTRX2(1)   ) , ( ROYAL( 79), DTS2     )
      EQUIVALENCE ( INTS(151), MTRX3(1)   ) , ( ROYAL( 81), PAR1(1)  )
      EQUIVALENCE ( INTS(226), MTRX4(1)   ) , ( ROYAL(156), PAR2(1)  )
      EQUIVALENCE ( INTS(301), MTRX5(1)   )
      EQUIVALENCE ( INTS(376), I          ) , ( ROYAL(231), PAR3(1)  )
      EQUIVALENCE ( INTS(377), J          ) , ( ROYAL(306), F(1,1)   )
      EQUIVALENCE ( INTS(378), K          ) , ( ROYAL(341), Y(1)     )
      EQUIVALENCE ( INTS(379), L          ) , ( ROYAL(366), DYDT(1)  )
      EQUIVALENCE ( INTS(395), KEY16      )
      EQUIVALENCE ( INTS(396), INTG(1)    )
      EQUIVALENCE ( INTS(449), ORDER(1)   )
      EQUIVALENCE ( INTS(529), TEST5      )
      EQUIVALENCE ( INTS(534), NLIST      )
      EQUIVALENCE ( INTS(540), NCON       )
      EQUIVALENCE ( INTS(542), NEQ        )
C
C          CALCULO PARA A METADE DA CONFIGURACAO.
C
  900 DO 901 INTNO = 1,NEQ
      N = INTG(INTNO)
  901 C(N) = Y(INTNO)
      NEXT = NCON
 1000 CONTINUE
      I = ORDER(NEXT)
      ITYPE = MTRX1(I)
      GO TO (140,56,140,53,140,53,54,56,51,10,11,53,56,56,54,56,55,52,
     1140,53,53,22,51,55,53,53,27,56,55,140,31,32,33,34,35), ITYPE
C
C          TRES PARAMETROS
   51 P3 = PAR3(I)
C          TRES ENTRADAS
   52 L = MTRX4(I)
      CL = C(L)
C          DOIS PARAMETROS
   53 P2 = PAR2(I)
C          UM PARAMETRO
   54 P1 = PAR1(I)
C          DUAS ENTRADAS
   55 K = MTRX3(I)
      CK = C(K)
```

SEITE 25

```
C        UMA ENTRADA
   56 J = MTRX2(I)
      CJ = C(J)
      GO TO (140,2,140,4,140,6,7,8,9,10,11,12,13,14,15,16,17,18,140,20,
     121,140,23,24,25,26,27,28,29), ITYPE
C
C        B           BANGUE-BANGUE
    2 IF (CJ) 102,98,95
  102 CI = -1.0
      GO TO 99
C
C        D           ESPACO MORTO
    4 IF(CJ) 304,98,104
  104 DIFF = CJ - P1
      IF (DIFF) 98,98,204
  204 CI = DIFF
      GO TO 99
  304 DIFF = CJ - P2
      IF (DIFF) 204,98,98
C
C        F           GERADOR DE FUNCAO
    6 NF = MTRX5(I)
      P3 = P1 - P2
      IF (P3) 140,140,506
  506 P1 = 10.0*(CJ - P2) / P3
      NSECT = P1
      IF (P1) 106,106,206
  106 CI =F(NF,1)
      GO TO 99
  206 IF (NSECT - 10) 406,306,306
  306 CI = F(NF,11)
      GO TO 99
  406 P2 = NSECT
      P3 = P1 - P2
      P1 = F(NF,NSECT+1)
      P2 = F(NF,NSECT+2)
      CI = P1 + P3*(P2 -P1)
      GO TO 99
C
C        G           GANHO     (MULTIPLICADOR CONSTANTE)
    7 CI = P1 * CJ
      GO TO 99
C
C        H           RAIZ QUADRADA
    8 CI = SQRT (CJ)
      GO TO 99
C
C        I           INTEGRADOR      ( MAXIMO 25 ELEMENTOS)
    9 INTNO = MTRX5(I)
      DYDT(INTNO) = CJ + P2*CK + P3*CL
      GO TO 100
C
C        J           GERADOR DE NUMEROS ALEATORIOS ENTRE + E - 1
   10 CONTINUE
      IR = 8307463 * IR
      CI = FLOAT(IR) / 999999999999.0
      GO TO 99
C
C        K           CONSTANTE
   11 GO TO 100
C
```

SEITE 26

