# *Gerando Polinômios e Funções Em Linguagem De Blocos De Função IEC-61131-3*

AUTOR: DANIEL VASCONCELOS GOMES
***danielgomes@gmail.com***

# ÍNDICE

# *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));
 }
}
```

***Listagem 1: Código em Java.***

Em c++:

```
#include <cmath>
#include <iostream>

using std:: cout;
using std:: endl;

int main( ) {
        double argumento;
        argumento = 60;
        cout << sin(argumento) << endl;
        return 0;
}
```

***Listagem 2: Código em C++.***

Como a linguagem de Blocos de Função IEC 61131-3 não é uma linguagem estruturada ou mesmo procedural, para implementar estes cálculos precisamos utilizar o recurso de aproximar a função por um polinômio.

No entanto, quando o programador deseja enviar e receber dados nas linguagens citadas, é preciso 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. Outra, é a aproximação por polinômios, ou, regressão polinomial. Inicialmente descreveremos alguns métodos para aproximar funções por polinômios e em seguida vamos descrever como programar um polinômio utilizando a linguagem Ladder.

Microcontroladores e microprocessadores não calculam funções como seno, cosseno, tangente, etc. Os desenvolvedores de compiladores simplesmente implementam essas funções utilizando aproximações por séries.

## A Série de Taylor

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

$$f(x) = g \qquad (1)$$

A série de *Taylor* [SIMMONS] é um recurso matemático que nos permite obter uma aproximação desta função.

A série pode ser representada por:

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

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

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

O valor n é a ordem da aproximação da série de Taylor. Ou seja, estamos fazendo n = 2 na equação (2).

O erro da aproximação acima é:

$$e(x) = sen(x) - x$$

Queremos que o erro 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.

## Onde utilizar?

Obviamente, o leitor se indagará onde poderá utilizar estas aproximações.

Funções senoidais são sempre úteis, pois estão presentes em processos naturais oscilatórios. Em circuitos elétricos, a função cosseno nos dá o fator de potência. Funções exponenciais estão presentes em cálculos que envolvam temperatura. Logaritmos são comuns para várias aplicações. Filtros podem ser implementados utilizando polinômios.

Há um outro caso. Quando obtemos dados experimentais que representam uma função. Por exemplo: levantamos um gráfico da temperatura de um tanque versus o tempo e queremos representar estes dados. Para obtermos uma função discreta de f(x) precisaremos interpolar os dados. Normalmente, as intepolações são lineares, isto é, os dados são ajustados à uma reta ax+b, mas polinômios de ordem maior podem ser encontrados.

### O que vamos precisar?

Um programa como o Microsoft Excel resolve a maioria de nossos problemas, mas com as descrições dos algoritmos, forneço alternativas se o leitor desejar implementar os códigos por si mesmo/a.

# ***ENCONTRANDO AS APROXIMAÇÕES***

### Introdução

Parte do problema consiste em encontrar um polinômio que represente uma função. Podemos fazê-lo de diversas maneiras, mas vou concentrar nossa análise em dois métodos: Série de Taylor e Interpolação polinomial

### A Série de Taylor

Vamos retomar a equação (2):

$$f(x) = f(0) + \frac{f^{i}(0) * x}{1!} + \frac{f^{ii}(0) * x^{2}}{2!} + \ldots \frac{f^{n}(0) x^{n}}{n!} \qquad \textbf{(4)}$$

Para uma prova do teorema acima, sugiro que o leitor verifique as referências [SIMMONS], ou ainda, envie-me um email e prontamente enviarei a demonstração.

***F(x)*** é uma função contínua em um intervalo finito e possui derivadas em x = 0.

Assim, neste artigo quando nos referirmos às derivadas de uma função f(x) utilizaremos a convenção:

$$F^{n}(x) = df(x)/d^{n}x$$

***F'(0)*** é a derivada de ordem n da função f(x) original calculada no ponto x = 0.

***n!*** é representa n fatorial. Por exemplo: 3! É igual a 3*2*1. Ou, 6! = 6*5*4*3*2*1.

O objetivo é fornecer ferramentas para calcular as aproximações e não ensinar cálculo diferencial. No entanto, forneceremos fórmulas prontas para uso para algumas funções mais comuns.

