PRIMEIRO PROJETO DE ANÁLISE NUMÉRICA II

Curvas de Bézier e Desenho de Fontes Tipográficas

ALUNO: Fabricio C. Machado
RA: 102174
fabcm1@gmail.com

Abril de 2013

Resumo

Neste projeto é feito um estudo das curvas de Bézier e suas principais propriedades. Também é observado que estas podem ser vistas como um caso de interpolação segundo Hermite. Em seguida é apresentado o programa METAFONTusado no designer de fontes e descrito como este programa usa as curvas de Bézier no desenho de fontes tipográficas.

Sumário

1 Introdução

A impressão de um documento é baseada em uma matriz de pontos preenchida pela impressora. Os objetos assim descritos são chamados *rasterizados* e os objetos salvos em um formato desse tipo são muito dependentes da resolução usada, pois não podem ser ampliados sem grande perda de qualidade.

Essa propriedade contrasta com os formatos ditos *vetoriais*, como o PostScript, onde tudo, até mesmo o texto, é descrito em termos de linhas retas e *curvas de Bézier*, o que permite que redimensionamentos, rotações e outras transformações possam ser feitas facilmente. Apenas no momento da impressão é que esses documentos são rasterizados, ao custo de certo processamento. O formato PDF é outro exemplo de formato vetorial, com a diferença de ser um pouco mais descritivo que o PostScript (este é uma linguagem de programação completa, com loops e condicionais), mas ainda assim portável, por possuir toda a informação necessária (como descrição das fontes) comprimida em um único arquivo.

Procurando evitar entrar na descrição da complexa variedade de extensões, formatos e padrões usados hoje em dia, este trabalho focará em um único programa que reúne todos os elementos de criação, descrição matemática e impressão de figuras e fontes. Muitas pessoas conhecem o programa TEX(ou sua versão posterior LATEX), muito usado na produção de textos, mas poucos conhecem seu irmão gêmeo: METAFONT, também criado por Donald E. Knuth. Este programa é uma linguagem de programação usada para descrever as fontes usadas pelo LATEX.

Na realidade, METAFONTfaz mais do que criar as fontes usadas nos documentos, a parte "META" de seu nome é mais interessante: o objetivo não é só desenhar uma fonte, mas criar uma descrição matemática desta em alto nível, de modo que novas fontes possam ser criadas apenas alterando alguns parâmetros e melhores resultados possam ser obtidos. Outra vantagem da fonte ser criada dessa forma é que ela se torna uma *fonte vetorial*, cuja qualidade não depende tanto da resolução usada na visualização (são fontes assim que permitem que arquivos PDF possam ser ampliados sem grande perda de qualidade, em oposição a um texto contido em uma figura).

Descrever uma fonte matematicamente é um trabalho difícil. É necessária uma ferramenta que dê bons resultados e que seja flexível para acomodar os objetivos do usuário, sem ser muito complicada. Para isso, METAFONTusa as curvas de Bézier, que permitem que uma curva seja gerada a partir de poucos pontos de controle fornecidos pelo designer.

O tema deste projeto será entender como METAFONTé capaz de desenhar uma curva "agradável" a partir de poucos pontos fornecidos. Para isso, o projeto está dividido em duas partes: primeiro será feito um estudo matemático das curvas de Bézier e em seguida será apresentado o programa METAFONTem maior detalhe, em especial, o funcionamento de seu comando `draw`.

2 Curvas de Bézier

2.1 Origem

A base teórica na qual as curvas de Bézier se baseiam reside no trabalho do matemático francês Charles Hermite (veremos a seguir que estas curvas são apenas um caso particular da interpolação segundo Hermite) e do matemático russo Sergei Bernstein. Entretanto, o potencial da aplicação dessas curvas no designer gráfico só foi descoberto em 1962 em um trabalho de Pierre Bézier, um engenheiro francês empregado da montadora de automóveis Renault interessado na descrição paramétrica de superfícies. Independentemente, Paul de Casteljau (trabalhando na Citroën) também produziu um trabalho importante alguns anos antes, em 1959, criando um algoritmo para a avaliação dessas curvas recursivamente (bissecção).

2.2 Definição

A *curva de Bézier* de ordem n é uma curva paramétrica definada por:

$$B(t) = \sum_{i=0}^{n} b_{i,n}(t)P_i = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i, \qquad t \in [0,1]$$

Os P_i são chamados de *pontos de controle* e a curva é definida a partir da base formada pelos $b_{i,n}(t)$, chamados *polinômios de Bernstein*:

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \qquad i = 0, 1, ..., n$$

2.3 Algumas Características

1. Note que P_0 e P_n são os pontos extremos da curva, isto é: $B(0) = P_0$ e $B(1) = P_1$.

2. Apesar da curva não passar pelos outros pontos de controle, ela está contida no fecho convexo destes pontos. Para ver isso, observe que $\sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} = (t + 1 - t)^n = 1$ e portanto todos os pontos de $B(t)$ são combinações convexas dos pontos de controle.

