Boletim da SPM - nº 42
Maio de 2000

TANGENTES, EXPONENCIAIS, ... COMO SE OBTÊM NUMA CALCULADORA?*

Maria de Jesus Sousa Vieira Bugalho
Esc. Sec. Seomara da Costa Primo

1. Introdução

As calculadoras são instrumentos cada vez mais utilizados no nosso quotidiano. Torna-se, por isso, importante perceber como funcionam, quais os seus mecanismos internos e qual o contributo da Matemática nesse campo. Na realidade, são muitas vezes utilizadas para cálculos que envolvem funções transcendentes, mas sem haver a preocupação, sequer, de saber como são feitos esses cálculos, de saber como é que as máquinas "pensam" e "calculam" o que lhes é pedido.

Um primeiro método que se pode usar é o das aproximações polinomiais que tem por objectivo encontrar um polinómio que, num determinado intervalo, esteja sempre o mais próximo possível da função em estudo. Efectivamente, o que se pretende é que o valor numérico obtido por aplicação de um certo polinómio ao valor sobre o qual se pretende efectuar os cálculos, seja uma "boa" aproximação do valor correspondente obtido por aplicação da função em questão, em qualquer ponto de um certo intervalo. Neste processo, um dos possíveis polinómios a utilizar é o de Taylor, embora para as calculadoras se utilizem outros polinómios que dão melhores aproximações sem aumentar o numero de operações a efectuar.

Outro método para o cálculo numérico de funções trigonométricas, é o método dos algoritmos Cordic que foi desenvolvido cm 1959 por J. E. Volder. A abreviatura significa COordinate Rotation DIgital Computer e, tal como o nome indica, esta técnica tem como base a rotação de coordenadas. Volder aplicou esta técnica à construção de um computador digital para a navegação aérea. Os algoritmos Cordic dão origem a boas aproximações das funções

*Este artigo é baseado num trabalho de seminário realizado no âmbito do Mestrado em Matemática na área de Matemática para o Ensino na FCUL, no ano lectivo 1997/98, sob a orientação do Dr. Carlos Albuquerque

trigonométricas (e suas inversas) e podem ser utilizados para as funções hiperbólicas (e suas inversas), logarítmica e exponencial.

As aproximações polinomiais e os algoritmos Cordic constituem dois métodos bastante utilizados, não só por calculadoras gráficas mas, também, por co-processadores de matemática.

As calculadoras TI (Texas Instruments), que são muito utilizadas no ensino secundário, usam, na sua maioria, um algoritmo Cordic. Como exemplo entre os co-processadores numéricos, que fornecem com rapidez e grande precisão valores numéricos de funções logarítmicas e trigonométricas, temos o Intel 8087 e sucessores, que utilizam algoritmos Cordic, e temos o Cyrix 83D87, que utiliza o método polinomial. Neste artigo vamos apresentar dois algoritmos Cordic que têm como objectivo o cálculo da tangente de um ângulo q do primeiro quadrante ($0 \leq \theta < \pi/2$). Para determinar a tangente de um ângulo fora desse intervalo, usam-se as fórmulas de redução ao primeiro quadrante. O primeiro destes algoritmos a ser analisado é o utilizado pela Texas Instruments nas calculadoras TI-82 e TI-83 ([2]) e o segundo é o apresentado no Dr. Dobb's Journal ([3]). Por fim será apresentado um algoritmo Cordic para o cálculo da exponencial de um valor X na calculadora TI-83. Todos os cálculos foram efectuados numa calculadora TI-83.

2. Algoritmos Cordic para cálculos com funções trigonométricas

2.1. Rotação de vectores

Para fixarmos ideias acerca dos métodos a usar convém começar por recordar as expressões que caracterizam a rotação de um ponto em torno da origem num referencial o.n..