```
C           L           LIMITADOR
   12 DIFF = CJ - P1
      IF (DIFF) 112,93,93
  112 DIFF = CJ - P2
      IF (DIFF) 94,97,97
C
C           M           MAGNITUDE      (VALOR ABSOLUTO)
   13 CI = ABS (CJ)
      GO TO 99
C
C           N           CLIPER NEGATIVO
   14 IF (CJ) 98,98,97
C
C           O           DESLOCAMENTO NO EIXO DAS ORDENADAS
   15 CI = CJ + P1
      GO TO 99
C
C           P           CLIPPER POSITIVO
   16 IF (CJ) 97,98,98
C
C           Q           ELEMENTO FINALIZADOR
   17 IF (CJ - CK) 100,100,170
C
C           R           RELE
   18 IF (CJ) 218,118,118
  118 CI =CK
      GO TO 99
  218 CI = CL
      GO TO 99
C
C           T           GERADOR DE PULSOS
   20 GO TO (120,220,220) , TEST5
  120 PAR2(I) = -P1
      IF (CJ) 98,95,95
  220 IF (CJ) 98,95,320
  320 PAR2(I) = P2 + DTS2
      IF (P2) 98,420,420
  420 PAR2(I) = -P1
      GO TO 95
C
C           U           RETARDO UNITARIO      (MAXIMO DE 25 ELEMENTOS)
   21 GO TO (121,221,221), TEST5
  121 CI = P1
      GO TO 321
  221 CI = P2
  321 PAR2(I) = CJ
      GO TO 99
C
C           V           VACUO          (USADO EM CONJUNTO COM O ELEMENTO Y)
   22 GO TO (122,100,100), TEST5
  122 MTRX5(I) = NEXT
      GO TO 100
C
C           W           SOMADOR PONDERADO
   23 CI = CJ*P1 + CK*P2 + CL*P3
      GO TO 99
C
C           X           MULTIPLICADOR
   24 CI = CJ * CK
      GO TO 99
C
```

```
                                                    SEITE  27
C          Y            TESTE DE FUNCAO IMPLICITA
   25 RELER = ABS(1.-CK/CJ)
      IF (RELER - P1) 97,97,125
  125 CONTINUE
  225 C(K) = (1.0 - P2)*CJ + P2*CK
      NEXT = MTRX5(K)
      GO TO 1000
C
C          Z            ORDEM ZERO
   26 GO TO (126,226,226), TEST5
  126 PAR2(I) = P1
      P2 = P1
  226 IF (CK) 98,94,326
  326 PAR2(I) = CJ
      GO TO 97
C
C          +            SOMADOR     (SAO PERMITIDAS TRES ENTRADAS)
   27 J = MTRX2(I)
      IF (J) 127,927,227
  127 J = -J
      CI = -C(J)
      GO TO 327
  227 CI = C(J)
  327 K = MTRX3(I)
      IF (K) 427,627,527
  427 K = -K
      CI = CI - C(K)
      GO TO 627
  527 CI = CI + C(K)
  627 L = MTRX4(I)
      IF (L) 727,99,827
  727 L = -L
      CI = CI -C(L)
      GO TO 99
  827 CI = CI + C(L)
      GO TO 99
  927 CI = 0.0
      GO TO 327
C
C          -            INVERSOR    (LIMITADO A UMA ENTRADA)
   28 GO TO 96
C
C          /            DIVISOR
   29 IF (CK) 129,140,129
  129 CI = CJ / CK
      GO TO 99
C
C          1            ELEMENTO ESPECIAL NUMERO 1
   31 CALL SUB1
      GO TO 100
C
C          2            ELEMENTO ESPECIAL NUMERO 2
   32 CALL SUB2
      GO TO 100
C
C          3            ELEMENTO ESPECIAL NUMERO 3
   33 CALL SUB3
      GO TO 100
C
C          4            ELEMENTO ESPECIAL NUMERO 4
   34 CALL SUB4
```

SEITE 28

```
      GO TO 100
C
C           5          ELEMENTO ESPECIAL NUMERO 5
   35 CALL SUB5
      GO TO 100
C
   93 CI = P1
      GO TO 99
   94 CI = P2
      GO TO 99
   95 CI = 1.0
      GO TO 99
   96 CI =  -CJ
      GO TO 99
   97 CI = CJ
      GO TO 99
   98 CI = 0.0
   99 C(I) = CI
  100 IF (NEXT - NLIST) 1100, 1200, 140
 1100 NEXT = NEXT + 1
      GO TO 1000
 1200 CONTINUE
      RETURN
C
C           ERRO
  140 CONTINUE
  150 TEST5 = 4
      GO TO 180
  160 TEST5 = 5
      GO TO 180
  170 TEST5 = 6
  180 RETURN
C
      END
```

SEITE 29

