Primitivas de n-splines

Praciano-Pereira, Tarcisio *

17 de setembro de 2015
préprints da Sobral Matemática
no. 2015.04
Editor Tarcisio Praciano-Pereira
tarcisio@member.ams.org

Resumo

Neste artigo é construído um exemplo de 3-splines à suporte compacto como primitiva de um 1-splines que satisfaz uma determinada condição que no presente caso é simplesmente energia nula, ou integral nula. A construção se dá pelo calculo de duas primitivas, sucessivamente. Na última seção se generaliza o processo mostrando a construção duma primitiva dum n-splines qualquer.

palavras chave: n-splines, primitiva dum n-splines, aproximação polinomial.

This paper deals with the construction of a 3-splines as the second order primitive of 1-splines defined by a condition, which in the present case is null energy or null integral. The construction is obtained by two successive integrations and in the last section this procedure is generalized to show the method to calculate a primitive of an n-splines.

keywords: n-splines, primitive of n-splines, polynomial approximation.

*tarcisio@member.ams.org

1 n-splines

Uma definição de $n-splines$, e você pode encontrar várias, todas equivalentes, é uma função polinomial por pedaços cuja classe de diferenciabilidade é $n-1$ sendo n o grau máximo entre os pedaços de polinômio que a definem. Um $n-splines$ é uma substituição cômoda para um polinômio de grau n, por exemplo, é fácil construir, e logo a seguir dou um exemplo, um $n-splines$ periódico.

Mas esta definição é difícil de ser usada, e poristo cada autora tenta encontrar uma que se adeque melhor ao seus trabalho. Para algumas autoras *splines* é feminino, para outras é um *splines*. Ninguém sabe ao certo quando foram inventados e nem quem os inventou, simplesmente apareceram! Dizem que Courant fazia uso deles, e nunca encontrei nenhum texto do Courant mencionando *splines*, o que também não significa que não exista, mas diversos autores afirmam que ele teria usado *splines* e este fato coloca o aparecimento dos n-splines na década de 40 do século passado. As funções definidas por pedaços são antigas, remontam ao século 19.

Pegue dois pedaços sucessivos dum $n-splines$, serão dois polinômios que podem ser do grau menor ou igual a n que estão colados com classe de diferenciabilidade $n-1$, nas pontas. Deixe-me introduzir alguma notação para tornar a redação mais concreta. Estou falando de f_{k-1}, f_k duas seções do gráfico do dum $n-splines$ f definidas, cada uma, em dois intervalos seguidos $[x_{k-1}, x_k], [x_k, x_{k+1}]$ e as derivadas sucessivas de f no ponto x_k coincidem $n-1$ vezes:

$$f_{k-1}(x_k) = f_k(x_k); \tag{1}$$

$$a_{k-1,0} + a_{k-1,1}(x-x_k) + a_{k-1,2}(x-x_k)^2 \ldots + a_{k-1,n}(x-x_k)^n = \tag{2}$$

$$= a_{k,0} + a_{k,1}(x-x_k) + a_{k,2}(x-x_k)^2 \ldots + a_{k,n}(x-x_k)^n \tag{3}$$

$$a_{k-1,1} + 2a_{k-1,2}(x-x_k) \ldots + na_{k-1,n}(x-x_k)^{n-1} = \tag{4}$$

$$= a_{k,1} + 2a_{k,2}(x-x_k) \ldots + na_{k,n}(x-x_k)^{n-1} \tag{5}$$

$$\ldots \tag{6}$$

$$(n-1)!a_{k-1,n} = (n-1)!a_{k,n}; \tag{7}$$

$$k \in \{0, \ldots, m\} \tag{8}$$

em que m é o número de segmentos de polinômio ou do número de intervalos.

Uma forma cômoda de definir consiste em considerar um intervalo $[0, \alpha]$ e uma partição deste intervalo determinada pelos $m+1$ nós

$$x_0 = 0, \ldots, x_m = \alpha \tag{9}$$

com um polinômio de grau menor ou igual n definido em cada intervalo satisfazendo ao sistema de equações (eq. 1.1) (eq. 1.8). que será a forma de definir $n-splines$ neste artigo.

