Primitivas de n-splines II

Praciano-Pereira, Tarcisio *

15 de fevereiro de 2016
préprints da Sobral Matemática
no. 2016.02
Editor Tarcisio Praciano-Pereira
tarcisio@member.ams.org

Resumo

Este artigo finaliza a construção inicializada em artigo anterior em se que encontra, com sucesso, a expressão da integral dum n-splines. Foi introduzida o conceito de n-quase-splines para tornar mais simples a demonstração final.

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

This paper finish the construction of the primitive of $n-splines$ started without success in a previous paper. The definition of an $n-quasi-splines$ has make possible to reach the desired result. A revision of the previous paper is included to make the ideas more easy to reader.

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

*tarcisio@member.ams.org

1 n-splines

Este artigo termina com sucesso a tentativa inicializada em [4] em que não consegui obter uma forma simples para a primitiva dum $n-splines$ qualquer que é o resultado obtido na última seção. Para levar os cálculos a cabo com sucesso, vi-me forçado a incluir a noção de $quase-splines$ na última seção.

Uma definição de $n-splinesf$, 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 $n-splinesf$ é uma substituição cômoda para um polinômio de grau n, por exemplo, é fácil construir, e ao final desta seção dou um exemplo, um $n-splines$ periódico.

Mas esta definição é difícil de ser usada, e por isto cada autora tenta encontrar uma que se adéque melhor ao seus trabalhos. 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. Wabba, [5], na introdução, se refere à criação do nome atribuindo-o a Schoenberg, em 1982, lembrando um instrumento de arquitetura de navios que teria este nome. Eu "*inventei splines*" em 1988 sem nada saber de Schoenberg ou do livro de Wabba que foi publicado em 1987, na verdade eu trabalhava com polinômios por pedaços que, depois de conhecer os *splines*, eu passei a denominar de *quase-splines* porque eram polinômios por pedaços de grau dois apenas contínuos, a diferenciabilidade falhava em cima dos nós. Estarei retornando aos *quase-splines* na última seção.

As funções definidas por pedaços são antigas, remontam ao século 19, eu estava trabalhando com *funções definidas por pedaços*. Eu vou voltar a definir *n-quase-splines* na última seção e justificar porque os vou introduzir.

A denominação também varia, alguns se referem a *splines de ordem* $n-1$ em que a ordem coincide com a classe de diferenciabilidade. Vou adotar como nomenclatura a referência a um $n-splines\ f$ em que n é o grau máximo dos pedaços de polinômio sendo $n-1$ a classe de diferenciabilidade. Esta variação na nomenclatura apenas enfatiza a imensa aplicação deste instrumento, confira [5].

Observe que o grau dos pedaços de polinômio não pode ser homogêneo porque entre os pedações de polinômios podem encontrar-se segmentos de reta, ou seja, polinômios de grau zero ou um.

Pegue dois pedaços sucessivos dum $n-splines\ f$, 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 sucessivos $[x_{k-1}, x_k], [x_k, x_{k+1}]$ medindo, respectivamente, $\Delta x_{k-1}, \Delta x_k$. As derivadas

sucessivas de f no ponto x_k coincidem $n-1$ vezes, ou mais precisamente,

$$\frac{d^p f_{k-1}}{dx^p}(x_k) = \frac{d^p f_k}{dx^p}(x_k); p \in \{0, \ldots, n-1\} \tag{1}$$

e, para $p = 0$ se tem $f_{k-1}(x_k) = f_k(x_k)$ de modo que a classe de continuidade é $\mathcal{C}^{n-1}$.

$$f_k(x) = \sum_{j=0}^{n} A_j^0 a_{k,j}(x - x_k)^j; \tag{2}$$

$$f_{k-1}(x_k) = \sum_{j=0}^{n} A_j^0 a_{k-1,j} \Delta x_{k-1}^j = A_0^0 a_{k,0} = f_k(x_k); \tag{3}$$

$$\frac{df_{k-1}}{dx}(x_k) = \sum_{j=0}^{n} A_j^1 a_{k-1,j} \Delta x_{k-1}^{j-1} = A_1^1 a_{k,1} = \frac{df_k}{dx}(x_k) \tag{4}$$

$$\frac{d^2 f_{k-1}}{dx^2}(x_k) = \sum_{j=0}^{n} A_j^2 a_{k-1,j} \Delta x_{k-1}^{j-2} = A_2^2 a_{k,2} = \frac{d^2 f_k}{dx^2}(x_k); \tag{5}$$

$$\ldots \tag{6}$$

$$\frac{d^p f_{k-1}}{dx^p}(x_k) = \sum_{j=0}^{n} A_j^p a_{k-1,j} \Delta x_{k-1}^{j-p} = A_p^p a_{k,p} = \frac{d^p f_k}{dx^p}(x_k) \tag{7}$$

$$\ldots \tag{8}$$

$$\frac{d^{n-1} f_{k-1}}{dx^{n-1}}(x_k) = A_{n-1}^{n-1} a_{k-1,n-1} \Delta x_{k-1}^0 + A_n^{n-1} a_{k-1,n} \Delta x_{k-1}^1; \tag{9}$$

$$\frac{d^{n-1} f_k}{dx^{n-1}}(x_k) = A_{n-1}^{n-1} a_{k,n-1}; \tag{10}$$

$$a_{k-1,n-1} + n a_{k-1,n} \Delta x_{k-1} = a_{k,n-1}; \tag{11}$$

$$a_{k-1,n}, a_{k,n} \text{ são independentes, } k \in \{1, \ldots, m\}; \tag{12}$$

em que m é o número de segmentos de polinômio ou do número de intervalos e estou usando a notação A_j^p, que representa o número de *arranjos de j elementos tomados p − a − p*, assumindo o valor zero quando $j < p$. Definindo assim, A_j^p é o $p-$ésimo coeficiente da soma dos termos da derivada de ordem p todos nulos até $A_p^p = 1; j = p$. Outra vantagem desta notação é que as linhas da matriz M vão progressivamente se anulando à direta para se tornar uma matriz não retangular, triangular inferior.

Não é uma matriz retangular porque as derivadas de ordem n são independentes: se um segmento à esquerda for um segmento de reta horizontal, sua matriz de coeficientes é identicamente nula mas o seguinte não precisa ser o que dá um *grau de liberdade* 1 para o sistema e, observe, é exatamente o que se precisa, no caso dum segmento de reta horizontal, que o próximo segmento de polinômio seja um múltiplo de $(x - x_k)^n$.