```
C
C       SUBROTINA CSM12.
C
C       IMPRIME LISTA DE OPCOES (DEPENDENDO DO VALOR DE KEY10).
C
C
      SUBROUTINE CSM12
C
      REAL  ROYAL(395)
      INTEGER*2  INTS(587), TEST9, KEY1, KEY2, KEY3, KEY4, KEY5, KEY6,
     1KEY7, KEY8, KEY9, KEY10, KEY11, KEY12, KEY13, KEY14, KEY15
      COMMON  ROYAL, INTS
      EQUIVALENCE ( INTS(380), KEY1  ) , ( INTS(381), KEY2  )
      EQUIVALENCE ( INTS(382), KEY3  ) , ( INTS(383), KEY4  )
      EQUIVALENCE ( INTS(384), KEY5  ) , ( INTS(385), KEY6  )
      EQUIVALENCE ( INTS(386), KEY7  ) , ( INTS(387), KEY8  )
      EQUIVALENCE ( INTS(388), KEY9  ) , ( INTS(389), KEY10 )
      EQUIVALENCE ( INTS(390), KEY11 ) , ( INTS(391), KEY12 )
      EQUIVALENCE ( INTS(392), KEY13 ) , ( INTS(393), KEY14 )
      EQUIVALENCE ( INTS(394), KEY15 ) , ( INTS(533), TEST9 )
C
      WRITE (1,300)
  300 FORMAT(/3X,'ENTRE COM KEY10 IGUAL A 2 SE DESEJAR A IMPRESSAO DE UM
     1A LISTA'/3X,'DAS OPCOES POSSIVEIS NO PROGRAMA. CASO CONTRARIO ENTR
     2E COM'/3X,'KEY10 IGUAL A 1.'//3X,'KEY10 = ( )')
      READ (1,301) KEY10
  301 FORMAT (11X,I1)
      GO TO (50, 1), KEY10
    1 WRITE (1,101)
  101 FORMAT(/3X,'     OPCAO',40X,'VARIAVEL'//3X,'CONFIGURACAO',40X,'KEY
     1'/3X,'CONDICOES INICIAIS E PARAMETROS',22X,'KEY2'/3X,'GERADORES D
     2E FUNCOES',31X,'KEY3'/3X,'ESPECIFICACOES DE INTEGRACAO',24X,'KEY4'
     3/3X,'INTERVALO DE IMPRESSAO',30X,'KEY5'/3X,'VARIAVEIS A SEREM IMPR
     4ESSAS',24X,'KEY6'/3X,'SUPRESSAO DA IMPRESSAO DOS DADOS',20X,'KEY11
     5'/3X,'PERFURACAO DE DECK DE CARTOES ATUALIZADO',12X,'KEY12'/3X,'INT
     6ERROGACAO DAS SAIDAS DOS BLOCOS',18X,'KEY13'/3X,'PRESERVACAO DO ES
     7TADO NO INSTANTE DA INTERRUPCAO',4X,'KEY14'/3X,'REINICIO NO PONTO
     8DE INTERRUPCAO PREVIO',13X,'KEY15')
   50 WRITE (1,150)
  150 FORMAT(//3X,'AS VARIAS OPCOES SERAO INTERROGADAS NO DECORRER DA SI
     1MULACAO.')
      RETURN
      END
```

SEITE 30

```
C
C       SUBROTINA CSM13.
C       INTERROGACAO DAS SAIDAS DE BLOCOS.
C
C
      SUBROUTINE CSM13
C
      REAL    ROYAL(395)
      INTEGER*2    INTS(587), KEY13, I
      DIMENSION   C(76)
      COMMON   ROYAL, INTS
      EQUIVALENCE ( INTS(392), KEY13    ) , ( ROYAL(  2),    C(1)    )
C
      WRITE (1,100)
    1 WRITE (1,101)
      READ (1,102) I
      IF( I )  20, 2, 10
   10 IF (I - 75)    30,30,20
   20 WRITE (1,104)
      GO TO 1
   30 CONTINUE
      WRITE (1,103)  I, C(I)
      WRITE (1,300)
  300 FORMAT (/3X,'ENTRE COM O VALOR DE KEY13 IGUAL A 1 SE DESEJAR INTER
     1ROGAR'/3X,'NOVAS SAIDAS. CASO CONTRARIO ENTRE COM KEY13 IGUAL A 2.
     2'//3X,'KEY13 = ( )')
      READ (1,301) KEY13
  301 FORMAT (11X,I1)
      GO TO (1, 2) , KEY13
    2 RETURN
  100 FORMAT (/3X,'OPCAO DE INTERROGACAO DAS SAIDAS'/)
  101 FORMAT (/3X,'(    ) BLOCO')
  102 FORMAT (3X,I3)
  103 FORMAT (/3X,'SAIDA DO BLOCO',I3,5X,F10.4)
  104 FORMAT (/3X,'ERRO NA ENTRADA')
      END
```

SEITE 31