As fórmulas teóricas para o erro são bastante complexas e pouco práticas. O erro pode ser calculado numericamente por:

$$Erro(x) = ( f_{original} - f_{aproximada}/f_{original}) * 100 \quad (5)$$

**As derivadas**

Abaixo fornecemos uma tabela com as derivadas de algumas funções mais comuns:

***Tabela 1- Derivadas de ordem 1 até 5***

| F(x) | F'(x) | F''(x) | F'''(x) | $F^{iv}(x)$ | $F^{v}(x)$ |
|---|---|---|---|---|---|
| sen(ax) | acos(ax) | $-a^2$sen(ax) | $-a^3$cos(ax) | $a^4$sen(ax) | $a^5$cos(ax) |
| cos(ax) | -asen(ax) | $-a^2$cos(ax) | $a^3$sen(ax) | $a^4$cos(ax) | $-a^5$sen(ax) |
| exp(ax) | aexp(ax) | $a^2$exp(ax) | $a^3$exp(ax) | $a^4$exp(ax) | $a^5$exp(ax) |

Vamos agora analisar a função f(x) = ln(x). Sua derivada primeira é 1/x, logo não existe um valor para ela no ponto x = 0. Assim, precisaremos expandir a função ln(x) em uma série de potências genérica. Novamente, deixo ao leitor curioso, a verificação desta relação (para isso, veja [SIMMONS] página 97).

**As séries**

Podemos fazer x = 0 na tabela 1 somente nas 3 primeiras linhas e obteremos a seguinte tabela:

***Tabela 2- Derivadas para x = 0***

| F(0) | F'(0) | F''(0) | F'''(0) | $F^{iv}(0)$ | $F^{v}(0)$ |
|---|---|---|---|---|---|
| sen(0) = 0 | acos(0) = a | $-a^2$sen(0) = 0 | $-a^3$cos(0) = $-a^3$ | $a^4$sen(0)=0 | $-a^5$cos(0) = $a^5$ |
| cos(0) = 1 | -asen(0) = 0 | $-a^2$cos(0) = $-a^2$ | $a^3$sen(0)=0 | $a^4$cos(0)= $a^4$ | $-a^5$sen(0)=0 |
| exp(0) = 1 | aexp(0) = a | $a^2$exp(0) = $a^2$ | $a^3$exp(0) = $a^3$ | $a^4$exp(0) = $a^4$ | $a^5$exp(0) = $a^5$ |

Assim:

***Tabela 3- Aproximações***

| F(x) | $F_{aproximada}$ |
|---|---|
| sen(ax) | $ax - a^3x^3/3! + a^5x^5/5!$ |
| cos(ax) | $1 - a^2x^2/2! + a^4x^4/4!$ |
| exp(ax) | $1 + ax + a^2x^2/2! + a^3x^3/3! + a^4x^4/4! + a^5x^5/5!$ |

Para calcular o erro da aproximação, basta utilizar (5).

Estas expansões de ordem igual a 5 são suficientes para obter excelente precisão, mas as expansões podem ser expressas em fórma de série e expandidas conforme a necessidade do leitor/a.

***Tabela 4- As Séries***

| F(x) | $F_{aproximada}$ |
|---|---|
| sen(x) | $\sum_{n=0}^{\infty}(-1)^n * \frac{x^{2n+1}}{(2n+1)!}$ |
| cos(x) | $\sum_{n=0}^{\infty}(-1)^n * \frac{x^{2n}}{(2n)!}$ |
| exp(x) | $\sum_{n=0}^{\infty}\frac{x^n}{n!}$ |

Vamos agora, mostrar as planilhas do Microsoft Excel criadas para fazer os cálculos. Para f(x) = sen(ax):

| | Graus: | Radianos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Digite o valor de a:** | 1 | | | | | | | | | |
| **Digite o valor de x:** | 30 | 0.5236 | | | | | | | | |
| | | | | | | | | | | |
| **Série:** | | | | | | | | | | |
| | ao | a1 | a3 | a4 | a5 | a6 | faprox(x) | f(x) | erro(x) | erro% |
| | 0 | 0.5236 | 0 | -0.0239 | 0 | 0.0003 | 0.5000021 | 0.5 | 2.13E-06 | 0.0004265 |
| | | | | | | | | | | |
| | | | | | | | | | | |
| *sen(ax) = a0(x) + a1(x) + a2(x) + a3(x) + a4(x) +a5(x) +a6(x)* | | | | | | | | | | |

***Fig 1- Planilha do Excel para Calcular A série de Taylor de sen(ax)***

Para f(x) = sen(ax):