Como cada seção é uma curva polinomial, você pode definir todas seções num único intervalo, mas isto pode tornar as coisas extremamente difíceis e deve ser usada apenas se facilitar de alguma forma o seu trabalho.

Um exemplo inicial é uma *função poligonal* cujo gráfico é constituido de pedaços de reta, então as derivadas à direita e à esquerda em cada nó coincidem 0 vezes então você tem uma função de classe $\mathcal{C}^0$, apenas contínua, um $1-splines$.

Se você calcular uma primitiva dum $1-splines$, resulta numa função de classe $\mathcal{C}^1$, um $2-splines$ cujos segmentos serão polinômios de grau no máximo dois. Se calcular uma primitiva dum $2-splines$, resulta numa função de classe $\mathcal{C}^2$, um $3-splines$ cujos segmentos serão polinômios de grau no máximo três.

Os $3-splines$ já foram consagrados como o melhor tipo de curva nas aplicações, fazemos com eles grande parte das aproximações que precisamos.

1.1 Construção geométrica dum 3-splines

O próximo exemplo mostra que se trata duma ferramento extremamente eficiente, mas também irá servir para evidenciar a existencia dum *elemento básico* que será explorado na próxima seção.

Suponha que você deseje uma função f de classe $\mathcal{C}^2$, um $3-splines$, que seja a suporte compacto. Então a integral de f' deve ser nula a partir duma condição inicial a, portanto um $2-splines$ formado de duas *bolhas sucessivas* cujas áreas se anulem: segmentos de parábolas formando as duas bolhas, uma positiva e a outra negativa. Confira a figura (1) página 2, em que aparece f' e serão

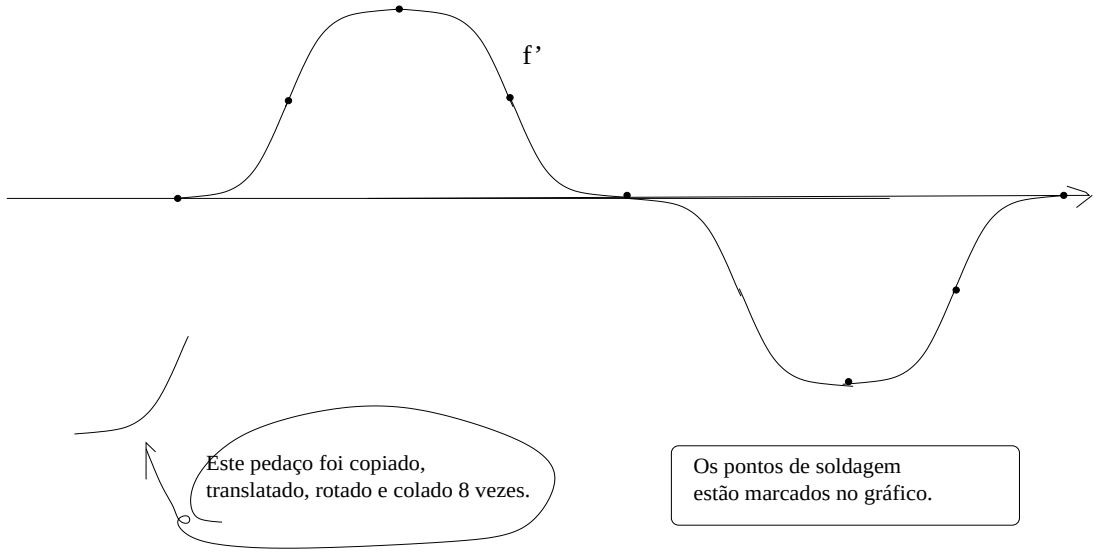

Figura 1: duas bolhas de integral zero

necessários 8 segmentos de parábola, portanto 8 intervalos. É possível alongar a bolha incluindo segmentos de reta entre os segmentos de parábola, a bolha de tamanho mínimo tem que ter 8 segmentos de parábola. Foi esta construção geométrica feita na figura (1). Não há ainda nenhum cálculo algébrico.

