Universidade Federal de Mato Grosso do Sul

Instituto de Matemática

Programa de Pós-Graduação

Mestrado Profissional em

Matemática em Rede Nacional

Estevão Vinícius Candia

MATEMÁTICA E O METAPOST

Campo Grande - MS

2018

Universidade Federal de Mato Grosso do Sul

Instituto de Matemática

Programa de Pós-Graduação

Mestrado Profissional em

Matemática em Rede Nacional

Estevão Vinícius Candia

MATEMÁTICA E O METAPOST

Orientadora: Profa. Dra. Elisabete Sousa Freitas

Dissertação apresentada ao Programa de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional do Instituto de Matemática da Universidade Federal de Mato Grosso do Sul - INMA/UFMS, como parte dos requisitos para obtenção do título de Mestre.

Campo Grande - MS

2018

MATEMÁTICA E O METAPOST

Estevão Vinícius Candia

Dissertação submetida ao Programa de Pós-Graduação em Matemática em Rede Nacional do Instituto de Matemática da Universidade Federal de Mato Grosso do Sul - INMA/UFMS, como parte dos requisitos para obtenção do título de Mestre.

Aprovada pela banca examinadora:

Prof. Dr. Claudemir Aniz
Universidade Federal de Mato Grosso do Sul - UFMS

Profa. Dra. Elisabete Sousa Freitas (Orientadora)
Universidade Federal de Mato Grosso do Sul - UFMS

Prof. Dr. Vanderlei Minori Horita
Universidade Estadual Paulista - UNESP

Campo Grande - MS, 01 de Novembro de 2018.

Aos meus pais, Eliane e Estevão, dedico.

"A essência da matemática reside na sua liberdade."

Georg Cantor

Agradecimentos

Aos meus pais, Eliane e Estevão, pela educação que me deram e pelos estímulos, desde criança, ao raciocínio e aos estudos. Agradeço pelo apoio que me deram durante todo meu processo acadêmico, desde a pré-escola até agora.

À minha amada esposa, Juliana, por sua compreensão e carinho mostrados durante todo o processo de pesquisa e elaboração dessa dissertação, período ao qual, sempre que possível, esteve ao meu lado.

A todos os professores do Instituto de Matemática da UFMS, que compartilharam comigo seu conhecimento e me ajudaram a aprender Matemática desde minha participação no Programa de Iniciação Científica Jr., em 2008, até agora. Em especial, aos professores Antonio Tamarozzi, Sonia Di Giácomo, Maura Candolo, Marilena Bittar, Flávia Zechineli, Janete Ferrareze, Rubia Santos e Claudemir Aniz. Muito obrigado!

À minha família e aos meus amigos, que com o incentivo, de certa forma, contribuíram na produção dessa dissertação. Em especial, agradeço a Nicole, por sempre deixar claro o orgulho que tem de ser minha irmã. O sentimento é recíproco.

Aos meus colegas da turma de 2016 do PROFMAT/UFMS, pelos momentos de estudo e também de descontração, pela troca de saberes e pelo apoio prestado durante esses anos. Também, à CAPES pelo apoio financeiro e ao PROFMAT por manter este programa de pós-graduação ao qual pude tirar proveito.

Ao professor Troy Henderson, pela atenção e suporte prestados com o previewer do METAPOST, a meu amigo Raphael Rocha e a Dailton Valverde, que me ajudaram muito com o LaTeX. Sem vocês, esse trabalho não seria o mesmo. Muito obrigado, mesmo!

Por fim, agradeço imensamente a professora Elisabete pela sugestão do tema e, posteriormente, pela dedicação, pela paciência, pela responsabilidade e comprometimento demonstradas durante a orientação deste trabalho. Seu conhecimento e paixão pela Matemática são cativantes e me motivaram a dar o meu melhor para o desenvolvimento dessa dissertação, ao quais serei eternamente grato.

Resumo

METAPOST é uma linguagem gráfica baseada no processamento batch. Proveniente do METAFONT, linguagem de programação criada pelo matemático e cientista da computação Donald Knuth, o METAPOST possui ferramentas que possibilitam a criação de imagens em alta qualidade geradas a partir de cálculos matemáticos. Este trabalho busca apresentar como funcionam os comandos básicos dessa linguagem, explicar alguns conceitos matemáticos inerentes à sua programação e mostrar como estes podem ser utilizados para criar figuras geométricas e gráficos de Funções Reais.

Palavras-chave: METAPOST; Geometria; Vetores; Transformações; Funções Reais.

Abstract

METAPOST is a graphical language based on batch processing. It comes from METAFONT, a programming language created by mathematician and computer scientist Donald Knuth. METAPOST provides tools that create high quality images generated from mathematical calculations. This thesis presents the basic commands of this language, explains some mathematical concepts related to its built-in functions and shows how they can be used to create geometric figures and graphs of Real Functions.

Keywords: METAPOST; Geometry; Vectors; Transformations; Real Functions.

Sumário

Lista de Figuras

Introdução

A comunicação escrita é base para o ensino e aprendizagem de várias áreas do conhecimento. Escrever um trabalho matemático legível e de forma elegante não era tão simples antigamente sem as ferramentas computacionais disponíveis hoje. A competência de expor conhecimento em conformidade com exigências técnicas, ou até mesmo normas pessoais, moveu pessoas a criar sistemas tipográficos de escrita que incluíssem símbolos e conceitos matemáticos. Donald Knuth[1] é um dos pioneiros nesse aspecto.

Knuth é autor de uma das principais referências da Ciência da Computação, o livro *The Art of Computer Programming*. Segundo Beebe [3], quando o primeiro volume apareceu em 1968, a maioria das composições tipográficas ainda era feita por digitação mecânica. Tipógrafos especialistas, mesmo com décadas de experiência, tinham de lidar com problemas de quebra de linha, quebra de página e layout de página. Em meados da década de 1970, os sistemas proprietários de editoração computacional haviam entrado no mercado e, na opinião de Knuth, haviam rebaixado seriamente a qualidade do material.

Alguns meses depois, Knuth veio a conhecer novos dispositivos que usavam técnicas digitais para criar imagens de letras e a sua estreita conexão com os algarismos binários da ciência da computação. Isso o levou a pensar em como ele próprio poderia projetar sistemas para colocar caracteres em uma página, desenhando os caracteres individuais como uma matriz de pontos pretos e brancos. Assim, surgem protótipos funcionais de dois programas de software para esse propósito, que Knuth descreve no livro *TEX e METAFONT: New Directions in Typesetting*.

A composição digital de seu projeto durou cerca de uma década e produziu vários outros livros. Também gerou teses de doutorado para Frank Liang [15], Michael Plass [22], Lynn Ruggles [24] e Ignacio Zabala Salelles [25]. Além disso, esses softwares viraram produto da indústria de formatação de documentos comerciais e das primeiras impressoras

[1] Professor Emérito da Universidade de Stanford.

a laser. O sistema TEX e o sistema LATEX, que foi baseado nele, tornaram-se os sistemas padrão de escrita tipográfica nas comunidades de ciência da computação, matemática e física. Ambos têm sido amplamente usados em muitos outros campos até hoje.

Outra tese de doutorado orientada por Knuth foi a de John Douglas Hobby [11]. Hobby continuou sua pesquisa sobre o METAFONT e criou, cerca de 10 anos depois, a linguagem gráfica METAPOST[2]. METAPOST é uma linguagem gráfica desenvolvida para gerar imagens com processamento batch, isto é, ela processa uma lista de comandos de texto para criar uma saída gráfica. Esse arquivo de saída é uma figura em PostScript que é baseada em cálculos matemáticos. Este tipo de imagem é excepcional pois pode ser redimensionada para qualquer resolução sem perda de qualidade.

Essa dissertação apresenta uma introdução aos comandos básicos dessa linguagem. Para isso, alguns conceitos matemáticos são explicados, pois fazem parte de sua programação. Espera-se mostrar como estes conceitos podem ser utilizados para criar figuras em alta qualidade utilizadas em áreas da Matemática como a Geometria e o estudo do gráfico de funções reais.

O Capítulo 1 aborda alguns comandos utilizados para criar figuras básicas com o METAPOST. Neste capítulo, é explicado o funcionamento do METAPOST, levando em conta como o seu fluxo de trabalho opera, quais são alguns tipos de variáveis utilizadas e quais são os operadores matemáticos disponíveis nele. Também é explicado como produzir uma figura e inseri-la em um documento TEX.

O Capítulo 2 discute conceitos matemáticos de vetores e de geometria que podem ser utilizados para fazer desenhos no METAPOST. Nele, é apresentada a base matemática para realizar algumas operações vetoriais disponíveis no METAPOST. Uma aplicação do produto interno para se traçar uma altura de um triângulo qualquer é exposta. São explanados também os conceitos geométricos utilizados para marcar os pontos notáveis de um triângulo, bem como desenhar um polígono regular.

O Capítulo 3 contém a base matemática utilizada para fazer transformações no plano. Alguns recursos disponíveis no METAPOST para isso são apresentados. Com esse recursos, é possível esboçar círculos e elipses. Além disso, é abordado como fazer os desenhos de dois problemas interessantes da Geometria.

[2]Segundo Vieth [27], Hobby pessoalmente prefere a escrita "MetaPost", escrita com fonte romana em vez da fonte `logo`, utilizada em textos editados em TEX. Porém, esse tipo de escrita foi introduzida pelo próprio criador do TEX, Donald Knuth. Logo, não há problemas em utilizar a escrita com a fonte `logo` e, assim, ela será utilizada nesta dissertação.

O Capítulo 4 discute a elaboração de gráficos de funções contínuas definidas em intervalos da reta. Inicialmente é feito um resumo com algumas proposições sobre funções contínuas. Em seguida, são apresentadas algumas técnicas do METAPOST para esboçar gráficos de funções, com diversos exemplos. Também são apresentadas macros do METAPOST que permitem utilizar operadores matemáticos não predefinidos.

O trabalho é finalizado com as considerações finais e algumas sugestões de trabalhos futuros.

Capítulo 1

Figuras Básicas

Neste capítulo, serão introduzidos alguns comandos básicos do METAPOST, bem como a produção de figuras e a inserção delas em documento TEX. O METAPOST é capaz de interpretar comandos de texto que lhe indicam o que, como e onde desenhar [1]. Para identificar as posições no desenho utiliza-se a correspondência biunívoca entre os pontos de um plano e o conjunto de pares ordenados de números reais.

Primeiramente, considere o conjunto $\mathbb{R}$ dos números reais e sua representação por meio dos pontos de uma reta. Para isso, tome um ponto qualquer da reta e a ele associe o número zero, denominado de origem. Em seguida, escolha uma unidade de comprimento e marque cada ponto P da reta com um número real r, dado pela medida do segmento OP em função da unidade de comprimento escolhida. Os pontos que ficam do lado direito da origem são marcados com números reais positivos e os do outro lado são marcados com números negativos. O número r que marca o ponto P é chamado coordenada de P ou abscissa de P. Uma reta onde cada ponto está associado à sua abscissa é chamada de reta numérica ou eixo real. Quando trabalha-se com um eixo isolado, é costume desenhá-lo na horizontal.

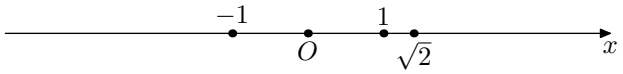


Figura 1.1: Reta numérica

Do mesmo modo que os pontos de uma reta podem ser caracterizados por números, também pode ser feita a associação de coordenadas aos pontos de um plano. É cos-

[1] O professor Troy Henderson criou uma ferramenta online [9] que possibilita ao usuário pré-visualizar uma imagem interpretada pelo METAPOST a partir de comandos digitados em uma caixa de texto.

tume desenhar um eixo na horizontal, representando uma reta numérica, e outro eixo verticalmente, orientado de baixo para cima, representando outra reta numérica.

Dado um ponto P do plano, traça-se por P, paralelas aos dois eixos, obtendo os pontos P_1 e P_2, de coordenadas x e y, respectivamente. Reciprocamente, sendo dado um par de coordenadas (x, y), determina-se P_1 e P_2 e traçando por estes pontos, paralelas aos eixos, encontra-se o ponto P em sua interseção. Existe, assim, uma correspondência biunívoca entre pares ordenados de números reais e pontos do plano.

Observe que o par (x, y) é distinto do par (y, x) a não ser que $x = y$, por isso que (x, y) é dito um par ordenado. Se um ponto P corresponde ao par ordenado (x, y), o número x, que ocorre em primeiro lugar, é chamado de abscissa de P, enquanto y, que aparece em segundo lugar, é a ordenada do ponto P. Um tal sistema de eixos é chamado sistema cartesiano ou sistemas de eixos ortogonais. O eixo das abscissas é representado por O_x e o das ordenadas por O_y.

Definição 1 *O conjunto formado por todos os pares ordenados de números reais é indicado por* $\mathbb{R} \times \mathbb{R}$ *ou* $\mathbb{R}^2$*. Assim,*

$$\mathbb{R}^2 = \{(x, y) \mid x, y \text{ são números reais}\}.$$

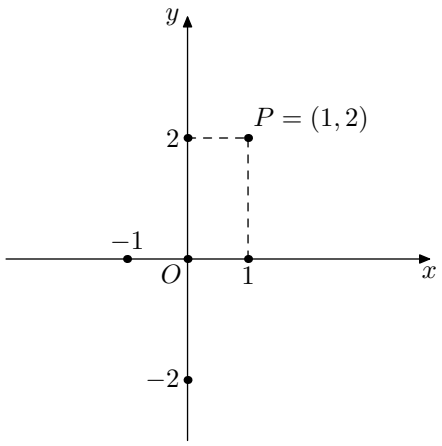


Figura 1.2: Sistema cartesiano

Para fazer um desenho, o METAPOST utiliza comandos do tipo `draw`. Por exemplo, o comando

```
drawdot (20,0)
```

desenha um ponto de coordenadas $(20, 0)$. Se não é indicada nenhuma unidade de medida

específica, o METAPOST interpreta como sendo um `bp`, sigla de "big points", que equivalem a unidade padrão de pontos PostScripts, cuja medida é igual a $\frac{1}{72}$ de uma polegada. Essa nomenclatura é utilizada para distinguir dos pontos de impressão, cuja sigla é `pt`, cuja medida é $\frac{1}{72{,}27}$ de uma polegada.

Além dessas unidades de medida, o METAPOST conta com outras mais tradicionais e algumas nem tão familiares no Brasil. Dentre as utilizadas no Brasil, as mais comuns são `cm` para centímetros, `in` para polegadas e `mm` para milímetros. Porém, é possível usar também outras unidades como `pc` para paicas (que equivalem a $\frac{1}{6}$ de uma polegada), `cc` para cíceros e `dd` para pontos Didot.

Unidade de Medida	Conversão
`pt`	0,035145 cm
`bp`	0,035278 cm
`in`	2,54 cm
`pc`	0,423333 cm
`mm`	0,1 cm
`cc`	0,451167 cm
`dd`	0,0376 cm

Tabela 1.1: Tabela de conversão das unidades de medida utilizadas no METAPOST

Para desenhar segmentos de reta, utiliza-se o comando `draw` seguido das coordenadas das extremidades do segmento unidas por `--`. Vários segmentos podem ser escritos na mesma linha de comando. Na Figura 1.3, por exemplo, três segmentos foram escritos no mesmo comando. Esses segmentos formaram um triângulo retângulo cujo a medida dos dois catetos é 3 cm.

```
draw (0,0)--(3cm,0)--(0,3cm)--(0,0);
```

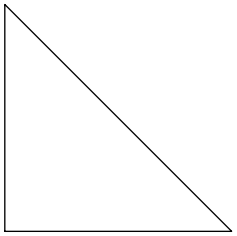


Figura 1.3: Triângulo Retângulo Isósceles

É cômodo manipular um fator de escala de sua preferência indicado por `u`. Assim, é possível definir as coordenadas em termos de `u` e, depois, fixar a medida desse fator. Para

gerar um fator `u` com `1 cm`, por exemplo, utiliza-se no início o comando

```
u=1cm;
```

Também, é possível utilizar a palavra `cycle` ao final de uma sequência de pontos para fechar a figura a ser formada, voltando para o ponto de início. O código utilizado para a Figura 1.3, poderia ser re-escrito como

```
u=1cm; draw (0,0)--(3u,0)--(0,3u)--cycle;
```

gerando a mesma figura.

Um detalhe importante é que, apesar de visualmente `(0,0)--(3u,0)--(0,3u)--(0,0);` ser um caminho fechado, o METAPOST não o interpreta assim. Estar com a extensão `cycle` no final é requisito para ser reconhecido como fechado. Alguns comandos discutidos adiante são aplicáveis apenas a figuras que o METAPOST considera como fechadas.

O METAPOST faz também linhas curvas. O comando utilizado é semelhante ao de linhas retas, com a diferença de usar `..` no lugar de `--`. Em uma mesma linha de comando, pode haver linhas curvas e retilínias. Por exemplo, considere os seis pontos marcados na Figura 1.4.

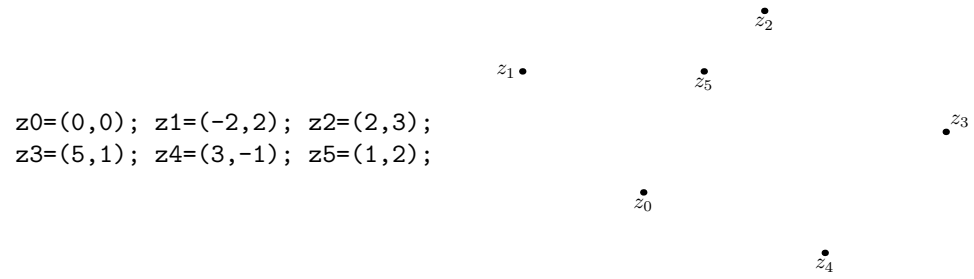
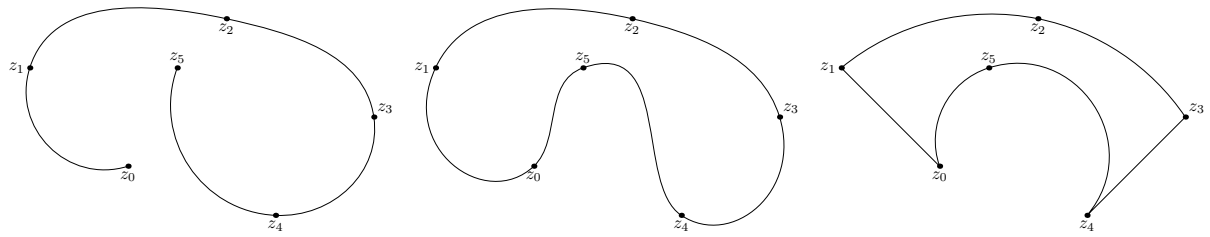


Figura 1.4: Seis pontos

Existem várias figuras que podem ser criadas a partir desses pontos. A Figura 1.5 mostra três dessas possibilidades. A Figura 1.5(a) mostra uma curva aberta que inicia em z_0 e termina em z_5. A Figura 1.5(b) mostra uma curva fechada, por isso é usado no final a palavra `cycle`. A Figura 1.5(c) mostra como é possível mesclar linhas retas e curvas num mesmo comando `draw`.