Consideremos um vector (X, Y) no primeiro quadrante, aplicado na origem do referencial. Da rotação deste vector em torno da origem resulta

um arco de circulo. Se tomarmos, como coordenadas iniciais, as coordenadas (X, Y), as coordenadas (X', Y'), obtidas por rotação a partir da posição inicial (X, Y), segundo um ângulo θ $(0 \leq \theta < \pi/2)$ e no sentido directo, podem ser descritas pelas equações:

$$\begin{cases} X' &= X\cos\theta - Y\,\mathrm{sen}\,\theta \\ Y' &= X\,\mathrm{sen}\,\theta + Y\cos\theta \end{cases}$$

$$\text{que são equivalentes a} \quad \begin{cases} \dfrac{X'}{\cos\theta} &= X - Y\cdot \mathrm{tg}\,\theta \\ \dfrac{Y'}{\cos\theta} &= Y + X\cdot \mathrm{tg}\,\theta \end{cases} \tag{1}$$

De modo análogo, no sentido inverso, temos as equações:

$$\begin{cases} \dfrac{X'}{\cos\theta} &= X + Y\cdot \mathrm{tg}\,\theta \\ \dfrac{Y'}{\cos\theta} &= Y - X\cdot \mathrm{tg}\,\theta \end{cases} \tag{2}$$

2.2. Algoritmo da TI-83

Tomamos um ângulo θ $(0 \leq \theta < \pi/2)$ do qual se pretende determinar a tangente. Consideramos um vector $R^N = (X^N, Y^N)$ definido a partir da origem, com comprimento fixo, $|R^N|$, e sobre o lado extremidade do ângulo θ. A rotação desse vector descreve um arco de circunferência de centro $(0, 0)$ e raio $|R^N|$. Por se tratar do estudo de uma razão trigonométrica, a tangente, é indiferente a medida do raio, pelo que podemos tomá-la como sendo uma unidade, isto é, $|R^N| = 1$. Para o cálculo da tangente do ângulo θ, "roda-se" o vector $(1, 0)$ sobre esse ângulo, no sentido directo, numa sequência

de passos, donde resulta uma sequência de ângulos de forma a que a soma de todos eles seja igual a θ. Nessa sequência de passos, a cada ângulo que se vai obtendo, corresponde uma posição (X, Y), pelo que, ficamos com uma sequência de valores X e Y de tal forma que, quando completamos o processo, $\frac{Y}{X} = \operatorname{tg}\theta$.

Nota: No texto que se segue, vão ser utilizadas duas notações: θ^h e θ_j. A notação θ^h representa os ângulos que entram na decomposição do ângulo inicial θ e que, por isso, se podem repetir. A notação θ_j representa os ângulos que surgem nas tabelas e que, por isso, não aparecem repetidos.

Neste algoritmo existem três passos fundamentais:

1. Decomposição do ângulo θ numa soma de ângulos θ_k positivos.

2. Determinação das coordenadas após cada rotação: tomando como coordenadas iniciais $(1, 0)$ e através de rotações no sentido directo, segundo os ângulos θ_k, obtemos um conjunto de coordenadas (X, Y), de tal forma que, quando completamos o processo, chegamos ao resultado pretendido.

3. Determinação de $\operatorname{tg}\theta$: $\operatorname{tg}\theta = \dfrac{Y^N}{X^N}$

Os ângulos θ_k, que entram na decomposição do ângulo θ, são da forma $\theta_k = \text{arctg}\, 10^{-k}$. Como veremos adiante (equações (3)),esta opção facilita muito a implementação das equações (1) já referidas. Nos nossos cálculos os valores numéricos obtidos para θ_k são sujeitos a aproximações com cinco casas decimais. Tal opção tornou viável a aplicação prática do algoritmo, tendo a máquina de calcular como único instrumento de trabalho. Tal procedimento deu origem a que k variasse, apenas, entre 0 e 5 (inclusivé) e permitiu que o algoritmo se executasse mais rapidamente sem se chegar a programá-lo. Constata-se que os resultados obtidos apresentam, todos, erros a partir da quinta casa decimal, o que é natural, dada a existência de erros de arredondamento.

Os valores de θ_k são apresentados na tabela seguinte:

K	10^{-k}	$\theta_k = \text{arctg}\, 10^{-k}$ (5 casas decimais)
0	1,00000	0,78540
1	0,10000	0,09967
2	0,01000	0,01000
3	0,00100	0,00100
4	0,00010	0,00010
5	0,00001	0,00001

Por conter um número pequeno de elementos, uma tabela deste tipo já faz parte do mecanismo interno da máquina. Note-se, no entanto, que devido aos arredondamentos para, apenas, 5 casas decimais efectuados nos valores de θ_k, a tabela aqui apresentada é mais pequena do que aquela que faz parte da calculadora.