Cada uma dessas bolhas satisfaz à condição inicial, de ser à suporte compacto portanto a construção poderia começar por cada uma delas donde se conclue que é preciso ter uma poligonal formando quatro triângulos de áreas iguais,

mas simétricas, (sinais contrários) duas a duas e você encontra f'' na figura (2) página 3,

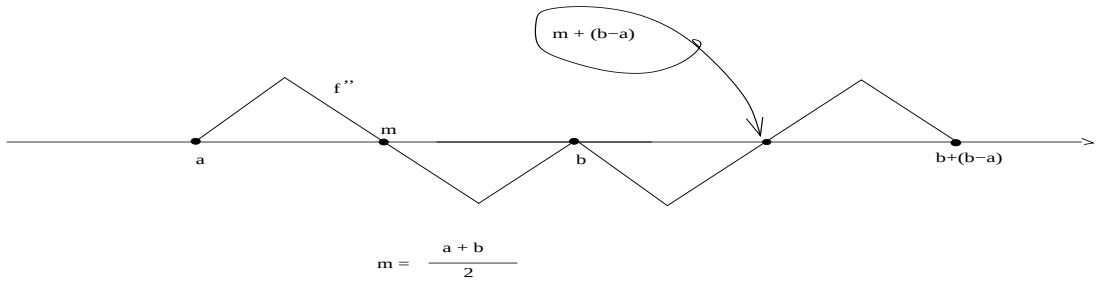

Figura 2: um splines a suporte compacto

A construção da figura (2) é também exclusivamente geométrica, nenhum cálculo algébrico foi feito, mas agora a interpretação geométrica vai ser transformada em *equações algébricas*:

$$m = \frac{a+b}{2}; \tag{10}$$

$$f''(x) \text{ é um 1-splines com suporte } [a, b]; \tag{11}$$

$$f'(x) = \int_a^x f''(t)dt; \tag{12}$$

$$x < b + (b - a); f'(x) = -\int_b^x f''(t - b + a)dt; \tag{13}$$

$$f' \text{ é um 2-splines a suporte compacto;} \tag{14}$$

$$f = \int_a^x f'(t)dt \text{ é um 3-splines a suporte compacto;} \tag{15}$$

Todas as justificativas são exercícios um pouco elaborados do Cálculo, são necessários apenas os conceitos de derivada e primitiva eis o esboço de demonstração seguido de alguma sugestões para outras construções:

- Como as áreas dos triângulos se anulam, a primitiva f' se anula também depois do quarto triângulo.
- também pela mesma razão f' tem duas bolhas 2-splines com áreas iguais e simétricas (sinais contrários).
- f será formada de segmentos de polinômios do terceiro grau e voltará a se anular depois da segunda bolha parabólica.
- Redefina f'' como uma função $2(b-a)-$periódica e experimente que pode resultar disto, é divertido!
- Escolha *uma propriedade geometrica* que você queira um $n - splines$ e faça o reverso duma *equação diferencial*, como no exemplo anterior, que você constrói o $n - splines$ desejado.