3. Relação Recursiva
   Denotanto $B_{P_0 P_1 ... P_n}$ a curva de Bézier com pontos de controle $P_0, P_1, ...P_n$, as curvas de ordem n podem ser definidas recursivamente da seguinte forma:

   $B_{P_0} = P_0$
   $B_{P_0 P_1 ... P_n} = (1-t) B_{P_0 P_1 ... P_{n-1}} + t B_{P_1 P_2 ... P_n}$

Essa equivalência pode ser demonstrada com a seguinte manipulação dos coeficientes:

$$\begin{aligned}
B(t) &= \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i = \\
&= \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-i} P_i + \sum_{i=1}^{n} \binom{n-1}{i-1} t^i (1-t)^{n-i} P_i \\
&= (1-t) \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i} P_i + t \sum_{i=1}^{n} \binom{n-1}{i-1} t^{i-1} (1-t)^{n-i} P_i \\
&= (1-t) \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i} P_i + t \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i} P_{i+1}
\end{aligned}$$

4. Relação Recursiva "Dinâmica"
   Aproveitando a última linha da demonstração anterior, podemos ainda escrever:

$$B(t) = \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i} . ((1-t)P_i + tP_{i+1})$$

   de modo que uma curva de ordem n pode ser vista como uma curva de ordem $n-1$ cujos pontos de controle variam dinamicamente. Esta propriedade é útil na construção das curvas, ilustraremos isso a seguir.

A partir daqui, restringiremos nossa análise às curvas de ordem 1, 2 e 3, que são as mais usadas na prática.

2.4 Bézier de ordem 1

A curva de ordem 1 é simplesmente $B(t) = (1-t)P_0 + tP_1$, um segmento de reta.

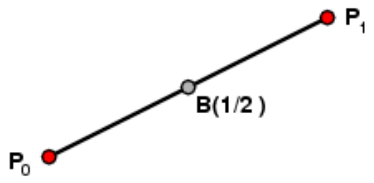

Figura 1: Curva de Bézier de ordem 1.

2.5 Bézier de ordem 2

A curva de ordem 2 é $B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2$.

Além das características já mencionadas no caso geral (P_0 e P_2 são os pontos extremos e a curva está contida no fecho convexo de P_0, P_1 e P_2), podemos interpretar o papel do ponto P_1 derivando a

curva:

$$\begin{aligned} B'(t) &= -2(1-t)P_0 + 2(1-t)P_1 - 2(1-t)tP_1 + 2tP_2 \\ &= 2(1-t)(P_1 - P_0) + 2t(P_2 - P_1) \end{aligned}$$

De modo que em $t = 0$ a tangente da curva é paralela a $P_1 - P_0$ e em $t = 1$ a tangente é paralela a $P_2 - P_1$.

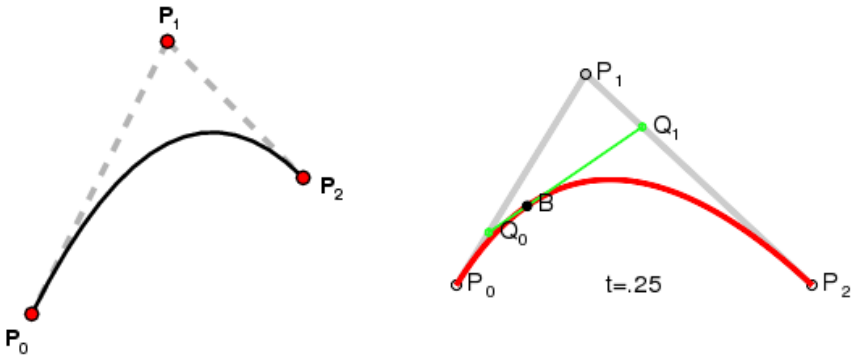

Figura 2: Curvas de Bézier de ordem 2. À direita, exemplo de sua construção recursiva.

2.6 Bézier de ordem 3

A curva de ordem 3 é $B(t) = (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3$.