Vamos agora ver como é que se aplica este algoritmo num caso concreto: cálculo de $\text{tg}\, 0,2$ ($\theta = 0,2$ radianos).

1) Determinação dos ângulos θ_k que decompõem $\theta = 0,2$.

Partimos do ângulo $\theta = 0,2$ e, utilizando a tabela auxiliar, vamos procurar o maior ângulo θ_k que "cabe" em $0,2$, ou seja, tal que $\theta - \theta_k \geq 0$. Verificamos então, que se trata do ângulo θ_1 e que $\alpha_1 = \theta - \theta_1$ é o ângulo que falta percorrer para atingirmos θ:

$$\begin{aligned} 0,2 &= \theta_1 + \alpha_1 \\ &= 0,09967 + 0,10033 \end{aligned}$$

De modo análogo, para o ângulo α_1, procuramos o maior θ_k que "cabe" em α_1 e verificamos que é novamente θ_1. Obtemos, então, $\alpha_2 = \alpha_1 - \theta_1 = \theta - 2 \cdot \theta_1$:

$$\begin{aligned} 0,2 &= \theta_1 + \theta_1 + \alpha_2 \\ &= 2 \cdot 0,09967 + 0,00066 \end{aligned}$$

Analogamente obtemos

$$\begin{aligned} 0,2 &= \theta_1 + \theta_1 + \theta_4 + \alpha_3 \\ &= 2 \cdot 0,09967 + 0,0001 + 0,00056 \end{aligned}$$

Continuando por este processo chegaríamos à conclusão de que, usando 5 casas decimais, a decomposição de $\theta = 0,2$ é

$$0,2 = 2\theta_1 + 6\theta_4 + 6\theta_5$$

Depois do ângulo inicial decomposto, o passo seguinte é a determinação das coordenadas após cada rotação.

2) Determinação das coordenadas após cada rotação

Este algoritmo pressupõe como coordenadas iniciais (1,0). Depois, procede-se a rotações, no sentido directo, segundo os ângulos $\theta_k = \operatorname{arctg} 10^{-k}$ determinados no ponto anterior. Assim, as equações que nos dão as coordenadas depois de cada rotação são:

$$\begin{cases} X^{i+1} &= \cos\theta_k (X^i - Y^i \cdot 10^{-k}) \\ Y^{i+1} &= \cos\theta_k (Y^i + X^i \cdot 10^{-k}) \end{cases} \Leftrightarrow \begin{cases} \dfrac{X^{i+1}}{\cos\theta_k} &= X^i - Y^i \cdot 10^{-k} \\ \dfrac{Y^{i+1}}{\cos\theta_k} &= Y^i + X^i \cdot 10^{-k} \end{cases} \quad (3)$$

Mas estas equações não são exactamente as que são utilizadas pelo algoritmo em estudo. Efectivamente, as equações que constam neste algoritmo omitem o factor multiplicativo $\cos\theta_k$ e são

$$\begin{cases} S^{i+1} &= S^i - T^i \cdot 10^{-k} \\ T^{i+1} &= T^i + S^i \cdot 10^{-k} \end{cases} \tag{4}$$

em que as duas variáveis são inicializadas em $(S^0, T^0) = (1, 0)$.

Como podemos constatar, no segundo membro destas equações, além de somas e diferenças existem, apenas, multiplicações por potências de base 10. Tal facto facilita bastante a implementação do algoritmo pois a calculadora trabalha no sistema decimal. É que, deste modo, as multiplicações por potências de base 10 correspondem, apenas, a alterações do expoente do número a multiplicar[1].

Aplicando (4) em vez de (3) vamos obter a seguinte sequência de coordenadas:

$$(1,0), \left(\frac{X^1}{\cos\theta^1}, \frac{Y^1}{\cos\theta^1}\right), \left(\frac{X^2}{\cos\theta^1 \cdot \cos\theta^2}, \frac{Y^2}{\cos\theta^1 \cdot \cos\theta^2}\right) \ldots \\ \ldots \left(\frac{X^n}{\cos\theta^1 \cdot \cos\theta^2 \ldots \cos\theta^N} \frac{Y^n}{\cos\theta^1 \cdot \cos\theta^2 \ldots \cos\theta^N}\right), \tag{5}$$

onde $\theta^i, \theta^j \in \{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$ e θ_i, θ_j não têm de ser diferentes, mesmo se $i \neq j$.