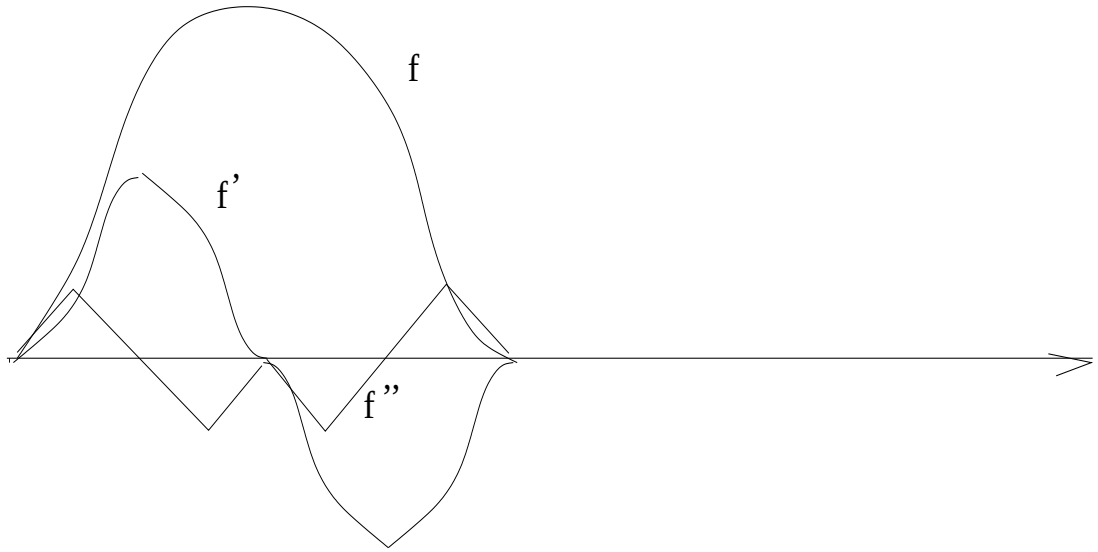

Figura 3: 3 splines a suporte compacto

A figura (3) página 4, que foi feita usando o editor de gráficos `xfig`, mostra o gráfico da construção formalizada nas equações (eq. 1.10)- (eq. 1.15).

Na próxima figura (4), página 4, você pode ver o mesmo gráfico *artístico*

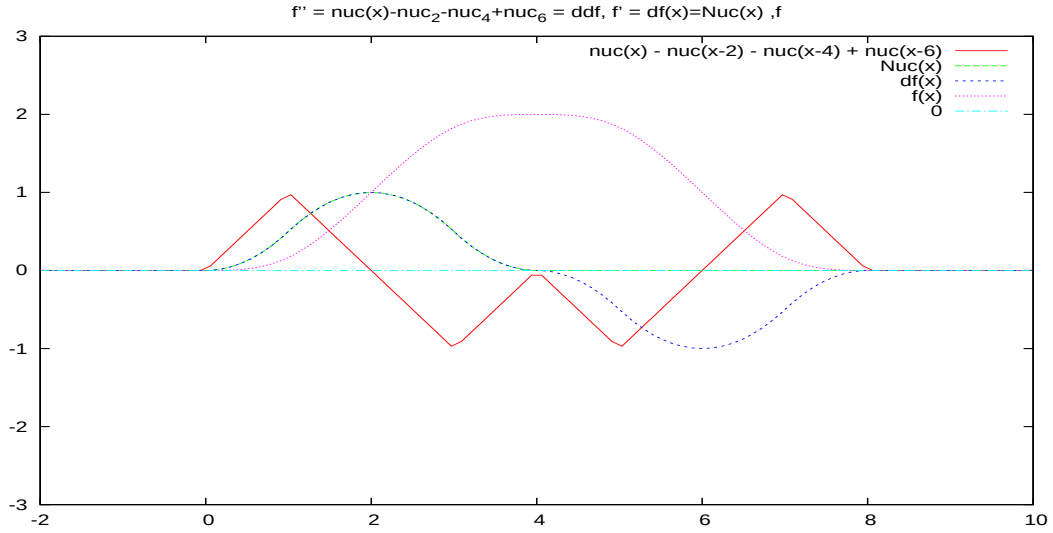

Figura 4: 3 splines a suporte compacto produzida por gnuplot

da figura (3), agora produzido por `gnuplot` à partir das equações calculadas, algebricamente, na próxima seção. E você pode reproduzir ou alterar o gráfico a partir do programa [2, PrimitivaQuaseSplines01.gnuplot], em que os cálculos estão todos guardados, e escondidos, como comentários. É um bom exercício tentar as sugestões feitas acima alterando o programa para obtê-las.

2 Os elementos

Da construção feita na seção anterior se pode concluir que o *elemento fundamental* é um triângulo isósceles que aparece quatro vezes na figura (2) página 3 e nas seguintes (3), (4).