Este sistema de equações pode ser apresentado como um produto duma matriz, não retangular, diagonal inferior, que é a representação matricial do

$n-splinesf$ em função de cada um dos m nós selecionados no intervalo,

$$M = \begin{pmatrix} A_0^0 a_{k-1,0} & A_1^0 a_{k-1,1} & \cdots & A_{n-1}^0 a_{k-1,n-1} & A_n^0 a_{k-1,n} \\ A_1^1 a_{k-1,1} & A_2^1 a_{k-1,2} & \cdots & A_n^1 a_{k-1,n} & 0 \\ A_2^2 a_{k-1,2} & A_2^3 a_{k-1,3} & \cdots & 0 & 0 \\ & \cdots & \cdots & 0 & 0 \\ A_p^p a_{k-1,p} & A_{p+1}^p a_{k-1,p+11} & \cdots & 0 & 0 \\ & \cdots & \cdots & 0 & 0 \\ A_{n-1}^{n-1} a_{k-1,n-1} & A_n^{n-1} a_{k-1,n-1} & \cdots & 0 & 0 \end{pmatrix} \tag{13}$$

$$M \begin{pmatrix} \Delta x_{k-1}^0 \\ \Delta x_{k-1} \\ \cdots \\ \Delta x_{k-1}^p \\ \cdots \\ \Delta x_{k-1}^{n-1} \\ \Delta x_{k-1}^n \end{pmatrix} = \begin{pmatrix} 0! a_{k,0} \\ 1! a_{k,1} \\ \cdots \\ p! a_{k,p} \\ \cdots \\ (n-1)! a_{k,n-1} \end{pmatrix}; \tag{14}$$

Observe, o *pior caso*, a matriz de dados à direita na equação (eq. 1.14), pode ser nula. Basta que os elementos da diagonal da matriz do sistema não sejam nulos e se pode ter solução, neste caso um $n-splines$ com um segmento de reta entre entre os gráficos de polinômios.

Esta expressão, em se tratando de $3-splines$, que vai ser o foco que vou dar ao longo do trabalho, se apresenta assim:

$$\begin{pmatrix} a_{k-1,0} & a_{k-1,1} & a_{k-1,2} & a_{k-1,3} \\ a_{k-1,1} & 2a_{k-1,2} & 3a_{k-1,3} & 0 \\ 2a_{k-1,2} & 6a_{k-1,3} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x_{k-1}^0 \\ \Delta x_{k-1} \\ \Delta x_{k-1}^2 \\ \Delta x_{k-1}^3 \end{pmatrix} = \begin{pmatrix} 0! a_{k,0} \\ 1! a_{k,1} \\ 2! a_{k,2} \end{pmatrix}; \tag{15}$$

de onde posso expandir em equações apresentando os coeficientes de f_k como função dos coeficientes de f_{k-1}

$$a_{k,0} = a_{k-1,0} + a_{k-1,1}\Delta x_{k-1} + a_{k-1,2}\Delta x_{k-1}^2 + a_{k-1,3}\Delta x_{k-1}^3; \tag{16}$$

$$a_{k,1} = a_{k-1,1} + 2a_{k-1,2}\Delta x_{k-1} + 3a_{k-1,3}\Delta x_{k-1}^2; \tag{17}$$

$$a_{k,2} = a_{k-1,2} + 3a_{k-1,3}\Delta x_{k-1}; \tag{18}$$

A matriz do $3-splines$ deixa claro a estrutura de qualquer matriz para $n-splines$ com o mesmo objetivo que tive neste exemplo, em cada linha aparecem as sucessivas derivadas da expressão polinomial produzindo uma matriz triangular inferior, não retangular, começando com a derivada de ordem zero, terminando com primeira derivada o que garante a classe de continuidade $n-1$. A ordem de liberdade deste sistema é 1.

Nos programas mencionados logo a seguir, defini as funções que calculam o valor dum polinômio em que os coeficientes são passados como parâmetros o que permite construir um laço que percorre os intervalos da partição do intervalo atualizando os coeficientes para cada intervalo e produzindo assim a imagem gráfica do $3-splines$, confira as figuras (fig. 2), página 6.

É fácil generalizar os programas para uma quantidade arbitrária de coeficientes, mas isto se encontra fora do foco deste artigo, ainda assim apresento um programa para o caso de $4-splines$.

Uma primeira consequência deste sistema de equações é que, se selecionarmos todos os coeficientes de f_k, ou explicando melhor, se dermos valores aos coeficientes f_k, o $n-splines f$ fica determinado a menos dos intervalos $[x_k, x_{k+1}]$. Confira o exemplo nos programas [3, `PrimitivaQuaseSplines*.calc`] em que o sistema de equações (eq. 1.15)- (eq. 1.18) está traduzido num algoritmo para produzir o gráfico dum $3-splines$ em que você pode selecionar os coeficientes do primeiro pedaço de polinômio f_0 e ver o gráfico do $3-splines$ construído automaticamente por um dos programas usando uma partição uniforme do intervalo $[0, \alpha]$.

Os programas citados foram escritos para a *linguagem de programação* `calc`,[1] e também usam a *linguagem de programação* `gnuplot`,[6] para produzir os gráficos.

Algumas modificações podem ser facilmente feitas nos programas para outras simulações como

- é possível incluir no programa uma saída de dados para ter a matriz do $3-splines$ e não foi feito porque não é este o objetivo aqui,
- introduzir uma medida randômica para os intervalos, e deixei isto comentado no programa, permitindo que a partição não seja uniforme,
- iniciar o processo por um intervalo qualquer, e deixei um exemplo em que o processo se inicia pelo intervalo médio.

Observe que estou citando uma classe de programas,
`PrimitivaQuaseSplines*.calc`
cuja raiz dos nomes é `PrimitivaQuaseSplines` e que podem ser baixados da página indicada, todos produzidos sob GPL, mesmo que esta indicação não apareça nos programas. Leia também os comentários, dentro dos programas, para complementar as informações dadas aqui.

A figura (fig. 2), página 6,

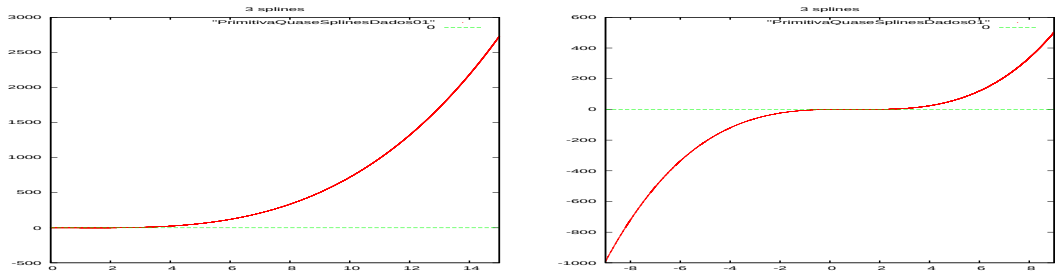

Figura 1: um $3-splines$ gerados pelo programa

exibe $3-splines$ gerados por um dos programas citados, num deles usando o intervalo do meio como gerador e neste caso o domínio é um intervalo $[-9, 9]$ e no outro gráfico o intervalo inicial é $[0, 3]$.