Também podemos interpretar o papel dos pontos P_1 e P_2 derivando a curva:

$$\begin{aligned} B'(t) &= -3(1-t)^2P_0 + 3(1-t)^2P_1 - 6(1-t)tP_1 + 6(1-t)tP_2 - 3t^2P_2 + 3t^2P_3 \\ &= 3(1-t)^2(P_1 - P_0) + 6(1-t)t(P_2 - P_1) + 3t^2(P_3 - P_2) \end{aligned}$$

Em $t = 0$ a tangente da curva é paralela a $P_1 - P_0$ e em $t = 1$ a tangente é paralela a $P_3 - P_2$. Observe que com 4 pontos de controle somos capazes de determinar *independentemente* as direções de início e fim da curva. Essa característica é um dos motivos pelos quais a curva de Bézier cúbica é tão usada na prática.

Observamos ainda que a curva de Bézier é o único polinômio de grau de 3 que satisfaz essas características. Quando definimos $B(0)$, $B'(0)$, $B(1)$ e $B'(1)$ estamos, na realidade, fazendo uma interpolação segundo Hermite em cada uma das coordenadas, de forma que a curva de Bézier é apenas uma forma conveniente de representar a interpolação segundo Hermite através dos pontos de controle.

2.7 Bissecção

A bissecção é uma outra propriedade recursiva da curva de Bézier estreitamente relacionada com a propriedade apresentada anteriormente. Observe na figura a seguir a construção de uma curva de

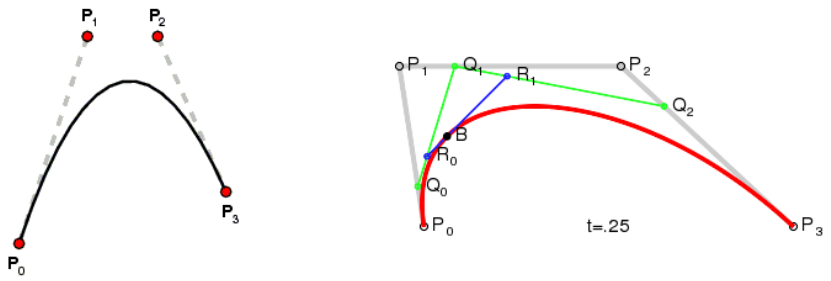

Figura 3: Curvas de Bézier de ordem 3. À direita, exemplo de sua construção recursiva.

ordem 3, com os pontos em $t = 1/2$:

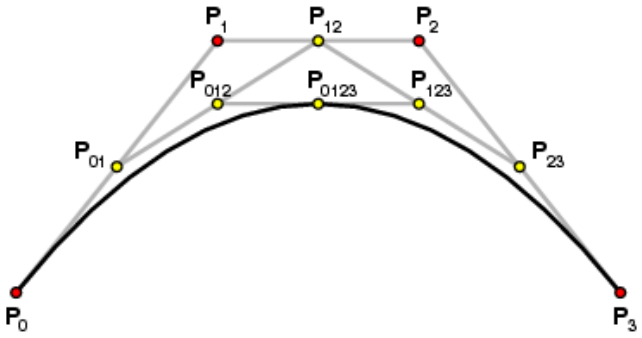

Figura 4: Bissecção de uma curva de Bézier.

Usando a notação apresentada na figura, vale a seguinte relação:

$$B_{P_0P_1P_2P_3}(t) = B_{P_0P_{01}P_{012}P_{0123}}(2t) = B_{P_{0123}P_{123}P_{23}P_3}(2t-1)$$

Assim, podemos dividir uma curva em duas curvas menores de mesma ordem, uma para ser avaliada em $t \in [0, 1/2]$ e a outra em $t \in [1/2, 1]$. Essa propriedade é muito útil em implementações, visto que não é necessário avaliar o polinômio com o qual a curva de Bézier foi definida e é fácil calcular os pontos de controle dessas novas curvas: basta manter uma lista dos pontos de controle e ir calculando seus pontos médios. Após algumas iterações, teremos uma lista de pontos que converge para a curva!

3 METAFONT

Como dito na introdução deste trabalho, METAFONTé uma linguagem de programação. Isso significa que para a produção de uma fonte ou mesmo um caracter, o usuário deve escrever um programa baseado numa sintaxe própria do METAFONTque possui suporte a funções, variáveis, loops e diversas características próprias, como a escolha do tipo de caneta a ser usado e a inclinação desta em cada ponto do desenho.