| | Graus: | Radianos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Digite o valor de a:** | 1 | | | | | | | | | |
| **Digite o valor de x:** | 30 | 0.5236 | | | | | | | | |
| | | | | | | | | | | |
| **Série:** | | | | | | | | | | |
| | ao | a1 | a3 | a4 | a5 | a6 | faprox(x) | f(x) | erro(x) | erro% |
| | 1 | 0 | -0.137 | 0 | 0.0031 | 0 | 0.866054 | 0.866 | -2.84796E-05 | 0.0033 |
| | | | | | | | | | | |
| | | | | | | | | | | |
| *cos(ax) = a0(x) + a1(x) + a2(x) + a3(x) + a4(x) +a5(x) +a6(x)* | | | | | | | | | | |

***Fig 2- Planilha do Excel para Calcular A série de Taylor de cos(ax)***

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Digite o valor de a:** | 1 | | | | | | | | | |
| **Digite o valor de x:** | 1 | | | | | | | | | |
| | | | | | | | | | | |
| Série: | | | | | | | | | | |
| | ao | a1 | a3 | a4 | a5 | a6 | faprox(x) | f(x) | erro(x) | erro% |
| | 1 | 1 | 0.5 | 0.1667 | 0.0417 | 0.0083 | 2.716666667 | 2.7183 | 0.0016 | 0.0594 |
| | | | | | | | | | | |
| *exp(ax) = a0(x) + a1(x) + a2(x) + a3(x) + a4(x) +a5(x) +a6(x)* | | | | | | | | | | |

***Fig 3- Planilha do Excel para Calcular A série de Taylor de exp(ax)***

# ***Interpolação Polinomial Discreta***

Mas o que acontece quando não temos uma função f(x) e sim um conjunto de dados experimentais? Nesse caso, utilizamos a interpolação polinomial.

A interpolação polinomial também é útil para funções complexas e podemos simplesmente representar esta função com um polinômio de ordem suficiente para que o erro seja mínimo.

Seja uma função dada por (1), queremos encontrar o polinômio:

$$Pm=a_o+a_1x+a_2x^2+.....a_nx^n \quad (6)$$

Onde n é o grau do polinômio Pm.

Vamos definir o produto interno:

$$(f,g)=\sum_{k=0}^{n} f(xk)g(xk) \qquad (7)$$

O problema se resume a calcular os coeficientes ao,a1,a2,....an.

Definindo:

$$y = \begin{pmatrix} yo \\ y1 \\ y2 \\ . \\ . \\ yn \end{pmatrix} \quad \text{e} \quad p = \begin{pmatrix} Pm(xo) \\ Pm(x1) \\ Pm(x2) \\ . \\ . \\ Pm(xn) \end{pmatrix} \qquad \textbf{(8)}$$

Substituindo (8) em (6):

$$p = a_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + a_1 \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} + a_2 \begin{pmatrix} x_0^2 \\ x_1^2 \\ \vdots \\ x_n^2 \end{pmatrix} + \ldots + a_m \begin{pmatrix} x_0^m \\ x_1^m \\ \vdots \\ x_n^m \end{pmatrix} .$$

Podemos então reescrever:

$$u_o = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix} \quad \text{e} \quad u_i = \begin{pmatrix} xo^i \\ x1^i \\ . \\ . \\ xn^i \end{pmatrix} \text{para i} = 1,2,....\text{m} \qquad \textbf{(9)}$$

Logo (6) se torna:

Pm = $a_o u_o + a_1 u_1 + \ldots\ldots\ldots a_m u_m$.

Os coeficientes ao,a1,a2….am são determinados através do sistema:

$$\begin{pmatrix} (uo,uo) & (u1,uo) & . & (um,uo) \\ (uo,u1) & (u1,u1) & . & (um,u1) \\ . & . & . & . \\ (uo,um) & (u1,um) & . & (um,um) \end{pmatrix} \begin{pmatrix} ao \\ a1 \\ . \\ am \end{pmatrix} = \begin{pmatrix} (y,uo) \\ (y,u1) \\ . \\ (y,um \end{pmatrix} \qquad \textbf{(10)}$$

Neste caso utilizamos uma base ortonormal :

$B = (1,x,x^2,x^3.....x^m)$.

Felizmente, o Microsoft Excel faz esta interpolação automaticamente.

