Polinômios e funções em Ladder

Introdução

A norma IEC 61131-3 regulamenta os blocos de função dentro de um diagrama Ladder. Dentre estes blocos estão aqueles que realizam operações de soma e multiplicação de uma variável. Em linguagens como C++ ou Java, bibliotecas devem ser referenciadas no código do programa para possibilitar cálculos de funções como SENO(X), COS(X), 1/(1-x), etc.

Por exemplo, em linguagem *Java* o problema se resolve facilmente:

```
class MeuCalculo {
  public static void main (String args[]) {

               double argumento = 60; //graus

       System.out.println(Math.sin(argumento));
 }
}
```

Como a linguagem de Blocos de Função IEC 61131-3 não é uma linguagem estruturada ou mesmo procedural, para implementar estes cálculos precisamos realizar uma programação extra.

No entanto, nas linguagens citadas, o programador precisa elaborar o código ou utilizar alguma API específica que lê o dado e então realiza o cálculo. Com blocos de função toda esta complexidade fica encapsulada. O usuário apenas terá que informar qual entrada do módulo de E/S deseja ler para em seguida processar o dado.

Uma abordagem interessante é utilizar as Séries de *Taylor* para aproximar funções mais complexas. Inicialmente, vamos descrever como programar um polinômio utilizando a linguagem Ladder e, em seguida mostrar um caso particular e implementar algumas redes lógicas que refletem este processo.

A Série de Taylor

Uma função f(x) é tal que para cada x real está associado um ponto y=f(x). Assim:

f(x) = g **(1)**

A série de *Taylor* [referência] é um recurso matemático que nos permite obter uma aproximação desta função. Na verdade, vamos utilizar a série de *Maclaurin*, que nada mais é do que um caso particular da Série de *Taylor.*

A série pode ser representada por:

$$f(x) = f(0) + \frac{f^{i}(0)*x}{1!} + \frac{f^{ii}(x)*x^{2}}{2!} + \ldots \frac{f^{n}(0)x^{n}}{n!}$$

(2)

Vamos calcular um exemplo bem simples para mostrar uma aplicação de (2).

$$\begin{aligned} &\text{Se } f(x) = \text{sen}(x) \text{ e } n = 2 \\ &f^{i}(x) = \cos(x) \\ &f^{ii}(x) = -sen(x) \\ &f(x) = \ sen(x) \cong sen(0) + \cos(0)x - sen(0)*x^{2}/2 \cong x \end{aligned} \quad \textbf{(3)}$$

O erro da aproximação acima é:

$$e(x) = sen(x) - x \quad \textbf{(4)}$$

Queremos que (4) seja igual a zero, ou seja, *e(x) = 0*. Isto ocorre quando x se aproxima de zero. Isto é, para valores bem pequenos a igualdade será verdadeira. Claramente percebemos que quanto maior a ordem da expansão da série de *Taylor* menor será o erro. Esta aproximação é comum na solução de problemas de Mecânica, particularmente, em oscilações quando linearizamos uma equação diferencial aproximando uma função senoidal por uma reta.

Podemos então, representar uma função por um polinômio. Assim, todo o problema consiste em implementar polinômios em Linguagem Ladder.

Forma geral do polinômio

Um polinômio pode ser representado na forma:

$Pn(x) = ao + a1*x^2 + a2*x^3 + \ldots\ldots\ldots\ldots an*x^n$ **(5)**

Ou simplesmente:

Pn(x) = v.u **(6)**

Onde:

v = (ao,a1,a2,a3.....an) **(7)**
u = $(1,x,x^2,x^3,\ldots\ldots\ldots x^n)$ **(8)**

(7) é o vetor dos coeficientes e (8) é a base ortonormal que gera o espaço vetorial dos polinômios de grau n.

O Polinômio Pn(x)

Vamos primeiro analisar o bloco de função ICT. Este bloco gera três valores inteiros constantes nas saídas OUT1, OUT2, e OUT3. A entrada EN apenas habilita o bloco, enquanto que a saída ENO indica que o bloco está ativo. Geralmente conectamos a entrada EN ao nível 1 e a saída ENO é conectada à uma bobina virtual.