Depois de se completar o processo, o quociente entre a ordenada e a abcissa do último par de coordenadas é igual ao valor da $\operatorname{tg}\theta$. No caso da sequência (5), temos então

$$\frac{S^N}{T^N} = \frac{\dfrac{Y^N}{\cos\theta^1 \cdot \cos\theta^2 \ldots \cos\theta^N}}{\dfrac{X^N}{\cos\theta^1 \cdot \cos\theta^2 \ldots \cos\theta^N}} = \frac{Y^N}{X^N} = \operatorname{tg}\theta.$$

Chegámos, assim, ao mesmo resultado que teríamos chegado se tivéssemos usado as equações (3) usando, desde o início de todo o processo iterativo, os factores $\cos\theta_k$. No entanto, embora se obtenha o mesmo resultado, deve-se notar que a sequência de coordenadas da forma $(S^i, T^i) = \left(\frac{X^i}{\cos\theta^1 \ldots \cos\theta^i}, \frac{Y^i}{\cos\theta^1 \ldots \cos\theta^i}\right)$ (sequência (5)), é diferente da sequência de coordenadas da forma (X^i, Y^i) obtida por aplicação das equações que constam

[1] A calculadora trabalha com a notação científica (base 10) de números reais. Esta notação permite exprimir um número real x na forma $x = \pm m \cdot 10^t$ onde $0 \leq m < 10$ é designado por mantissa, 10 é a base e t é um inteiro designado por expoente ([5]).

no primeiro sistema. Enquanto os pontos (X^i, Y^i) estão a uma distância 1 da origem, os pontos (S^i, T^i) estão a uma distância superior ou igual a 1, pois

$$\begin{aligned}\|(S^i, T^i)\| &= \sqrt{(S^i)^2 + (T^i)^2} = \sqrt{\frac{(X^i)^2}{\cos^2\theta^1 \ldots \cos^2\theta^i} + \frac{(Y^i)^2}{\cos^2\theta^1 \ldots \cos^2\theta^i}} \\ &= \frac{\|(X\ i, Y^i)\|}{|\cos\theta^1 \ldots \cos\theta^i}\end{aligned}$$

e como $\|(X^i, Y^i)\| = 1$ e $0 < \cos\theta^i \leq 1$ então $\|(S^i, T^i)\| = \frac{1}{\cos\theta^1 \ldots \cos\theta^i} \geq 1$.

Mas como o objectivo deste algoritmo é o cálculo da tangente de um ângulo θ, pode-se omitir o factor $\cos\theta_k$ nas equações (3). É que, para além de se chegar ao mesmo resultado $\mathrm{tg}\,\theta = \frac{Y^N}{X^N}$, o algoritmo torna-se mais fácil de implementar, executa-se mais rapidamente e provoca menos erros de arredondamento.

Voltando, agora, ao caso em que $\theta = 0,2$, em que a decomposição de θ é $0,2 = 2\theta_1 + 6\theta_4 + 6\ \theta_5$, a situação é a seguinte:

A partir das coordenadas iniciais (1,0) e, por aplicação das equações (4), obtemos as coordenadas (S^1, T^1) que são, respectivamente, X^1 e Y^1 a dividir por $\cos\theta_1$, onde θ_1 é o ângulo segundo o qual se deu a "rotação". Depois, por "rotação" segundo o ângulo seguinte, θ_1, obtemos (S^2, T^2). E assim sucessivamente, até se obter o último par de coordenadas (S^{14}, T^{14}) (14 é o número de ângulos em que se decompõe $\theta = 0,2$).

Os resultados obtidos para as coordenadas (S^k, T^k), para os ângulos α_k, para os quocientes $\dfrac{T^k}{S^k}$ e para as distâncias de (S^k, T^k) à origem são apresentados na Tabela 1.