(a) `draw z0..z1..z2..z3..z4..z5` (b) `draw z0..z1..z2..z3..z4..z5..cycle` (c) `draw z0--z1..z2..z3--z4..z5..cycle`

Figura 1.5: Algumas possibilidades de curvas

1.1 Estilos gráficos

Num trabalho do METAPOST você talvez queira mudar a largura da linha, dos pontos, ou até mesmo a fonte a ser utilizada. Diversos comandos dão essas ferramentas. Um comando de muita utilidade nesse aspecto é `pencircle`. Este comando refere-se a forma da "caneta" utilizada para traçar as linhas das figuras. Por exemplo, escrever a linha de comando

```
pickup pencircle scaled 1.5mm
```

indica que a caneta terá ponta circular e a largura com que as linhas do desenho serão desenhadas dali para frente será `1.5mm`. É como se, literalmente, você tomasse uma caneta esferográfica com essa largura no bico e a utilizasse para fazer o desenho. Esse comando pode ser utilizado para marcar no desenho pontos de tamanhos diferentes. A caneta padrão utilizada possui $\frac{1}{2}$ pt de largura. Assim, na Figura 1.6 a seguir, foi utilizado esse tamanho padrão para os lados do triângulo e um tamanho diferente para cada vértice do mesmo.

```
u:=3cm;
draw (0,0)--(u,0)--(0,u)--(0,0);
pickup pencircle scaled 1mm;
drawdot (0,0);
pickup pencircle scaled 1.5mm;
drawdot (u,0);
pickup pencircle scaled 2mm;
drawdot (0,u);
```

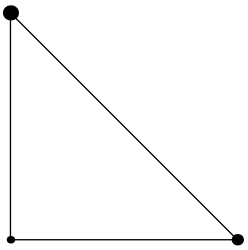


Figura 1.6: Triângulo Retângulo Isósceles com vértices marcados em escalas diferentes

Além de mudar a largura das linhas, é possível tracejá-las com comandos `dashed`. A linha de comando fica da forma

`draw` ⟨caminho⟩ `dashed` ⟨tipo de tracejado⟩;

onde ⟨caminho⟩ refere-se à reta ou curva a ser desenhada e ⟨tipo de tracejado⟩ refere-se a um padrão predefinido de traço. Os dois padrões comuns são `evenly` que faz traços e `withdots` que faz pontos. Veja como ficaria o triângulo da Figura 1.6 com os lados tracejados na Figura 1.7. Já a Figura 1.8 mostra o mesmo triângulo, porém com as linhas pontilhadas.

```
u:=3cm;
draw (0,0)--(u,0) dashed evenly scaled 4;
draw (u,0)--(0,u) dashed evenly scaled 2;
draw (0,u)--(0,0) dashed evenly;
```

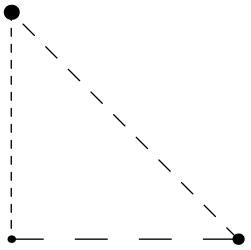


Figura 1.7: Triângulo Retângulo Isósceles com lados tracejados

```
u:=3cm;
draw (0,0)--(u,0) dashed withdots scaled 0.2;
draw (u,0)--(0,u) dashed withdots scaled 0.6;
draw (0,u)--(0,0) dashed withdots;
```

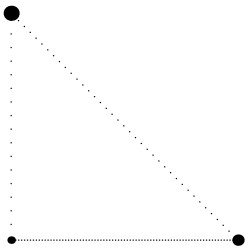


Figura 1.8: Triângulo Retângulo Isósceles com lados pontilhados

1.2 Rótulos

Com o METAPOST, é possível integrar texto e figuras. Assim, com alguns comandos, como `label` e `dotlabel`, você pode utilizar letras, símbolos matemáticos e, até mesmo, frases e equações matemáticas em sua imagem. O comando `label` é utilizado para rotular objetos em geral no trabalho. Já o comando `dotlabel` marca o ponto nas coordenadas onde deseja-se rotular algo, além de rotular o mesmo. No exemplo da Figura 1.3, pode-se rotular os vértices e os lados do triângulo retângulo como é mostrado na Figura 1.9.

Os rótulos são postos nas linhas de comando na forma

`label.`⟨sufixo⟩ (⟨rótulo⟩`,`⟨posição⟩) ;

Após o comando `dotlabel` ou `label`, utiliza-se um sufixo que determine a posição onde o rótulo será colocado. Se essa posição não for especificada, o rótulo será centralizado.

```
u:=3cm;
draw (0,0)--(u,0)--(0,u)--(0,0);
dotlabel.llft("A", (0,0));
dotlabel.lrt("B", (u,0));
dotlabel.ulft("C", (0,u));
label.bot(btex $c$ etex, (u/2,0));
label.urt(btex $a$ etex, (u/2,u/2));
label.lft(btex $b$ etex, (0,u/2));
```

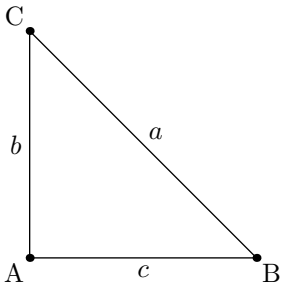


Figura 1.9: Triângulo Retângulo Isósceles ABC

Os sufixos utilizados para determinar a posição do rótulo são oito: `lft, rt, top, bot, ulft, urt, llft` e `lrt`. Quando se usa o sufixo `bot`, por exemplo, o rótulo do objeto em questão é posto abaixo da posição escrita. A Figura 1.10 mostra uma ilustração para o uso de cada tipo de sufixo em relação a um ponto.

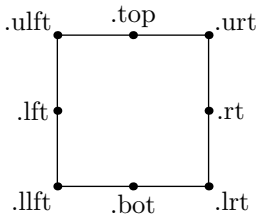


Figura 1.10: Sufixos do comando `label`

O rótulo pode ser escrito com a letra padrão se posto entre aspas. No caso da Figura 1.9, por exemplo, os vértices foram rotulados com letras maiúsculas A, B e C. Isso justifica o uso das aspas. No entanto, os lados do triângulo foram escritos utilizando-se os comandos `btex etex`. Esses comandos fazem com que o interpretador METAPOST escreva o que estiver entre eles utilizando o ambiente matemático do TEX.

A posição também pode ser escrita de diversas maneiras. No caso da Figura 1.9, foi utilizado o ponto médio de cada segmento. Dessa forma, por exemplo, o rótulo c teve como posição definida o ponto médio de AB.

1.3 Setas

O uso de setas é essencial para o esboço de certos desenhos, gráficos e vetores. O METAPOST possui dois comandos que podem ser utilizados para esse fim: `drawarrow` e `drawdblarrow`. Ambos podem ser utilizados com caminhos retilíneos ou curvos. Por exemplo, o comando

```
drawarrow origin--(1,1);
```

desenha uma seta no segmento de início na origem e fim no ponto $(1, 1)$. A diferença entre os dois comandos citados é que o primeiro coloca a seta apenas na extremidade final do segmento, enquanto o segundo põe nas duas extremidades.

Com as setas, é possível desenhar os eixos do plano cartesiano, como ilustrado na Figura 1.11. Os códigos utilizados para rotular os eixos e para fazer a malha pontilhada já podem ser entendidos com o que foi discutido até então. O código

```
u:=1cm; drawarrow ((-4u,0)--(4u,0)); drawarrow ((0,-4u)--(0,4u));
```

foi utilizado para criar os eixos.

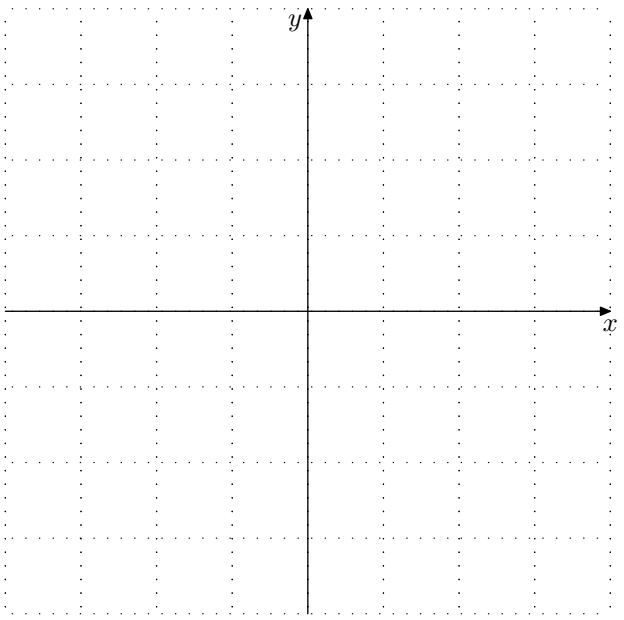


Figura 1.11: Plano cartesiano em malha quadriculada de 1 cm

1.4 Tipos de dados

O METAPOST trabalha com dez tipos de dados: `numeric`, `pair`, `path`, `transform`, `rgbcolor`, `cmykcolor`, `string`, `boolean`, `picture` e `pen`. Esses tipos de dados possibilitam que o usuário armazene partes do desenho ou do código para uso posterior. Nessa seção serão explorados alguns desses tipos, bem como a forma em que eles podem ser utilizados em um desenho gerado pelo METAPOST.

Para representar uma variável, pode-se utilizar qualquer combinação de caracteres simples maiúsculos ou minúsculos. Para declarar de qual tipo será uma variável a ser

utilizada no trabalho, basta digitar o tipo, seguido da variável a se utilizar e um sinal de "`;`". Utilizando-se de colchetes, o METAPOST cria uma família de variáveis. Sendo assim,

```
numeric u; path p[];
```

indica que `u` é uma variável do tipo `numeric` e que qualquer variável da forma `p[i]`, onde `i` é um número real, é do tipo `path`.

O tipo `numeric` é utilizado para se armazenar variáveis numéricas. Pode-se declarar que `raio` será uma variável numérica simplesmente por escrever uma linha de comando

```
numeric raio;
```

no começo do trabalho. Após isso, é possível definir o valor dessa variável. Para fazer isso, é necessário utilizar "`:=`" e não apenas o sinal de "`=`". Por exemplo,

```
raio:= 1cm;
```

define que a variável `raio` equivalerá a 1 cm. Quando se usa apenas o sinal de `=`, o METAPOST não interpreta como uma declaração de variável, mas como uma equação. Assim, se a variável for ter o mesmo valor no código inteiro, apenas `=` é suficiente. Mas, se o valor for alterado no meio do código, é necessário declará-lo com `:=` para que não haja erro de compilação.

Pode-se declarar mais de uma variável em uma única linha de comando. Para isso, basta separar por vírgula cada variável no mesmo comando e finalizar com o ponto e vírgula. Na Figura 1.12, por exemplo, foram definidas duas variáveis numéricas, uma `raio` e outra `diametro`, para desenhar uma circunferência. Essa circunferência foi feita por meio do comando `fullcircle`, que se refere a um caminho circular fechado de diâmetro unitário centrado na origem. Para que a circunferência tenha 1 cm de raio, é necessário redimensioná-la pelo tamanho de seu diâmetro com a transformação já citada `scaled`.

```
numeric raio, diametro;
raio:=1cm;
diametro:=2*raio;
draw fullcircle scaled diametro;
```

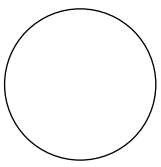


Figura 1.12: Circunferência com raio de 1 cm

O tipo `pair` define um par de duas variáveis numéricas. Como visto, essas variáveis representam um ponto do $\mathbb{R}^2$. É possível fazer operações com esse tipo de variável, como adição e subtração entre pares. É possível também multiplicar ou dividir um par por uma variável numérica. Essas operações serão discutidas mais a frente. Cada par possui uma abscissa x e uma ordenada y. O METAPOST usa a estrutura padrão para definir os pontos no plano, sendo essa (x, y). Isso significa que se é definido um par `P:=(1,2)`, por exemplo, o METAPOST gera automaticamente duas variáveis numéricas que podem ser escritas como `xpart P` e `ypart P`, valendo, respectivamente, 1 e 2. Não é preciso declarar no começo do código que as variáveis da forma `z`⟨índice⟩ são do tipo `pair`. Essa informação é intrínseca ao METAPOST. No entanto, para outras variáveis, como, por exemplo, `A1`, `A2`, `A3`, etc., é necessário fazer a declaração de variáveis como `pair`. Para facilitar o trabalho, o comando

```
pair A[];
```

define que qualquer variável do tipo `A[`⟨índice⟩`]` ou `A`⟨índice⟩ será um par.

O tipo `path` é utilizado para armazenar caminhos entre pontos agrupados com os comandos já citados "`--`" e "`..`", ou caminhos predefinidos como o `fullcircle`, já citado. Os pontos agrupados com "`--`" formam segmentos de reta. Já os pontos agrupados com "`..`" formam curvas "suaves", interpolando esses pontos par a par. É possível com o METAPOST alterar a curva entre uma lista de pontos. Isso é feito por se determinar uma direção angular específica para o caminho em um determinado ponto, utilizando o comando `dir` α, onde α representa o ângulo em graus. A Figura 1.13 mostra três curvas que sofrem modificações no ponto de partida A. O código referente às cores dessas linhas e também os tipos referentes a cores são o assunto da próxima seção.

```
pair A, B, C; path p, q, r;
A:=origin; B:=(1cm, 4cm); C:=(0,2cm);
p:=A{dir 45}..B..C;
draw p withcolor blue;
q:=A{dir 0}..B..C;
draw q withcolor red;
r:=A{dir 130}..B..C;
draw r withcolor green;
```

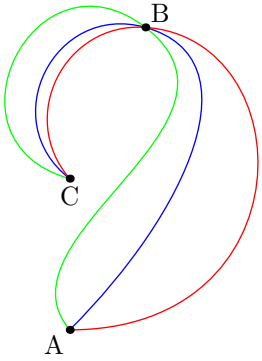


Figura 1.13: Exemplos de curvas modificadas pela direção no ponto inicial

O tipo `picture` é utilizado para armazenar figuras. Para isso, é utilizada uma variável

interna pré-definida `currentpicture`. Essa variável armazena a figura atual no modo que já está desenhada com os comandos `fill`, `draw`, `filldraw` e seus respectivos comandos de desfazer a ação `unfill`, `undraw` e `unfilldraw`. Esse comando não é muito utilizado em desenhos básicos, mas é indispensável para os mais complexos.

O tipo `transform` será tratado no Capítulo 3. Os tipos `pen`, `boolean` e `string` não agregam informações que serão práticas para essa dissertação, por isso não serão discutidos aqui.

1.5 Cores e preenchimento de áreas

Outro comando simples e prático de se usar no METAPOST é o `fill`. Este é utilizado para preencher caminhos e curvas fechadas, aqueles com `cycle` no final. É possível preencher com cor vermelha a Figura 1.5(b), por exemplo, adicionando aos seus comandos a linha

```
fill (z0..z1..z2..z3..z4..z5..cycle) withcolor red;,
```

o resultado é mostrado na Figura 1.14. Note que o METAPOST trabalha seguindo a ordem das linhas de comando. Isso significa que se o comando `fill` for posto depois dos comandos `dotlabel` e `draw`, por exemplo, o preenchimento ficará sobreposto ao contorno da curva e aos rótulos dos pontos.

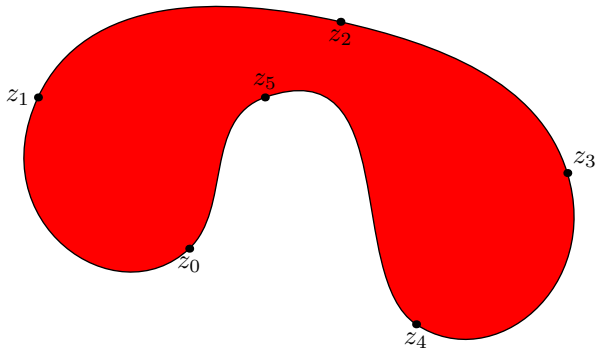


Figura 1.14: Curva fechada preenchida com vermelho

Com o exemplo da Figura 1.14, entende-se que `withcolor` seleciona a cor a ser utilizada pelo comando `fill`. O que é digitado após esse comando pode ser o nome da cor pré-definida pelo METAPOST, em inglês. Essas cores pré-definidas são: `red`, para **vermelho**; `blue`, para **azul**; `green`, para **verde**; `cyan`, para **ciano**; `magenta`, para **magenta**; `yellow`, para **amarelo**; `black`, para **preto**; e `white`, para branco.

Para utilizar outras cores diferentes das pré-definidas, é possível utilizar-se de uma variável do tipo `rgbcolor` (que pode também ser escrito como `color`) ou do tipo `cmykcolor`. Esses tipos funcionam como o tipo `pair`, com a diferença de que em vez de usar duas variáveis numéricas, usam três ou quatro. Esses valores variam de 0 a 1. Além disso, operações podem ser feitas como adição, subtração e multiplicação por quantidades numéricas.

O tipo `color` utiliza três variáveis. A primeira componente se refere à quantidade de vermelho; a segunda, de verde; e a terceira, de azul. Assim, a cor preta é `(0, 0, 0)` e branco é `(1, 1, 1)`. Como citado, algumas cores já são pré-definidas. Pode-se definir uma cor a partir de outra utilizando comandos operatórios. Por exemplo, algumas escalas de cinza podem ser definidas a partir da cor pré-definida `white` apenas multiplicando-a por um fator entre 0 e 1. O quadro da Figura 1.15 é apresentado em [7] com onze tons de cinza, sendo estes os resultados de `i*0.1*white`, para $i = 0, 1, .., 10$.

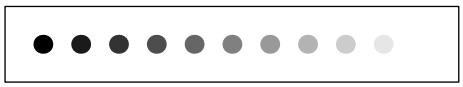
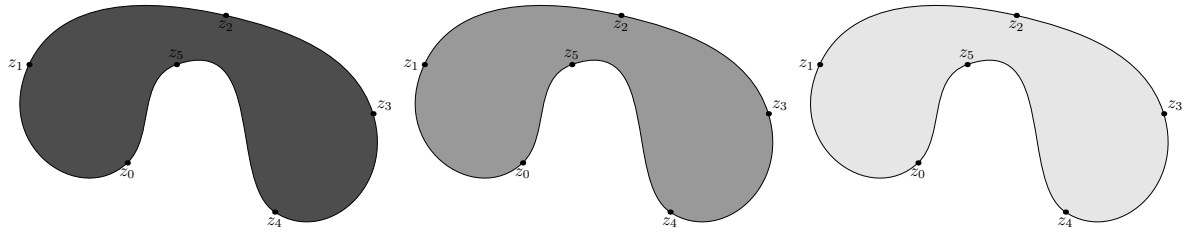


Figura 1.15: Variação de tons de cinza

Como `white` é a forma escrita da cor `(1, 1, 1)`, escrever `0.3white` é uma forma abreviada de `(0.3, 0.3, 0.3)` e resulta num tom escuro de cinza. Veja como fica a curva da Figura 1.14 preenchida com algumas variações de cinza na Figura 1.16.



(a) `withcolor 0.3white` (b) `withcolor 0.6white` (c) `withcolor 0.9white`

Figura 1.16: Formas preenchidas com variações de cinza

O tipo `cmykcolor` utiliza o sistema CMYK de cores. Diferente do sistema RGB, que é aditivo, CMYK é um sistema subtrativo de cores. São utilizadas quatro valores numéricos ordenados entre 0 e 1. [2]

[2] O objetivo deste trabalho não contempla a explanação dos sistemas de cores CMYK e RGB. Para informações mais detalhadas, consulte [21].

Para utilizar cores que não são pré-definidas, é preciso saber suas coordenadas RGB ou CMYK. A cor roxa, por exemplo, no sistema RGB é definida pela tripla (128, 0, 128). Para fazer a conversão para as coordenadas que o METAPOST interpreta, divide-se cada coordenada por 256. Assim, a `color (0.5, 0, 0.5)` representa a cor roxa. Para gravar uma variável dessa cor e utilizá-la numa figura, um exemplo de sintaxe é

```
rgbcolor roxo; roxo := (0.5, 0, 0.5);.
```

Com essa linha de comando, é possível recolorir a curva da Figura 1.14 como mostrado na Figura 1.17.

```
rgbcolor roxo;
roxo := (0.5, 0, 0.5);
u:=1cm;
z0=(0,0); z1=(-2u,2u); z2=(2u,3u);
z3=(5u,1u); z4=(3u,-u); z5=(u,2u);
path p;
p:= z0..z1..z2..z3..z4..z5..cycle;
fill p withcolor roxo;
draw p;
```

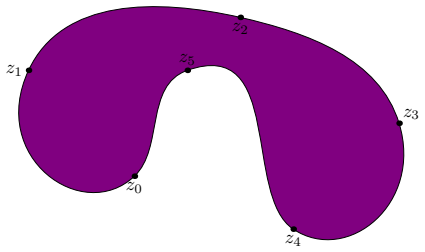


Figura 1.17: Preenchimento com cor roxa não predefinida

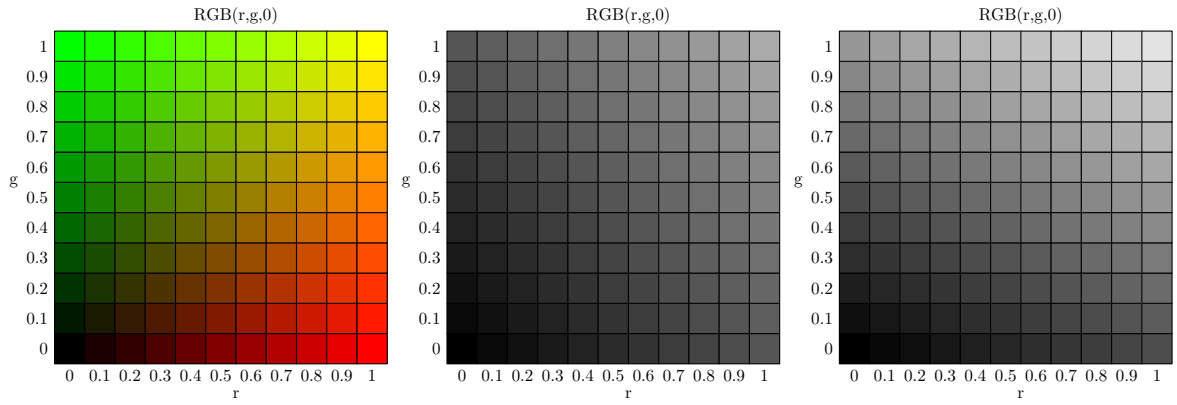
Heck [7] criou o quadro ilustrativo da Figura 1.18(a) com a variação que ocorre entre o verde e vermelho em tons sem azul. Ele também propôs um exercício com o objetivo de que, com o METAPOST, fosse criado um quadro como esse que mostra a conversão real dessas cores em preto e branco, definida pela função

$$(r, g, b) \mapsto \frac{r + g + b}{3} \cdot (1, 1, 1)$$

e, depois, outro com as cores utilizadas nas televisões em preto e branco, que é dada pela fórmula de conversão

$$(r, g, b) \mapsto (0,30r + 0,59g + 0,11g) \cdot (1, 1, 1)\,.$$

E, além disso, que ao final se fizesse uma comparação entre os dois. Com uma simples mudanças no código utilizado para o quadro original, é possível analisar os dois quadros do exercício, dispostos nas Figuras 1.18(b) e 1.18(c). O resultado aponta que a conversão utilizada na televisão é um pouco mais clara que o natural.