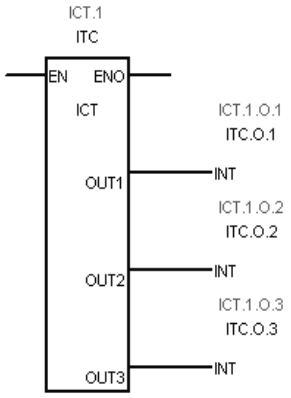

Fig 1- O Bloco de Função ICT

Vamos expandir o bloco acima de modo que possamos gerar todos os elementos do vetor *v*:

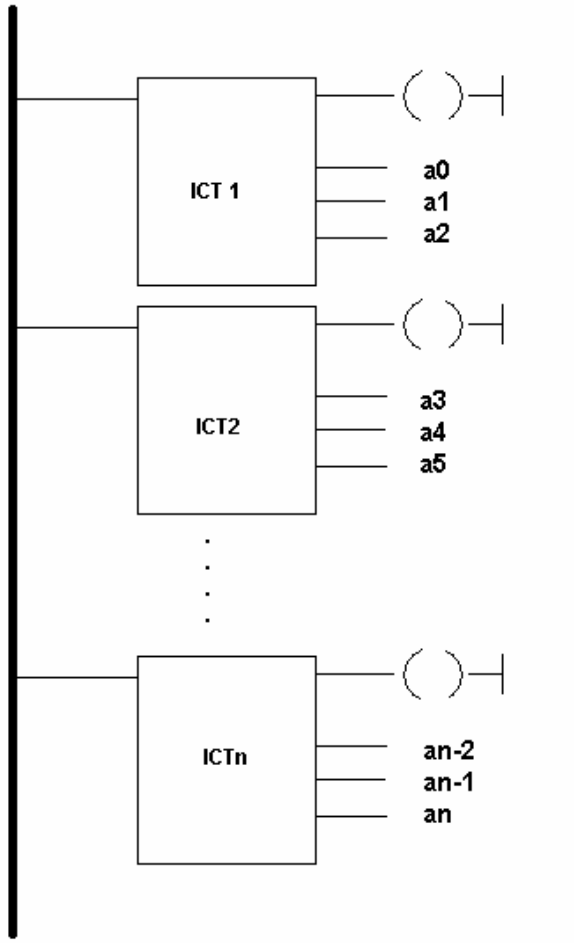

Fig 2- Gerando o vetor de coeficientes através do bloco ICT

Vamos representar a figura 2 como um bloco simples. Vamos utilizar uma saída do bloco ICT para gerar um valor 1 constante da base *u* . Assim, podemos genericamente representar o esquema da figura 2 como:

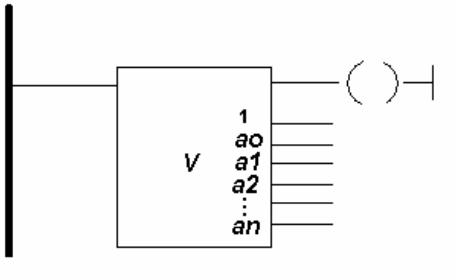

Fig 3- O Vetor v

Vamos agora gerar o vetor *u*, que é na verdade uma base. Para isso vamos utilizar o bloco de função MUL. Este bloco multiplica duas ou mais entradas. Este bloco pode ser conectado a uma entrada analógica real de um módulo de entrada e saída.

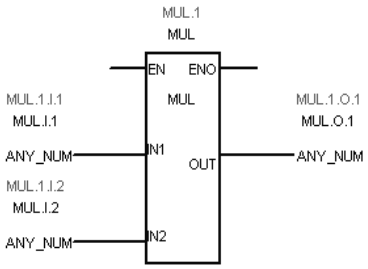

Fig 4- O Bloco de Função MUL

As entradas IN1 e IN2 podem ser logicamente ligadas às entradas analógicas. O que faremos é fazer IN1=IN2 para obtermos a função quadrática. De modo análogo, obtemos as outras potências x^3, x^4,.....x^n.