K	α_k (5 casas decimais)	S^K	T^K	T^K/S^K	$\|(S^K, T^K)\|$
0	0.2	1	0	0	1.0000000000
1	0.10033	1.0000000000	0.1000000000	0.1000000000	1.0049875621
1	0.00066	0.9900000000	0.2000000000	0.2020202020	1.0100000000
4	0.00056	0.9899800000	0.2000990000	0.2021242853	1.0100000051
4	0.00046	0.9899599901	0.2001979980	0.2022283729	1.0100000101
4	0.00036	0.9899399703	0.2002969940	0.2023324646	1.0100000152
4	0.00026	0.9899199406	0.2003959880	0.2024365606	1.0100000202
4	0.00016	0.9898999010	0.2004949800	0.2025406607	1.0100000253
4	0.00006	0.9898798515	0.2005939700	0.2026447651	1.0100000303
5	0.00005	0.9898778456	0.2006038688	0.2026551758	1.0100000304
5	0.00004	0.9898758395	0.2006137676	0.2026655865	1.0100000304
5	0.00003	0.9898738334	0.2006236663	0.2026759972	1.0100000305
5	0.00002	0.9898718272	0.2006335651	0.2026864081	1.0100000306
5	0.00001	0.9898698208	0.2006434638	0.2026968189	1.0100000305
5	0	0.9898678144	0.2006533625	0.2027072298	1.0100000306

Tabela 1

Notemos que, efectivamente, as distâncias de (S^k, T^k) à origem são superiores ou iguais a um. No entanto, o valor da tangente é igual ao que seria obtido pela sequência de coordenadas (X^k, Y^k):

$$\operatorname{tg} 0,2 = \frac{T^{14}}{S^{14}} = \frac{Y^{14}}{X^{14}} = 0,2027072298$$

O valor dado pela função tangente da TI-83 é $\operatorname{tg} 0,2 = 0,2027100355$.

Tal como se pretendia, o resultado obtido por $\dfrac{Y}{X}$ no final do algoritmo é aproximadamente igual ao resultado obtido na calculadora TI-83 para $\operatorname{tg} 0,2$ (o erro é inferior a $3 \cdot 10^{-6}$).

Acabámos de analisar a aplicação do algoritmo da TI-83 a um caso concreto. Para qualquer outro caso o procedimento seria o mesmo. Podemos assim apresentar de uma forma geral o algoritmo da TI-83.

Algoritmo da TI-83
(adaptado a 5 casas decimais)

---

1) Determinação dos valores $n_k \in N_0$ a partir da decomposição do ângulo inicial ($\theta \in [0, \pi/2[$), usando a tabela da pag. 5

$$\theta \simeq \sum_{k=0}^{5} n_k \theta_k \quad \text{onde} \quad \theta_k = \operatorname{arctg} 10^{-k}$$

2) Inicialização de duas variáveis com os valores $\begin{cases} S^0 = 1 \\ T^0 = 0 \end{cases}$

3) Aplicação recursiva das equações (4):

$$\begin{cases} S^{i+1} = S^i - T^i \cdot 10^k \\ T^{i+1} = T^i + S^i \cdot 10^{-k} \end{cases} \qquad \text{onde } i \text{ varia de 0 até } \sum_{k=0}^{5} n_k - 1.$$

4) O processo termina com o cálculo do resultado pretendido:

$$\operatorname{Tg} \theta = \frac{T^N}{S^N}$$

Outras funções trigonométricas

Para obter resultados relativos a outras funções trigonométricas, não são necessários outros algoritmos. A partir do cálculo da tangente de um ângulo θ podem obter-se, facilmente, os valores de $\operatorname{cotg} \theta, \operatorname{sen} \theta$ e $\cos \theta$ utilizando, apenas, fórmulas trigonométricas que envolvem as quatro operações e a raiz quadrada.