Vou partir destes exemplos para apresentar uma forma cômoda de definir $n-splinesf$ associado a um intervalo $[0,\alpha]$ e a uma partição deste intervalo determinada pelos $m+1$ nós

$$x_0 = 0, \ldots, x_m = \alpha \tag{19}$$

com um polinômio de grau menor ou igual n definido em cada intervalo satisfazendo ao sistema de equações (eq. 1.2)- (eq. 1.9). que será a forma de definir $n-splines$ neste artigo. Entretanto não necessariamente usando o sistema de equações (eq. 1.15).

Como cada seção dum $n-splines$ 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 outro exemplo simples e de uso corriqueiro, é uma *função poligonal* cujo gráfico é constituído 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. Vou me direcionar na continuação para uma primitiva específica e também fixar os *splines* definidos num intervalo $[0,\alpha]$ de modo que também a primitiva seja um *splines* definidos num intervalo $[0,\alpha]$ todos com a condição inicial $(0,0)$. E vou referir-me, então, *à primitiva*. Facilmente a leitora poderá adaptar as contas para uma situação diferente desta, espero que o meu modo de escrever torne o texto mais legível e as contas muito mais simples.

O método que acabei de descrever no parágrafo anterior, difere fundamentalmente do que descrevi anteriormente e tem uma vantagem que você deve observar: agora estou selecionando a segunda derivada dum $3-splines$, um $1-splines$, como ponto de partida, em vez de usar uma seleção de coeficientes para um primeiro segmento como foi foco nos programas descritos anteriormente. A diferença: selecionado a segunda derivada você decide como será o comportamento do $3-splines$. Na próxima seção vou exatamente explorar este método.

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

2 Construção geométrica dum 3-splines

Nesta seção vou adotar uma estratégia diferente na construção dum outro tipo de exemplo, uma ferramento extremamente eficiente e também irá servir para evidenciar a existência dum *elemento básico* que será explorado na terceira seção que será um complemento desta e inteiramente independente da última onde vou tratar das primitivas de *splines*. A ferramenta central aqui é o cálculo de primitivas que será retomada na última seção.

Depois de fazer uma construção geométrica para produzir um determinado tipo de *splines* vou retomar a construção "*algébrica*" e calcular a primitiva dum *1-splines* e em seguida a primitiva desta para chegar a um *3-splines*. Um dos objetivos desta seção, e da que se lhe segue, se encontra na tentativa de evidenciar a semelhança entre *splines* e polinômios:

- uma primitiva dum *n-splines* é um *(n+1)-splines*,
- a derivada dum *n-splines* é um *(n-1)-splines*.
- Entretanto, um *3-splines* substitui de forma muito mais eficiente qualquer polinômio de grau n em qualquer circunstância.

Interessam-me as semelhanças entre *splines* e polinômios, porque as diferenças são muito grandes ... Se não lhe parecer interessante o projeto, pode saltar, sem prejuízo, diretamente para a última 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 ou mais *bolhas sucessivas* cujas áreas se anulem: por exemplo, segmentos de parábolas formando duas bolhas, uma positiva e a outra negativa. Confira a figura (fig. 2) página 6,

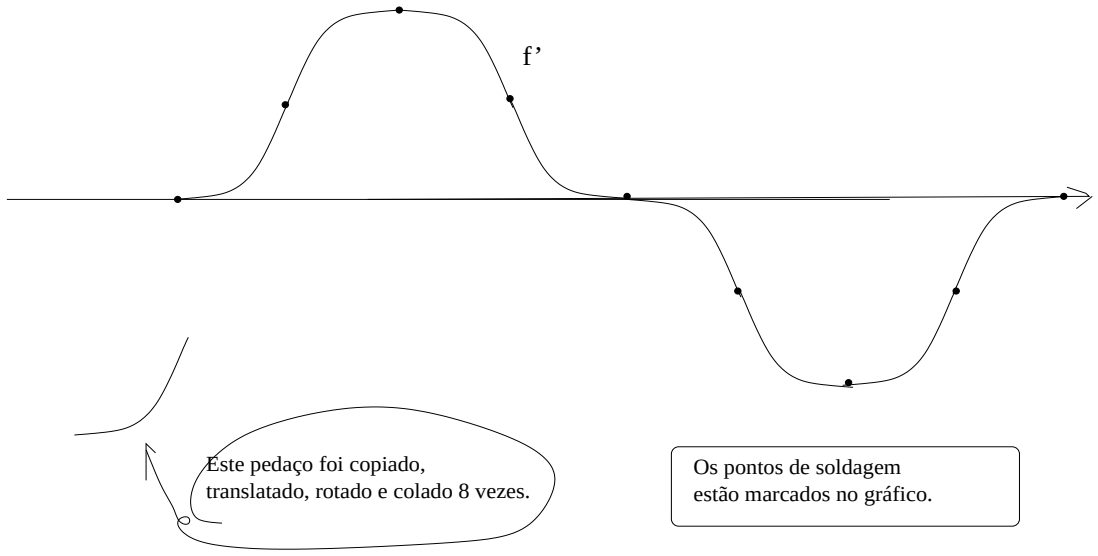

Figura 2: duas bolhas de integral zero

em que aparece f' e serão 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 (fig. 2). 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 conclui que

é preciso ter uma poligonal, um *1-splines*, formando quatro triângulos de áreas iguais, mas simétricas, (sinais contrários) duas a duas e você encontra f'' na figura (fig. 3) página 7,