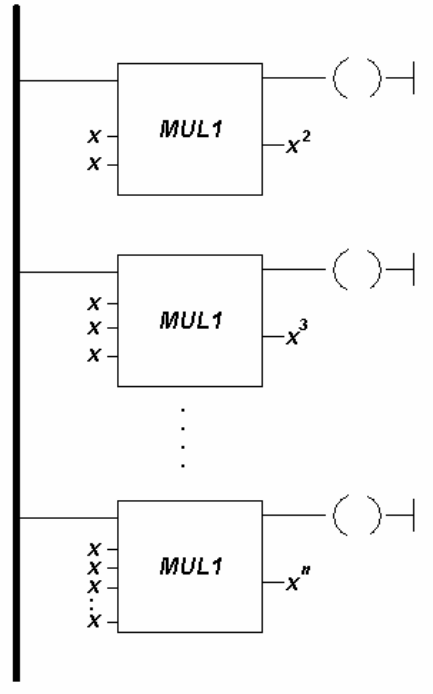

Fig 5- Gerando a base u através do bloco MUL

Ou ainda:

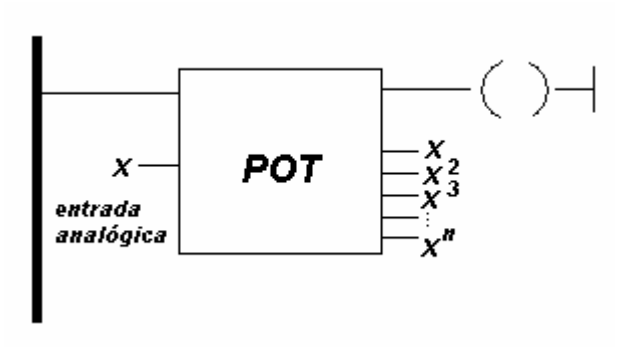

Fig 6- O Vetor u

Utilizando novamente blocos multiplicadores podemos multiplicar cada elemento de *v* por cada elemento de *u*.

Assim geramos os pares:

$ao*1+a1*x+a2*x^2+a3*x^3+.....an*x^n$

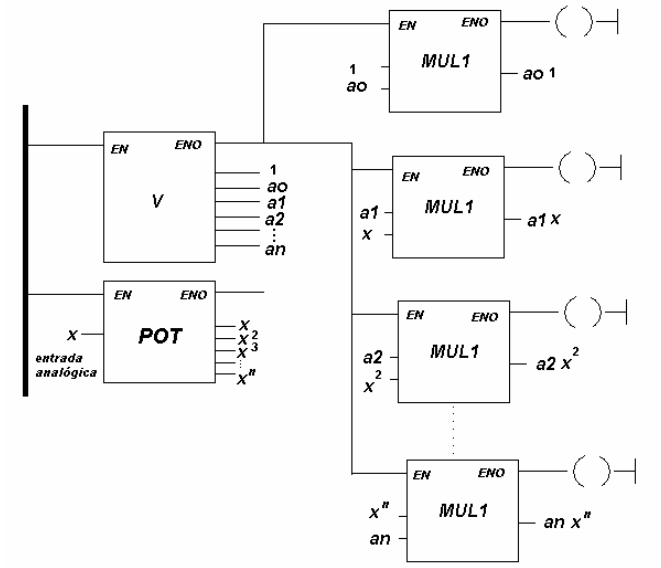

Fig 7- Gerando os pares an*xn

Vamos encapsular todo o diagrama da figura acima em apenas um bloco ilustrativo:

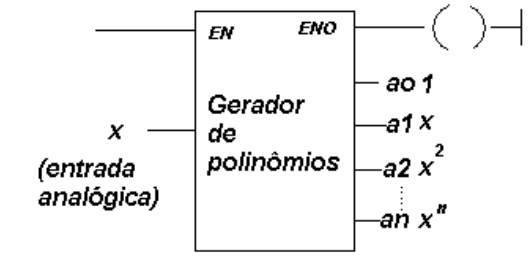

Fig 8- Gerador de polinômios

Falta apenas agora somar os pares *an*xn*. Para isso vamos utilizar o bloco de adição ADD.