Tenha a curiosidade ler o programa citado, e procure entender o significado das tranlações de `nuc()` que aparecem no comando para executar o gráfico

```
plot nuc(x) - nuc(x-2) - nuc(x-4) + nuc(x-6) , Nuc(x), df(x),f(x), 0
```

A função `nuc()` é o "triângulo isósceles".

Esta observação nos conduz a uma construção mais simples e inclusive a um programa para construir estes $3-$ *splines* nas condições mais variadas. Este elemento fundamental vai ser um triângulo de área 1 que está definido como a função $nuc()$, à suporte compacto $[0,2]$ a seguir.

Observe que como o suporte de $nuc()$ é disjunto do suporte de sua translação $nuc_2()$, (tem apenas um ponto em comum, portanto a interseção tem medida zero, e é a medida que conta em se tratando de integral), então a primitiva de $nuc - nuc_2$ vai ser a diferença das primitivas. Claro que tudo foi calculado para que as contas ficassem simples.

A expressão nuc_2 significa a translação de nuc à direita, ou ainda $nuc_2(x) = nuc(x-2)$.

Definindo,

$$nuc(x) = \begin{cases} x < 0 & \Rightarrow & 0; \\ x < 1 & \Rightarrow & x; \\ x < 2 & \Rightarrow & 2 - x; \\ x \geq 2 & \Rightarrow & 0; \end{cases} \tag{16}$$

$$f''(x) = nuc(x) - nuc(x-2); \tag{17}$$

$$f''(x) = \begin{cases} x < 0 & \Rightarrow & 0; \\ x < 1 & \Rightarrow & x; \\ x < 2 & \Rightarrow & 2 - x; \\ x < 3 & \Rightarrow & 2 - x; \\ x < 4 & \Rightarrow & -4 + x; \\ x \geq 5 & \Rightarrow & 0; \end{cases} \tag{18}$$

$$f'(x) = Nuc - Nuc_2; Nuc \text{ é uma primitiva de } nuc; \tag{19}$$

$$Nuc(x) = \begin{cases} 0 & \Rightarrow & x < 0; \\ x < 1 & \Rightarrow & \frac{x^2}{2}; \\ x < 3 & \Rightarrow & 1 - \frac{(2-x)^2}{2}; \\ x < 4 & \Rightarrow & \frac{(4-x)^2}{2}; \\ x \geq 5 & \Rightarrow & 0; \end{cases} \tag{20}$$

$$f'(x) = Nuc - Nuc_2 = \begin{cases} x < 0 & \Rightarrow & 0; \\ x < 1 & \Rightarrow & \frac{x^2}{2}; \\ x < 3 & \Rightarrow & 1 - \frac{(2-x)^2}{2}; \\ x < 4 & \Rightarrow & \frac{(4-x)^2}{2}; \\ x < 5 & \Rightarrow & -\frac{(x-4)^2}{2}; \\ x < 7 & \Rightarrow & -1 + \frac{(6-x)^2}{2}; \\ x < 8 & \Rightarrow & -\frac{(8-x)^2}{2}; \\ x \geq 9 & & 0; \end{cases} \tag{21}$$

$$f(x) = \begin{cases} x < 0 & \Rightarrow & 0; \\ x < 1 & \Rightarrow & \frac{x^3}{6}; \\ x < 3 & \Rightarrow & \frac{(2-x)^3}{6} + x - 1; \\ x < 4 & \Rightarrow & 2 - \frac{(x-4)^3}{6}; \\ x < 5 & \Rightarrow & 2 - \frac{(4-x)^3}{6}; \\ x < 7 & \Rightarrow & 7 - x - \frac{(6-x)^3}{6}; \\ x < 8 & \Rightarrow & \frac{(8-x)^3}{6}; \\ x \geq 9 & \Rightarrow & 0; \end{cases} \tag{22}$$

O código em `gnuplot` para visualisar estes splines pode ser baixado do link [2, PrimitivaQuaseSplines01.gnuplot]