```
C
C       SUBROTINA CSM8.
C       CONTROLE PRINCIPAL DA SAIDA PELA IMPRESSORA.
C
C
      SUBROUTINE CSM8
      REAL    ROYAL(395)
      INTEGER*2  INTS(587), K1, K2, K3, K4, K5, IARG
      DIMENSION  C(76)
      COMMON   ROYAL , INTS
      EQUIVALENCE ( INTS(535),   K1    ) , ( ROYAL( 2) , C(1)   )
      EQUIVALENCE ( INTS(536),   K2    ) , ( ROYAL( 77) , T     )
      EQUIVALENCE ( INTS(537),   K3    )
      EQUIVALENCE ( INTS(538),   K4    )
      EQUIVALENCE ( INTS(539),   K5    )
      EQUIVALENCE ( INTS(587),   IARG  )
      GO TO ( 1, 2 ) , IARG
    1 CALL  CSM8A
      RETURN
    2 WRITE (3,201) T, C(K1), C(K2), C(K3), C(K4), C(K5)
  201 FORMAT (/5X,F10.3,5(3X,F13.4))
      RETURN
      END
```

SEITE 32

```
C
C       SUBROTINA CSM8A.
C       PEDE INFORMACOES SOBRE A IMPRESSAO.
C
      SUBROUTINE CSM8A
C
      REAL ROYAL(395)
      INTEGER*2  INTS(587), PRINT(5), TEST8, KEY5, KEY6, N
      COMMON ROYAL, INTS
      EQUIVALENCE                                    ( ROYAL( 78),  DT    )
      EQUIVALENCE  ( INTS(384),  KEY5          ) , ( ROYAL(391),  TSAMP )
      EQUIVALENCE  (INTS(385),  KEY6           )
      EQUIVALENCE  (INTS(532),  TEST8          )
      EQUIVALENCE  (INTS(535),  PRINT(1)       )
C
      GO TO (2, 1), TEST8
    1 GO TO (2, 4), KEY5
    2 WRITE(1,101)
      READ (1,102) TSAMP
      IF (TSAMP - DT )  80,4,4
C
C       PROCURA DE QUE BLOCOS DEVEM SER IMPRESSOS.
C
    4 GO TO (6, 5), TEST8
    5 GO TO (7, 40), KEY6
    6 TEST8 = 2
    7 WRITE (1, 103)
      READ (1, 105)  (PRINT(N),N=1,5 )
      WRITE (3,104) PRINT
C
C       SE O NUMERO DO BLOCO NAO ESTA ENTRE 1 E 75, IMPRIMA 0.0
C
      DO 30 N=1,5
      IF ( PRINT(N) )  10, 10, 20
   10 PRINT(N) = 0
      GO TO 30
   20 IF ( PRINT(N) - 75 )  30, 30, 10
   30 CONTINUE
      GO TO 50
   40 WRITE (3,104) PRINT
   50 CONTINUE
      WRITE (3,110)  TSAMP
      RETURN
   80 WRITE (1,109)
      GO TO 2
C
  101 FORMAT (//3X,'(          ) INTERVALO DE IMPRESSAO.')
  102 FORMAT (3X,F10.0)
  103 FORMAT (3X,'TEMPO',2X,5(2X,'SAIDA(  )'))
  104 FORMAT (//////9X,'TEMPO',6X,5(2X,'SAIDA',I3,6X),//)
  105 FORMAT (6X,5(9X,I2))
  109 FORMAT (/3X,'O INTERVALO DE IMPRESSAO NAO PODE SER MENOR QUE O INT
     1ERVALO'/3X,'DE INTEGRACAO')
  110 FORMAT(///5X,'INTERVALO DE IMPRESSAO =',F13.4)
C
```

SEITE 33

```
C       SUBROTINA SUB1
C       SUBROTINA ESPECIAL PARA A FUNCAO SENO.
C
      SUBROUTINE SUB1
C
      REAL ROYAL(395)
      INTEGER*2  INTS(587), MTRX2, I, J
      DIMENSION  C(76), MTRX2(75)
      COMMON  ROYAL, INTS
C
      EQUIVALENCE  ( INTS( 76), MTRX2(1) ) , ( ROYAL(  2), C(1) )
      EQUIVALENCE  ( INTS(376), I        )
C
CCC       I CONTEM O INDICE DA VARIAVEL DE SAIDA
CCC       MTRX2(I) CONTEM J, O NUMERO DO BLOCO DA PRIMEIRA VARIAVEL DE
CCC       ENTRADA AO BLOCO I
CCC       C(J) CONTEM O VALOR ATUAL DA PRIMEIRA ENTRADA AO BLOCO I
CCC       O VALOR ATUAL DA SAIDA DO BLOCO I DEVE SER ARMAZENADO EM C(I)
C
      J = MTRX2(I)
      CJ = C(J)
C       SENO   (ARGUMENTO EM RADIANOS)
      C(I) = SIN (CJ)
C
      RETURN
      END
```

SCITE 34