(a) Variações entre vermelho e verde

(b) Conversão real

(c) Conversão feita pelas televisões

Figura 1.18: Conversão de cores em preto e branco

1.6 Operadores matemáticos

O METAPOST possui intrínseco a sua linguagem de programação alguns operadores matemáticos que são práticos para criação de gráficos em geral. Os mais básicos são `+`, `-`, `*` e `/`, utilizados para as operações de adição, subtração, multiplicação e divisão, respectivamente. A exponenciação é representada por `**`, isto é, `2**3` representa 2^3.

É importante ressaltar que o METAPOST foi programado para entender as operações numa determinada ordem. Para não ter dúvidas, é recomendável utilizar parênteses. Por exemplo, `5*a**2` significa $(5a)^2$, não $5a^2$. Para não haver esse tipo de confusão, basta digitar `5*(a**2)`.

Existem também operadores para o que Hobby [13] chama de "Adição Pitagórica"e "Diferença Pitagórica".

Definição 2 *Sejam b e c dois números reais positivos. Dizemos que a é a adição pitagórica entre b e c se a é a medida da hipotenusa de um triângulo retângulo de catetos b e c. Assim, pelo Teorema de Pitágoras,*

$$a = \sqrt{b^2 + c^2}$$

Definição 3 *Sejam a e b dois números reais positivos, com $a > b$. Dizemos que c é a diferença pitagórica entre a e b se c é a medida do cateto desconhecido de um triângulo*

retângulo de hipotenusa a e cateto b. Assim, pelo Teorema de Pitágoras,

$$c = \sqrt{a^2 - b^2}$$

A sintaxe para a adição pitagórica e subtração pitagórica no METAPOST é `++` e `+-+`, respectivamente. Assim, `a++b` significa $\sqrt{a^2 + b^2}$ e `a+-+b` significa $\sqrt{a^2 - b^2}$.

Alguns operadores conhecidos e também utilizados por outros softwares são `abs`, `sqrt`, `ceiling`, `floor` e `mod`. O comando `abs a` computa $|a|$, enquanto `abs (x1,y1)` é o mesmo que $\sqrt{x_1^2 + y_1^2}$, isso é, a soma pitagórica de seus dois componentes. Já `sqrt x` é utilizado para representar $\sqrt{x}$. Os comandos `ceiling a` e `floor a` computam, respectivamente, $\lceil a \rceil$ e $\lfloor a \rfloor$, onde $\lceil a \rceil$ é o menor número inteiro maior ou igual a a e $\lfloor a \rfloor$ é o maior número inteiro menor ou igual a a. O comando `a mod b` retorna o resto da divisão euclidiana de a por b.

Cálculos trigonométricos são feitos com os comandos `sind a` e `cosd a`, que computam o seno e cosseno, respectivamente, de um ângulo a em graus. Para usar a variável em radianos, o uso de macros pode facilitar o trabalho. Essas macros serão explanadas no Capítulo 4. Hobby [13] aborda uma lista completa com outros operadores não citados aqui, inclusive operadores lógicos, além da ordem em que a linguagem foi programada para compreendê-los.

1.7 Fluxo de Trabalho do METAPOST

Para produzir uma figura, o usuário precisa criar uma entrada contendo as instruções escritas na linguagem METAPOST em algum editor TEX. A extensão de arquivo mais comum para esse propósito é a ".mp". O interpretador do METAPOST processa esse arquivo e retorna como uma saída PostScript que pode ser incluída num documento TEX ou visto em algum leitor PostScript. Hobby [13] utiliza a Figura 1.19 para ilustrar esse processo.

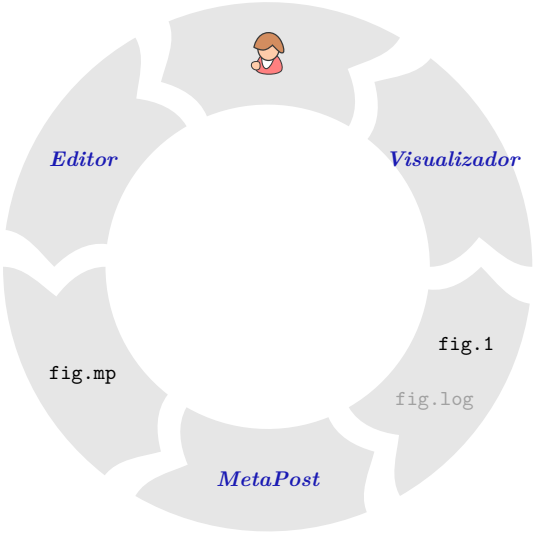


Figura 1.19: Fluxo de trabalho do METAPOST

Para compreender os detalhes do arquivo de entrada, analise o código completo mostrado na Figura 1.20. Observe que cada linha é terminada por um ponto-e-vírgula. Como um arquivo no METAPOST funciona como uma lista de instruções, quando ele lê um `;`, ele entende que o que está escrito é um comando e o faz o que ali é solicitado antes de passar para o próximo na fila. As exceções são os casos em que a instrução tem um fim claro, como é o caso de `end` e `endfig`. Nesses casos, não há problemas em omitir o ponto-e-vírgula. Uma linha do código pode ter vários comandos separados por `;`, ou seja, não é necessário que esteja um em cada linha.

```
outputformat:="mps";
outputtemplate:=("%j-%c.mps");
beginfig(1); % triângulo retângulo
draw (0,0)--(3cm,0)--(0,3cm)--(0,0);
endfig;
end
```

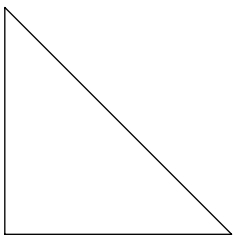


Figura 1.20: Código de entrada para a Figura 1.3

Para facilitar a escrita e a leitura do código é aceitável separar um único comando em diversas linhas. Com o símbolo de porcentagem, `%`, pode-se adicionar comentários às linhas do código. Qualquer coisa escrita depois desse símbolo em uma linha é ignorada durante o processamento que o METAPOST faz a partir da leitura do arquivo.

As variáveis utilizadas antes da chave `beginfig(1)` servem para facilitar o trabalho do usuário ao importar a imagem para um trabalho em TEX. A primeira atribuição apresentada, `outputformat` permite que o METAPOST grave as saídas conforme se queira.

Como o foco aqui é fazer uso dessas figuras em arquivos TEX, convém atribuir o formato ".mps" que é o utilizado pelo TEX para imagens PostScripts.

É digno de nota como foi atribuída a estrutura do nome do arquivo na variável `outputtemplate`. O código `%j` refere-se ao *jobname*, ou seja, ao nome do arquivo original. Já o código `%c` refere-se ao *counter*, isto é, ao contador das figuras dispostas no arquivo. Esse código é útil pois um mesmo documento pode conter diversas figuras. Todas as figuras podem ser criadas em um mesmo arquivo. Cria-se entre as chaves `beginfig(1);... endfig;` o código para a primeira figura, depois entre `beginfig(2);... endfig;` o código para a segunda, e assim sucessivamente. Quando todas as figuras estão digitadas, basta encerrar o arquivo com a chave `end`. Todas as figuras serão exportadas para o diretório de origem do arquivo quando invocado o METAPOST e o código for compilado.

Com os arquivos ".mps" gerados, incluí-los em um documento TEX é fácil. Para importar a figura gerada pelo METAPOST, utiliza-se o comando

$$\texttt{\textbackslash includegraphics}\,\{\langle\text{figura.mps}\rangle\}$$

onde ⟨figura.mps⟩ se refere ao arquivo da imagem. Este arquivo precisa estar no mesmo diretório em que o arquivo ".tex" será salvo.

Capítulo 2

Vetores e Geometria

Alguns conceitos matemáticos de vetores e de geometria podem ser utilizados para fazer desenhos no METAPOST. O METAPOST possui operadores vetoriais e é capaz de entender os dados do tipo pair como vetores. A base matemática para isso, bem como algumas aplicações dessas operações na geometria, serão abordadas neste capítulo.

2.1 Vetores

Considere o conjunto de pares ordenados de números reais

$$\mathbb{R}^2 = \{(x, y) \mid x, y \text{ são números reais}\}.$$

Dados um número real λ e dois pontos quaisquer $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ de $\mathbb{R}^2$, define-se

$P_1 + P_2 = (x_1 + x_2, y_1 + y_2)$ (adição de pontos)

$\lambda \cdot P = (\lambda x_1, \lambda x_2)$ (multiplicação por um número real)

As conhecidas propriedades algébricas dos números reais permitem demonstrar facilmente as seguintes propriedades: para quaisquer que sejam os pontos $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ e $P_3 = (x_3, y_3)$, tem-se que

$P_1 + P_2 = P_2 + P_1$

$(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$

$P_1 + (0,0) = P_1$

$P_1 + (-P_1) = (0,0)$, onde $-P_1 = (-x_1, -y_1)$.

Também, quaisquer que sejam $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ e os números reais r e s, tem-se

$(rs)P_1 = r(sP_1)$

$(r+s)P_1 = rP_1 + sP_2$

$r\,(P_1 + P_2) = rP_1 + sP_2$

$1 \cdot P_1 = P_1$

Assim, o conjunto $\mathbb{R}^2$ com estas operações, uma interna e a outra externa, é um espaço vetorial, seus elementos passam a ser chamados de vetores e os números reais são chamados de escalares.

Graficamente, fixado um sistema de eixos ortogonais, um vetor $P = (x, y)$ é representado pelo segmento OP orientado de O para P, indicado por $\overrightarrow{OP}$ e desenhado como uma flexa dirigida de O para P. Também, será indicado um vetor por uma única letra encimada por uma seta, como $\vec{v}$.

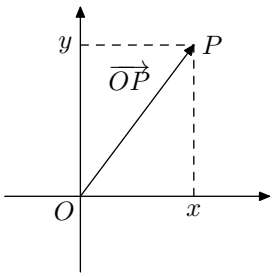


Figura 2.1: Vetor $\overrightarrow{OP}$

Considere dois vetores $A = (x_1, y_1)$ e $B = (x_2, y_2)$, segue que

$$B - A = B + (-A) = (y_2 - y_1, x_2 - x_1).$$

Se $(x_1, y_1) \neq (x_2, y_2)$, o segmento orientado $\overrightarrow{AB}$ é paralelo ao segmento orientado $\overrightarrow{OP}$, com $P = (x_2 - x_1, y_2 - y_1)$. Assim, o vetor $\overrightarrow{OP}$ pode ser representado por qualquer segmento orientado $\overrightarrow{AB}$, paralelo a $\overrightarrow{OP}$, de mesmo comprimento e mesma orientação.

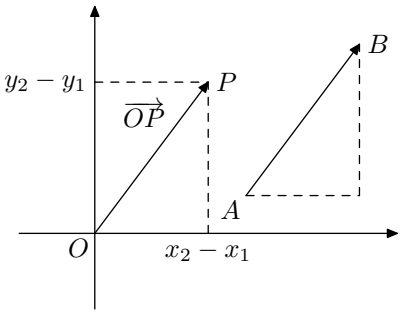


Figura 2.2: Vetor $\overrightarrow{AB}$

Definição 4 *Dois vetores* $\overrightarrow{u} = (x_1, y_1)$ *e* $\overrightarrow{v} = (x_2, y_2)$ *são colineares quando existe* $t \in \mathbb{R}$ *tal que* $\overrightarrow{u} = t\overrightarrow{v}$*.*

O vetor nulo $\overrightarrow{0} = (0, 0)$ é colinear com qualquer outro vetor, pois $\overrightarrow{0} = 0 \cdot \overrightarrow{v}$.

Geometricamente, a soma de dois vetores $\overrightarrow{u}$ e $\overrightarrow{v}$ é calculada pela regra do paralelogramo.

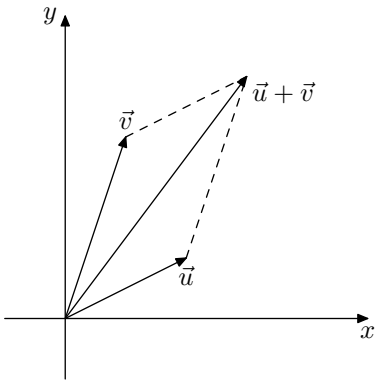


Figura 2.3: Regra do paralelogramo

2.1.1 Produto Interno de Vetores

Definição 5 *Dado um vetor* $\vec{v} = (x, y)$*, define-se o seu módulo, denotado por* $|\vec{v}|$*, como sendo*

$$|\vec{v}| = \sqrt{x^2 + y^2}$$

Assim, dado o vetor $\vec{v} = \overrightarrow{OP}$, o seu módulo é a distância da extremidade P à origem. Observa-se, também, que o módulo de um vetor $\vec{v} = (x, y)$ é o comprimento de qualquer segmento que o represente, isto é, qualquer segmento $\overrightarrow{AB}$ tal que $B - A = (x, y)$.

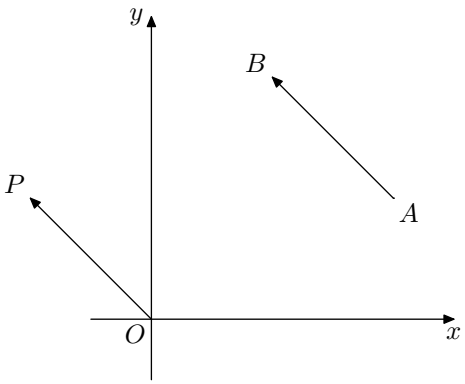


Figura 2.4: Vetores $\overrightarrow{AB}$ e $\overrightarrow{OP}$

Dado um vetor $\vec{v} = (x, y)$, não nulo, considerando o ângulo θ do eixo O_x com a direção de $\vec{v}$, $0 \leq \theta \leq 2\pi$, tem-se que

$$x = |\vec{v}|\cos\,\theta \;\text{ e }\; y = |\vec{v}|\text{sen}\,\theta$$

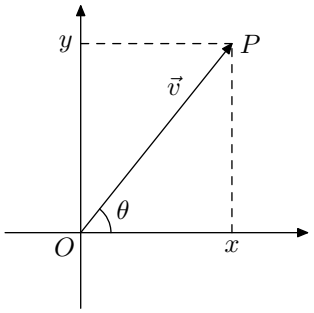


Figura 2.5: Vetor $\overrightarrow{OP}$ e ângulo θ

Definição 6 *O produto interno ou produto escalar de dois vetores* $\vec{v}_1 = (x_1, y_1)$ *e* $\vec{v}_2 = (x_2, y_2)$*, denotado por* $\vec{v}_1 \cdot \vec{v}_2$*, é definido por*

$$\vec{v}_1 \cdot \vec{v}_2 = x_1x_2 + y_1y_2$$

Considerando os ângulos θ_1 e θ_2 de $\vec{v_1}$ e $\vec{v_2}$, respectivamente, com o eixo O_x segue que

$$\vec{v}_1 \cdot \vec{v}_2 = x_1x_2 + y_1y_2 = |\vec{v_1}||\vec{v_2}|\;(\cos\,\theta_1\,\cos\,\theta_2 + \;\text{sen}\,\theta_1\,\text{sen}\,\theta_2)$$

$$\vec{v}_1 \cdot \vec{v}_2 = |\vec{v_1}||\vec{v_2}|\,\cos\,(\theta_1 - \theta_2)$$

Observando que, se θ é o ângulo entre os vetores $\vec{v}_1$ e $\vec{v}_2$, tem-se que

$$\vec{v}_1 \cdot \vec{v}_2 = |\vec{v_1}||\vec{v_2}| \cos\theta$$

e conclui-se que o produto interno $\vec{v}_1 \cdot \vec{v}_2$ só se anula em dois casos: quando um dos vetores é o vetor nulo ou quando $\cos\theta = 0$. No segundo caso, as direções dos vetores são perpendiculares.

Definição 7 *Dois vetores são ortogonais quando o seu produto interno é igual a zero.*

Seguem facilmente as seguintes propriedades do produto interno:

$\vec{v} \cdot \vec{v} = |\vec{v}|^2$

$\vec{u} \cdot (\vec{v} + \vec{w}) = \vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w}$

$(\vec{u} + \vec{v}) \cdot \vec{w} = \vec{u} \cdot \vec{w} + \vec{v} \cdot \vec{w}$

$(t\vec{u}) \cdot \vec{v} = t(\vec{u} \cdot \vec{v}) = \vec{u} \cdot (t\vec{v})$

Antes de prosseguir para o próximo tópico, vale a pena discutir uma notação predefinida no METAPOST para pares, bem como formas de defini-las implicitamente. Para tanto, considere um triângulo ABC, como o da Figura 2.6, cujo código está transcrito ao seu lado.

```
u:=1cm;
z1=origin; z2=(5u,-u); z3=(2u,2u);
draw z1--z2--z3--cycle;
dotlabel.llft(btex $A$ etex, z1);
dotlabel.lrt(btex $B$ etex, z2);
dotlabel.top(btex $C$ etex, z3);
```

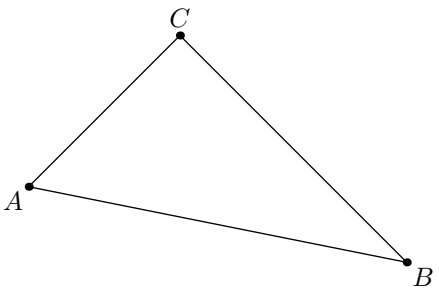


Figura 2.6: Triângulo ABC

É interessante analisar um pouco as variáveis utilizadas para a construção do triângulo da Figura 2.6. Quando foram definidas as coordenadas dos vértices do triângulo com o comando `z1=origin; z2=(5u,-u); z3=(2u,2u)`, poderiam ser utilizadas quaisquer letras ou combinações de letras e algarismos numéricos. A forma `z`⟨índice⟩ é muito conveniente porque é uma abreviação predefinida no METAPOST para

`(x`⟨índice⟩`, y`⟨índice⟩`)`.