2.3. Algoritmo apresentado no Dr Dobb's Journal

Tal como o anterior, este algoritmo tem como objectivo o cálculo da tangente de um ângulo θ do primeiro quadrante ($0 \leq \theta < \pi/2$). A execução deste algoritmo tem uma estrutura muito semelhante à do primeiro mas com algumas diferenças interessantes. Uma primeira diferença á a utilização de potências de 2 onde antes usávamos potências de 10. O objectivo é facilitar a implementação num computador com aritmética binária. Outra diferença é que, enquanto no algoritmo da TI-83 o ângulo θ fica decomposto numa

soma de ângulos da forma $\theta_k = \text{arctg}\, 10^{-k}$, neste caso, consegue-se uma decomposição em somas e diferenças de ângulos da forma $a_k = \text{arctg}\, 2^{-k}$, isto é, θ fica decomposto numa soma de parcelas de forma $w_k \cdot a_k$ onde $w_k = \pm 1$ e $a_k = \text{arctg}\, 2^{-k}$. Como tal, se no primeiro caso, as rotações são feitas apenas no sentido directo, no segundo, elas podem ser feitas em ambos os sentidos. Assim, a partir das coordenadas iniciais $(1, 0)$, as restantes coordenadas vão ser obtidas por aplicação das equações

$$\begin{cases} S^{k+1} = S^k - T^k \cdot 2^{-k} \\ T^{k+1} = T^k + S^k \cdot 2^{-k} \end{cases} \quad (\text{se } w_k = 1; \text{ sentido directo})$$

$$\text{e} \quad \begin{cases} S^{k+1} = S^k + T^k \cdot 2^{-k} \\ T^{k+1} = T^k - S^k \cdot 2^{-k} \end{cases} \quad (\text{se } w_k = -1; \text{ sentido inverso})$$

Notemos que no segundo membro destas equações, além de somas e diferenças existem, apenas, multiplicações por potências de base 2. É este facto que favorece a implementação numa máquina que utilize o sistema binário. Efectivamente, neste caso, as multiplicações de um número por potências de base 2 correspondem, apenas, a alterações do expoente desse número. Uma diferença importante é que, enquanto no algoritmo da TI-83, na decomposição do ângulo θ, os valores de θ_k podem intervir mais do que uma vez ou, simplesmente, não intervir, no algoritmo apresentado no Dr Dobb´s Journal, os valores a_k intervêm todos, uma e uma só vez ([3]). Assim, enquanto o número de passos a executar no algoritmo da TI-83 varia de acordo com o ângulo θ em estudo, no caso deste segundo algoritmo (usando uma aproximação de 5 casas decimais nos valores de a_k), sabemos que, qualquer que seja o ângulo θ em estudo, o número de passos a executar é 18 ($0 \leq K \leq 17$) e, como tal, obtemos 18 coordenadas de rotação. O último par de coordenadas obtido é da forma

$$(S^{18}, T^{18}) = \left(\frac{X^{18}}{\cos a_0 \cdot \cos a_1 \cdots \cos a_{17}} \frac{Y^{18}}{\cos a_0 \cdot \cos a_1 \cdots \cos a_{17}} \right)$$

onde o produto (P_{18}) que consta no denominador se mantêm constante, qualquer que seja o ângulo θ em estudo. Efectivamente, tendo em conta que o ângulo θ fica decomposto numa soma de parcelas da forma $w_k \cdot a_k$ onde $wk = \pm 1$, o produto P_{18} é dado por $P_{18} = \Pi_{k=0} 17 \cos(w_k a_k)$ onde $w_k = \pm 1$. Mas como $\cos(-\alpha) = \cos \alpha$, a expressão de P_{18} simplifica-se ficando $P_{18} = \Pi_{k=0}^{17} \cos a_k$. Assim, sabemos à partida que o seu valor numérico é $P_{18} = \cos a_0 \cdot \cos a_1 \cdots \cos a_{17} \simeq 0,607251261$.

Note-se que, no caso de $\theta = 0,2$,

$$(S^{18}, T^{18}) = (1,6139351352; 0,3271585657).$$

Por observação da figura da pag. 3, sabemos que $\operatorname{sen}\theta = Y^{18}$ e $\cos\theta = X^{18}$.