Para o contexto deste trabalho, vamos nos limitar ao fato de pontos poderem ser determinados através de coordenadas e gravados em variáveis, a partir das quais podemos passar instruções de desenho, como o comando `draw`:

3.1 Comando draw

A sintaxe básica do comando draw é bem simples. Dados, digamos, 5 pontos z_0, z_1, z_2, z_3 e z_4, podemos desenhar uma linha que passe por esses pontos com o comando:

$$\mathbf{draw} z_0..z_1..z_2..z_3..z_4$$

É possível ainda passar informações extras sobre a forma da curva, como a direção que ela deve ter ao passar por certo ponto:

$$\mathbf{draw} z0..z1..z2\{(1,-1)\}..z3\{(-1,1)\}..z4$$

Figura 5: As duas curvas geradas pelo comando draw.

O METAFONTusa curvas de Bézier de ordem 3 em seus desenhos, mas como estas curvas são usadas se os pontos de controle não são dados? De fato, é preciso dar apenas pontos onde a curva deve passar. Para resolver esse problema, internamente o programa possui uma rotina que calcula os pontos de controle restantes, de modo que uma curva de Bézier de ordem 3 é definida a cada 2 pontos.

3.2 Splines

De fato, o que o programa faz é calcular os pontos de controle de uma spline cúbica através dos pontos dados, de modo que a curva resultante e suas derivadas primeira e segunda sejam contínuas (o programa associa esta última a uma chamada “mock curvature”, que não é exatamente a derivada segunda mas algo correspondente). A formulação é semelhante à vista em aula, com a diferença de que no lugar dos coeficientes dos polinômios, as variáveis são os ângulos do vetor tangente à curva em cada ponto. O sistema resultante também é tridiagonal e diagonalmente dominante.

Aqui também são necessárias duas condições de contorno para definir a spline. Caso o usuário dê informações extras sobre a curva, o programa irá subdividir o caminho em diversas splines, usando as condições dadas como contorno. Por exemplo, no segundo comando `draw` da seção anterior, o

programa irá calcular uma spline de z_0 a z_2 (composta de 2 curvas de Bézier de ordem 3), outra de z_2 a z_3 (aqui só há uma curva, o único trabalho será calcular os pontos de controle) e outra de z_3 a z_4. Nos contornos onde não são dadas condições, o programa define a spline impondo condições sobre sua curvatura.

3.3 Desenhando uma curva

Calculados os pontos de controle, a próxima tarefa do programa é digitalizá-la[1]. Considerando $B(z_0, z_1, z_2, z_3; t)$ a curva a ser calculada, esta corresponde a dois polinômios, um em cada coordenada: $x(t) = B(x_0, x_1, x_2, x_3; t)$ e $y(t) = B(y_0, y_1, y_2, y_3; t)$. O objetivo é encontrar o conjunto de pontos inteiros $P = \{(\lfloor x(t) \rfloor, \lfloor y(t) \rfloor) \mid 0 \leq t \leq 1\}$.

Para isto, o programa faz uma simplificação: supõe que $x'(t)$ e $y'(t)$ são não decrescentes (é claro que esse não é o caso geral, mas em outra parte do código o programa consegue reduzir os outros casos a este) e decompõe o caminho em uma sequência de movimentos para a direita e para cima, usando a propriedade da bissecção comentada anteriormente.

De forma simplificada, o algoritmo é:

1. Se $\lfloor x_0 \rfloor = \lfloor x_3 \rfloor$, dê $\lfloor y_3 \rfloor - \lfloor y_0 \rfloor$ passos para cima.
2. Se $\lfloor y_0 \rfloor = \lfloor y_3 \rfloor$, dê $\lfloor x_3 \rfloor - \lfloor x_0 \rfloor$ passos para a direita.
3. Caso contrário, bisseccione a curva e repita o processo nas duas metades.

3.4 Exemplo

A seguir, um exemplo de um programa que descreve um caracter usando METAFONT:

```
1. u#:=4/8pt#;                                   Definição de um parâmetro de escala.
2. define_pixels(u);
3. beginchar(66,14u#,17u#,4.5u#);"Letter beta";
4. x1=2u; x2=x3=3u;                              Definição das coodenadas dos pontos.
5. bot y1=-5u; y2=8u; y3=14u;
6. x4=6.5u; top y4=h;
7. z5=(10u,12u);
8. z6=(7.5u,7.5u); z8=z6;
9. z7=(4u,7.5u);
```