Isso possibilita dar valores para variáveis `z` por meio de equações lineares que envolvam suas coordenadas. Por exemplo, os pontos `z1`, `z2`, `z3`, e `z4` de um quadrado de 2 cm de lado, podem ser definidos pelas seguintes equações:

```
z1=-z3=(1cm,1cm);
x4=-y4=-x2=y2=1cm;
```

Exatamente os mesmos pontos podem ser obtidos definindo seus valores diretamente:

```
z1=(1cm,1cm); z2=(-1cm,1cm); z3=(-1cm,-1cm); z4=(1cm,-1cm);
```

Depois de ler as equações, o MetaPost conhece os valores de `z1`, `z2`, `z3`, e `z4`. Com esses valores, utilizando um comando `draw`, você é capaz de traçar um quadrado de 2 cm de lado. Essa representação é possível pois uma característica importante que provém do METAFONT é a capacidade de resolver equações lineares para que os programas também possam vir a ser escritos de uma forma não estritamente explícita. Por exemplo, o METAPOST pode ler

```
a+b=3; 2a=b+3;
```

e deduzir que $a = 2$ e $b = 1$. Essas mesmas equações também poderiam ser escritas de maneira um pouco mais compacta, sendo unidas com vários sinais de igual:

```
a+b = 2a-b = 3;
```

Com esse assunto da interpretação do METAPOST de equações lineares explanado, é possível entender bem como uma altura do triângulo ABC pode ser traçada.

2.1.2 Uma aplicação do produto interno no METAPOST

Como visto, se dois vetores são perpendiculares o produto interno entre eles é nulo. Na Geometria, isso pode ser aplicado quando se quer traçar a altura de um triângulo qualquer. Para desenhar a altura relativamente ao vértice C, considere o ponto H_C, pé da perpendicular do vértice C ao lado AB. O ponto H_C é caracterizado por pertencer à reta determinada pelos pontos A e B e por $\overrightarrow{CH_C}$ ser perpendicular a $\overrightarrow{AB}$. Vetorialmente,

significa que os vetores $\overrightarrow{AB}$ e $\overrightarrow{AH_C}$ são colineares e o produto interno entre os vetores $B - A$ e $C - H_C$ é igual a zero.

O METAPOST possui um comando operador que calcula o produto interno entre dois vetores, que é o `dotprod`. Assim, `z1 dotprod z2` é equivalente a $z_1 \cdot z_2$, isto é, como `z1=(x1,y1)` e `z2=(x2,y2)`, então `z1 dotprod z2` é equivalente a `x1*x2 + y1*y2`.

Considerando que A, B e C foram definidos, respectivamente, como `z1`, `z2` e `z3`, ao utilizar-se o comando

```
(z2-z1) dotprod (z4-z3)=0;,
```

que representa o produto interno entre $\overrightarrow{AB}$ e $\overrightarrow{CH_C}$, onde `z4` refere-se ao ponto H_C, o METAPOST ainda não é capaz de definir exatamente suas duas coordenadas. De fato, existe uma infinidade de pontos H tal que $\overrightarrow{CH}$ seja perpendicular a $\overrightarrow{AB}$.

Para o METAPOST determinar precisamente as coordenadas do ponto H_C procurado é preciso que, com outro comando, seja definido que os vetores $\overrightarrow{AB}$ e $\overrightarrow{AH_C}$ são colineares. Como as coordenadas dos pontos H tais que $\overrightarrow{AH}$ é um múltiplo de $\overrightarrow{AB}$ formam outra equação linear, o METAPOST conseguirá definir exatamente a localização do ponto H_C. Para essa informação, o comando utilizado é `z4=whatever[z1,z2]`. Matematicamente, o METAPOST entende esse comando como sendo `z4=z1+t(z2-z1)`, isto é,

$$H_C = A + t(B - A) \iff H_C - A = t(B - A)$$

ou seja, os vetores $\overrightarrow{AB}$ e $\overrightarrow{AH_C}$ são colineares. O comando `whatever` serve para representar alguma variável qualquer cuja qual não se é preciso saber o valor exato. Se for utilizado um número real em vez de `whatever`, o METAPOST efetua este cálculo e gera um par. Por exemplo, `0.5[A,B]` gera o ponto médio entre `A` e `B`, pois equivale a $A+\frac{1}{2}(B-A) = \frac{A+B}{2}$.

Com o ponto H_C já definido, basta um comando `dotlabel` para marcar e rotulá-lo e um `draw` para traçar a altura CH_C. A Figura 2.7 mostra o código, juntamente com o resultado gráfico.

```
u:=1cm;
z1=origin; z2=(5u,-u); z3=(2u,2u);
draw z1--z2--z3--cycle;
dotlabel.llft(btex $A$ etex, z1);
dotlabel.lrt(btex $B$ etex, z2);
dotlabel.top(btex $C$ etex, z3);
(z2-z1) dotprod (z4-z3)=0;
z4=whatever[z1,z2];
draw z3--z4 dashed evenly;
dotlabel.llft(btex $H_C$ etex, z4);
```

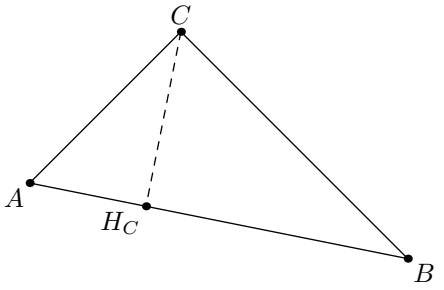


Figura 2.7: Triângulo ABC com altura relativa ao lado AB traçada

Note que se o triângulo em questão for obtusângulo, a altura será traçada fora do triângulo. No caso do triângulo ABC trabalhado até então, se $C = (-2, 2)$, então o triângulo é obtusângulo. É possível, nesse caso, desenhar tracejadamente o segmento CH_C e o prolongamento do lado AB, unindo os pontos A e H_C, como exemplificado na Figura 2.8.

```
u:=1cm; z1=origin;
z2=(5u,-u); z3=(-2u,2u);
draw z1--z2--z3--cycle;
dotlabel.bot(btex $A$ etex, z1);
dotlabel.lrt(btex $B$ etex, z2);
dotlabel.top(btex $C$ etex, z3);
(z2-z1) dotprod (z4-z3)=0;
z4=whatever[z1,z2];
draw z3--z4 dashed evenly;
draw z1--z4 dashed evenly;
dotlabel.llft(btex $H_C$ etex, z4);
```

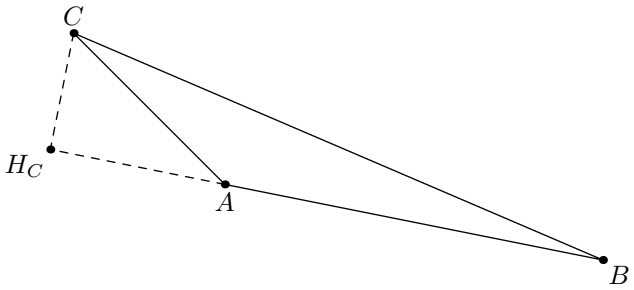


Figura 2.8: Triângulo ABC obtusângulo com altura relativa ao lado AB traçada

2.2 Pontos Notáveis do Triângulo

Nessa seção serão apresentadas algumas definições e proposições utilizadas para os pontos notáveis do triângulo. Todas elas têm como referência Muniz Neto [18].

Definição 8 *Uma ceviana de um triângulo é qualquer segmento (ou a reta ou a semirreta correspondente) que une um vértice do triângulo a um ponto sobre a reta suporte do lado oposto a tal vértice.*

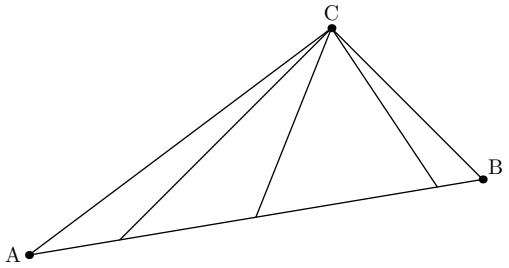


Figura 2.9: Triângulo ABC e três cevianas

Traçar uma ceviana com o METAPOST é simples. Na Figura 2.9, por exemplo, os pontos A, B e C foram definidos, respectivamente, como `A`, `B` e `C`. Depois, com comandos `draw` foram traçados os segmentos `C--0.2[A,B]`, `C--0.5[A,B]` e `C--0.9[A,B]`.

Definição 9 *Uma mediana de um triângulo é um segmento que une um vértice do mesmo ao ponto médio do lado oposto a esse vértice.*

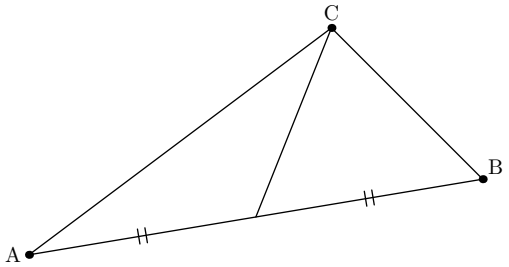


Figura 2.10: Triângulo ABC e uma mediana

Uma mediana é uma ceviana particular de um triângulo. Logo, o código utilizado para fazer esse segmento no METAPOST é o mesmo citado anteriormente. No entanto, é preciso especificar que o ponto definido por `C` é unido ao ponto médio de AB. A sintaxe utilizada para esse segmento é `C--0.5[A,B]`. Todo triângulo possui três medianas.

Definição 10 *Em um triângulo ABC, a altura relativa ao lado AB (ou ao vértice C) é o segmento que une o vértice C ao pé da perpendicular baixada de C à reta* $\overleftrightarrow{AB}$.

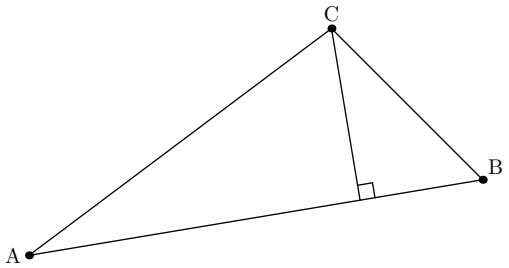


Figura 2.11: Triângulo ABC e a altura relativa ao vértice C

Os comandos utilizados para traçar uma altura num triângulo qualquer foram discutidos na Subseção 2.1.2. Obviamente, todo triângulo possui três alturas relativas a cada lado (ou a cada vértice).

Definição 11 *Dado um ângulo $\angle ACB$, a bissetriz de $\angle ACB$ é a semirreta $\overrightarrow{CD}$ que o divide em dois ângulos iguais. Neste caso, diz-se ainda que $\overrightarrow{CD}$ bissecta $\angle ACB$. Assim,*

$$\overrightarrow{CD} \text{ bissecta } \angle ACB \iff A\widehat{C}D = B\widehat{C}D.$$

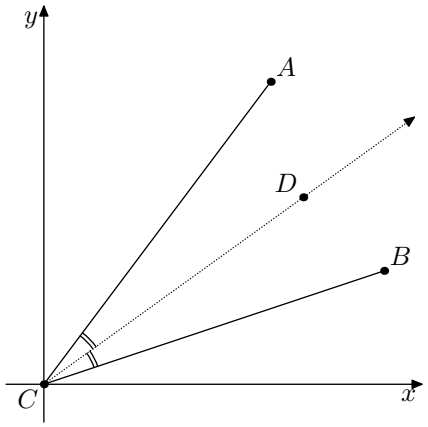


Figura 2.12: Bissetriz $\overrightarrow{CD}$ de $\angle ACB$

Para entender como traçar uma bissetriz com o METAPOST, é necessário utilizar-se dos comandos `dir` e `angle`. Utilizar o operador `dir` é uma maneira simples de definir um ponto no círculo unitário em um dado ângulo com o eixo horizontal. Por exemplo, `dir(30)` gera o par $\left(\frac{\sqrt{3}}{2}, \frac{1}{2}\right)$. Matematicamente, o METAPOST traduz `dir(a)` como sendo o par `(cosd(a), sind(a))`. Já o operador `angle` se refere ao operador inverso de `dir`. Este comando toma um par, interpreta-o como um vetor e calcula o arco-tangente de dois argumentos, isto é, dá o ângulo, em graus, correspondente ao formado pelo vetor em relação ao eixo horizontal.

Sendo assim, dado um ângulo $\angle ACB$, considere um sistema de eixos com origem em C, como na Figura 2.12. Sejam α e β os ângulos dos vetores $\overrightarrow{CB}$ e $\overrightarrow{CA}$ em relação ao eixo horizontal, respectivamente. Rotacionando β graus o ponto B em torno de C no sentido anti-horário, obtém-se o ponto B', tal que o ângulo do vetor $\overrightarrow{CB'}$ em relação ao eixo das abscissas é $\alpha + \beta$.

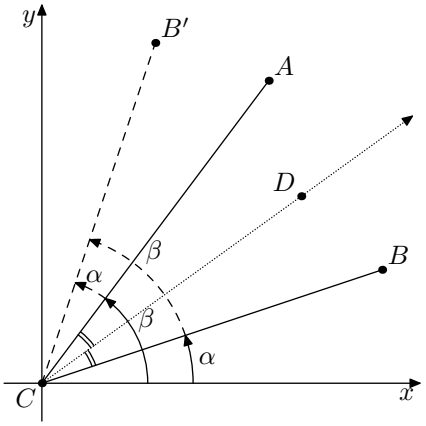


Figura 2.13: Determinação da medida do ângulo da bissetriz $\overrightarrow{CD}$ de $\angle ACB$

Com isso, é fácil perceber que o ângulo formado pelo vetor $\overrightarrow{CD}$ e pelo eixo horizontal é a metade de $\alpha + \beta$. O comando para se referir a esse ângulo no METAPOST é

```
1/2*(angle(A-C)+angle(B-C)).
```

Para determinar algum ponto D que esteja nessa bissetriz, basta utilizar a seguinte linha de comando

```
D = C + whatever*dir(1/2*(angle(A-C)+angle(B-C))),
```

sendo `whatever` qualquer constante. Determinado o ponto, basta fazer o desenho da bissetriz com um comando `draw`.

Em um triângulo ABC, a bissetriz interna relativa a AB (ou ao vértice C) é a porção CD da bissetriz do ângulo interno $\widehat{C}$ do triângulo, desde C até o lado AB. O ponto $D \in AB$ é o pé da bissetriz interna relativa a AB. Assim, todo triângulo possui três bissetrizes internas. É preciso utilizar outra linha de comando, além da já citada no parágrafo anterior, para traçar apenas a bissetriz interna pois requer-se um ponto específico da bissetriz que esteja no lado AB. Este comando, já discutido, é `D = whatever[A,B]`.

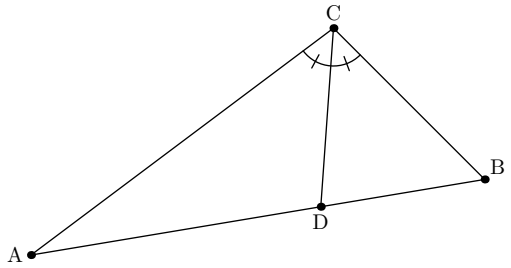


Figura 2.14: Triângulo ABC e a bissetriz relativa ao ângulo $\widehat{C}$.

Apesar de não ser uma ceviana, outra reta também importante para o estudo dos pontos notáveis do triângulo é a mediatriz.

Definição 12 *Dados os pontos A e B no plano, define-se como sendo a mediatriz do segmento AB a reta perpendicular a AB e que passa por seu ponto médio.*

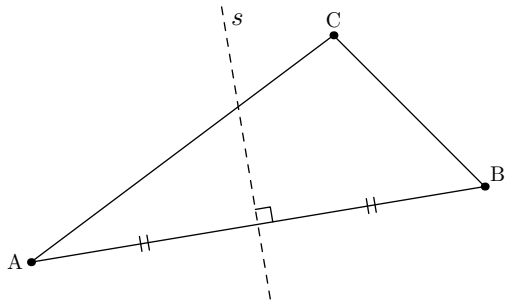


Figura 2.15: Triângulo ABC e a mediatriz s do lado AB.

Para fazer o desenho da mediatriz s do lado AB de um triângulo ABC com o METAPOST, primeiramente defina com a linha de comando

```
pair M; numeric AB; path t;
```

três variáveis que serão utilizadas. A variável `M` se referirá ao ponto médio de AB; o valor numérico `AB`, ao ângulo, em graus, do vetor $\overrightarrow{AB}$ em relação ao eixo horizontal; e o caminho `t`, à mediatriz do segmento AB. Essas variáveis facilitarão a digitação do código para a figura.

Para definir `M` como o ponto médio de AB, basta utilizar a sintaxe `M=0.5[A,B]`. Da mesma forma, como já visto anteriormente, para definir `AB` como a medida do ângulo, graus, do vetor $\overrightarrow{AB}$, a sintaxe é `AB=angle(B-A)`. Para definir a reta s, mediatriz de AB, é necessário traçar um segmento de reta que passe pelo ponto médio de AB. No caso da

Figura 2.15, foi utilizado um com uma unidade abaixo do ponto médio e três unidades acima. Como `dir(90+AB)` gera um vetor perpendicular ao segmento AB, o código usado para definir a mediatriz foi

```
t:=(M-u*dir(90+AB))--(M+3u*dir(90+AB)).
```

Com isso, basta um comando `draw t dashed evenly` para desenhá-la tracejada. Utilizar essa notação vetorial também é prática para rotular a reta. A linha de comando utilizada para escrever o s no desenho foi

```
label.urt(btex $s$ etex,M+2.6u*dir(90+AB));.
```

Vale observar, também, que como um triângulo possui três segmentos, cada um deles possui uma mediatriz. A intersecção das três mediatrizes, bem como a das cevianas particulares discutidas até então geram os pontos notáveis dos triângulos. Esses pontos possuem algumas características interessantes que serão discutidas a seguir.

2.2.1 Baricentro

Proposição 1 *Em todo triângulo, as três medianas passam por um único ponto chamado baricentro. Ademais, o baricentro divide cada mediana, a partir do vértice correspondente, na razão 2:1.*

Demonstração: Sejam N e P os pontos médios dos lados AC e AB, respectivamente, e seja $BN \cap CP = \{G\}$, como mostrado na Figura 2.16.

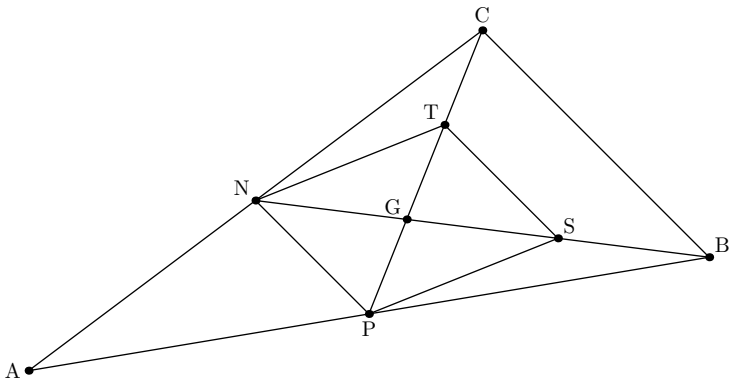


Figura 2.16: Medianas e o baricentro de ABC

Considere, também, os pontos médios S e T dos segmentos BG e CG, respectivamente. Sendo assim, NP é a base média de ABC relativa a BC, enquanto ST é a base média de BCG relativa a BC.

O Teorema da Base Média, afirma que um segmento que é base média referente a um dos lado do triângulo é paralelo a esse lado. Logo, tanto NP quanto ST são paralelos a BC. Esse teorema também afirma que essa base média mede metade da medida do lado ao qual ela é paralela. Logo, conclui-se também que $\overline{NP} = \overline{ST} = \frac{1}{2}\overline{BC}$. Isso é suficiente para garantir que $NPST$ é um paralelogramo. Como as diagonais de um paralelogramo se interscetam nos seus respectivos pontos médios, segue que $\overline{PG} = \overline{GT}$ e $\overline{NG} = \overline{GS}$. Mas, como $\overline{BS} = \overline{SG}$ e $\overline{CT} = \overline{TG}$, segue que $\overline{BS} = \overline{SG} = \overline{GN}$ e $\overline{CT} = \overline{TG} = \overline{GP}$, igualdades que fornecem $\overline{BG} = 2\overline{GN}$ e $\overline{CG} = 2\overline{GP}$.