Assim, neste segundo algoritmo, multiplicando o último par de coordenadas obtido por P^{18}, obtemos os valores do seno e do co-seno:

$$\cos\theta = P_{18}\cdot S^{18} = X^{18} \simeq 0,1986674516$$
$$\operatorname{sen}\theta = P_{18}\cdot T^{18} = Y^{18} \simeq 0,980064146$$

Outra forma de se obter estes valores é inicializando o algoritmo em $(P_{18}, 0)$. Com esta inicialização, o algoritmo dá-nos, directamente, os valores do $\operatorname{sen}\theta$ e $\cos\theta$ pois o último par de coordenadas obtido é (X^{18}, Y^{18}). Efectivamente, neste caso, as coordenadas obtidas por rotação são:

$$\begin{aligned}(P_N,0) = P_N(1,0) &\to P_N\left(\frac{X^1}{\cos a_0}, \frac{Y^1}{\cos a_0}\right)\\ &\to P_N\left(\frac{X^2}{\cos a_0\cdot\cos a_1}, \frac{Y^2}{\cos a_0\cdot\cos a_1}\right)\to\cdots\\ \cdots\to P_N&\left(\frac{X^{N-1}}{\cos a_0\cdot\cos a_1\cdots a_{N-2}}, \frac{Y^{N-1}}{\cos a_0\cdot\cos a_1\cdots a_{N-2}}\right)\\ &\to P_N\left(\frac{X^N}{\cos a_0\cdot\cos a_1\cdots a_{N-1}}, \frac{Y^N}{\cos a_0\cdot\cos a_1\cdots a_{N-1}}\right)\end{aligned}$$

o que é equivalente a

$$(P_N,0)\to\cos a_1\cdot\cos a_2\cdots\cos a_{N-1}(X^1,Y^1)\to\cos a_2\cdots\cos a_{N-1}(X^2,Y^2)$$
$$\to\cdots\to\cos a_{N-1}\,(X^{N-1},Y^{N-1})\to(X^N,Y^N)$$

Notemos que os valores dados pelas funções seno e co-seno na TI-83 são $\operatorname{sen}0,2 \simeq 0,1986693308$ e $\cos 0,2 \simeq 0,9800665778$. Tal como se pretendia, os valores obtidos pelo algoritmo e pela calculadora TI-83 são aproximados (o erro surge na sexta casa decimal).

Também no algoritmo da TI-83, a inicialização em $(K,0)$ com $K = \cos^2\theta_1\cdot\cos^6\theta_4\cdot\cos^6\theta_5 \simeq 0,9900987131$ (caso de $\theta = 0,2$), permitiria obter, directamente, os valores de $\operatorname{sen}\theta$ e de $\cos\theta$. No entanto, no caso deste algoritmo, existe a desvantagem de K variar com o ângulo em estudo.

3. Algoritmo Cordic para cálculos com a função exponencial

Para o cálculo de outras funções transcendentes existem também algoritmos do tipo Cordic. Vamos ver como funciona um algoritmo Cordic para o cálculo da exponencial de um valor X na calculadora TI-83.

Este algoritmo tem bastantes semelhanças com os anteriores:

---

1) Determinação dos valores $a_k \in N_0$ a partir da decomposição do valor inicial X ($X \geq 0$)

$$X \simeq \sum_{k=0}^{5} a_k c_k \quad \text{onde} \quad c_k = \log(1 + 10^{-k})$$

2) Inicialização de uma variável com o valor $Y^0 = 1$

3) Aplicação recursiva da equação

$$Y^{i+1} = Y^i(1 + 10^{-k}) \quad \text{ou} \quad Y^{i+1} = Y^i + 10^{-k} \cdot Y^i$$

onde i varia de 0 até $\sum_{k=0}^{5} a_k - 1$

4) O processo termina e Y^N contém o pretendido:

$$e^X = Y^N$$

Neste algoritmo, o valor X começa por ser decomposto numa soma de valores, numa sequência de passos, onde a cada valor da decomposição de X que se vai obtendo, corresponde um determinado valor de Y. Assim, ficamos com uma sequência de valores de Y, de tal forma que, quando completamos o processo, $Y = e^X$.