```
C       SUBROTINA SUB2
C       SUBROTINA ESPECIAL PARA A FUNCAO COSENO
C
C
      SUBROUTINE SUB2
C
      REAL    ROYAL(395)
      INTEGER*2   INTS(587), MTRX2, I, J
      DIMENSION   C(76), MTRX2(75)
      COMMON  ROYAL, INTS
C
      EQUIVALENCE  ( INTS( 76), MTRX2(1) ) , ( ROYAL(  2), C(1) ) ,
      EQUIVALENCE  ( INTS(376), I        )
C
C       I CONTEM O INDICE DA VARIAVEL DE SAIDA
C       MTRX2(I) CONTEM J, O NUMERO DO BLOCO DA PRIMEIRA VARIAVEL DE
C       ENTRADA AO BLOCO I
C       C(J) CONTEM O VALOR ATUAL DA PRIMEIRA ENTRADA AO BLOCO I
C       O VALOR ATUAL DA SAIDA DO BLOCO I DEVE SER ARMAZENADO EM C(I)
C
      J = MTRX2(I)
      CJ = C(J)
C       COSENO    (ARGUMENTO EM RADIANOS)
      C(I) = COS (CJ)
C
      RETURN
      END
```

SEITE 35

```
C       SUBROTINA SUB3
C       SUBROTINA ESPECIAL PARA A FUNCAO EXPONENCIAL
C
      SUBROUTINE SUB3
C
      REAL   ROYAL(395)
      INTEGER*2   INTS(587), MTRX2, I, J
C
      DIMENSION   C(76), MTRX2(75)
      COMMON   ROYAL, INTS
C
       EQUIVALENCE  ( INTS( 76), MTRX2(1) ) , ( ROYAL(  2), C(1) )
      EQUIVALENCE  ( INTS(376), I        )
C
C       I CONTEM O INDICE DA VARIAVEL DE SAIDA
C       MTRX2(I) CONTEM J, O NUMERO DO BLOCO DA PRIMEIRA VARIAVEL DE
C       ENTRADA AO BLOCO I
C       C(J) CONTEM O VALOR ATUAL DA PRIMEIRA ENTRADA AO BLOCO I
C       O VALOR ATUAL DA SAIDA DO BLOCO I DEVE SER ARMAZENADO EM C(I)
C
      J = MTRX2(I)
      CJ = C(J)
      C(I) = EXP (CJ)
C
      RETURN
      END
C       SUBROTINA SUB4
C       SUBROTINA ESPECIAL PARA A FUNCAO ARCOTANGENTE
C
```

SEITE 36

```
      SUBROUTINE SUB4
C
      REAL    ROYAL(395)
      INTEGER*2    INTS(587), MTRX2, I, J
      DIMENSION    C(76), MTRX2(75)
      COMMON    ROYAL, INTS
C
       EQUIVALENCE   ( INTS( 76), MTRX2(1) ) , ( ROYAL(  2), C(1) )
      EQUIVALENCE   ( INTS(376), I         )
C
C        I CONTEM O INDICE DA VARIAVEL DE SAIDA.
C        MTRX2(I) CONTEM J, O NUMERO DO BLOCO DA PRIMEIRA VARIAVEL DE
C        ENTRADA AO BLOCO I
C        C(J) CONTEM O VALOR ATUAL DA PRIMEIRA ENTRADA AO BLOCO I
C        O VALOR ATUAL DA SAIDA DO BLOCO I DEVE SER ARMAZENADO EM C(I)
C
      J = MTRX2(I)
      CJ = C(J)
      C(I) = ATAN (CJ)
C
      RETURN
      END
C        SUBROTINA SUB5
C        SUBROTINA ESPECIAL PARA A GERACAO DA FUNCAO LOGARITMICA
C
```

SEITE 37

```
      SUBROUTINE SUBS
C
      REAL ROYAL(395)
      INTEGER*2   INTS(587), MTRX2, I, J
      DIMENSION   C(76), MTRX2(75)
      COMMON   ROYAL , INTS
C
      EQUIVALENCE   ( INTS( 76), MTRX2(1) ) , ( ROYAL(  2), C(1) )
      EQUIVALENCE   ( INTS(376), I        )
C
C        I CONTEM O INDICE PARA A VARIAVEL DE SAIDA
C        MTRX2(I) CONTEM O NUMERO DO BLOCO DA PRIMEIRA VARIAVEL DE
C        ENTRADA AO BLOCO I
C        C(J) CONTEM O VALOR ATUAL DA PRIMEIRA ENTRADA AO BLOCO I
C        O VALOR ATUAL DA SAIDA DO BLOCO I DEVE SER ARMAZENADO EM C(I)
C
      J = MTRX2(I)
      CJ = C(J)
      C(I) = ALOG(CJ)
C
      RETURN
      END
,
HP          FEHLERFREI
KERNSPEICHERBEDARF:    630 ZELLEN OHNE COMMON