Note, agora, que se M for o ponto médio de BC e G' for o ponto de interseção das medianas AM e BN, conclui-se, analogamente, que G' divide AM e BN na razão $2:1$ a partir de cada vértice. Daí, segue que os pontos G e G' são tais que $\overline{BG} = 2\overline{GN}$ e $\overline{BG'} = 2\overline{G'N}$, ou seja, $G \equiv G'$. Assim, AM, BN e CP se intersectam no mesmo ponto G, baricentro do triângulo ABC, de forma que G divide cada uma das medianas na razão $2:1$, a partir do vértice correspondente. ∎

Desenhar um triângulo e seu baricentro no METAPOST é uma tarefa muito simples. A Figura 2.17 mostra o código utilizado para desenhar as medianas e o baricentro de um triângulo com vértices nos pontos $A(-2,1)$, $B(4,0)$ e $C(2,2)$. A declaração das variáveis numéricas, das famílias de pares `A[]`, `B[]` e `C[]` e os comandos de rotular foram omitidos do código aqui mostrado. O mesmo será feito para os códigos dos outros pontos notáveis. Nos outros códigos, será omitida também a definição das coordenadas dos vértices, que são os pares `A0`, `B0` e `C0`.

```
pair G;
A0 = (-2u,u); B0 = (4u,0); C0 = (2u,2u);
A1 = 0.5[C0,B0]; B1 = 0.5[A0,C0];
C1 = 0.5[B0,A0];
G = whatever[A0,A1] = whatever[B0,B1];
draw A0--B0--C0--cycle;
draw C0--C1; draw A0--A1; draw B0--B1;
```

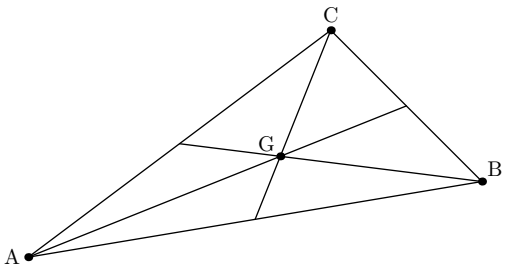


Figura 2.17: Baricentro

Perceba que basta definir os pontos médios de cada lado com os comandos e definir

que o baricentro `G` será a interseção de apenas duas dessas medianas, visto que a Proposição 1 mostrou que a terceira mediana também passará por esse ponto. Como não há possibilidades de o baricentro estar para fora do triângulo, outra forma que poderia ser utilizada para defini-lo seria com o comando `intersectionpoint`. Esse comando operador é utilizado entre caminhos e gera as coordenadas do ponto de interseção deles. No caso do baricentro, poderia ser utilizada a linha de comando

```
G = (A0--A1) intersectionpoint (B0--B1).
```

Outra forma de determinar as coordenadas baricentro G de um triângulo ABC é calculá-las por meio das coordenadas do vértice. Considerando P e N os pontos médios dos lados AB e AC, respectivamente, tem-se que $P = \dfrac{A+B}{2}$ e $N = \dfrac{A+C}{2}$. Como G está nos segmentos BN e CP, existem números t e λ tais que

$$G = B + t(N - B) = (1 - t)B + tN \text{ e } G = C + \lambda(P - C) = (1 - \lambda)C + \lambda P$$

Substituindo os valores de $P = \dfrac{A+B}{2}$ e $N = \dfrac{A+C}{2}$ obtém-se

$$G = \frac{t}{2}A + (1 - t)B + \frac{t}{2}C = \frac{\lambda}{2}A + (1 - \lambda)C + \frac{\lambda}{2}B.$$

As duas equações fornecem o mesmo ponto G quando $t = \lambda = \dfrac{2}{3}$ e, assim,

$$G = \frac{1}{3}A + \frac{1}{3}B + \frac{1}{3}C = \frac{A+B+C}{3}.$$

Como os pontos A, B e C foram definidos com os pares `A0`, `B0` e `C0`, usando a equação anterior tem-se que

```
G:=(A0+B0+C0)/3;.
```

2.2.2 Circuncentro

Proposição 2 *Em todo triângulo, as mediatrizes dos lados passam todas por um mesmo ponto chamado circuncentro.*

Demonstração: Sejam ABC um triângulo qualquer, r, s e t, respectivamente, as mediatrizes dos lados BC, CA e AB, e O o ponto de interseção das retas s e t, como na Figura

2.18. A mediatriz de um segmento pode ser caracterizada como o lugar geométrico dos pontos equidistante das suas extremidades. Assim, tem-se que, como $O \in t$, $\overline{OB} = \overline{OA}$ e, também, como $O \in s$, $\overline{OA} = \overline{OC}$. Portanto, $\overline{OB} = \overline{OC}$ e segue, de novo pela caracterização da mediatriz como lugar geométrico, que $O \in r$. ∎

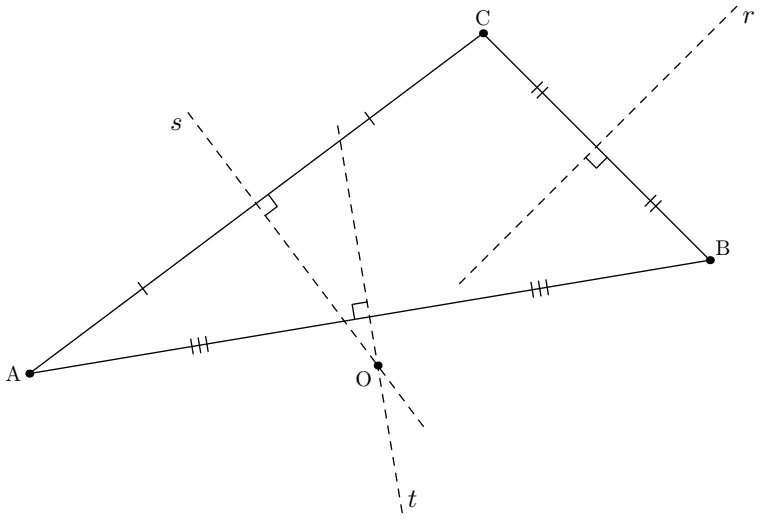


Figura 2.18: Mediatrizes e o circuncentro de ABC

Para esboçar as mediatrizes e o circuncentro de um triângulo ABC será necessário declarar um par `O` que representará o circuncentro; três caminhos `r, s, t`, referentes às mediatrizes; e três valores numéricos `AB`, `BC` e `CA`, referentes aos ângulos, em graus, formado pelo eixo horizontal e os vetores $\overrightarrow{AB}$, $\overrightarrow{BC}$ e $\overrightarrow{CA}$.

O primeiro passo é definir todas as coordenadas dos vértices e dos pontos médios do triângulo ABC. Como no exemplo do baricentro, os pares `A0`, `B0` e `C0` são os vértices do triângulo e os pares `A1`, `B1` e `C1`, os pontos médios dos lados BC, CA e AB, respectivamente. Depois, com o comando `angle`, atribuir os valores das variáveis numéricas `AB, BC` e `CA`. Para o METAPOST computar as coordenadas do circuncentro, basta analisar que se `O` será um ponto tal que o produto interno entre os vetores `(O-A1)` e `(C0-B0)` e entre os vetores `(O-B1)` e `(A0-C0)` serão nulos, pois são perpendiculares. A sintaxe para escrever isso no METAPOST é

```
(O-A1) dotprod (C0-B0) = (O-B1) dotprod (A0-C0) = 0.
```

Os comandos discutidos até aqui já são suficientes para marcar o circuncentro no triângulo. Porém, se há interesse em traçar, também, as mediatrizes do triângulo, basta

utilizar os comandos citados no início dessa Seção. Como as coordenadas do circuncentro e dos pontos médios já estão definidas, basta com um comando `draw` traçar segmentos de reta que passem por esses pontos. A Figura 2.19 mostra os códigos utilizados para definir cada uma das retas, dos ângulos e do circuncentro.

```
pair O; path r, s, t;
numeric AB, BC, CA;
AB=angle(B0-A0); BC=angle(C0-B0);
CA=angle(A0-C0);
(O-A1) dotprod (C0-B0) =
(O-B1) dotprod (A0-C0) = 0;
t:=(O+u*dir(90+AB))--(C1-u*dir(90+AB));
r:=(O+u*dir(90+BC))--(A1-u*dir(90+BC));
s:=(O+u*dir(90+CA))--(B1-u*dir(90+CA));
```

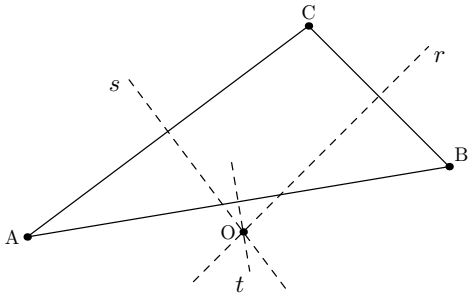


Figura 2.19: Circuncentro

2.2.3 Ortocentro

Proposição 3 *Em todo triângulo, as três alturas se intersectam em um só ponto chamado ortocentro.*

Demonstração: Seja ABC um triângulo qualquer. Há três casos a serem levados em consideração:

(a) ABC é retângulo, como na Figura 2.20. Nesse caso, suponha sem perda de generalidade, que o triângulo ABC é retângulo em A. Dessa forma, A é o pé das alturas relativas aos lados AB e AC. Como a altura relativa ao lado BC passa (por definição) por A, segue que as alturas de ABC concorrem em A.

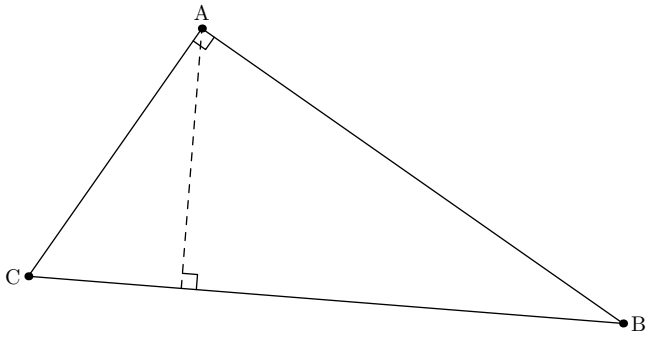


Figura 2.20: Ortocentro no triângulo retângulo

(b) ABC é acutângulo, como na Figura 2.21. Sejam r, s e t retas paralelas a BC, CA e AB, respectivamente, passando por A, B e C, também respectivamente. Sejam

também $r \cap s = \{P\}$, $s \cap t = \{M\}$ e $t \cap r = \{N\}$. Então, os quadriláteros $ABCN$ e $ABMC$ são paralelogramos, de forma que $\overline{CN} = \overline{AB} = \overline{CM}$ e, assim, C é o ponto médio de MN. Da mesma forma, B é o ponto médio de MP e A o ponto médio de NP.

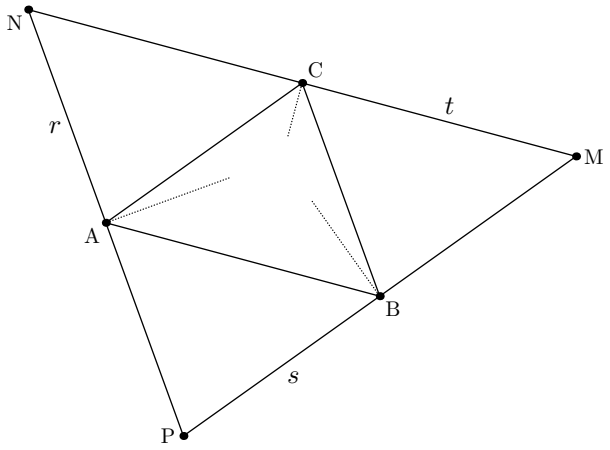


Figura 2.21: Ortocentro no triângulo acutângulo

Por outro lado, a altura relativa ao lado BC também é perpendicular ao lado NP, visto que $\overleftrightarrow{BC}$ e $\overleftrightarrow{NP}$ são paralelas. Do mesmo modo, as alturas relativas a AC e AB são respectivamente perpendiculares a MP e MN. Com isso, as alturas do triângulo ABC são as mediatrizes dos lados do triângulo MNP. Mas, a Proposição 2 provou que as mediatrizes dos lados de um triângulo são concorrentes, logo as alturas de ABC assim também serão.

(c) ABC é um triângulo obtusângulo. Neste caso, a prova é análoga à do caso (b). ∎

Agora, veja como será possível, com o METAPOST, traçar o ortocentro dos triângulos das Figuras 2.17 e 2.19. Além das variáveis já declaradas anteriormente, para esse desenho será necessário declarar um par `H`, que será o ortocentro, e três caminhos `r, s, t`, para traçar as alturas, se desejado.

Declaradas as variáveis e lembrando que `A0`, `B0` e `C0` são os vértices do triângulo, a nova família de pontos a ser utilizada aqui, `A2`, `B2` e `C2`, será a dos pés das alturas relativas a cada lado. A sintaxe para se obter esses pontos foi discutida na Subseção 2.1.2. Por exemplo, a linha de comando

```
A2 = whatever[B0,C0]; (A2-A0) dotprod (C0-B0) =0;
```

é utilizada para gerar `A2`, o pé da altura relativa ao vértice A. Depois de definidos os três pontos `A2`, `B2` e `C2`, para definir o par de coordenadas do ortocentro `H`, basta utilizar a linha de comando

```
H = whatever[A0,A2] = whatever[B0,B2];
```

pois, como visto, as três alturas se intersectam num mesmo ponto.

Com `H` definido, já é possível marcá-lo no desenho com um comando de rótulo. Mas se for de interesse traçar as alturas, basta fazê-las com os caminhos declarados `r, s, t`, como mostrado no código na Figura 2.22. Os segmentos do final do código, `C0--(C0+2u*dir(BC))` e `C0--(C0-2u*dir(CA))` são os prolongamentos dos lados CA e BC do triângulo.

```
pair H; path r, s, t;
A2 = whatever[B0,C0];
B2 = whatever[C0,A0];
C2 = whatever[A0,B0];
(A2-A0) dotprod (C0-B0) =
(B2-B0) dotprod (A0-C0) =
(C2-C0) dotprod (B0-A0) = 0;
H=whatever[A0,A2]=whatever[B0,B2];
t:=
(H+u*dir(90+AB))--(C2-u*dir(90+AB));
r:=
(H-u*dir(90+BC))--(A0+u*dir(90+BC));
s:=
(H-u*dir(90+CA))--(B0+u*dir(90+CA));
draw
C0--(C0+2u*dir(BC)) dashed evenly;
draw
C0--(C0-2u*dir(CA)) dashed evenly;
```

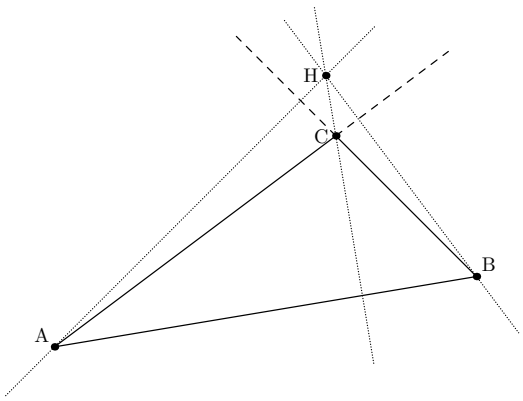


Figura 2.22: Ortocentro

2.2.4 Incentro

Proposição 4 *As bissetrizes internas de todo triângulo concorrem em um único ponto, o incentro do triângulo.*

Demonstração: Sejam r, s e t, respectivamente, as bissetrizes internas dos ângulos $\widehat{A}$, $\widehat{B}$ e $\widehat{C}$ do triângulo ABC e I o ponto de interseção das retas s e t.

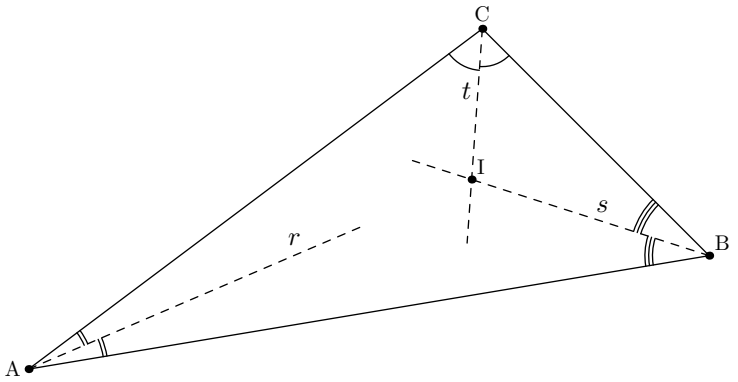


Figura 2.23: Incentro de um triângulo

A bissetriz de um ângulo pode ser caracterizada como o lugar geométrico dos pontos que equidistam dos lados desse ângulo. Como $I \in s$, segue da caracterização das bissetrizes como lugar geométrico que I equidista dos lados AB e BC de ABC. Analogamente, $I \in t$ garante que I equidista dos lados BC e AC. Portanto, I equidista de AC e AB e, usando novamente a caracterização como lugar geométrico das bissetrizes, conclui-se que I está na bissetriz do ângulo $\widehat{C}$, ou seja, $I \in r$. Assim, r, s e t concorrem em I. ∎

Para determinar o incentro de um triângulo com o METAPOST, será necessário determinar os três pés das bissetrizes internas, `A3`, `B3` e `C3`, de cada lado e determinar a interseção entre os segmentos com uma extremidade em um vértice do triângulo e outra no pé da bissetriz interna relativa a esse vértice. O código utilizado para determinar as coordenadas de pontos da bissetriz referente a um determinado ângulo já foi discutido anteriormente. Para marcar o incentro, a única variável diferente que precisará ser declarada é um par `I` que representará o incentro do triângulo.

Para determinar `A3`, o pé da bissetriz interna referente ao ângulo $\widehat{A}$, por exemplo, o código utilizado terá duas equações. A primeira é

```
A3 = whatever[C0,B0];
```

que determina que esse ponto está na reta que passa por BC. A segunda é

```
A3 = A0+whatever*dir(1/2*angle(C0-A0)+1/2*angle(B0-A0));,
```

que indica que esse ponto está na bissetriz do ângulo formado pelos vetores $\overrightarrow{AC}$ e $\overrightarrow{AB}$.

Depois de definidos os três pés das bisstrizes, o incentro `I` pode ser definido por meio

da forma vetorial

```
I = whatever[C0,C3] = whatever[A0,A3];
```

ou utilizando o comando de intersecção entre caminhos

```
I = (B0--B3) intersectionpoint (A0--A3);,
```

visto que as bissetrizes sempre se intersectam dentro do triângulo.

```
pair I;
A3 = whatever[C0,B0] = A0 + whatever*
dir(1/2*angle(C0-A0)+1/2*angle(B0-A0));
B3 = whatever[A0,C0] = B0 + whatever*
dir(1/2*angle(A0-B0)+1/2*angle(C0-B0));
C3 = whatever[B0,A0] = C0 + whatever*
dir(1/2*angle(A0-C0)+1/2*angle(B0-C0));
I = whatever[C0,C3] = whatever[A0,A3];
```

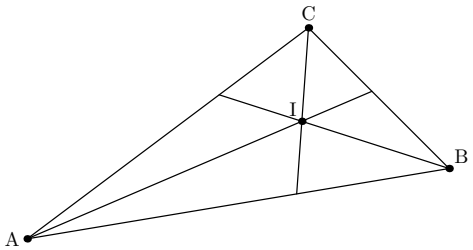


Figura 2.24: Incentro

2.3 Polígonos Regulares

Definição 13 *Sejam $n \geq 3$ um natural e $A_1, A_2, \ldots, A_n$ pontos distintos do plano. Diz-se que $A_1A_2 \ldots A_n$ é um polígono (convexo) se, para $1 \leq i \leq n$, a reta $\overleftrightarrow{A_iA_{i+1}}$ não contém nenhum outro ponto A_j, mas deixa todos eles em um mesmo semiplano, dentre os que ela determina (aqui e no que segue, $A_0 = A_n$, $A_{n+1} = A_1$ e $A_{n+2} = A_2$).*

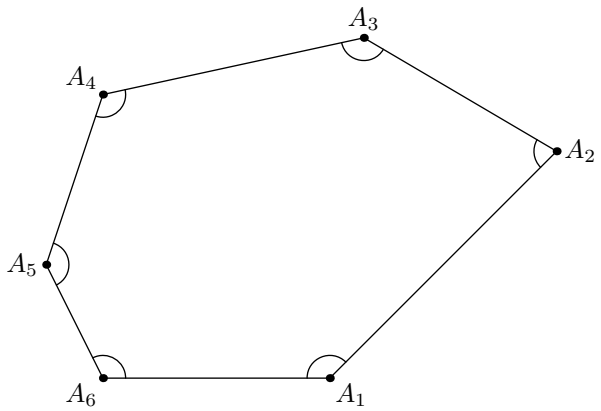


Figura 2.25: Polígono convexo de 6 lados

Traçar um polígono com o METAPOST é uma tarefa simples e pode ser feita com os

recursos discutidos até agora. Basta definir as coordenadas dos vértices e fazer o traçado dos lados.