[1] Vale a pena notar que a linguagem PostScript só foi criada em 1982, sendo que Donald Knuth começou a trabalhar no METAFONTem 1977, chegando em sua versão final em 1984. Por isso, o autor teve que se preocupar com todos os estágios da criação da fonte, inclusive a rasterização. Uma grande preocupação de Knuth era que seu programa obtivesse os mesmos resultados, independentemente das características da máquina, tendo que sofrer poucas adaptações para ser implementado em diferentes máquinas.

```
10. z9=(11.5u,2u);
11. z0=(5u,u);
12. penpos1(2u,10);
13. penpos2(u,20);
14. penpos3(u,-45);
15. penpos4(.8u,-90);
16. penpos5(1.5u,-180);
17. penpos6(.5u,120);
18. penpos7(.5u,0);
19. penpos8(.5u,210);
20. penpos9(1.5u,-180);
21. penpos0(.3u,20);
22. pickup pencircle;
23. penstroke z1e..z2e..z3e..z4e..z5e..z6e..{up}z7e..z8e..z9e..{up}z0e;
24. labels(range 0 thru 9);
25. endchar;
26. end
```

Definição da direção da caneta durante o traço.

O comando de desenho usado na linha 23 funciona de forma semelhante ao comando draw discutido, apenas também leva em conta o preenchimento do interior da letra, de acordo com as curvas feitas pela caneta.

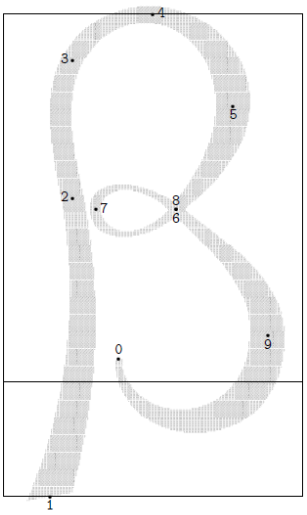

Figura 6: O caractere ß criado pelo programa exibido.

4 Considerações Finais

Neste projeto foi possível estudar as principais características das curvas de βézier e como suas propriedades as tornam úteis no desenho de curvas, sendo assim usadas em diversos softwares de designer gráfico.

Considerei especialmente interessantes as propriedades recursivas das curvas de Bézier e a forma como os polinômios de Bernstein conseguem relacionar combinações convexas com propriedades dos números binomiais. Uma dificuldade enfrentada neste estudo foi a pesquisa sobre o programa METAFONTque hoje em dia é apenas usado como subrotina do LaTeX, tendo então, poucos tutoriais disponíveis para o usuário final.

Vejo várias possibilidades para se aprofundar o trabalho apresentado, entre elas:

1. Um estudo mais aprofundado dos polinômios de Bernstein e suas aplicações.
2. Um estudo das superfícies de Bézier.
3. Um estudo mais aprofundado sobre o programa METAFONT, em especial aprender a usá-lo de forma eficiente para o desenvolvimento de uma fonte completa e compreender como tirar proveito de seu lado "META", isto é, como especificar uma fonte em termos de parâmetros ajustáveis e as sutilezas relacionadas.
4. Um estudo sobre os formatos usados na compressão e descrição de documentos hoje em dia. Em especial, as características dos arquivos PDF e Djvu.
5. Um estudo sobre o designer de fontes atual. Quais programas são usados hoje em dia e quais as sutilezas gráficas envolvidas (serifs, proporções, overshoot...).

Referências Bibliográficas

[1] KNUTH, Donald E. *The METAFONTbook.* Reading, Mass.; Wokingham: Addison-Wesley, c1986. (Computers & typesetting; v. C).

[2] KNUTH, Donald E. *METAFONT: the program.* Reading, Mass.; Wokingham: Addison-Wesley, c1986. (Computers & typesetting; v. D).

[3] Kaspar, Catherine. *Using Bezier Curves for Geometric Transformations* [monograph] MSM Creative Component, Fall 2009

[4] *Bézier curve.* (19 de abril de 2013). Em Wikipedia, The Free Encyclopedia. Acessado em 22:08, 22 de abril de 2013, em `http://en.wikipedia.org/w/index.php?title=B%C3%A9zier_curve&oldid=551068111`

[5] *PostScript* (18 de abril de 2013). Em Wikipedia, The Free Encyclopedia. Acessado em 22:14, 22 de abril de 2013, em `http://en.wikipedia.org/w/index.php?title=PostScript&oldid=550998725`

[6] Grandsire, Christophe. *The METAFONTtutorial*. (30 de dezembro de 2004) Em METAFONT Tutorial Page. Acessado em 22:33, 22 de abril de 2013, em `http://metafont.tutorial.free.fr/`

[7] Casselman, Bill. *From Bézier to Bernstein*. (novembro de 2008) Em Feature Column, Americam Mathematical Society. Acessado em 22:42, 22 de abril de 2013, em `http://www.ams.org/samplings/feature-column/fcarc-bezier`