UP CSM0   FEHLERFREI
KERNSPEICHERBEDARF:    366 ZELLEN OHNE COMMON

UP CSM1   FEHLERFREI
KERNSPEICHERBEDARF:    891 ZELLEN OHNE COMMON

UP CSM2   FEHLERFREI
KERNSPEICHERBEDARF:    561 ZELLEN OHNE COMMON

UP CSM3   FEHLERFREI
KERNSPEICHERBEDARF:   1100 ZELLEN OHNE COMMON

UP CSM4   FEHLERFREI
KERNSPEICHERBEDARF:    669 ZELLEN OHNE COMMON

UP CSM5   FEHLERFREI
KERNSPEICHERBEDARF:   1096 ZELLEN OHNE COMMON

UP CSM6   FEHLERFREI
KERNSPEICHERBEDARF:    470 ZELLEN OHNE COMMON

UP CSM7   FEHLERFREI
KERNSPEICHERBEDARF:    185 ZELLEN OHNE COMMON

UP CSM10  FEHLERFREI
KERNSPEICHERBEDARF:    510 ZELLEN OHNE COMMON
```

```
UP CSM11  FEHLERFREI
KERNSPEICHERBEDARF:    932 ZELLEN OHNE COMMON

UP CSM12  FEHLERFREI
KERNSPEICHERBEDARF:    447 ZELLEN OHNE COMMON

UP CSM13  FEHLERFREI
KERNSPEICHERBEDARF:    237 ZELLEN OHNE COMMON

UP CSM8   FEHLERFREI
KERNSPEICHERBEDARF:    119 ZELLEN OHNE COMMON

UP CSM8A  FEHLERFREI
KERNSPEICHERBEDARF:    289 ZELLEN OHNE COMMON

UP SUB1   FEHLERFREI
KERNSPEICHERBEDARF:     51 ZELLEN OHNE COMMON

UP SUB2   FEHLERFREI
KERNSPEICHERBEDARF:     51 ZELLEN OHNE COMMON

UP SUB3   FEHLERFREI
KERNSPEICHERBEDARF:     51 ZELLEN OHNE COMMON

UP SUB4   FEHLERFREI
KERNSPEICHERBEDARF:     51 ZELLEN OHNE COMMON

UP SUB5   FEHLERFREI
KERNSPEICHERBEDARF:     51 ZELLEN OHNE COMMON
```

# CONCLUSÕES E COMENTÁRIOS

# CONCLUSÕES E COMENTÁRIOS

Nosso objetivo, ao iniciarmos este trabalho, era dotar o Laboratório de Computação e Simulação da EFEI de um processador para a simulação de sistemas contínuos.