Vamos mostrar um exemplo de regressão polinomial de ordem 4 (isto é, a função original será representada por um polinômio de ordem 4) da função $f(x) = 1/(1+x)^2$.

| x | 1/(1+x)^2 | faproximada | erro |
|---|---|---|---|
| 0.2 | 0.694444444 | 0.65110416 | 0.04334028 |
| 0.4 | 0.510204082 | 0.52456176 | -0.01435768 |
| 0.6 | 0.390625 | 0.41971856 | -0.02909356 |
| 0.8 | 0.308641975 | 0.33397776 | -0.02533578 |
| 1 | 0.25 | 0.2649 | -0.0149 |
| 1.2 | 0.20661157 | 0.21020336 | -0.00359179 |
| 1.4 | 0.173611111 | 0.16776336 | 0.00584775 |
| 1.6 | 0.147928994 | 0.13561296 | 0.01231603 |
| 1.8 | 0.12755102 | 0.11194256 | 0.01560846 |
| 2 | 0.111111111 | 0.0951 | 0.01601111 |
| 2.2 | 0.09765625 | 0.08359056 | 0.01406569 |
| 2.4 | 0.08650519 | 0.07607696 | 0.01042823 |
| 2.6 | 0.077160494 | 0.07137936 | 0.00578113 |
| 2.8 | 0.069252078 | 0.06847536 | 0.00077672 |
| 3 | 0.0625 | 0.0665 | -0.004 |
| 3.2 | 0.056689342 | 0.06474576 | -0.00805642 |
| 3.4 | 0.051652893 | 0.06266256 | -0.01100967 |
| 3.6 | 0.047258979 | 0.05985776 | -0.01259878 |
| 3.8 | 0.043402778 | 0.05609616 | -0.01269338 |
| 4 | 0.04 | 0.0513 | -0.0113 |
| 4.2 | 0.036982249 | 0.04554896 | -0.00856671 |
| 4.4 | 0.034293553 | 0.03908016 | -0.00478661 |
| 4.6 | 0.031887755 | 0.03228816 | -0.0004004 |
| 4.8 | 0.029726516 | 0.02572496 | 0.00400156 |
| 5 | 0.027777778 | 0.0201 | 0.00767778 |
| 5.2 | 0.026014568 | 0.01628016 | 0.00973441 |
| 5.4 | 0.024414063 | 0.01528976 | 0.0091243 |
| 5.6 | 0.022956841 | 0.01831056 | 0.00464628 |
| 5.8 | 0.021626298 | 0.02668176 | -0.00505546 |
| 6 | 0.020408163 | 0.0419 | -0.02149184 |

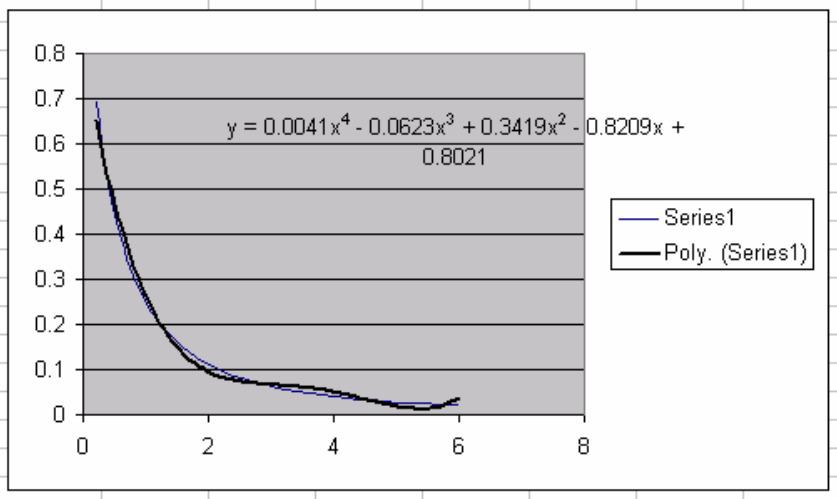


***Fig 4- Interpolação polinomial de ordem 4 no Microsoft Excel***

# ***POLINÔMIOS EM LADDER***

## Forma geral do polinômio

Um polinômio pode ser representado na forma:

$Pn(x) = ao + a1*x^2 + a2*x^3 + .....an*x^n$ **(11)**

Ou simplesmente:

$Pn(x) = v.u$ **(12)**

Onde:

$v = (ao,a1,a2,a3.....an)$ **(13)**

$u = (1,x,x^2,x^3,.....x^n)$ **(14)**

(13) é o vetor dos coeficientes e (14) 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.