O algoritmo considera a função exponencial na seguinte forma factorizada:

$$e^x = e^{c_1} \cdot e^{c_2} \cdots e^{c_N} \quad \text{onde} \quad x = c_1 + c_2 + \cdots + c_N.$$

Como podemos constatar, tendo em conta esta factorização, a opção tomada para c_k (passo 1) favorece a implementação do algoritmo:

$$\begin{aligned} Y = e^X &\simeq= e^{\sum_{i=0}^{5} a_i c_i} = \Pi_{i=0}^{5} e^{a_i c_i} = \Pi_{i=0}^{5} (e^{c_i})^{a_i} \\ &= \Pi_{i=0}^{5} (1 + 10^{-i}) a_i = (1 + 10^{-0})^{a_0} \cdot (1 + 10^{-1})^{a_1} \cdots (1 + 10^{-5})^{a_5} \end{aligned}$$

Ao longo de todo o processo, o valor X vai-se decompondo numa soma de valores, a cada um dos quais, vai corresponder um valor de Y. Assim, o valor pretendido Y vai-se obter através da aplicação repetida da equação

$$Y^{i+1} = Y^i(1 + 10^{-k}) \quad \text{ou} \quad Y^{i+1} = Y^i + 10^{-k} \cdot Y^i \ , \quad K \in [0, 5]$$

onde Y^{i+1} é o valor obtido a partir de Y^i neste processo iterativo.

Podemos observar no segundo membro desta última equação que, além de somas e subtracções existem, apenas, multiplicações por potências de base 10. Tendo em conta que a calculadora trabalha no sistema decimal, estas multiplicações correspondem, apenas, a alterações do expoente de Y^i.

Na aplicação deste algoritmo a casos concretos utiliza-se, também, uma tabela auxiliar:

K	$1+10^{-k}$	$c_k = \log(1+10^{-k})$ (5 casas decimais)
0	2,00000	0,69315
1	1,10000	0,09531
2	1,01000	0,00995
3	1,00100	0,00100
4	1,00010	0,00010
5	1,00001	0,00001

Vejamos o caso concreto do cálculo da exponencial de 2,45, isto é, do valor $e^{2,45}$ na Tabela 2.

Observando a tabela, concluímos que, segundo o algoritmo, $Y = e^{2,45} \simeq 11,58825667$ enquanto, o valor obtido pela função exponencial da TI-83 é $e^{2,45} \simeq 11,58834672$ (o erro surge na quarta casa decimal).

Terminamos referindo que estes são, apenas, três algoritmos Cordic. Para estas mesmas funções existem outros que, embora muito semelhantes aos que foram referidos, apresentam diferenças bastante interessantes. Existem, ainda, outros algoritmos deste tipo para outras funções.

Referências

[1] Bugalho, Maria de Jesus Sousa Vieira, "Alguns algoritmos Cordic", Seminário de Mestrado em Matemática na área de Matemática para o Ensino na FCUL, 1998.

[2] Texas Instruments, "Trig functions - how do TI products calculate them?", `http://www.ti.com/calc/docs/faq/83faq086.htm`, Novembro 98.

[3] Jarvis, Pitts, "Implementing Cordic Algorithms", Dr. Dobb's Journal, October 1990, pag. 152-156.

K	Y	X	$c_k = \log(1 + 10^{-k})$ (5 casas decimais)
0	1	2,45	
0	2	1,75685	0,69315
0	4,00000000	1,06370	0,69315
0	8,00000000	0,37055	0,69315
1	8,80000000	0,27524	0,09531
1	9,68000000	0,17993	0,09531
1	10,64800000	0,08462	0,09531
2	10,75448000	0,07467	0,00995
2	10,86202480	0,06472	0,00995
2	10,97064505	0,05477	0,00995
2	11,08035150	0,04482	0,00995
2	11,19115501	0,03487	0,00995
2	11,30306656	0,02492	0,00995
2	11,41609723	0,01497	0,00995
2	11,53025820	0,00502	0,00995
3	11,54178846	0,00402	0,00100
3	11,55333025	0,00302	0,00100
3	11,56488358	0,00202	0,00100
3	11,57644846	0,00102	0,00100
3	11,58802491	0,00002	0,00100
5	11,58814079	0,00001	0,00001
5	11,58825667	0,00000	0,00001

Tabela 2

[4] Nave, Rafi, "Implementation of Transcendental Functions on a Numerics Processor", Microprocessing and Microprogramming 11, 1983, pag. 221-225.

[5] Pina, Heitor, "Métodos Numéricos", cap. 1, McGraw-Hill.

[6] Ruckdeschel,F.R., "Basic Scientific Subroutines", Vol. II, cap. 4, Byte/McGraw-Hill Publications, 1981.