Definição 14 *Um polígono é dito regular se todos os seus lados e todos os seus ângulos internos tiverem medidas iguais.*

Para desenhar com o METAPOST um polígono regular de três lados, ou seja, um triângulo equilátero, por exemplo, pode-se definir as coordenadas de um vértice e depois calcular as coordenadas dos outros dois em função das do primeiro. Considere, o código e o resultado na Figura 2.26.

```
numeric l; pair A[]; l:=3cm; A1=origin;
A2=A1+l*dir(10); A3=A1+l*dir(70);
draw A1--A2--A3--cycle;
dotlabel.bot(btex $A_1$ etex, A1);
dotlabel.rt(btex $A_2$ etex, A2);
dotlabel.lft(btex $A_3$ etex, A3);
```

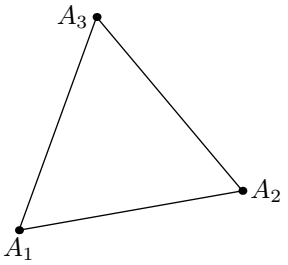


Figura 2.26: Polígono regular de três lados

Primeiramente foram declaradas as variáveis a serem utilizadas para o desenho. A família `A` de pares se refere aos vértices do triângulo e o valor numérico `l`, à medida dos lados do mesmo. O primeiro par, referente ao vértice A_1, foi definido na origem, mas poderia ter sido definido em qualquer ponto. Os pontos `A2` e `A3` foram definidos de forma que `A2` estivesse a `l` unidades na direção de 10° e `A3`, 70°.

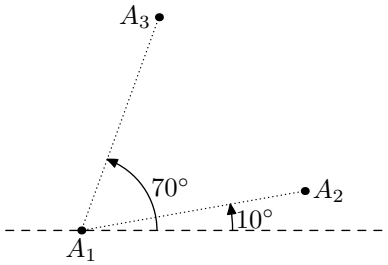


Figura 2.27: Construção do triângulo equilátero

Determinados os pontos, basta traçar o triângulo com o comando `draw` e marcar os pontos com os comandos de `dotlabel`. Veja que, como os ângulos internos de um triângulo equilátero medem 60°, o ângulo utilizado para determinar os outros dois vértices `A2` e `A3` poderiam ser quaisquer tais que a diferença fosse 60°.

Para traçar um quadrado, isto é, um polígono regular de quatro lados, o METAPOST contém um comando predefinido `unitsquare`. Esse comando refere-se ao caminho

```
(0,0)--(1,0)--(1,1)--(0,1)--cycle,
```

que é um quadrado de lado unitário com vértice na origem desenhado no primeiro quadrante. Para fazer um quadrado de 3 *cm* de lado, por exemplo, basta utilizar a linha de comandos

```
draw unitsquare scaled 3cm;
```

que o resultado é o mostrado na Figura 2.28.

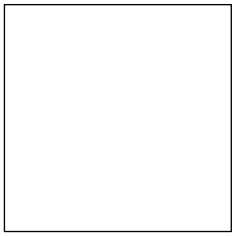


Figura 2.28: Quadrado com 3 *cm* de lado

Para construir um polígono regular de n lados, basta dividir um círculo de centro O em n arcos iguais, obtendo os pontos $A_1, A_2, \ldots, A_n$. Assim, o polígono $A_1A_2 \ldots A_n$ tem n lados iguais e os ângulos correspondente iguais, portanto é um polígono regular de n lados.

Para traçar um polígono regular de n lados dividindo um círculo em n partes iguais com o METAPOST é necessário, apenas, determinar as coordenadas destes pontos e, daí, traçar os lados do polígono. Considere o pentágono da Figura 2.29 e o código utilizado para determinar as coordenadas dos vértices e o utilizado para traçar os seus lados.

```
numeric n, theta, raio; path P;
n:=5; theta := 360/n; raio:=2cm;
for i = 1 upto n:
z[i] = dir(i*theta)*raio; endfor
P := z[1] for i = 1 upto n:
--z[i] endfor --cycle;
draw P;
```

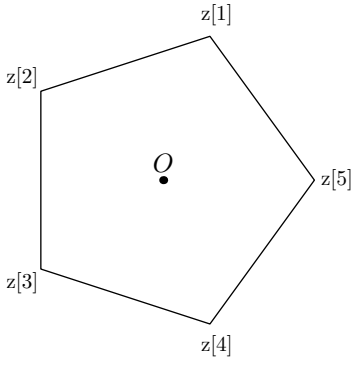


Figura 2.29: Pentágono.

Note que, primeiramente, foram definidas variáveis numéricas. A variável `n` foi definida para ser a quantidade de lados. Como um círculo equivale a um arco de 360°, se este será dividido em n partes, os arcos formados pelos vértices serão de $\theta = \frac{360^\circ}{n}$. Por essa razão, a variável numérica `theta` foi definida como

```
theta := 360/n.
```

A variável numérica `raio` se refere à medida do raio r do círculo que será dividido e o caminho `P` ao polígono regular.

Definidas as variáveis a serem utilizadas, o próximo passo é definir as coordenadas dos vértices. Para facilitar a escrita, foi utilizado um loop que define todos os vértices. O comando

```
for i = 1 upto n:  z[i]=dir(i*theta)*raio; endfor
```

define que o primeiro ponto, `z[1]`, terá as coordenadas calculadas pelo vetor que forma $1 \cdot \theta$ graus com a horizontal, a uma distância r da origem. Já o segundo vértice, `z[2]`, é definido da mesma forma, só que o ângulo é $2 \cdot \theta$. Definidos todos os n vértices, os arcos formados por cada par de vértices consecutivos medem θ graus e, assim, o polígono é regular. Na Figura 2.29, foram nomeados os vértices e o ponto O, que é a origem, foi marcado para facilitar o entendimento.

O segundo passo é definir o caminho `P`, que nada mais é do que a junção dos vértices por traços (`--`), formando os segmentos de reta que serão os lados do polígono. Uma forma de definir esse caminho seria

```
P:=z[1]--z[2]--z[3]--z[4]--z[5]--cycle;,
```

pois o polígono em questão é um pentágono. Como o objetivo é criar um código que possa ser utilizado para qualquer número n de lados, o comando utilizado,

```
P:=z[1] for i=1 upto n:  --z[i] endfor --cycle;,
```

gera justamente o caminho citado anteriormente, só que para qualquer valor de n. Com isso, basta um comando `draw P` que o polígono regular será desenhado. Com uma simples mudança na variável `n` de 5 para 15, o resultado é o mostrado na 2.30.

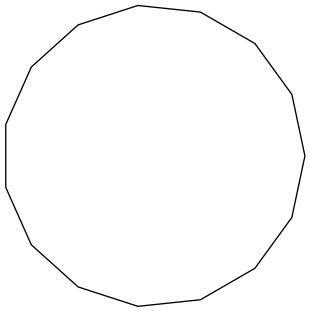


Figura 2.30: Pentadecágono regular

Note que com esse código é preciso definir a medida do raio do círculo circunscrito ao polígono. Porém, utilizando alguns conhecimentos geométricos, é possível traçar um polígono regular com uma medida l de lado fixada. Para isso é necessário escrever a medida r do raio em função da medida l do lado. Essa tarefa não é difícil, pois como pode ser visto na Figura 2.31, pela Lei dos Cossenos, tem-se que

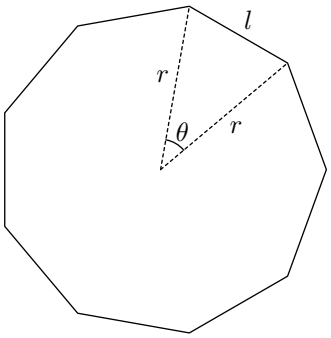


Figura 2.31: Eneágono regular

$$
\begin{aligned}
l^2 &= r^2 + r^2 - 2\cos\theta \cdot r^2 \\
l^2 &= 2r^2\left(1 - \cos\theta\right) \\
l^2 &= 2r^2\left(1 - \cos^2\left(\frac{\theta}{2}\right) + \operatorname{sen}^2\left(\frac{\theta}{2}\right)\right) \\
l^2 &= 2r^2\left(1 - \cos^2\left(\frac{\theta}{2}\right) + 1 - \cos^2\left(\frac{\theta}{2}\right)\right) \\
l^2 &= 4r^2\left(1 - \cos^2\left(\frac{\theta}{2}\right)\right) \\
l^2 &= 4r^2\left(\operatorname{sen}^2\left(\frac{\theta}{2}\right)\right)
\end{aligned}
$$

Como $\theta = \dfrac{360^\circ}{n}$ e $n \geq 3$, então $\theta \leq 120^\circ$. Assim, $\operatorname{sen} \dfrac{\theta}{2} \geq 0$ e, por sua vez,

$$l = 2r \operatorname{sen} \frac{\theta}{2}.$$

Com essa informação, se for desejada uma medida específica do lado do polígono regular a ser desenhado, basta definir outra variável numérica `l`, referente a medida do lado e escrever a variável numérica `raio` em função de `l`, isto é,

$$r = \frac{l}{2 \operatorname{sen} \frac{\theta}{2}}.$$

A Figura 2.32 mostra um exemplo em que um decágono regular é desenhado com seus lados medindo exatamente 2 cm.

```
numeric n, l, theta, raio; path P;
n:=10; l:= 2cm; theta := 360/n;
raio:=l/(2*sind(theta/2));
for i = 1 upto n:
z[i]=dir(i*theta)*raio; endfor
P:=z[1] for i=1 upto n:
--z[i] endfor --cycle;
draw P;
```

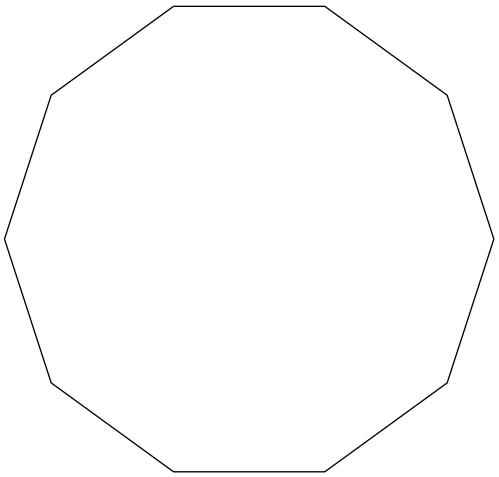


Figura 2.32: Decágono regular com 2 cm de lado.

Capítulo 3

Tranformações

Neste capítulo serão discutidas algumas transformações no plano, bem como alguns recursos que permitem utilizá-las no METAPOST para elaboração de alguns gráficos. Também serão apresentadas algumas aplicações dessas transformações para o esboço de círculos e elipses, sendo estas exemplificadas com alguns problemas interessantes de Geometria.

3.1 Transformações no Plano

Definição 15 *Uma transformação no plano $\mathbb{R}^2$ é uma função $T : \mathbb{R}^2 \to \mathbb{R}^2$, isto é, uma correspondência que associa a cada ponto P do plano outro ponto $P_1 = T(P)$ do plano, chamado sua imagem por T.*

A transformação identidade $T : \mathbb{R}^2 \to \mathbb{R}^2$, definida por $T(x, y) = (x, y)$ é indicada no Metapost pelo comando `identity`.

Exemplo 1 *Considere $T : \mathbb{R}^2 \to \mathbb{R}^2$ definida por $T(x, y) = \left(x, \frac{1}{2}y\right)$. Tal transformação é denominada compressão vertical do plano. Aplicada a um ponto (x, y) do plano, ela preserva a abscissa e reduz a ordenada à metade, comprimindo verticalmente as figuras planas.*

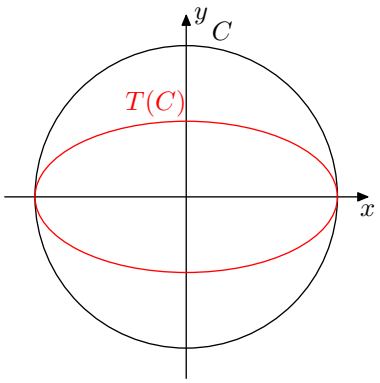


Figura 3.1: Compressão vertical

Exemplo 2 *Considere* $T : \mathbb{R}^2 \to \mathbb{R}^2$ *definida por* $T(x, y) = \left(\frac{1}{2}x, y\right)$. *Tal transformação é denominada compressão horizontal do plano. Aplicada a um ponto* (x, y) *do plano, ela preserva a ordenada e reduz a abscissa à metade, comprimindo horizontalmente as figuras planas.*

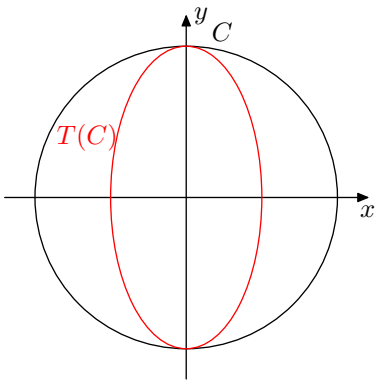


Figura 3.2: Compressão horizontal

3.1.1 Tipo `transform`

Além dos tipos básicos de dados já explanados anteriormente, o METAPOST possui um tipo de variável avançado que faz transformações. Algumas transformações são bem práticas para recursos gráficos matemáticos. O tipo `transform` é utilizado para guardar transformações em uma única variável. Uma transformação pode ser qualquer combinação de rotação, dimensionamento, inclinação e deslocamento. Essas transformações podem ser aplicadas tanto a pontos como a caminhos definidos com `path`. No caso de caminhos, elas transformam cada um de seus pontos conforme as equações predefinidas de cada uma.

Quando se quer declarar uma variável, por exemplo, `T`, como sendo uma tranformação, escreve-se `transform T;`. Em seguida a transformação precisa ser descrita. A trans-

formação predefinida `identity` é um ponto de partida necessário para esse processo. Ela não altera o ponto ou caminho inicial. Apenas define que a partir dele, as outras transformações serão realizadas. Veja como essas transformações funcionam com alguns exemplos gráficos básicos.

Transformação `yscaled a`: A transformação indicada por `yscaled a` é a transformação que a cada ponto (x, y) do plano associa o ponto (x, ay). Assim,

$$(x, y) \texttt{ yscaled } a = (x, ay)$$

Transformação `xscaled a`: A transformação indicada por `xscaled a` é a transformação que a cada ponto (x, y) do plano associa o ponto (ax, y). Assim,

$$(x, y) \texttt{ xscaled } a = (ax, y)$$

No Exemplo 1, o código utilizado para gerar as curvas C e $T(C)$ foi:

```
transform T;
T:= identity yscaled (1/2);
path f, g;
u:=0.8cm; r:=2*u;
f:=fullcircle scaled 2.5r;
g:=f transformed T;
draw f withcolor black;
draw g withcolor red;
```

Transformação `scaled a`: A transformação `scaled a` já foi abordada no Capítulo 1 e é a mais comum dentre as transformações. O comando `scaled a` indica a transformação $T(x, y) = (ax, ay)$, assim

$$(x, y) \texttt{ scaled } a = (ax, ay)$$

Exemplo 3 *Considere um triângulo com vértices em* $A = (0, 0), B = (3, 5)$ *e* $C = (4, 1)$ *e a transformação* $T(x, y) = \left(\frac{x}{2}, \frac{y}{2}\right)$. *Aplicando a transformação* T, *obtém-se o triângulo de vértices* $D = (0, 0), E = \left(\frac{3}{2}, \frac{5}{2}\right)$ *e* $F = \left(2, \frac{1}{2}\right)$. *No METAPOST, essa transformação é feita com o comando* `scaled 1/2`.

```
numeric u; u:=1cm;
pair A, B, C, D, E, F;
path ABC, DEF;
transform T; T:= identity scaled (1/2);
A:=(0,0); B:=(3u,5u); C:=(4u,u);
ABC:= A--B--C--cycle;
DEF:= ABC transformed T;
draw ABC; draw DEF withcolor red;
```

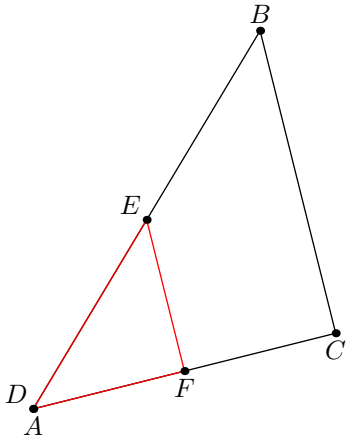


Figura 3.3: Transformação no Triângulo ABC

Transformação `rotated` θ: O comando de transformação `rotated` θ indica a transformação em $\mathbb{R}^2$ definida por $T(x, y) = (x \cos\theta - y \text{ sen } \theta, x \text{ sen } \theta + y \cos\theta)$ e efetua uma rotação de centro na origem e ângulo θ graus.

```
numeric u; u:=1cm;
pair A, B, C, D, E, F;
path ABC, DEF;
transform T;
T:= identity rotated (90);
A:=(0,0); B:=(u,4u); C:=(3u,u/2);
ABC:= A--B--C--cycle;
DEF:= ABC transformed T;
draw ABC; draw DEF withcolor red;
```

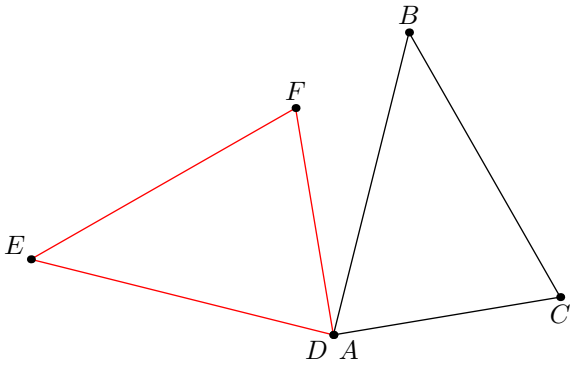


Figura 3.4: Rotação de 90° do Triângulo ABC

Transformação `shifted (a,b)`: O comando `shifted (a,b)` indica a transformação em $\mathbb{R}^2$ denominada Translação, definida por $T(x, y) = (x + a, y + b)$. A translação transforma toda figura F numa figura $T(F) = \{(x + a, y + b) \mid (x, y) \in F\}$. Um sistema de eixos ortogonais Ox e Oy é transformado no sistema Ox_1 e Oy_1, cujos eixos são paralelos a Ox e Oy preservando também os sentidos.

```
numeric u; u:=1cm;
path eixox, eixoy, eixor, eixos;
eixox:=(-2u, 0)--(3u,0);
eixoy:=(0,-2u)--(0,3u);
drawarrow eixox; drawarrow eixoy;
transform T;
T:= identity shifted (u, 2u);
eixor:= eixox transformed T;
eixos:= eixoy transformed T;
drawarrow eixor dashed evenly;
drawarrow eixos dashed evenly;
```

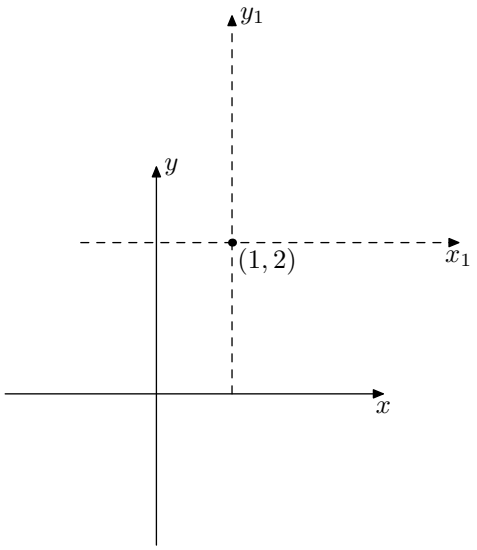


Figura 3.5: Eixos transladados

Exemplo 4 *Considere um triângulo de vértices* $A = (0,0), B = (3,5)$ *e* $C = (4,1)$ *e* $T(x,y) = (x+2, y-2)$. *Aplicando* T *em* ABC, *obtém-se o triângulo de vértices* $D = (2,-2), E = (5,3)$ *e* $F = (6,-1)$. *No* METAPOST *essa transformação é feita com o comando* `shifted (2,-2)`.

```
numeric u; u:=1cm;
pair A, B, C, D, E, F;
path ABC, DEF;
transform T;
T:= identity shifted (2u,-2u);
A:=(0,0); B:=(3u,5u);  C:=(4u,u);
ABC:= A--B--C--cycle;
DEF:= ABC transformed T;
draw ABC; draw DEF withcolor red;
```