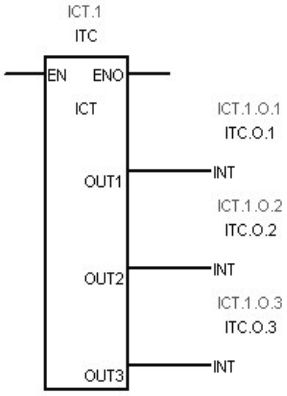


***Fig 5- O Bloco de Função ICT***

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

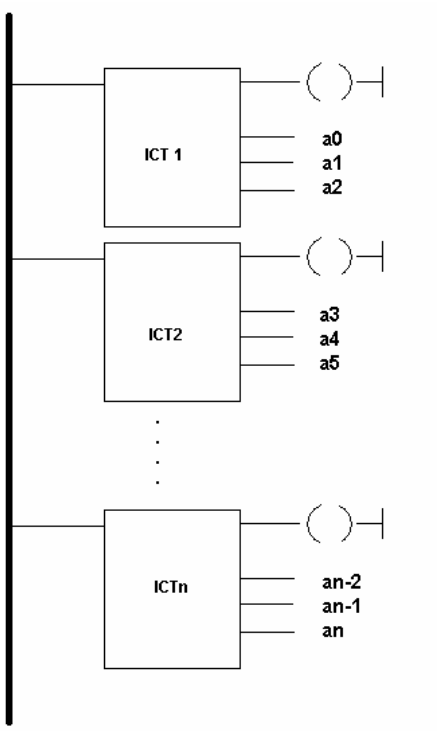


***Fig 6- Gerando o vetor de coeficientes através do bloco ICT***

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:

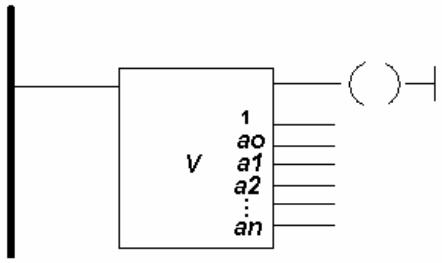


***Fig 7- O Vetor v***

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.

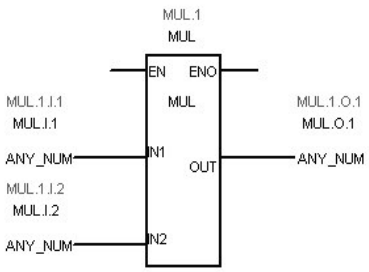


***Fig 8- O Bloco de Função MUL***

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 $x^3$, $x^4$,.....$x^n$.

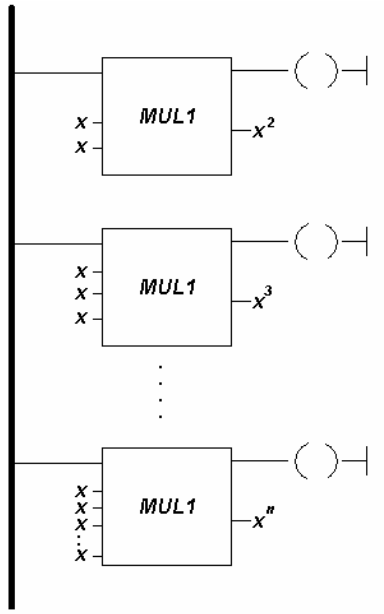


***Fig 9- Gerando a base u através do bloco MUL***

Ou ainda:

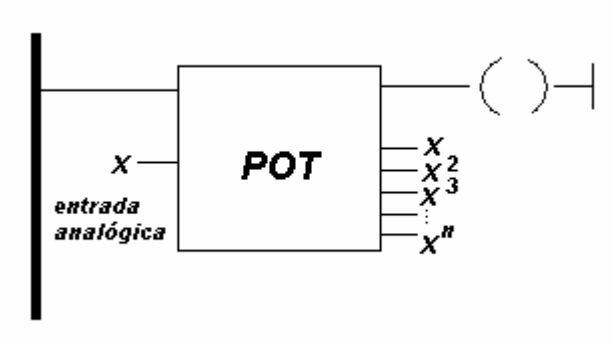


***Fig 10- O Vetor u***

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

Assim geramos os pares:

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

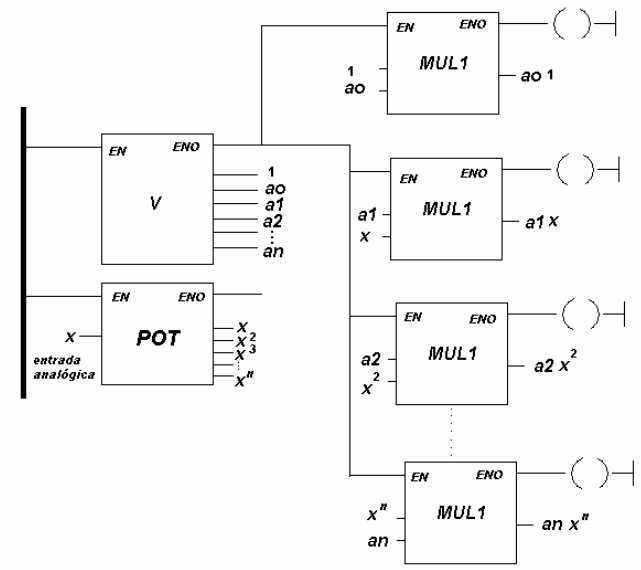


***Fig 11- Gerando os pares an*xn***

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

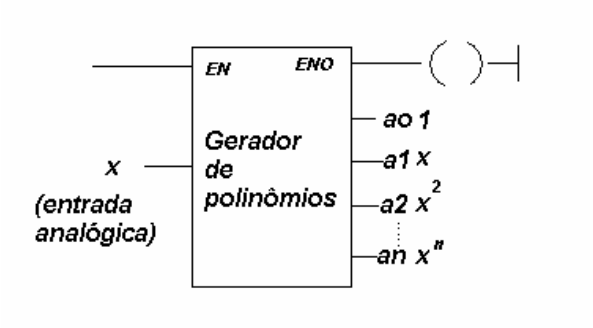