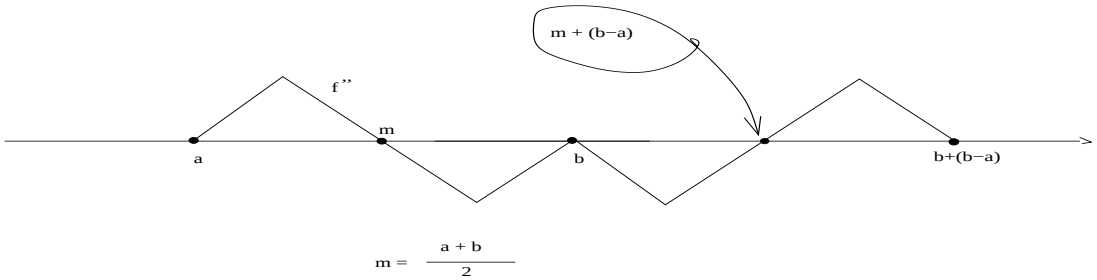

Figura 3: um splines a suporte compacto

A construção da figura (fig. 3) é também exclusivamente geométrica, nenhum cálculo algébrico foi feito, mas agora a interpretação geométrica vai ser transformada em *equações algébricas*:

$$m = \frac{a+b}{2}; \tag{20}$$

$$f''(x) \text{ é um 1-splines com suporte } [a, b]; \tag{21}$$

$$f'(x) = \int_a^x f''(t)dt; \tag{22}$$

$$x < b + (b - a); f'(x) = -\int_b^x f''(t - b + a)dt; \tag{23}$$

$$f' \text{ é um 2-splines a suporte compacto;} \tag{24}$$

$$f = \int_a^x f'(t)dt \text{ é um 3-splines a suporte compacto;} \tag{25}$$

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 geométrica* que você queira um $n - splines$ e faça o reverso duma *equação diferencial*, como no exemplo anterior, que você constrói o $n - splines$ desejado.

A figura (fig. 4) página 8,

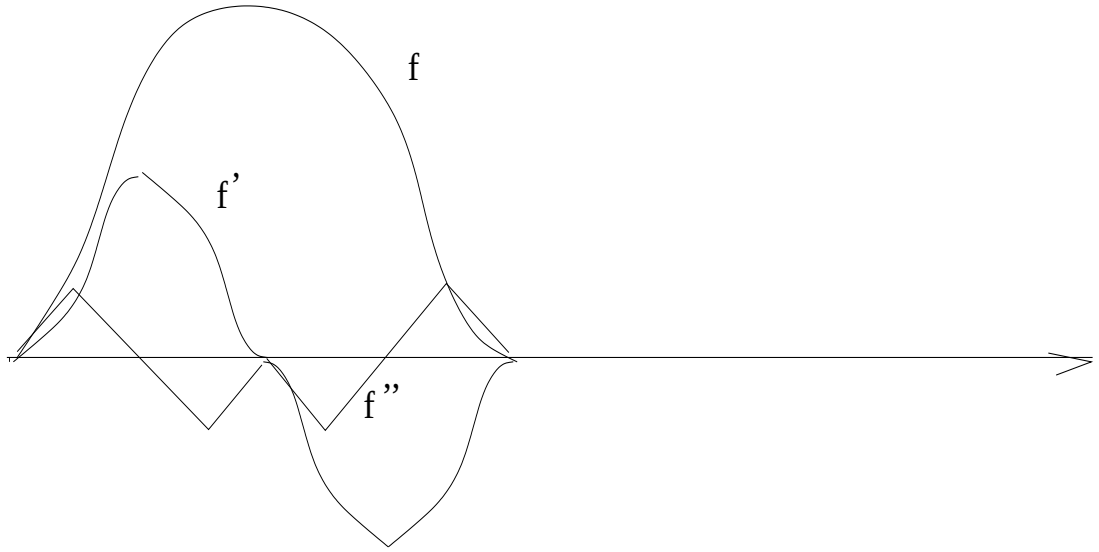

Figura 4: 3 splines a suporte compacto

que foi feita usando o editor de gráficos `xfig`, mostra o gráfico da construção formalizada nas equações (eq. 2.20)- (eq. 2.25).

Na figura (fig. 5), página 8,

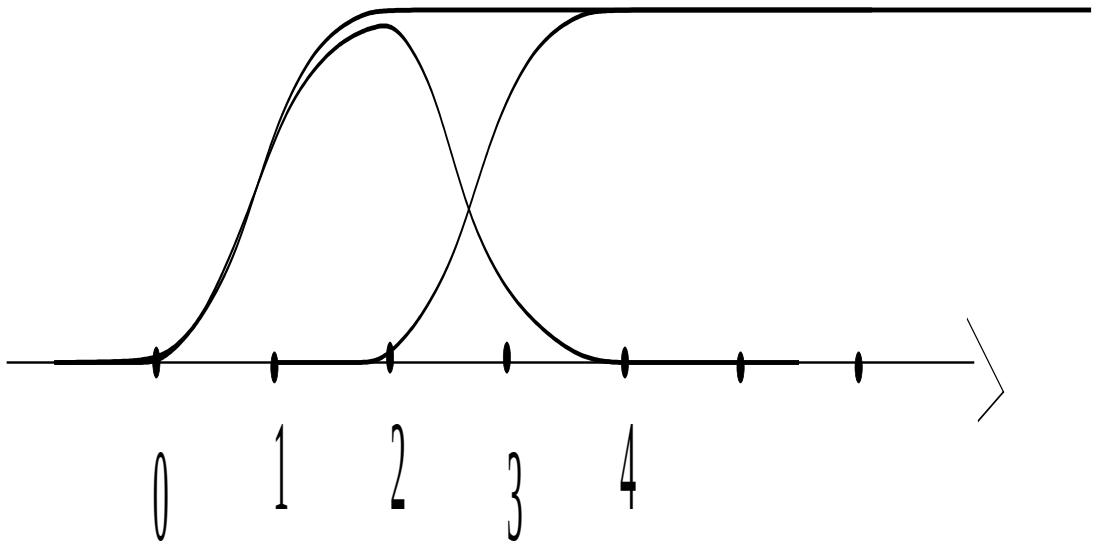

Figura 5: 3 splines a suporte compacto produzida por gnuplot

você pode ver gráfico *artístico* da figura da integral dum *splines* a suporte compacto, (fig. 4), produzido com o editor gráfico `xfig` e na figura (fig. 6), página

9,

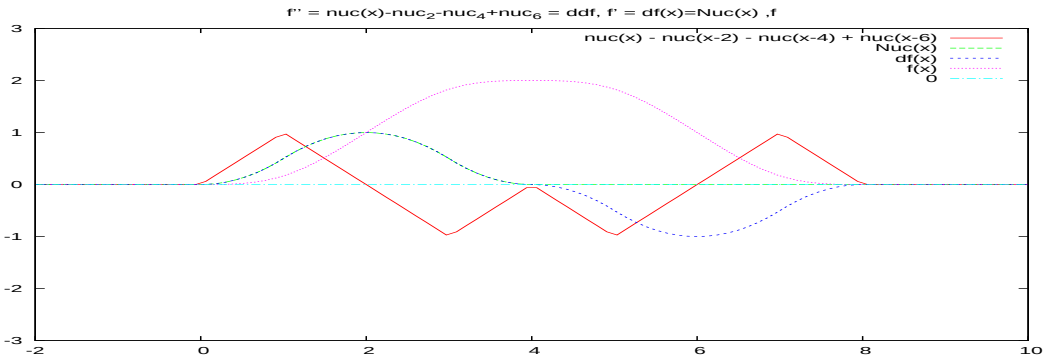

Figura 6: primeira e segunda derivada, $3-$ *splines*

você pode ver a sequência gráfica dum $3-$ *splines* a suporte compacto, ou alternativamente, a sucessão de primitivas dum splines a suporte compacto com a condição de energia zero imposta. O gráfico foi feito com `gnuplot` à partir das equações calculadas, algebricamente, na próxima seção. Você pode reproduzir ou alterar o gráfico a partir do programa [3, 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.