A definição de funções no `gnuplot`, `nuc()` por exemplo, é *um pouco* críptica mas a seguinte explicação deve quebrar o segredo. Se trata duma construção oriunda da linguagem de programação `C` em que a *interrogação* representa um `if()` e os dois pontos representa o `else` de cada `if()`, isto dá em Matemática:

$$nuc(x) = \begin{cases} \text{Se } x < 0 & 0; \\ \text{Se } x < 1 & x; \\ \text{Se } x < 2 & 2 - x; \\ \text{Se } x > 2 & 0; \end{cases} \tag{23}$$

e agora "`;`" estão representando `else` e na linguagem do `gnuplot` fica

```
nuc(x) = (x<0)?0:(x<1)?x:(x<2)?2-x:0;
```

Ao ler o programa você irá encontrar esta expressão editada de forma *mais humana* porque usei a facilidade de edição em `LinuX` que me permite usar a contrabarra para quebrar uma linha ao longo de quatro linhas.

Na definição dos splines `spl2(x), spl3(x)`, também usei, no progrma, a facilidade de edição de `LinuX` em que a `contrabarra` elimina o fim de linha permitindo que uma linha se propague por diversas linhas tornando o código todo visível. Esta facilidade permite de quebrar-se o código em cada `else` e praticamente escrever como a Matemática da equação (eq. 2.23). É preciso que você coloque o fim de linha exatamente depois da contra barra, e "*fim de linha*" quer dizer "`enter`", pelo menos em `LinuX`[1].

[1] É uma injustiça que infelizmente é difícil de ser corrigida, ninguém usa `LinuX`, todos usamos `bash` que é uma linguagem interpretada que faz a ligação `periféricos-LinuX`.

Os cálculos se encontram todos guardados como `comentários` dentro dos programas citados.

3 $n-$splines periódicos

Vou mostrar nesta seção como construir um $n-$splines periódico basicamente uma amplicação do que foi feito na construção do $3-$splines na seção anterior.

Se f for um $n-$splines periódico então sua derivada será periódica. A recíproca não é verdadeira, é preciso acrescentar à condição de *energia periódicamente nula*, na derivada, para que a primitiva seja periódica.

Com estas duas condições sobre a derivada um $n-$splines periódico se origina dum $1-$splines periódico ou melhor dum $0-$splines periodico sendo f obtida por n integrações sucessivas deste splines inicial. Em geral não se fala de $0-$splines uma vez que se pensa sempre que *splines* sejam contínuos embora o radical n já resolveria este problema, claro criando outro "*de classe de continuídade -1*", assim têm razão as autoras que não fazem referência aos $0-$splines. Mas feita esta ressalva, sinto-me com a liberdade de usá-los.

Um exemplo de $0-$splines é a função característica de algum intervalo, ou melhor, qualquer combinação linear de funções características de intervalos é um $0-$splines, e isto pode ser tomado como definição, embora haja o risco de uma tal combinação linear resulte numa função constante que seria localmente um $1-$splines, mas globalmente seria um $0-$splines.

Então

$$g_0 = \chi_{[0,1]} - \chi_{[1,2]} - \chi_{[2,3]} + \chi_{[3,4]} = \eta - \eta_1 - \eta_2 + \eta_3; \eta = \chi_{[0,1]}; \tag{24}$$

é um $0-$ *splines* cuja integral é nula. Se este "padrão" for repetido com

$$f_0(x) = \sum_{k\in\mathbf{Z}} g_{0,k}(x); \tag{25}$$

indefinidamente, se obtémum $0-$ *splines* periódico de período 2, ou,

$$g_{0,\alpha} = \chi_{[0,\alpha]} - \chi_{[\alpha,2\alpha]} = \nu - \nu_\alpha; \nu = \chi_{[0,\alpha]}; \tag{26}$$

é um $0-$ *splines* periódico de período α cuja integral é nula que ira produzir

$$f_0(x) = \sum_{k\in\mathbf{Z}} (-1)^k g_{0,k\alpha}(x); \tag{27}$$

e se obtém um $0-$ *splines* periódico de período α. É exatamente aplicar recurrência à construção feita na seção anterior.

