H. P. NEVES *

SUMÁRIO

O trabalho a seguir descreve um analisador espectral baseado em microcomputador, com aplicação dirigida mais especificamente à análise de padrões vocais. Apesar dessa especificidade, não se procurou desenvolver os algoritmos propriamente ditos no que se refere à identificação dos padrões da voz humana, mas sim fornecer uma ferramenta a essa análise. Cabe ressaltar que o analisador poderia encontrar aplicações diversas, tais como análise espectral em física nuclear, reconhecimento de imagens, análise de padrões em radar e sonar, etc.

ABSTRACT

The paper describes a microcomputer-based spectral analyzer, designed specifically for vocal pattern analysis. In spite of its specificity, there had been no intention to deal with the pattern analysis algorithms themselves, but to make a tool for such an analysis. It should be stressed that our analyzer can find several different applications , like spectral analysis applied to nuclear physics, image recognition, pattern analysis applied to radar and sonar, etc.

* Engenheiro eletricista (Universidade Federal de Minas Gerais, 1985); microeletrônica e interfaces especiais para microcomputadores; trabalho desenvolvido no Depto. de Engenharia Eletrônica da UFMG em 1984; Rua Síria, 116 - Floresta - 30000 Belo Horizonte, MG

## INTRODUÇÃO

No desenvolvimento desse trabalho, foi utilizado um microprocessador Z80, o qual foi escolhido por sua grande versatilidade no que se refere ao seu conjunto de instruções, e pela facilidade de encontrá-lo no mercado brasileiro. A interface para a entrada analógica de sinal é realizada por um conversor analógico digital fabricado pela Analog Devices, o AD571, que é um conversor rápido de 10 bits, capaz de efetuar uma conversão em apenas 25 microssegundos. O microprocessador executa uma rotina de transformada rápida de Fourier para a conversão de domínio tempo-frequência, além de gerenciar a aquisição de dados. Tal rotina de cálculo foi escolhida devido à sua grande velocidade, o que permite sua utilização em tempo real.

## HARDWARE

Utilizou-se para a conversão analógico-digital um conversor AD571, que é um conversor rápido de 10 bits (tempo de conversão = 25 μs), tendo-se em vista que, para aplicações diferentes da original (processamento de voz) velocidades de conversão maiores podem ser necessárias. Pode-se demonstrar que

$$F = \frac{1}{(t \cdot N)}$$

onde F é a distância (em Hertz) entre as raias espectrais desejadas, t o intervalo de amostragem e N o número de pontos a serem processados. No nosso caso, serão processados 1024 pontos por amostragem. Se tivermos t = 1 ms, teremos uma separação entre raias de cerca de 1 Hz, o que nos dá uma excelente resolução espectral. Utilizou-se como circuito de entrada um microfone de eletreto com dispositivo ativo incorporado e dois amplificadores operacionais. O primeiro deles (AO1) encarrega-se de amplificar o sinal recebido do microfone, ajustando-o a um nível adequado, enquanto o segundo (AO2) atua como buffer a fim de adaptar a impedância do circuito com a entrada do AD571, cuja impedância é relativamente baixa (aproximadamente 5000 Ω). Os CI's 2 e 3 (74LS373) são latches octais que travam o resultado da leitura do conversor A/D. Isso se faz necessário pois o conversor trabalha com dados de 10 bits enquanto que o barramento de dados do Z80 é de 8 bits. Assim, toda vez que uma conversão se completar, os latches são trancados através do sinal proveniente do pino 17 do conversor (aviso de fim de conversão, "READY"), já que este encontra-se ligado ao pino ENABLE G de cada um dos latches. Observe-se que tal pino encontra-se também ligado ao pino INT do Z80. Isso porque toda vez que o computador estiver pronto para carregar dados novos na memória, ele entra em Halt, ou seja, pára de processar e fica aguardando uma interrupção. Ao entrar em Halt, a linha HALT do Z80 é ativada e, como ela está conectada à linha 11 do conversor ( que comanda o início de uma nova conversão, "CLEAR/CONVERT"), ela comanda-o a executar uma