O caminho escolhido para a sua realização foi a adaptação de um programa já existente, desenvolvido para outro tipo de computador.

Ao término do trabalho, acreditamos ter conseguido o nosso objetivo.

O programa P.S.S.C., tal como o apresentamos, possibilita a realização de estudos de simulação abrangendo toda a área de interêsse científico e didático da EFEI.

Com relação ao programa original, podemos afirmar ter sido preservada a grande versatilidade de opções, possíveis em uma sessão de simulação.

Quanto à velocidade de processamento, acreditamos que foi conseguido um razoável balanço, pois, apesar do fato de as opções durante uma corrida serem feitas por meio do teletipo do console, que é um dispositivo bastante lento, a saida de resultados via impressora é muito mais rápida.

O programa pode ser utilizado para fins didáticos e científicos.

Na parte didática, poderá servir como meio de acesso ao computador para a obtenção de soluções, a cursos como: análise de sistemas, controle e Servomecanismos e outros.

Pode ainda servir para a verificação de soluções ob-

tidas nos computadores analógicos.

Quanto à parte científica, acreditamos ser o programa um instrumento de utilidade para todos aqueles que tem em simulação a sua área de interesse.

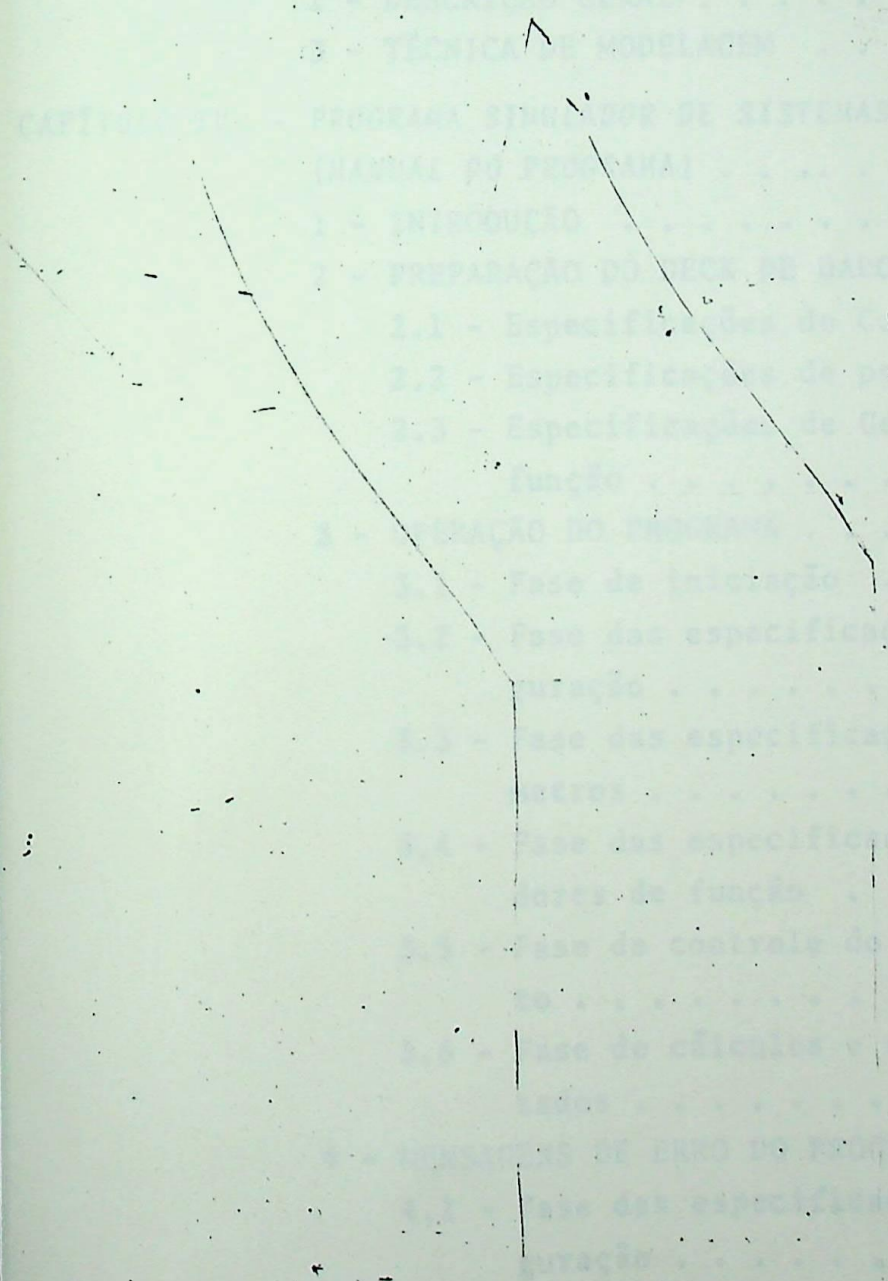
Pretendíamos, conforme foi dito anteriormente, incluir no trabalho, um capítulo com exemplos de aplicação do programa, abrangendo o mais possível a gama de utilização do mesmo, de forma a permitir a usuários na familiarizados com o programa, uma visualização mais completa do seu uso.