2.1 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 (fig. 3) página 7 e nas seguintes (4), (5).

Tenha a curiosidade ler o programa citado, e procure entender o significado das translações 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. Permita-me insistir na importância deste *elemento*:

- Chamei de *nuc* porque ele tem a propriedade que se espera dos núcleos em diversos lugares na *análise funcional*, *teoria de operadores*, *wavelets* - integral 1 e uma *unidade aproximada* relativamente à convolução,
- A equação

$$\rho \overset{t\mapsto nuc(t)}{\mapsto} \rho nuc(\rho t) \tag{26}$$

transforma o suporte do *nuc* naquele que você preferir. Observe que se trata duma *família a um parâmetro* indicado com o símbolo ρ.

- Incluindo translações, *dilações* como se prefere dizer quando se fala no dialeto de *wavelets* você cria as perturbações que você desejar incluídas na derivada se propagando para o $3-$*splines*.

Observe que, nas contas que fiz, 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,

$$nuc(x) = \begin{cases} x<0 & \Rightarrow & 0; \\ x<1 & \Rightarrow & x; \\ x<2 & \Rightarrow & 2-x; \\ x\geq 2 & \Rightarrow & 0; \end{cases} \tag{27}$$

$$f''(x) = nuc(x) - nuc(x-2); \tag{28}$$

$$f''(x) = \begin{cases} x<0 & \Rightarrow & 0; \\ x<1 & \Rightarrow & x; \\ x<2 & \Rightarrow & 2-x; \\ x<3 & \Rightarrow & 2-x; \\ x<4 & \Rightarrow & -4+x; \\ x\geq 5 & \Rightarrow & 0; \end{cases} \tag{29}$$

$$f'(x) = Nuc - Nuc_2; Nuc \text{ é uma primitiva de } nuc; \tag{30}$$

$$Nuc(x) = \begin{cases} 0 & \Rightarrow & x<0; \\ x<1 & \Rightarrow & \frac{x^2}{2}; \\ x<3 & \Rightarrow & 1-\frac{(2-x)^2}{2}; \\ x<4 & \Rightarrow & \frac{(4-x)^2}{2}; \\ x\geq 5 & \Rightarrow & 0; \end{cases} \tag{31}$$

$$f'(x) = Nuc - Nuc_2 = \begin{cases} x<0 & \Rightarrow & 0; \\ x<1 & \Rightarrow & \frac{x^2}{2}; \\ x<3 & \Rightarrow & 1-\frac{(2-x)^2}{2}; \\ x<4 & \Rightarrow & \frac{(4-x)^2}{2}; \\ x<5 & \Rightarrow & -\frac{(x-4)^2}{2}; \\ x<7 & \Rightarrow & -1+\frac{(6-x)^2}{2}; \\ x<8 & \Rightarrow & -\frac{(8-x)^2}{2}; \\ x\geq 9 & & 0; \end{cases} \tag{32}$$

$$f(x)=\begin{cases} x<0 & \Rightarrow & 0; \\ x<1 & \Rightarrow & \frac{x^3}{6}; \\ x<3 & \Rightarrow & \frac{(2-x)^3}{6}+x-1; \\ x<4 & \Rightarrow & 2-\frac{(x-4)^3}{6}; \\ x<5 & \Rightarrow & 2-\frac{(4-x)^3}{6}; \\ x<7 & \Rightarrow & 7-x-\frac{(6-x)^3}{6}; \\ x<8 & \Rightarrow & \frac{(8-x)^3}{6}; \\ x\geq 9 & \Rightarrow & 0; \end{cases} \tag{33}$$

O código em `gnuplot` para visualizar estes splines pode ser baixado do link [3, 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:

$$nuc(x)=\begin{cases} \text{Se } x<0 & 0; \\ \text{Se } x<1 & x; \\ \text{Se } x<2 & 2-x; \\ \text{Se } x>2 & 0; \end{cases} \tag{34}$$

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 comum dos editores para programação em `LinuX` que me permite usar a contra barra para quebrar uma linha ao longo de várias linhas.

Na definição dos splines `spl2(x), spl3(x)`, também usei, no programa, a facilidade de edição de `LinuX` em que a `contra barra` 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. 2.34). 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].

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 ampliação do que foi feito na construção do $3-$splines na seção anterior.

[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`.

Esta seção é apenas uma complementação à anterior e sem nenhuma ligação com a seguinte que é a última e voltada para cálculo de primitivas de *splines*. Esta seção pode ser ignorada se não lhe parecer interessante sem prejuízo para a próxima, mas você perde uma construção interessante feita usando "*primitivas de splines*"...

Deixo como exercício o teorema seguinte:

Teorema 1 () *Se f for um $n-$splines periódico então sua derivada será periódica. A recíproca não é verdadeira, é preciso acrescentar à condição de* energia periodicamente nula*, na derivada, para que a primitiva seja periódica. A derivada tem que ter valor médio nulo sobre qualquer intervalo cuja medida seja um período.*

Com estas duas condições sobre a derivada um $n-$splines periódico se origina dum $1-$splines periódico ou melhor dum $0-$splines periódicos 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, obviamente, criando outro "*de classe de continuidade -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, porque as combinações lineares não poderiam eliminar pelo menos um dos saltos.

Então

$$g_0 = \chi_{[0,1]} - \chi_{[1,2]} - \chi_{[2,3]} + \chi_{[3,4]}; \tag{35}$$

$$g_0 = \eta - \eta_1 - \eta_2 + \eta_3; \tag{36}$$

$$\eta = \chi_{[0,1]}; \tag{37}$$

é um $0-splines$ cuja integral é nula. Se este "padrão" for repetido, quer dizer transladado usando a medida do suporte, $\underline{a}$, como parâmetro da translação, o resultado é um $0-splines$ de período $\underline{a}$:

$$f_0(x) = \sum_{k\in\mathbf{Z}} g_{0,4k}(x); \tag{38}$$

neste caso se obtém um $0-splines$ de período 4.