conversão. Terminada esta, a linha 17 do conversor é ativada para sinalizar "fim de conversão". Estando ela ligada à linha INT do computador, aciona então uma interrupção, fazendo com que o dado recém entrado seja armazenado. Foi usado para armazenamento dos dados um banco de memória dinâmica (8 CI's 4116) de 8 Kbytes, já que são processados 1024 pontos, os quais, em notação de ponto flutuante de 4 bytes, ocupam 4096 bytes, devendo-se ainda considerar que tais pontos são tratados como números complexos, necessitando pois de uma área igual à anterior para armazenar a parte imaginária. Foi ainda incorporada RAM estática de 1 Kbyte (2 CI's 2114) destinado às variáveis de cálculo, à pilha do Z80 e à pilha operacional destinada às operações com ponto flutuante. Além disso, foi usada uma EPROM 2732 (8 K x 8), cuja capacidade de armazenamento é mais que suficiente para abrigar o programa total (foi aqui levado em consideração o fato de que o programa em questão é normalmente utilizado como suporte de um programa global, devendo-se portanto destinar-lhe espaço). Utilizou-se ainda um multiplexador (MUX) para o endereçamento e refrescamento das memórias dinâmicas. A figura 1 apresenta o hardware do sistema.

SOFTWARE

a) Descrição da transformada rápida de Fourier

A transformada rápida de Fourier (em inglês "Fast Fourier Transform"- FFT) é um poderosíssimo recurso surgido em 1965 para se calcular a transformada discreta de Fourier com uma velocidade incrivelmente maior que no processo convencional, facilitando assim sua aplicação em tempo real. Para que se tenha uma base de comparação, o processo normal de cálculo de uma transformada de Fourier exigiria, para uma função amostrada em 1024 pontos (como no nosso caso) cerca de quatro milhões de multiplicações, enquanto que no cálculo através da FFT esse número cai para menos de 20000 multiplicações. A razão entre os tempos de processamento dos dois métodos pode ser demonstrada como sendo

$$R = 2.N / \gamma$$

onde N é o número de pontos e $\gamma$ é o logaritmo na base 2 de N. Para N=1024, $\gamma$=10 e consequentemente R=204,8; portanto o programa usando FFT é 200 vezes mais rápido que o convencional. A figura 2 ilustra através de um gráfico a diferença entre os tempos.

Mostraremos a seguir os fundamentos nos quais se baseia a transformada rápida de Fourier. Seja a transformada convencional contínua de Fourier de uma função contínua s(t):

$$S(f) = \int_{-\infty}^{\infty} s(t)\, e^{-j2\pi ft}\, dt$$

onde S(f) é a transformada s(t). Fazendo-se a convolução da função s(t) com a função

com a função delta de Dirac, teremos s(t) discretizada no tempo. Após truncarmos tal função amostrada num intervalo T, obtemos sua transformada e fazemos a convolução desta com a função delta de Dirac no domínio da frequência, obtendo assim a definição da transformada discreta de Fourier, ou seja,

$$G(\frac{n}{NT}) = \sum_{k=0}^{N-1} g(kT)e^{-j2\pi nk/N}$$

onde k=0,1,2,3,...,N-1. Escrevendo-se k em vez de kT e n em vez de n/NT, obtemos a representação simplificada abaixo:

$$X(n) = \sum_{k=0}^{N-1} x_0(k)e^{-j2\pi nk/N}$$

onde n=0,1,2,3,...,N-1. Podemos notar que a equação acima descreve o cálculo de N equações. Se N=4 e se nós escrevermos W=exp(-jπ/N), podemos reescrevê-la como:

$$\begin{aligned}
X(0)&=x_0(0)W^0+x_0(1)W^0+x_0(2)W^0+x_0(3)W^0\\
X(1)&=x_0(0)W^0+x_0(1)W^1+x_0(2)W^2+x_0(3)W^3\\
X(2)&=x_0(0)W^0+x_0(1)W^2+x_0(2)W^4+x_0(3)W^6\\
X(3)&=x_0(0)W^0+x_0(1)W^3+x_0(2)W^6+x_0(3)W^9
\end{aligned}$$

que é o mesmo que

$$\begin{bmatrix} X(0)\\ X(1)\\ X(2)\\ X(3)\end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0\\ W^0 & W^1 & W^2 & W^3\\ W^0 & W^2 & W^4 & W^6\\ W^0 & W^3 & W^6 & W^9\end{bmatrix} \cdot \begin{bmatrix} x_0(0)\\ x_0(1)\\ x_0(2)\\ x_0(3)\end{bmatrix} \qquad (1)$$

É importante observar aqui que, por razões que serão vistas posteriormente, é conveniente escolher o número de pontos de amostragem sempre como uma potência de 2. Baseados na propriedade de que $W^{nk}=W^{(nk\ MOD\ N)}$ onde nk MOD N é o resto da divisão de nk por N , podemos reescrever a equação (1) como:

$$\begin{bmatrix} X(0)\\ X(1)\\ X(2)\\ X(3)\end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1\\ 1 & W^1 & W^2 & W^3\\ 1 & W^2 & W^4 & W^6\\ 1 & W^3 & W^6 & W^9\end{bmatrix} \cdot \begin{bmatrix} x_0(0)\\ x_0(1)\\ x_0(2)\\ x_0(3)\end{bmatrix}$$

podemos demonstrar essa propriedade para, por exemplo, nk=6:

$$W^6 = e^{(-j2\pi/4).6} = e^{-j3\pi} = e^{-j\pi} = e^{(-j2\pi/4).2} = W^2$$

fatorando-se a matriz temos

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \cdot \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (2)$$

a forma de se fatorar a matriz é que caracteriza a rotina de FFT. Multiplicando-se as duas últimas matrizes, temos:

$$\begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \cdot \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix}$$

podemos notar que, no cálculo de $x_1(0)$ e $x_1(2)$, temos

$$x_1(0) = x_0(0) + W^0 x_0(2)$$
$$x_1(2) = x_0(0) + W^2 x_0(2) = x_0(0) - W^0 x_0(2)$$

o que implica em 2 somas e 1 multiplicação; o mesmo se verifica para $x_1(1)$ e $x_1(3)$. Portanto economizamos 2 multiplicações redundantes. Continuando o cálculo, multiplicamos as duas matrizes restantes:

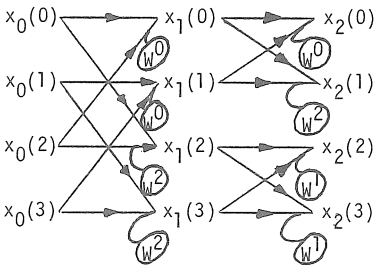
$$\begin{bmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \cdot \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix}$$

Observe-se que a fatoração obtida em (2) levou a uma inversão entre as linhas 2 e 3; tal fenômeno faz parte da fatoração e devemos ter o cuidado de, após terminar os cálculos obtendo a equação acima, desinverter as linhas adequadamente. Para isso, é suficiente que se escreva a ordem de cada linha em binário e que se reescreva cada ordem de trás para frente, conforme abaixo:

0 → 00 → inversão → 00 (0)
2 → 10 → inversão → 01 (1)
1 → 01 → inversão → 10 (2)
3 → 11 → inversão → 11 (3)

Vejamos agora como é feita a fatoração da matriz principal. Podemos representar cada etapa do processo graficamente, sendo que os resultados intermediários, a partir da matriz das variáveis independentes, são escritos verticalmente e as operações representa-

das por setas. Dessa forma, o cálculo da transformada acima exemplificado (N=4) é representado da seguinte forma:




Observe-se que a primeira coluna representa a matriz das variáveis independentes, a segunda representa o primeiro resultado parcial, correspondente ao produto das duas últimas matrizes da equação (2) e a terceira coluna representa o produto final. As linhas que convergem para $x_1(0)$, por exemplo, significam que ele é obtido a partir da soma de $x_0(0)$ e $x_0(2)$, este último multiplicado por $W^0$ (note que a seta proveniente de $x_0(2)$ tem $W^0$escrito debaixo dela). Podemos também notar que existem pares de valores que são calculados a partir dos mesmos valores anteriores, como no caso de $x_1(0)$ e $x_1(2)$, que são calculados a partir de $x_0(0)$ e $x_0(2)$. Como estes últimos dois só são utilizados no cálculo de $x_1(0)$ e $x_1(2)$, podemos logo deduzir que cada coluna vertical de dados acima pode ser armazenada no mesmo local aonde estavam os dados que foram utilizados para calculá-la; dessa forma, se estamos trabalhando com uma amostragem de N pontos, necessitaremos somente de N posições de memória para armazenar os dados, desde a função original até a função transformada. A cada par de valores como descrito acima, chamamos nó dual. É importante dizer que os nós duais aparecem sempre com um espaçamento de $N/2^{\ell}$, onde N é o número de pontos amostrados e $\ell$ é a ordem do cálculo, isto é o número da coluna vertical com a qual estamos operando.

Outro passo importante é a determinação do expoente de W em cada operação. Para isso devemos seguir três regras:

1) Escrever o índice do elemento que desejamos calcular em binário usando $\gamma$ bits;
2) Deslocar o número $\gamma$-$\ell$ bits para a direita zerando os espaços vagos;
3) Reverter a ordem dos bits.

Por exemplo, para $\gamma$=4, $\ell$=3 e índice do elemento =2, temos:

0010 → deslocamento → 0001 → reversão → 1000

Na figura 3 temos um algoritmo simplificado da transformada rápida de Fourier.

b) Representação dos números em ponto flutuante

Foi utilizada para finalidades de cálculo uma notação de ponto flutuante, onde os números são representados por números de 4 bytes, sendo que o primeiro deles é o expoente e os demais são a mantissa. Tal representação é mais que suficiente em termos de precisão para a maioria das aplicações conhecidas de FFT. Para representarmos um número em notação de ponto flutante da maneira que foi utilizada em nosso programa, basta que sigamos os seguintes procedimentos:

1) O zero é representado como 0000 0000 0000 0000 0000 0000 0000 0000.

2) Sem considerar o sinal, escreve-se o número em binário. Exemplos:

   -10 → 1010

   0,375 → 0,011

   4,25 → 100,01

   7 → 111

3) Se o número à esquerda da vírgula binária for zero e o primeiro dígito à sua direita for 1, o expoente será zero.

4) Se o número à esquerda da vírgula binária for zero e o primeiro dígito à sua direita for 0, o expoente será igual ao negativo do número de zeros à direita da vírgula decimal até chegar no primeiro dígito igual a um.

5) Se o número à esquerda da vírgula binária for diferente de zero, o expoente será igual ao número de dígitos. Exemplos:

   -10 → expoente 4

   0,375 → expoente -1

   4,25 → expoente 5

   7 → expoente 3

6) Se o número a ser representado for negativo, deixamo-lo tal como ele está, se for positivo, trocamos por zero o primeiro bit da mantissa. Exemplos:

-10 → 0000 0100 1010 0000 0000 0000 0000 0000

0,375 → 1111 1111 0100 0000 0000 0000 0000 0000

4,25 → 0000 0101 1000 1000 0000 0000 0000 0000

7 → 0000 0011 0110 0000 0000 0000 0000 0000

c) Cálculo das funções trigonométricas

Para o cálculo das funções trigonométricas, foram utilizados polinômios de Chebyshev, que consistem em funções elementares que se aproximam com muita exatidão das funções desejadas. Eles podem ser definidos da seguinte forma:

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z)$$

onde n é a ordem do polinômio e z a variável independente. Para o cálculo de seno e cosseno, foi utilizado um valor de n=6. Podemos obter os coeficientes para as potências de z em qualquer compêndio de cálculo numérico avançado. Na figura 4 temos o gráfico dos

polinômios de Chebyshev até a quarta ordem. Cabe ressaltar que optou-se pelo uso desses polinômios pela velocidade de cálculo razoavelmente alta em relação aos processos convencionais, como por exemplo através de séries de Taylor, sendo que sua precisão é maior.

d) Programa principal

Na figura 5 temos o fluxograma geral do programa. Ele foi desenvolvido utilizando-se o programa Assembler EDTASM da Apparat Inc. Note-se que a parte correspondente ao FFT foi resumida em vista de já ter sido apresentada em maiores detalhes na figura 3. O programa inicia com a habilitação de interrupção para a aquisição de dados. Uma vez em Halt, o computador aguarda uma requisição de interrupção e fica em estado de espera. Cada vez que uma interrupção é solicitada (o que corresponde a um fim de conversão, como já vimos na seção de Hardware), o programa passa a ser executado no endereço 0038H (devido ao modo de interrupção utilizado), aonde temos a aquisição de dados propriamente dita. Cada vez que os dados são carregados entra-se novamente em Halt para aguardar outros dados. Isso não ocorre somente quando a área de dados já está cheia, o que faz com que a execução seja desviada para o corpo principal do programa, aonde os dados são convertidos para a notação de ponto flutuante, são processados pela rotina de FFT e são novamente reconvertidos para números inteiros, desta vez de 16 bits. Após essa reconversão, retorna-se ao início do programa para que novos dados sejam armazenados e convertidos. Como já foi dito no início das explicações, nosso programa funciona como subsídio a um programa global de reconhecimento de padrões vocais; tal programa deve, pois, ser inserido antes desse retorno ao início do programa.

## CONCLUSÃO

Verificou-se, no decorrer do trabalho, ser desnecessário o emprego de DMA (hipótese essa que foi inicialmente levantada), uma vez que este só se justifica para taxas de transferência superiores a 10000 bit/s; portanto, com o AD571 a transferência via CPU atende plenamente. Para aplicações que exijam maior velocidade de cálculo, recomenda-se o uso de microprocessadores mais rápidos, como o Z80H. No decorrer do projeto estudou-se ainda a possibilidade de se usar o microprocessador 8086. Ele teria especial vantagem pelo fato de possuir instruções do tipo MUL e DIV que fazem multiplicações e divisões de 16 bits diretamente, bem como um poderoso conjunto de instruções multibyte que indubitavelmente tornariam nosso programa bem mais rápido. Uma utilização ainda mais interessante do 8086 incluiria sua configuração em multiprocessamento (aterrando-se seu pino MN/$\overline{MX}$) com um co-processador de ponto flutuante e uma supervisão de controle por um arbitrador de barramentos.

## BIBLIOGRAFIA

ALBRECHT et alii - Electronics Engineer's Handbook - McGraw-Hill
BRIGHAM, E.O. - The Fast Fourier Transform - Prentice Hall
ABRAMOWITZ & STEGUN - Handbook of Mathematical Functions - Dover
LEVENTHAL, Lance A. - Z80 Assembly Language Programming - Osborne-McGraw-Hill
LOGAN, I. - ZX81 ROM Disassembly
RUSSEL & RECTOR - 8086 Handbook - Osborne-McGraw-Hill
UNIVERSITY OF TEXAS - Practical Use of FFT