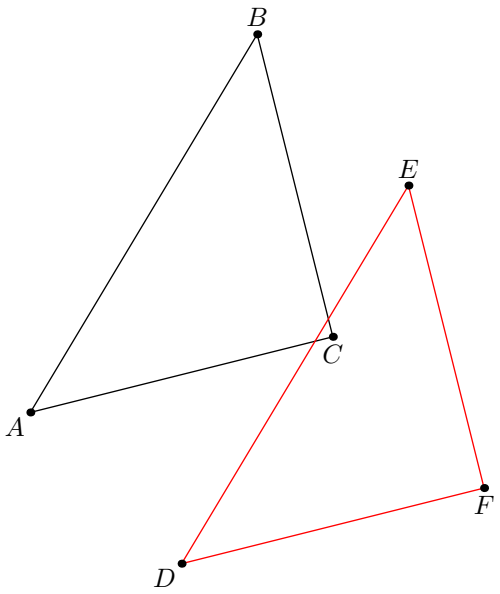


Figura 3.6: Triângulo ABC transladado

A partir de transformações dadas pode-se fazer composições destas transformações, como mostra a Figura 3.7. Quando esse recurso for utilizado, é preciso se atentar à ordem em que as transformações são compostas. Dependendo da posição em que elas são dispostas na linha de comando, pode ser que o resultado não seja o esperado.

```
u:=2cm;
pair A, B, C, D, E, F;
path ABC, DEF; transform T;
T:= identity rotated -45 shifted (0,2);
A:=origin; B:=(1,1); C:=(-1,2);
ABC:=A--B--C--cycle;
DEF:=ABC transformed T;
draw ABC scaled u;
draw DEF scaled u withcolor red;
```

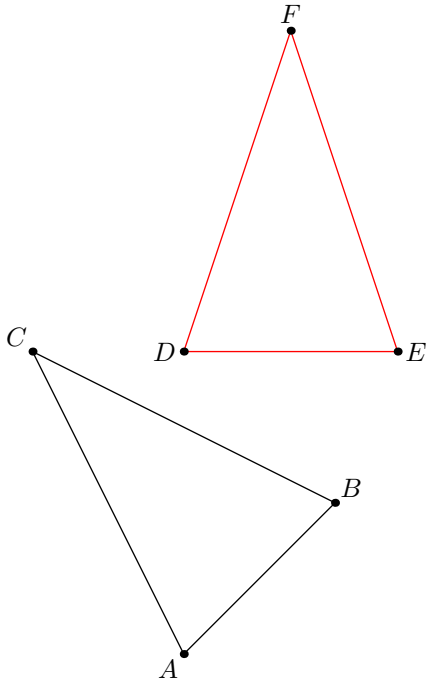


Figura 3.7: Triângulo ABC rotacionado e transladado

Nas seções a seguir, serão apresentadas algumas aplicações de transformações para o traçado de círculos e elipses com o METAPOST. Para isso, vale a pena retomar alguns conceitos e resultados da Geometria Analítica, encontrados em [5]. Como visto na Seção 1.4, o comando `fullcircle` desenha um círculo de raio unitário centrado na origem. Esse comando e algumas das transformações citadas nessa seção serão úteis para criar círculos e elipses no METAPOST.

3.2 Círculos

Definição 16 *O círculo $\mathcal{C}$ de centro no ponto $A \in \pi$ e raio $r > 0$ é o conjunto dos pontos do plano π situados à distância r do ponto A, ou seja:*

$$\mathcal{C} = \{P \in \pi \mid d(P, A) = r\}.$$

Considerando um sistema de coordenadas ortogonal OXY do plano π, suponha que o centro A do círculo esteja sobre o ponto (a, b). Segue que

$$P = (x, y) \in \mathcal{C} \Leftrightarrow d(P, A) = r \Leftrightarrow d(P, A)^2 = r^2 \Leftrightarrow (x - a)^2 + (y - b)^2 = r^2.$$

Com isso, a equação do círculo com centro em (a, b) e raio r é dada por

$$(x-a)^2 + (y-b)^2 = r^2.$$

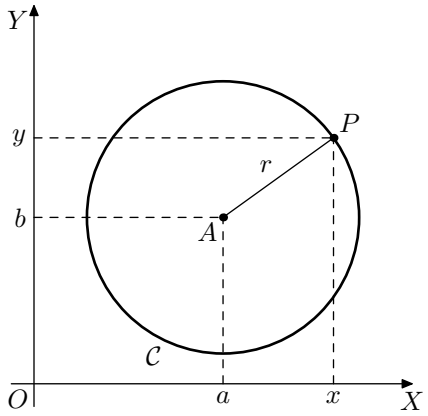


Figura 3.8: Círculo $\mathcal{C}$ de centro A e raio O

O comando utilizado para representar um círculo com o METAPOST é o `fullcircle`, comentado na Seção 1.4. Se o objetivo for traçar um círculo de raio r, centrado em (a, b) em um plano cartesiano como o comentado na Seção 1.3, basta utilizar aplicar a transformação

```
T:=identity scaled 2r shifted (a,b);
```

a um caminho definido por `fullcircle`. Isso acontece porque o comando `fullcircle` gera um círculo de diâmetro unitário, ou seja, de raio $\frac{1}{2}$. Os pontos desse círculo, por sua vez, satisfazem a equação

$$x^2 + y^2 = \left(\frac{1}{2}\right)^2 \tag{3.1}$$

A primeira transformação, `scaled 2r`, multiplica as duas coordenadas, x e y, pelo fator $2r$. Assim, os pontos (x, y) são transformados em pontos (x', y') tais que

$$(x', y') = (2r \cdot x, 2r \cdot y) \Rightarrow \begin{cases} x = \dfrac{x'}{2r} \\ \\ y = \dfrac{y'}{2r} \end{cases} \tag{3.2}$$

Substituindo os valores da Equação 3.2 na Equação 3.1, segue que

$$\left(\frac{x'}{2r}\right)^2 + \left(\frac{y'}{2r}\right)^2 = \left(\frac{1}{2}\right)^2 \Rightarrow \frac{(x')^2}{4r^2} + \frac{(y')^2}{4r^2} = \frac{1}{4} \Rightarrow (x')^2 + (y')^2 = r^2 \tag{3.3}$$

que é a equação de um círculo centrado na origem com raio r.

A segunda transformação, `shifted (a,b)`, soma a primeira coordenada com a e a segunda com b. Desta maneira, os pontos (x', y') são transformados em pontos (x'', y'') de forma que

$$(x'', y'') = (x' + a, y' + b) \Rightarrow \begin{cases} x' = x'' - a \\ y' = y'' - b \end{cases} \tag{3.4}$$

Substituindo os valores da Equação 3.4 na Equação 3.3, segue que o resultado final será os pontos (x'', y'') tais que

$$(x'' - a)^2 + (y'' - b)^2 = r^2$$

que é a equação do círculo esperado, de raio r e centro (a, b).

Exemplo 5 *Traçar um círculo num plano cartesiano com raio* 2 *e centro em* $(3, 1)$.

```
path C; pair c; transform T;
u:=1cm; r:= 2u; c:=(3u,u);
T:= identity scaled 2r shifted c;
C:= fullcircle transformed T;
draw C withpen pencircle scaled 1bp;
```

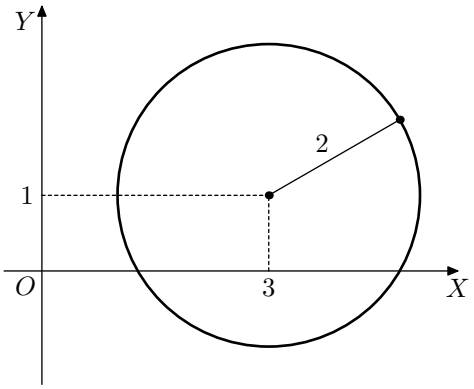


Figura 3.9: Círculo com raio 2 e centro em $(3, 1)$.

3.3 Elipses

Na Figura 3.1, do início deste capítulo, foi aplicada uma compressão vertical em um círculo, resultando numa elipse. Agora será mostrado como a Geometria Analítica comprova que, de fato, as elipses no plano são resultados de transformações do círculo. Conse-

quentemente, será possível traçar quaisquer elipses utilizando-se apenas de transformações no comando `fullcircle`.

Definição 17 *Uma elipse $\mathcal{E}$ de focos F_1 e F_2 é o conjunto dos pontos P do plano cuja soma das distâncias a F_1 e F_2 é igual a uma constante $2a > 0$, maior do que a distância entre os focos $2c \geq 0$. Ou seja, sendo $0 \leq c < a$ e $d(F_1, F_2) = 2c$,*

$$\mathcal{E} = \{P \mid d(P, F_1) + d(P, F_2) = 2a\}.$$

A reta que contém os dois focos, F_1 e F_2, da elipse é chamada reta focal. O centro da elipse é o ponto médio do segmento F_1F_2. O eixo focal é o segmento cujas extremidades são os dois pontos de intersecção da reta focal com a elipse. A reta não focal é a reta perpendicular à reta focal que passa pelo centro da elipse. O eixo não focal é o segmento com extremidades nas intersecções da reta não focal com a elipse.

Para determinar uma equação que satisfaça qualquer ponto $P = (x, y) \in \mathcal{E}$, considere um sistema de coordenadas ortogonal OXY, onde a origem, O, está no centro da elipse e os eixos focal e não focal sejam, respectivamente, o eixo OX e OY. Considere os números reais a, b e c conforme a Figura 3.10.

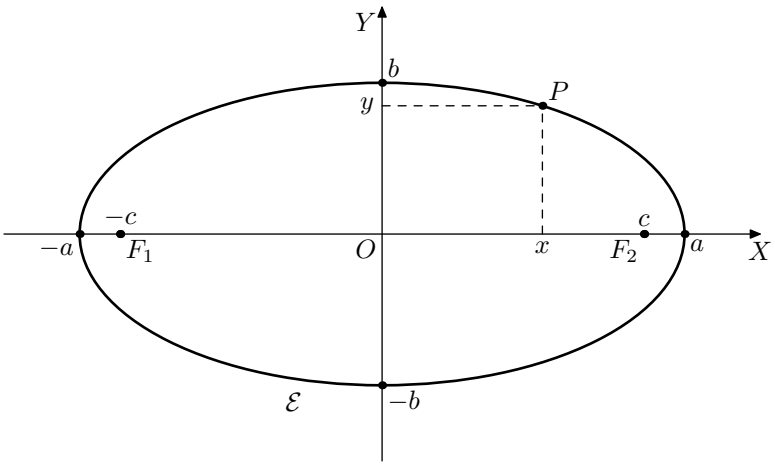


Figura 3.10: Elipse $\mathcal{E}$ de centro O e focos F_1 e F_2 e eixo focal $2a$

Primeiramente, observe que $F_1 = (-c, 0)$ e $F_2 = (c, 0)$. Isso posto, seja $B_1 = (0, b)$.

Note que, como $B_1 \in \mathcal{E}$, é possível afirmar que

$$\begin{aligned}
& B_1 = (0,b) \in \mathcal{E} \Leftrightarrow d(B_1,F_1) + d(B_1,F_2) = 2a \\
\Leftrightarrow\ & \sqrt{(0+c)^2 + (b-0)^2} + \sqrt{(0-c)^2 + (b-0)^2} = 2a \\
\Leftrightarrow\ & \sqrt{c^2+b^2} + \sqrt{c^2+b^2} = 2a \\
\Leftrightarrow\ & 2\sqrt{c^2+b^2} = 2a \\
\Leftrightarrow\ & c^2 + b^2 = a^2 \qquad (3.5)
\end{aligned}$$

A Equação 3.5 revela que o segmento B_1F_1 possui a medida a. De fato, isso faz sentido, pois como os triângulos OF_1B_1 e OF_2B_1 são congruentes pelo caso LAL, F_1B_1 e F_2B_1 possuem a mesma medida. Como a soma de suas medidas deve ser igual a $2a$, cada um deve medir a. Tendo isso em mente, tome um ponto $P \in \mathcal{E}$ e seja $P = (x,y)$. Segue que

$$\begin{aligned}
& P = (x,y) \in \mathcal{E} \Leftrightarrow d(P,F_1) + d(P,F_2) = 2a \\
\Leftrightarrow\ & \sqrt{(x+c)^2 + y^2} + \sqrt{(x-c)^2 + y^2} = 2a \\
\Leftrightarrow\ & \sqrt{(x+c)^2 + y^2} = 2a - \sqrt{(x-c)^2 + y^2} \qquad (3.6) \\
\Leftrightarrow\ & (x+c)^2 + y^2 = 4a^2 - 4a\sqrt{(x-c)^2 + y^2} + (x-c)^2 + y^2 \qquad (3.7) \\
\Leftrightarrow\ & x^2 + 2cx + c^2 = 4a^2 - 4a\sqrt{(x-c)^2 + y^2} + x^2 - 2cx + c^2 \\
\Leftrightarrow\ & 4cx = 4a^2 - 4a\sqrt{(x-c)^2 + y^2} \\
\Leftrightarrow\ & a\sqrt{(x-c)^2 + y^2} = a^2 - cx \qquad (3.8) \\
\Leftrightarrow\ & a^2\left((x-c)^2 + y^2\right) = \left(a^2 - cx\right)^2 \qquad (3.9) \\
\Leftrightarrow\ & a^2\left(x^2 - 2cx + c^2 + y^2\right) = a^4 - 2a^2cx + c^2x^2 \\
\Leftrightarrow\ & x^2\left(a^2 - c^2\right) + a^2y^2 = a^4 - a^2c^2 \\
\Leftrightarrow\ & x^2\left(a^2 - c^2\right) + a^2y^2 = a^2\left(a^2 - c^2\right) \\
\Leftrightarrow\ & b^2x^2 + a^2y^2 = a^2b^2 \\
\Leftrightarrow\ & \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \qquad (3.10)
\end{aligned}$$

Quase todas as equivalências são imediatas. Porém, é preciso provar ainda que 3.7 $\Rightarrow$

3.6 e que 3.9 $\Rightarrow$ 3.8. Para a segunda implicação, é preciso garantir que

$$a^2 - cx \geq 0$$

quando $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. Sendo assim, levando em consideração que $0 \leq c < a$, tem-se

$$\begin{aligned} & \frac{x^2}{a^2} \leq \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \Rightarrow \frac{x^2}{a^2} \leq 1 \Rightarrow x^2 \leq a^2 \Rightarrow |x| \leq a \Rightarrow -a \leq x \leq a \\ \Rightarrow \quad & a^2 - cx \geq a^2 - ca > a^2 - a^2 \\ \Rightarrow \quad & a^2 - cx > 0 \end{aligned}$$

Para mostrar que 3.7 $\Rightarrow$ 3.6 é preciso verificar que

$$2a - \sqrt{(x-c)^2 + y^2} \geq 0$$

quando $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. Então, lembrando que $0 \leq c < a$ e $a^2 = b^2 + c^2$, tem-se

$$\begin{aligned} & \frac{y^2}{b^2} \leq \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \Rightarrow \frac{y^2}{b^2} \leq 1 \Rightarrow y^2 \leq b^2 \Rightarrow y^2 - b^2 \leq 0 \\ \Rightarrow \quad & (x-c)^2 + y^2 = x^2 - 2cx + c^2 + y^2 \leq a^2 + 2a^2 + a^2 - b^2 + y^2 \leq 4a^2 \\ \Rightarrow \quad & \sqrt{(x+c)^2 + y^2} \leq \sqrt{4a^2} \\ \Rightarrow \quad & 2a - \sqrt{(x+c)^2 + y^2} \geq 0 \end{aligned}$$

Com todas as equivalências e implicações demonstradas, conclui-se que a Equação 3.10 é a forma canônica da elipse de centro na origem e reta focal coincidente com o eixo OX. Esse será o único caso considerado aqui porque o foco é traçar uma elipse com o METAPOST. Os outros casos de elipses, como a com eixo focal coincidente ao eixo OY, elipses com centro fora da origem ou, ainda, elipses rotacionadas são todos resultados de transformações nos pontos da elipse. Saber traçar uma elipse com centro na origem e eixo focal coincidente ao eixo OX é suficiente para traçar os outros casos.

Com o METAPOST a transformação `T:=identity xscaled 2a yscaled 2b` é suficiente para transformar o caminho `fullcircle` em uma elipse com as características descritas até então. Isso acontece porque, como citado, `fullcircle` gera o círculo descrito

por

$$x^2 + y^2 = \left(\frac{1}{2}\right)^2. \tag{3.11}$$

Depois disso, a primeira transformação, `xscaled 2a`, mantém as coordenadas y com o mesmo valor e multiplica as coordenadas x por $2a$. Assim, os pontos (x, y) de `fullcircle` são transformados em pontos (x', y') tais que

$$(x', y') = (2a \cdot x, y) \Rightarrow \begin{cases} x = \dfrac{x'}{2a} \\ y = y' \end{cases} \tag{3.12}$$

Ao substituir os valores de x e y de 3.12 na Equação 3.11, tem-se

$$\left(\frac{x'}{2a}\right)^2 + (y')^2 = \left(\frac{1}{2}\right)^2. \tag{3.13}$$

A segunda transformação, `yscaled 2b`, multiplica as ordenadas dos pontos por $2b$ e mantém as abscissas com os mesmos valores. Logo, cada ponto (x', y') é transformado em (x'', y''), tal que

$$(x'', y'') = (x', 2b \cdot y') \Rightarrow \begin{cases} x' = x'' \\ y' = \dfrac{y''}{2b} \end{cases} \tag{3.14}$$

Fazendo a substituição de 3.14 na Equação 3.13, o caminho final da transformação será dado pela equação

$$\left(\frac{x''}{2a}\right)^2 + \left(\frac{y''}{2b}\right)^2 = \left(\frac{1}{2}\right)^2 \Rightarrow \frac{(x'')^2}{a^2} + \frac{(y'')^2}{b^2} = 1$$

que é, justamente, a equação da elipse de centro na origem, reta focal coincidente com o eixo OX, eixo focal com medida igual a $2a$ e eixo não focal com medida igual a $2b$.

Exemplo 6 *Traçar a elipse com focos $F_1 = (1, 2)$ e $F_2 = (3, 1)$ que passa por $P = (3, 2)$.*

Note que essa elipse não possui o centro na origem e também está rotacionada. Então, além de utilizar as transformações `xscaled 2a yscaled 2b` para transformar o `fullcircle` na elipse, será necessário rotacionar e transladá-la. A medida `a` pode ser calculada implicitamente pelo METAPOST com a linha de comando

```
2a=abs(F1-P)+abs(F2-P);
```

enquanto `c` e `b` são calculados explicitamente por

```
c=abs(F1-C); b=a+-+c;.
```

O ângulo `theta` que é preciso rotacionar este caminho é o do vetor formado pelos focos, ou seja,

```
theta:=angle(F2-F1);.
```

Assim, o código e o resultado final dessas transformações são os mostrados na Figura 3.11.

```
path E; pair F[], P, C; transform T;
numeric a, b, c, theta;
F1:=(u,2u); F2:=(3u,u); P:=(3u,2u);
C=0.5[F1,F2];
2a=abs(F1-P)+abs(F2-P);
c=abs(F1-C); b=a+-+c;
theta:=angle(F2-F1);
T:= identity xscaled 2a yscaled 2b
rotated theta shifted C;
E:= fullcircle transformed T;
draw E withpen pencircle scaled 1bp;
```

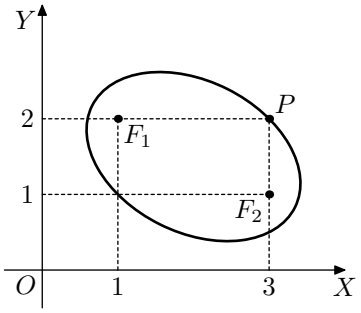


Figura 3.11: Elipse com focos $F_1 = (1, 2)$ e $F_2 = (3, 1)$ que passa por $P = (3, 2)$.

3.4 Problemas clássicos

Utilizando os conceitos de círculos, elipses e transformações vistos até então, é possível esboçar a solução de alguns problemas interessantes da Geometria. Esta seção trata de dois deles.

3.4.1 Círculo dos nove pontos

Em 1821, os matemáticos franceses Poncelet e Brianchon [4] demonstraram que existe, em qualquer triângulo, um círculo que passa pelos três pontos médios dos lados, pelos três pés das alturas e pelos três pontos médios dos segmentos com uma extemidade nos vértices do triângulo e outra no ortocentro do mesmo. Esse círculo é conhecido como círculo dos nove pontos.