Não nos foi possível, entretanto, incluir este capítulo pelo fato do computador TR-86 se encontrar, no momento, fora de uso, por motivos técnicos.

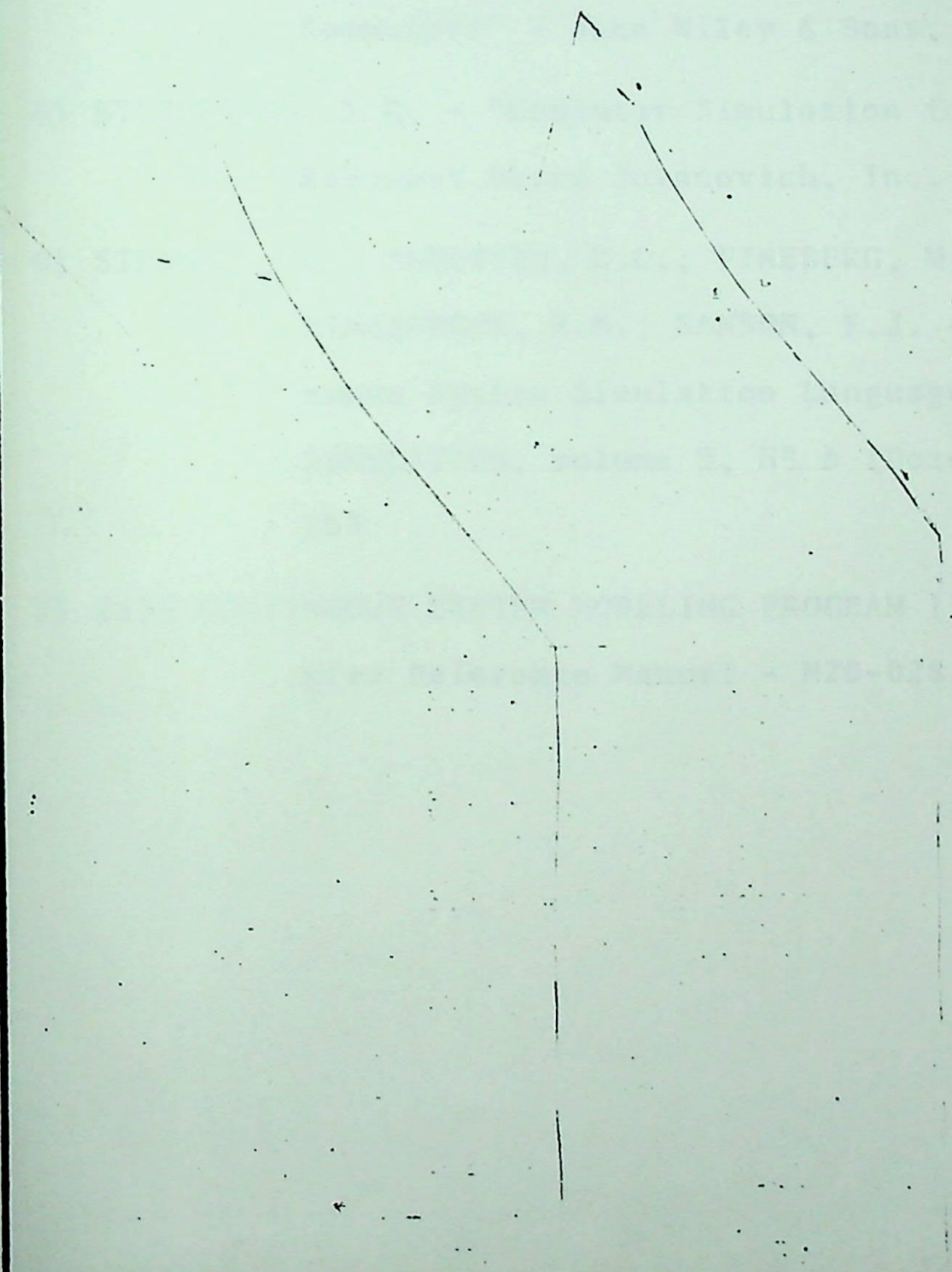
Colocamo-nos, entretanto, à disposição dos interessados para o esclarecimento de quaisquer dúvidas com respeito à utilização do P.S.S.C.

Finalmente, gostaríamos de ressaltar que este trabalho é apenas o resultado da primeira tentativa de dotar o LCS da EPEI de um processador para a simulação digital de sistemas contínuos.

Nos sentiremos mais do que recompensados se esta nossa iniciativa puder servir de estímulo para que outros e melhores trabalhos venham a ser realizados neste campo.



# I N D I C E



# B I B L I O G R A F I A

1) CHU, Y. - "Digital Simulation of Continuous Systems" - Mc Graw-Hill, Inc. - New York, 1969

2) JACKSON, A.S. - "Analog Computation" - Mc Graw-Hill, Inc. - New York, 1960

3) JOHNSON, C.L. - "Analog Computer Techniques" - Mc Graw-Hill Inc. - New York, 1963

4) RALSTON, A; WILF, H.S. - "Mathematical Methods for Digital Computers" - John Wiley & Sons, - Inc. N.Y. 1967

5) STEPHENSON, R.E. - "Computer Simulation for Engineers" - Harcourt Brace Jovanovich, Inc. - New York, 1971

6) STRAUS, J.C.; AUGUSTIN, D.C.; FINEBERG, M.S., JOHNSON, B.B. LINEBARGER, R.N.; SANSON, F.J. - "The Sci Conti - nuous System Simulation Language (C.S.S.L)" - SIMULATION, volume 9, Nº 6 (Dezembro, 1967), 281-283

7) 1130 CONTINUOUS SYSTEM MODELING PROGRAM (1130 - CX-13X)Program Reference Manual - H20-0282-0-IBM, 1966.