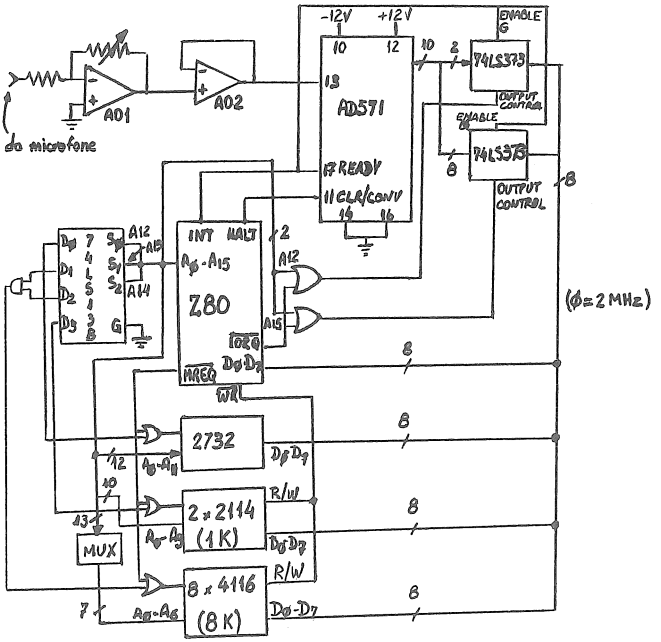



FIGURA 1

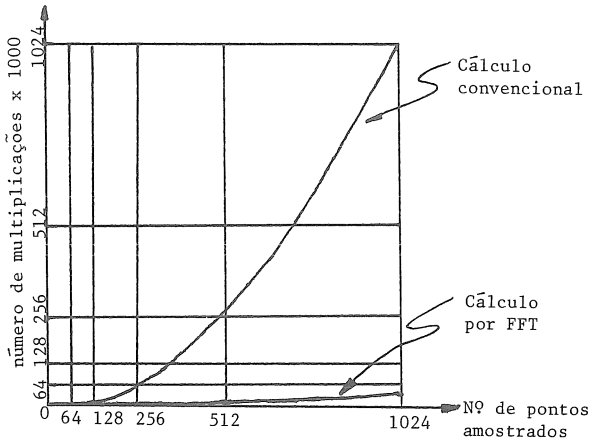



FIGURA 2

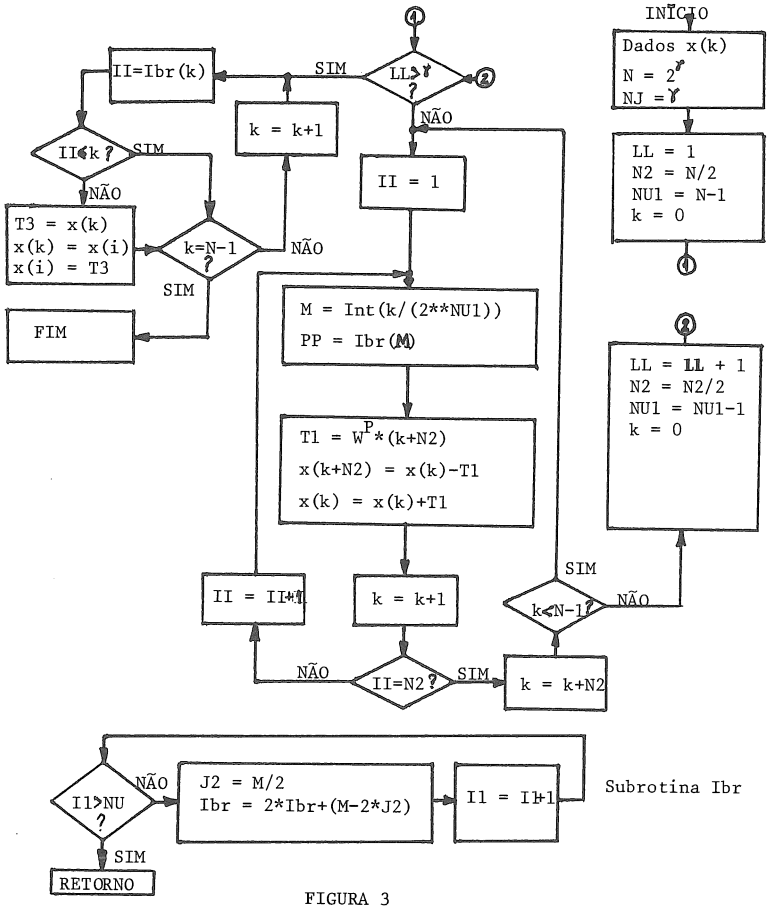



FIGURA 3

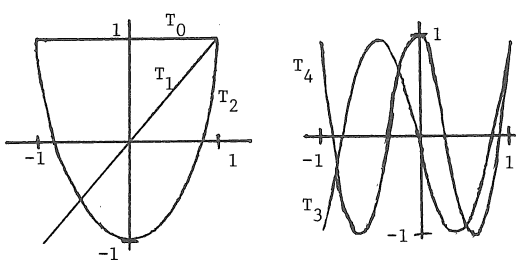



FIGURA 4

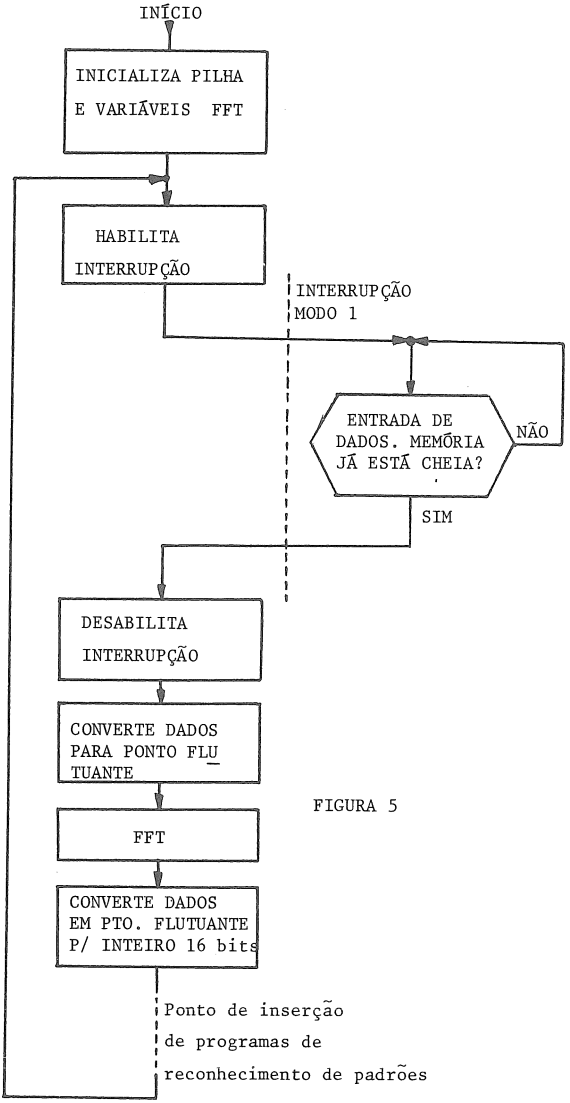

INÍCIO
INICIALIZA PILHA
E VARIÁVEIS FFT
HABILITA
INTERRUPÇÃO
INTERRUPÇÃO
MODO 1
ENTRADA DE
DADOS. MEMÓRIA
JÁ ESTÁ CHEIA?
NÃO
SIM
DESABILITA
INTERRUPÇÃO
CONVERTE DADOS
PARA PONTO FLU
TUANTE
FFT
CONVERTE DADOS
EM PTO. FLUTUANTE
P/ INTEIRO 16 bits
Ponto de inserção
de programas de
reconhecimento de padrões

FIGURA 5