***Fig 12- Gerador de polinômios***

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

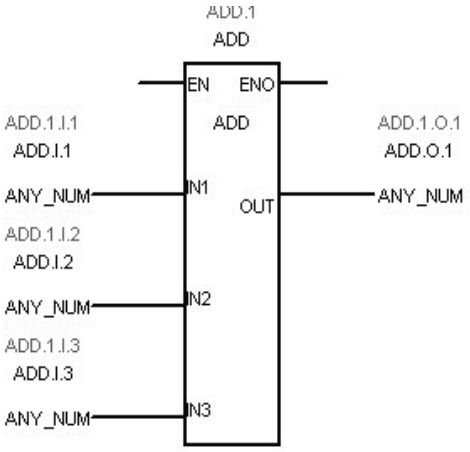


***Fig 13- O Bloco ADD***

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

Assim:

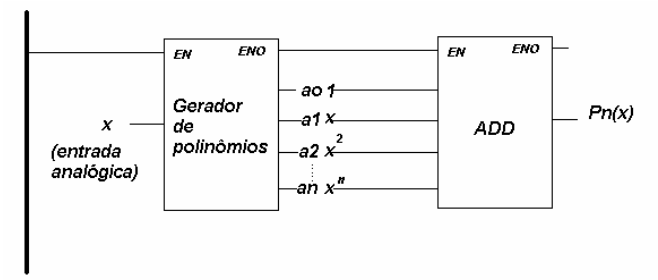


***Fig 14- O Gerador de polinômios***

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.

## 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.

# *NOTAS FINAIS*

Caso seja do interesse do leitor/a, por favor envie-me um e-mail solicitando informações, pois implementei a maioria desses algoritmos em C++, Java e Pascal além de outras técnicas e posso, sem problema algum, enviar ao leitor/a curioso/a. Além disso, quaisquer problemas com o uso do Excel e dicas sobre esse software, podem ser solicitados.

Meu web site é: http://www.tecnologiaeoutros.hpg.ig.com.br

# *REFERÊNCIAS*

**[SIMMONS]**
**Simmons, George F.** ; ***Cálculo Com Geometria Analítica vol. 2*** . Makron Books Do Brasil Editora Ltda. São Paulo 1993.

**[NEIDE]**
**M. B. Franco, Neide.; Cálculo Numérico**. http://www.lcad.icmc.usp.br/~siae98/