Na seção anterior mostrei como se pode obter um $n-splines$ positivo à suporte compacto. Seja η este $n-splines$ e $\underline{a}$ a medida do suporte, então

$$f_0(x) = \sum_{k\in\mathbf{Z}} \eta_{ka}(x) - \eta_{(k+1)a}; \tag{39}$$

é um $n-splines$ de período $\underline{2a}$ cuja figura artística pode ser vista na (fig. 7) página 13,

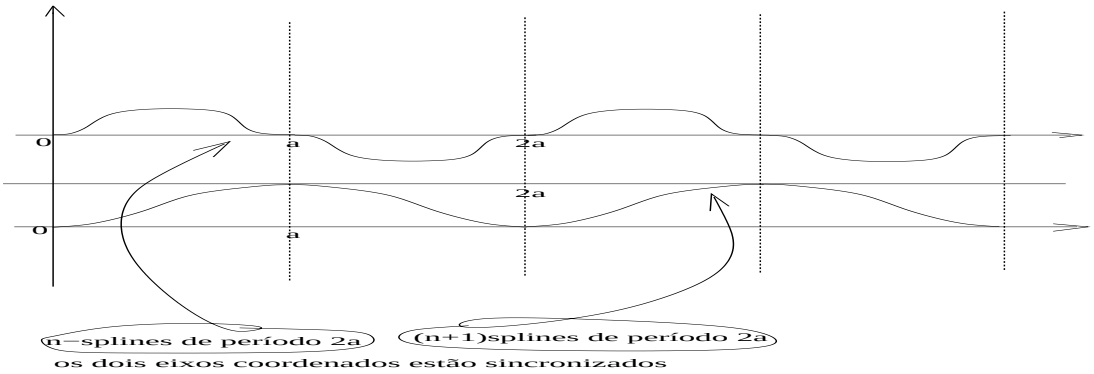

Figura 7: splines periódicos

junto com sua primitiva, no sistema de eixos abaixo, que é um $(n+1)$*splines* de período $\underline{2a}$

O cálculo duma primitiva do $(n+1)-$*splines* que aparece na figura (fig. 7) dá um exemplo para condição insuficiente do teorema 1.

Observe que a soma na equação (eq. 3.39) é "*formal*" e na verdade nunca haverá mais do 1 parcela no cálculo de $f_0(x)$.

Penso que fica evidente que apenas repeti, geometricamente, à construção da seção anterior, se o $n-$*splines* for a suporte compacto $\underline{a}$, a soma com a inversa aditiva de sua a-translação é um $n-$*splines* for a suporte compacto $\underline{2a}$ de *energia nula* e consequente sua primitiva é um $n-$*splines* a suporte compacto $\underline{2a}$. E as somas formais a medida do suporte como parâmetro de translação será *splines* periódico de período $\underline{2a}$.

A leitora poderia opor uma pergunta, ou questionar o método: houve um salto do caso $3-$*splines* para o caso $n-$*splines*? Na verdade houve uma omissão, se trata dum processo indutivo cujo primeiro elo foi construído. A hipótese de indução é a de que dado $(n-1)-$*splines* f com energia nula e período a, definido no intervalo $[0,a$, então a sua primitiva

$$F(x)=\int_0^x f(t)dt \tag{40}$$

é um $(n)-$*splines* de período $2a$.

Então a primitiva

$$\int_0^x F(t)-F_{2a}(t)dt \tag{41}$$

é um $(n+1)-$*splines* de período $4a$.

Fechada a omissão que não creio que mereça a distinção de *teorema*.

É interessante mencionar que existe uma alternativa a esta construção usando convolução:

- o produto de convolução dum $0-$*splines* por ele mesmo, ainda chamado de segunda potência por convolução, é $1-$*splines*. Por exemplo se $f=\chi_{[0,a]}$

então $f * f$ é um $1 - splines$ com suporte $[0, 2a]$, e $f - f_{2a}$ tem energia nula.

- se g for um $n - splines$ a suporte compacto $[0, a]$ então $h = f * g$ é um $(n + 1) - splines$ com suporte $[0, 2a]$ e $h - h_{2a}$ tem energia nula.
- A $n-$ésima potência por convolução de $f = \chi_{[0,a]}$ é $(n - 1) - splines$ com suporte $[0, na]$.

Esta sequência de ideias mostra que posso construir diretamente um $n - splines$ à suporte compacto predizendo o seu comportamento a partir de sua $n-$ésima derivada, que será uma combinação linear de funções características de intervalos, apenas tenho que usar convolução como metodologia. No artigo [2] você pode encontrar o meio para calcular a enésima potência por convolução de $f = \chi_{[0,a]}$ com a indicação de programas para fazê-lo.

Na próxima seção vou mostrar como calcular a $n-$ésima primitiva de um $n-$splines com condição inicial $(0, 0)$.

4 Primitiva dum n-splines

Vou introduzir uma versão mais fraca de polinomial por pedaços, *n-quase-splines* como sendo uma função polinomial por pedaços que, eventualmente, não é de classe $\mathcal{C}^{n-1}$ nos extremos do intervalo: tem pelo menos um ponto de descontinuidade. Então um $n - splines$ é *n-quase-splines*. Como isto vou me permitir ao uso de polinômios de grau n para cortar-lhes os gráficos em pedaços e os colar de volta. O resultado não seria um $n - splines$. Todos os resultados desta seção foram construídos para este novo objeto mas valem *verbatim* para os $n - splines$ e assim evito comentários aborrecidos sobre descontinuidade.