Para o caso α o programa

`PrimitivaQuaseSplines03.gnuplot` que pode ser baixado da página [2, PrimitivaQuaseSplines03.gnuplot] mostra explicitamente a construção dum $3-$ *splines* periódico usando uma função que cria as classes de equivalência modulo 8 para números reais definida no programa.

A "*soma*" na equação (eq. 3.27) é o que algumas vezes se chama de "*soma formal*" uma vez que as parcelas são translações de funções à suporte compacto a *real soma* se reduz à quatro parcelas sucessivas.

Qualquer primitiva de f_0 ou de $g_{0,\alpha}$ é um $2-$splines porém o período é duplicado a cada nova integração relativamente ao período da função inicial. Então a $n-$ésima primitiva de qualquer destes $0-splines$ terá o período $np; p \in \{1, \alpha\}$ desejado. Para obter um período específico, basta controlar o período dos "*núcleos*" iniciais dividindo-se por uma constante adequada.

Na próxima seçao vou mostrar como calcular a $n-$ésima primitiva de um $n-$splines.

4 Primitiva dum n-splines

Neste seção vou considerar um $n-$splines genérico f e apresentar o algoritmo para o cálculo duma sua primitiva. A notação é a seguinte:

$$(x_k, y_k)_{k\in\{0,\ldots,n\}} \text{ uma sucessão de nós;} \tag{28}$$

$$(x_k)_{k=0}^{n} \text{ os nós duma partição de } [0, \alpha]; \tag{29}$$

$$(x_k, y_k)_{k=0}^{n} \text{ são os pontos de precisão de } f \tag{30}$$

$$f_k : [x_k, x_{k+1}] \to \mathbf{R};\ \text{é um polinômio de grau } n; \tag{31}$$

O cálculo duma primitiva de f se passa como se estivesse sendo calculado a primitiva de cada uma das seções f_k apenas considerando a condição inicial (x_k, y_{k-1}) quer dizer

$$F_k(x) = y_{k-1} + \int_{x_k}^{x} f_k(t)dt; \tag{32}$$

Na prática a expressão de F_k pode ser facilmente obtida calculando-se a integral $\int_0^{x-x_k} f_k(t)dt$ em que o integrando foi alterado por *uma mudança de variável* adequada, essencialmente a troca dos limites de integração:

$$F_k(x) = y_{k-1} + \int_0^{x-x_k} f_k(t)dt; y_{k-1} = f_{k-1}(x_k) \tag{33}$$

ou seja, F_k tem como condição inicial (x_k, y_{k-1}).

Você pode ver isto na prática lendo os programas [2, PrimitivaQuaseSplines01.gnuplot] ou [2, PrimitivaQuaseSplines03.gnuplot] em que as contas ficaram detalhadas, protegidas por comentários, depois do que foi feita a simplificação algébrica. Você poderá observar que no cálculo da integral da equação (eq. 4.33), nos programa, simplesmene copiei a expressão de cada item anterior ao definir o seguinte dando à variável o valor extremo do intervalo anterior: $y_{k-1} = f_{k-1}(x_k)$.

Possivelmente uma boa programadora conseguirá escrever o algoritmo de modo mais limpo elas que dominam a *arte de programar computadores*...

Para terminar uma observação. É possível que a leitora tenha guardado a impressão de que para obter uma primitiva de ordem n se tenha pela frente um trabalho gigantesco. E é verdade! Mas em geral o objetivo não é este, e para começar $n - splines$ podem ser obtidos, e em geral o são, por outros métodos, por exemplo, por um produto de convolução, analise [1] e confira como fazer.

Referências

[1] A.J. Neves and T. Praciano-Pereira. Convolutions power of a characteristic function. *arxiv.org*, 2012, April, 22:16, 2012.

[2] T Praciano-Pereira. Programas para cálculo numérico. Technical report, 2009. http://www.calculo-numerico.sobralmatematica.org/programas/.