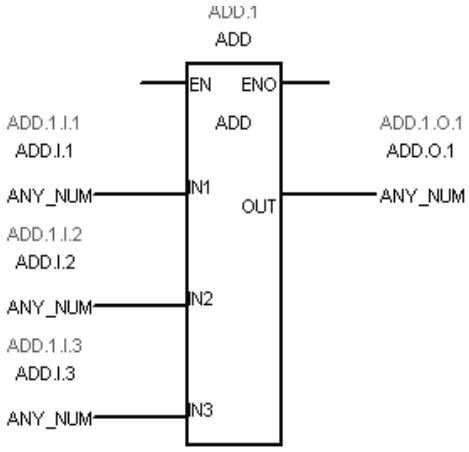

Fig 9- O Bloco ADD

Este bloco soma as entradas IN1, IN2 e retorna o resultado em OUT. Normalmente, existe limitação do número de entradas disponíveis.

Assim:

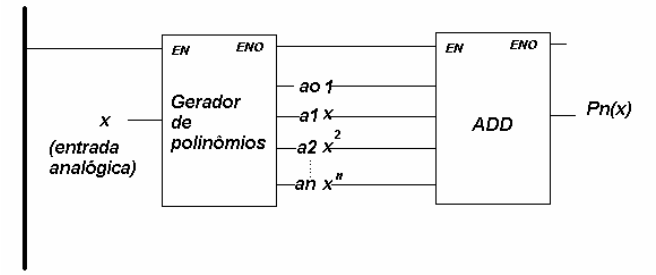

Fig 10- O Gerador de polinômios

Devemos saber claramente que o diagrama acima representa uma abstração. Em uma aplicação real, o usuário deverá dividir os blocos de função por várias redes lógicas.

Vamos mostrar apenas ilustrativamente, o mesmo programa em linguagem Java.

```
public class MeuPolinomio {

 private double v[];
 private int N;

 // construtor
 public MeuPolinomio(double coef[], int n) {
  v= new double[n];
  N = n;
  for(int i = 0; i < N; i++) {
         v[i]= coef[i];
  }
}

 public double MeuValor(double x) {
```

```
  double pot[] = new double[N];
  double aux[] = new double[N];
  double resultado = 0;

  // gera os valores de 1, x, x^2, x^3...x^n

  for(int i = 0; i < N; i++) {
         pot[i]= Math.pow(x,i);
  };

 // gera os pares aox,a1x,a2x^2...anx^n
 // o resultado é Pn(x)

  for(int  i = 0; i < N; i++) {
   aux[i] = pot[i]*v[i];
   resultado = aux[i] + resultado;
  };

  return resultado;
 }
}

class Polinomio {
 public static void main(String args[]) {

  int Dimensao = 2;
  double meusdados = {1,2,3};
  double x = 0.4;

  MeuPolinomio meupol = new
MeuPolinomio(meusdados,Dimensao);

  System.out.println(meupol.MeuValor(x));
}
```

Encontrando o valor de uma função *f(x)*

Até o momento, nós elaboramos um programa utilizando blocos de função para calcular o valor de um polinômio Pn para um determinado valor de x.

Vamos agora utilizar a equação (3) para aproximar o valor de uma função *f(x)= sen(x)* por um uma série de *Taylor* de ordem 2.

Através de (3) verificamos que o valor de u é:

u = (0,1,0)
v = (1,x,x^2)

O valor de x é determinado pela entrada analógica. Normalmente se trata de um valor lido de um instrumento de campo. Este valor é conectado a um módulo de entrada analógica e este valor é passado para a entrada dos blocos de função.