Na figura (fig. 8), página 14,

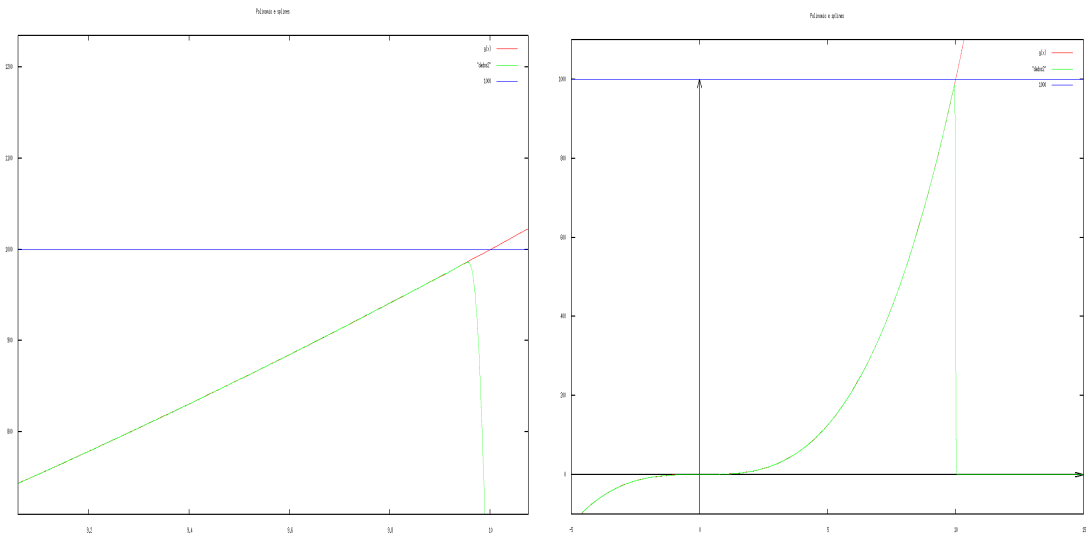

Figura 8: função polinomial enjanelada e $n-$splines

você pode ver, à direita um detalhe do que se vê à esquerda, o gráfico do polinômio $P(x) = x^3$ numa *janela* $[0, 10] \times [-100, 1000]$ junto com um gráfico dum $5 - splines$ cujo gráfico é uma aproximação para o gráfico do polinômio no intervalo $[0, 10]$. O $5 - splines$ foi obtido *por convolução do polinômio com um* $2 - splines$ que é uma *unidade aproximada* por convolução, *uma aproximação* da medida de Dirac que é a unidade no produto por convolução. Eu poderia ter definido uma função por pedaços, igual ao polinômio dentro da janela e com valor zero fora da mesma janela e o resultado seria o mesmo do gráfico. Esta função por pedaços é um exemplo de *quase-splines* que estou usando neste artigo, eu não poderia chamar o polinômio enjanelado de *splines* portanto preciso de $quase - splines$. Todas as afirmações envolvendo $quase - splines$ valem para *splines* se não for imposta a condição de continuidade, e um *n-splines* é *n-quase-splines* com classe de continuidade $n - 1$.

Relembrando, a notação:

$$(x_k, y_k)_{k\in\{0,\dots,n\}} \text{ uma sucessão de nós;} \tag{42}$$

$$(x_k)_{k=0}^{n} \text{ os nós da partição } \Pi([0, \alpha]); \tag{43}$$

$$(x_k, y_k)_{k=0}^{n} \text{ são os pontos de precisão de } f \tag{44}$$

$$f_k : [x_k, x_{k+1}] \to \mathbf{R}; \text{ é um polinômio de grau } n; \tag{45}$$

$$f_k = (a_{k,0}, a_{k,1}, \dots, a_{k,n}); f_k(x) = \sum_{j=0}^{n} a_{k,j}(x - x_k)^j; \tag{46}$$

uma repetição parcial das equações (1.2) e seguintes restringindo-me ao que interessa para esta seção.

A função f_k é a restrição de f ao intervalo $[x_k, x_{k+1}]$, f, um $n - quase - splines$ que interpola os pontos do plano na equação (eq. 4.44).

A leitora pode observar, com alguma surpresa, que o intervalo escolhido como domínio é um intervalo específico, $[0, \alpha]$, e consequentemente questionar a generalidade do método. Ocorre que, dado um intervalo qualquer de medida α onde esteja definido um $n - quase - splines$, eu posso obter uma sua imagem via transformação linear bijetiva e contínua, um isomorfismo dum espaço vetorial adequado de *splines*, noutro $n - quase - splines$ definido no intervalo $[0, \alpha]$ de tal modo que ambos tenham as mesmas propriedades geométricas, em particular a mesma integral, um será translação do outro. Como o cálculo da integral considerando a condição inicial $(0, 0)$ é mais simples, se justifica a pseudo particularização selecionada,

$$f(x) = (x - a)^k; \int_a^{a+\Delta} (x - a)^k dx; \tag{47}$$

$$\int_a^{a+\Delta} (x - a)^k dx = \int_0^{\Delta} x^k dx = \tfrac{\Delta^{k+1}}{k+1}; \tag{48}$$

$$[_{(1)}f](\alpha) = \tfrac{\alpha^{k+1}}{k+1}; \tag{49}$$

porque a condição inicial é $(0, 0)$.

Vou usar seguidamente esta *mudança de variável* nos cálculos que se seguem e na última linha da sequência de equações (eq. 4.47)- (eq. 4.49) estou usando a notação $[_{(1)}f]$ para representar a primitiva particular de f com condição inicial $(0,0)$ uma *imitação da notação para derivadas que não foi inventada por mim.*

Algumas, ou todas as listas de coeficientes, exceto uma, podem ser *identicamente zero*, porque, entre os segmentos do gráfico dum $n-quase-splines$ pode haver segmentos de reta, um polinômio de grau zero que pode, inclusive ser o polinômio identicamente zero, mas pelo menos um dos segmentos tem que ser um polinômio do grau n.

Enfim um $n-quase-splines$ fica caracterizado por uma matriz $m\times(n+1)$ de coeficientes em que cada linha está associada a um dos intervalos da partição de $[0,\alpha]$. Esta matriz não é um *operador* no sentido da Álgebra Linear, é *apenas uma lista de coeficientes.* Cada linha traz os coeficientes correspondentes à f_k em que $k\in\{0,\ldots,m-1\}$ e funciona bem computacionalmente ao programar usando listas, ao exaurir a lista, foram usados todos os coeficientes do $n-quase-splines$. `Python` é uma linguagem bem apropriada para este trabalho, porque trabalha com listas como um dado básico, também o fazem `scilab` e as linguagens da família `LISP`.

Qualquer $n-quase-splines$ a suporte compacto satisfaz às condições das equações (eq. 4.42) - (eq. 4.46), com exceção da condição inicial mas a observação anterior mostra que se trata duma restrição passível de ser desfeita. A recíproca não é verdadeira e estas equações não servem como definição de $n-quase-splines$ como ficou evidenciado na primeira seção. Os coeficientes tem que ser calculados para que as sucessivas derivadas, em cada nó, sejam coincidentes para os segmentos f_{k-1},f_k. Mas, feitos os cálculos, é suficiente apresentar os coeficientes

$$a_{k,0},\ldots,a_{k,n};k\in\{0,\ldots,m-1\}; \tag{50}$$

de f_k e servem para construir exemplos simples de $n-quase-splines$ como já mostrei na primeira seção.

Um exemplo bem simples de $n-quase-splines$ é obtido cortando segmentos do gráfico dum polinômio P associados aos intervalos duma partição do intervalo $[0,\alpha]$,

$$P(x)=x^n; \tag{51}$$

$$f_k(x);k\in\{0,\ldots,m-1\}; \tag{52}$$

$$f_k=P|_{[x_k,x_{k+1}]};x_m=\alpha;x_0=0; \tag{53}$$

e colando-os de volta para obter um *quase-splines* cujo gráfico coincide com o gráfico de P sobre o intervalo $[0,\alpha]$.

É um exemplo de $n-quase-splines$ a suporte compacto cujo gráfico coincide com o gráfico de P sobre o intervalo $[0,\alpha]$, ou, colocado de outra forma, cortei o gráfico de $x\mapsto x^n$ sobre cada subintervalo da partição e os colei de volta construindo um $n-quase-splines$.

A figura (fig. 8), página 14, é uma construção deste tipo.

Vale fazer isto para qualquer função polinomial de grau n e observe aqui uma aplicação interessante da *Formula de Taylor*: a matriz dos coeficientes dum tal $n-quase-splines$ são exatamente os coeficientes de Taylor do polinômio desenvolvido em cada extremo x_k dos intervalos da partição. Como a *fórmula de Taylor dum polinômio* é outro polinômio com o mesmo grau, esta afirmação prova o teorema,

Teorema 2 (coeficientes dum $n-quase-splines$**)** *Fórmula de Taylor Se f for um $n-quase-splines$ então os coeficientes relativos ao intervalo $[x_k, x_{k+1}]$ são os coeficientes do desenvolvimento de Taylor no ponto x_k*

$$a_{k,0} = f(x_k), \ldots, a_{k,p} = \frac{d^p f}{dx^p}(x_k)/p!, \ldots, a_{k,n} = \frac{d^n f}{dx^n}(x_k)/n!; \tag{54}$$

$$f_k(x) = \sum_{j=0}^{n} a_{k,j}(x - x_k)^j \tag{55}$$

Este teorema em geral é um instrumento teórico com grande interesse prático uma vez que vale a igualdade na fórmula de Taylor, que é um polinômio de grau n fornecendo um método para calcular os coeficientes do segmento f_k do $n-quase-splines$ f.

Na verificação e construção do algoritmo descrito no próximo teorema eu usei o $n-quase-splines$ obtido ao cortar o gráfico de $f(x) = x^n$ em cima dos nós duma partição, colocando de volta para produzir o $quase-splines$ como *teste* para metodologia implementada aqui uma vez que a integral de f é fácil de calcular servindo para verificar os resultados na construção do algoritmo desejado.

Teorema 3 (integral) *dum $n-quase-splines$ Seja f um $n-quase-splines$ a suporte compacto, como caracterizado pelas equações (eq. 4.42)- (eq. 4.46). Sua primitiva com condição inicial $(0,0)$, $[_{(1)}f]$, é a soma*

$$0 \leq j \leq n; x \in [x_k, x_{k+1}]; k \in \{0, \ldots, m-1\}; \tag{56}$$

$$y = f(x); f_k = f|_{[x_k, x_{k+1}]}; f_k = \sum_{j=0}^{n} a_{k,j}x^j; (a_{k,0}, a_{k,1}, \ldots a_{k,n}); \tag{57}$$

$$s \in [x_k, x_{k+1}]; [_{(1)}f](s) = [_{(1)}f](x_k) + \int_{x_k}^{s} f_k(t)dt = \tag{58}$$

$$= [_{(1)}f](x_k) + \int_{0}^{\Delta} f_k(t + x_k)dt; \Delta = s - x_k; \tag{59}$$

$$[_{(1)}f](s) = \sum_{j=0}^{k-1} \int_{0}^{\Delta x_k} f_j(t + x_k)dt + \int_{0}^{\Delta} f_k(t + x_k)dt; \Delta = s - x_k; \tag{60}$$

$$[_{(1)}f](s) = \sum_{p=0}^{k-1} \sum_{j=0}^{n} \frac{a_{p,j}\Delta x_j^{j+1}}{j+1} + \sum_{j=0}^{n} a_{k,j}\frac{\Delta^{j+1}}{j+1}; \tag{61}$$

$$[_{(1)}f](\alpha) = \sum_{k=0}^{m-1} \sum_{j=0}^{n} a_{k,j}\frac{\Delta x_k^{j+1}}{j+1}; \tag{62}$$

Dem: *Porque a integral dum $n-quase-splines$ é a soma das integrais de polinômios de grau menor ou igual a n. A diferença entre as duas últimas expressões se deve a que na última se tem a integral sobre o intervalo $[0,\alpha]$ enquanto que na anterior a integral sobre o intervalo $[N-1,s]$ é somada às integrais sobre os subintervalos a anteriores a este.* **q.e.d .**

Observe que $[_{(1)}f](x)$ não é mais um $n-splines$ a suporte compacto como mostram as seções anteriores uma vez que não existe a condição de energia zero para f.

Os programas que calculam estas integrais estão desenvolvido no artigo em preparação, *Mínimos Quadrados, como aplicação do Cálculo Variacional* onde será feita a citação com link para os programas sejam baixados.

Teorema 4 (n−ésima) *primitiva dum $n-splines$ Seja f um $n-splines$ como caracterizado pelas equações (eq. 4.42)- (eq. 4.46). Sua $N-$ primitiva com condição inicial $(0,0)$, $[_{(N)}f]$, é a soma*

$$[_{(N)}f](\alpha)=\sum_{k=0}^{m-1}\sum_{j=0}^{n}a_{k,j}\frac{\Delta x_k^{j+N}}{j+N}; \tag{63}$$

Dem: *É o mesmo que calcular a primitiva de ordem N dum polinômio com condição inicial $(0,0)$.* **q.e.d .**

Referências

[1] David I. Bell Landon Curt Noll and other. Calc - arbitrary precision calculator. Technical report, http://www.isthe.com/chongo/, 2011.

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

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

[4] T. Praciano-Pereira. Primitivas dum n-splines. *sobralmatematica.org/preprints/*, 2015, 04:11, 2015.

[5] Grace Wahba. *Spline models for observational data*, volume 59. Siam, 1990.

[6] Thomas Williams, Colin Kelley, and many others. gnuplot, software to make graphics. Technical report, http://www.gnuplot.info, 2010.