Exemplo 7 ***(Círculo dos nove pontos)*** *Traçar o círculo dos nove pontos de um triângulo com vértices* $A = (0, 1)$, $B = (5, 1)$ *e* $C = (1, 10)$.

O esboço desse desenho, bem como o código utilizado no METAPOST encontra-se na Figura 3.12. Para determinar os pontos, primeiramente, foram declaradas as variáveis a serem utilizadas. Serão necessários três pontos para o vértice (`A,B,C`), os nove pontos do círculo (`D,E,F,G,H,I,J,K,L`), um ponto para o centro do círculo (`O`) e um ponto para o ortocentro (`X`). Também foi definida a unidade de medida como sendo `u=1cm`.

Declaradas as variáveis, o próximo passo é definir as coordenadas de cada ponto, seja de forma explícita ou por meio de equações lineares. Os pontos dos vértices do triângulo podem ser dados de forma explícita, com os comandos

```
A=(0,u); B=(5u,u); C=(u,10u);.
```

Escrever todos os outros pontos em função desses faz com que o código possa ser utilizado para qualquer outro triângulo, alterando-se apenas as coordenadas do vértice.

Os seis primeiros pontos do círculo são os três pontos médios dos lados e os três pés das alturas. Os comandos e equações utilizadas para este fim foram explanadas na Seção 2.2. Os outros três pontos, os pontos médios dos segmentos com uma extemidade nos vértices do triângulo e outra no ortocentro, precisam das coordenadas do ponto `X`, ortocentro do triângulo. Esse comando já foi discutido também na Subseção 2.2.3. As coordenadas dos últimos três pontos serão calculadas pelos comandos

```
J=.5[A,X]; K=.5[B,X]; L=.5[C,X];.
```

Com as coordenadas de todos os pontos do círculo determinadas, falta determinar as coordenadas do centro do círculo. Poncelet e Brianchon [4] provaram que esse círculo existe pois existe um ponto que equidista desses nove pontos. Escolhendo-se dois pares desses pontos, basta determinar as coordenadas da intersecção entre suas mediatrizes. Na Figura 3.12, foram utilizados os pontos D e I e os pontos G e K para esse cálculo. A linha de comandos

```
(O-0.5(D+I)) dotprod (D-I) = (O-0.5(G+K)) dotprod (G-K) = 0;
```

determina que `O` é um ponto tanto da mediatriz do segmento DI quanto do segmento GK.

Todos as variáveis de pares estão definidas. Agora basta desenhar o triângulo e o

círculo com comandos `draw`. O triângulo é simples, basta escrever

```
draw A--B--C--cycle;.
```

Para traçar o círculo, é só usar a sequência de transformações no comando `fullcircle` citadas anteriormente. Porém, é preciso determinar a medida do raio do círculo. Como o centro equidista dos nove pontos, o raio dele pode ser calculado por `abs(O-D)`, por exemplo. Assim, a linha de comandos

```
draw fullcircle scaled (2*abs(O-D)) shifted O;
```

desenhará o círculo desejado. No código mostrado na Figura 3.12, os comandos de rótulo foram omitidos.

```
numeric u; u=1cm;
pair A,B,C,D,E,F,G,H,I,J,K,L,O,X;
A=(0,u); B=(5u,u); C=(u,10u);
D=.5[A,B]; E=.5[B,C]; F=.5[A,C];
G=whatever[C,B];
H=whatever[A,C];
I=whatever[B,A];
(A-G) dotprod (B-C) =
(B-H) dotprod (C-A) =
(C-I) dotprod (A-B) = 0;
X = whatever[A,G] = whatever[B,H];
J=.5[A,X]; K=.5[B,X]; L=.5[C,X];
(O-0.5(D+I)) dotprod (D-I) =
(O-0.5(G+K)) dotprod (G-K) = 0;
draw A--B--C--cycle;
draw fullcircle
scaled (2*abs(O-D)) shifted O;
```

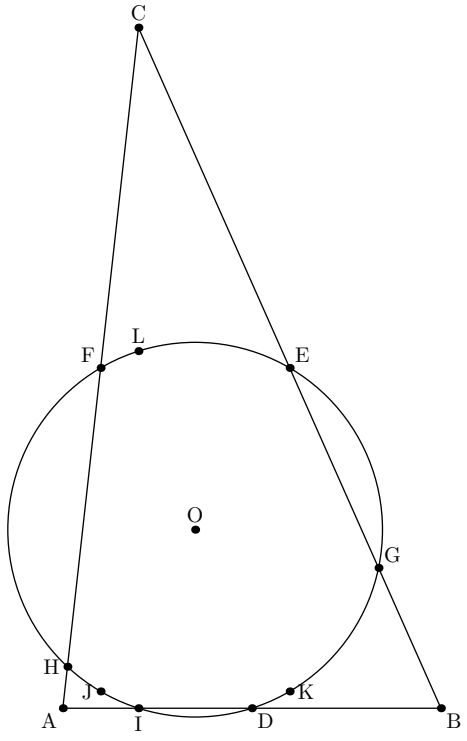
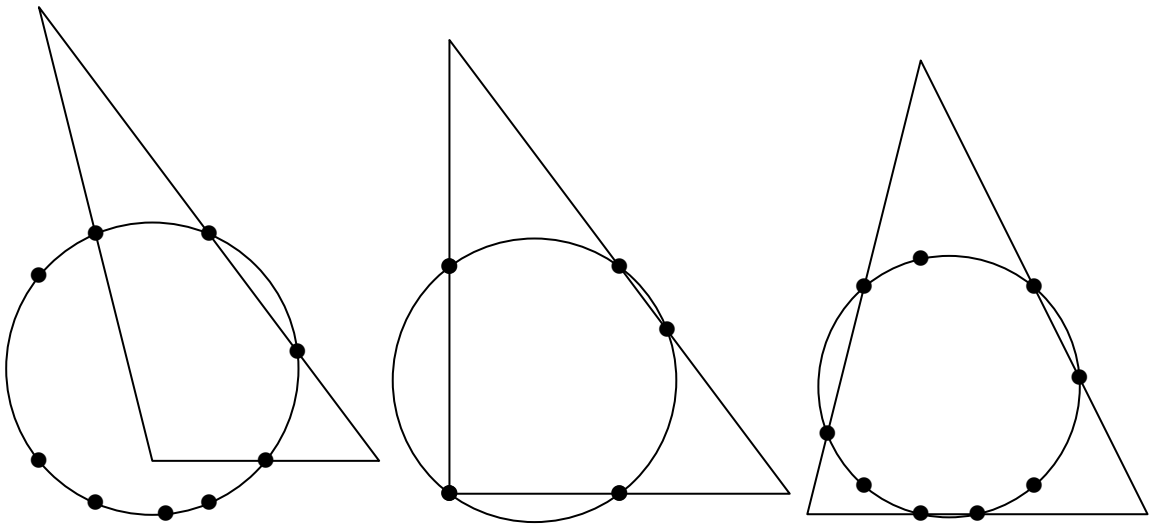


Figura 3.12: Círculo dos 9 pontos

A Figura 3.13 mostra como é possível traçar o círculo dos nove pontos com esse código apenas por alterar as coordenadas dos vértices do triângulo para os três tipos de triângulos (obtusângulo, retângulo e acutângulo). Note, na Figura 3.13(b), que no Triângulo Retângulo vários pontos são coincidentes.



(a) Triângulo obtusângulo (b) Triângulo retângulo (c) Triângulo acutângulo

Figura 3.13: Círculo dos nove pontos nos três tipos de triângulos.

3.4.2 Elipses inscritas no triângulo

Alves [1] mostrou que existem infinitas elipses inscritas em um triângulo qualquer. Na verdade, dado um ponto interior a um triângulo, sempre é possível traçar uma elipse inscrita a esse triângulo com um dos focos neste ponto. Dados um triângulo ABC e um ponto interno F, a construção dessa elipse é descrita da seguinte forma:

1. Obtenha os pontos simétricos G_1, G_2 e G_3 do foco F em relação às retas que contém os lados do triângulo ABC.

2. Marque o centro F' do círculo que passa por G_1, G_2 e G_3. Este será o outro foco da elipse.

3. Marque os pontos P_1, P_2 e P_3 na intersecção dos lados do triângulo com os segmentos que possuem uma extremidade em F' e a outra em G_1, G_2 e G_3.

4. Trace a elipse que possui focos em F e F' e passa pelo ponto P_1. Essa elipse é inscrita ao triângulo ABC e os pontos de tangência são P_1, P_2 e P_3.

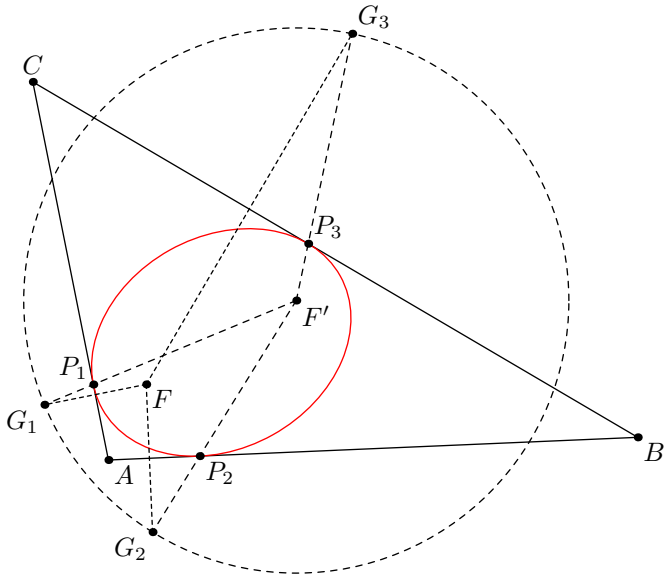


Figura 3.14: Construção da elipse inscrita no triângulo ABC com um foco em F.

Exemplo 8 *Construa a elipse inscrita no triângulo de vértices* $A = (-1, -3)$, $B = (3, 0)$ *e* $C = (1, 7)$ *que tem como um dos focos o ponto* $F = (1, -1)$.

```
numeric u,a,b,c,theta;  transform T;
pair A,B,C,F[],G[],O,P[]; path E;
u=1cm; A=(-u,-3u); B=(3u,0); C=(u,7u);
draw A--B--C--cycle;
F1=(u,-u); F3=whatever[A,C];
F4=whatever[A,B]; F5=whatever[C,B];
(F3-F1) dotprod (A-C)=
(F4-F1) dotprod (A-B)=
(F5-F1) dotprod (C-B)=0;
G1 = F1 rotatedaround (F3,180);
G2 = F1 rotatedaround (F4,180);
G3 = F1 rotatedaround (F5,180);
(F2-0.5(G2+G3)) dotprod (G2-G3) =
(F2-0.5(G1+G2)) dotprod (G1-G2) = 0;
P1=whatever[A,C]=whatever[F2,G1];
O=0.5[F1,F2];
2a=abs(F1-P1)+ abs(F2--P1);
c=abs(F1--O); b=a+-+c;
theta=angle(F2-F1);
T:= identity xscaled 2a yscaled 2b
rotated theta shifted O;
E:= fullcircle transformed T; draw E;
```

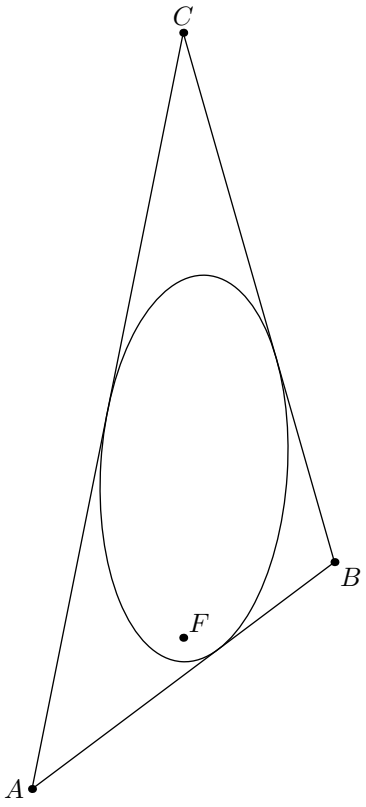


Figura 3.15: Elipse inscrita no triângulo ABC com focos F e F'

O código e a figura desse exemplo estão na Figura 3.15. Note que foram definidas as variáveis numéricas e a de pares primeiro. Os pares `A`, `B` e `C` são os vértices do triângulo.

Os da família `F[]` são os dois focos e os pés das perpendiculares baixada a partir de F sobre os lados do triângulo ABC. Os da família `G[]` são os simétricos do foco F em relação aos lados do triângulo. Por último, `O` refere-se ao centro da elipse e a família `P[]` aos pontos de tangência dela com o triângulo.

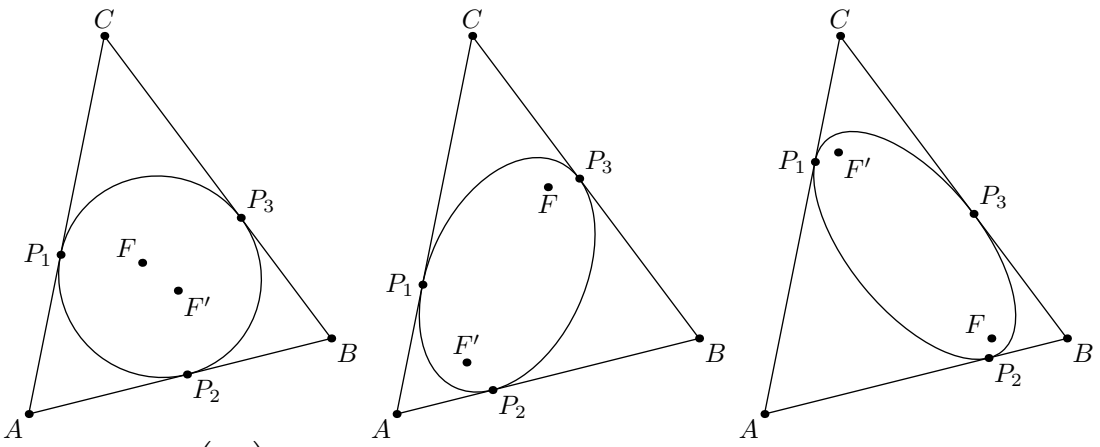
Definidos os vértices e as coordenadas do ponto interno `F1` que será um dos focos da elipse, é preciso determinar os pontos `F3`, `F4` e `F5` que serão, respectivamente, os pés das perpendiculares baixadas por F sobre os lados AC, BA e CB. Feito isso, determinar os simétricos de F em relação a cada lado é fácil de fazer com uma transformação disponível no METAPOST, a `rotatedaround`. A linha de comandos

```
G1 = F1 rotatedaround (F3,180);
```

determina que o par `G1` é `F1` rotacionado 180° em torno do ponto `F3`. Ou seja, `G1` é, justamente, o simétrico de F em relação ao lado AC.

Com as coordenadas dos pontos `G1`, `G2` e `G3` calculadas, os outros comandos utilizados para fazer os próximos passos da construção já foram discutidos anteriormente. Determinado o outro foco `F2` e um ponto `P1` que será o ponto de tangência do lado AC com a elipse, basta utilizar a transformação utilizada no Exemplo 6.

A Figura 3.16 mostra o triângulo com vértices em $A = (0, 0)$, $B = (4, 1)$ e $C = (1, 5)$ e o ponto F em três posições diferentes. A diferença entre os códigos de uma imagem para outra é a simples mudança das coordenadas do foco F e, obviamente, a posição dos rótulos.



(a) $F = \left(\frac{3}{2}, 2\right)$ (b) $F = (2, 3)$ (c) $F = (3, 1)$

Figura 3.16: Triângulos ABC e a elipse inscrita com focos em diversos pontos.

Exemplo 9 *Traçar, com o METAPOST, a elipse inscrita de Steiner de um triângulo retângulo isósceles.*

Dado um triângulo qualquer, o Teorema de Steiner garante a existência de uma única elipse inscrita que tangencia os lados do triângulo nos seus pontos médios. A elipse com essas características é chamada de Elipse Inscrita de Steiner e seu centro é o baricentro do triângulo. Este Teorema foi tema da dissertação de Nascimento [19].

Considere o plano $\mathbb{R}^2$ como o conjunto $\mathbb{C}$ dos números complexos, identificando $(x, y) \in \mathbb{R}^2$ com o número complexo $z = x + yi$. Suponha que os números complexos z_1, z_2 e z_3 sejam os vértices de um triângulo. Segundo Minda e Phelps [16], os focos da elipse inscrita de Steiner são as raízes da derivada do polinômio de coeficientes complexos e grau três,

$$p(z) = (z - z_1)(z - z_2)(z - z_3).$$

Como provado por Aniz e Nascimento [2], considerando o caso particular de um triângulo retângulo isósceles, as coordenadas dos focos da Elipse de Steiner são obtidas explicitamente como

$$F_1 = \left(\frac{\sqrt{2}d}{3}, \frac{d}{3} \right) \quad \text{e} \quad F_2 = \left(-\frac{\sqrt{2}d}{3}, \frac{d}{3} \right),$$

onde o sistema de eixos foi escolhido de forma que os vértices do triângulo sejam

$$z_1 = (0, d) = di, z_2 = (-d, 0) = -d, z_3 = (d, 0) = d.$$

Neste caso, com os focos conhecidos e sabendo que ela passará pelo ponto $(0, 0)$, ponto médio dos pontos z_2 e z_3, a elipse pode ser desenhada com o METAPOST, conforme mostrado na Figura 3.17.

```
pair A,B,C,F[],P,M[],M,G;
numeric u,a,b,c,d; path E;
transform T;
u=1.5cm; d=3u;
A:=(0,d); B:=(-d,0);
C:=(d,0);
draw A--B--C--cycle;
F1=(((sqrt 2)*d)/3, d/3);
F2=((-(sqrt 2)*d)/3, d/3);
P:=(0,0); M:=0.5[F1,F2];
2a=abs(F1-P)+abs(F2-P);
c=abs(F1-M); b=a+-+c;
T:= identity xscaled 2a
yscaled 2b shifted M;
E:= fullcircle
transformed T;
draw E withpen
pencircle scaled 1bp;
M1=(A+B)/2; M2=(A+C)/2;
M3=(B+C)/2; G=(A+B+C)/3;
draw F1--F2 withcolor red;
draw A--M3 dashed evenly;
draw B--M2 dashed evenly;
draw C--M1 dashed evenly;
```

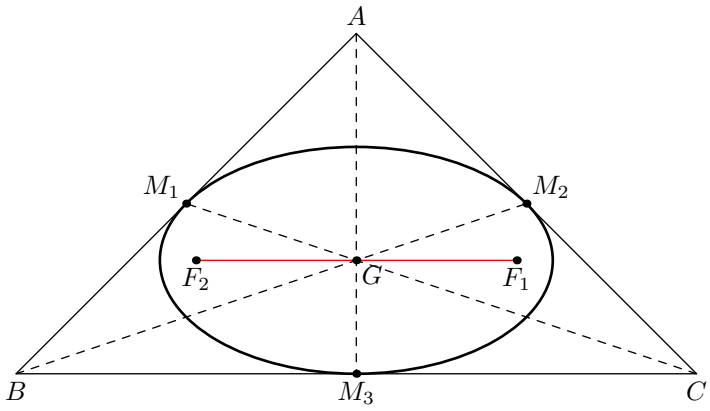


Figura 3.17: Elipse inscrita de Steiner num triângulo retângulo isósceles ABC

Capítulo 4

Funções Reais

Neste capítulo será discutida a elaboração de gráficos de funções contínuas definidas em intervalos da reta. Inicialmente é feito um resumo com algumas proposições sobre funções contínuas. Em seguida, são apresentadas algumas técnicas do METAPOST para esboçar gráficos de funções.

Definição 18 *Uma função real f é uma terna*

$$(A, B, a \mapsto b)$$

onde A e B são dois subconjuntos não vazios de $\mathbb{R}$ e $a \mapsto b$ é uma regra que associa a cada elemento de $a \in A$ um único elemento $b \in B$. O único $b \in B$ associado ao elemento $a \in A$ é indicado por $f(a)$.

Exemplo 10 *Considere a função $f : (0, +\infty) \to \mathbb{R}$, dada por $f(x) = \dfrac{1}{x}$. Nessa função, o conjunto de partida, chamado de domínio, é $(0, +\infty)$ e o conjunto de chegada, chamado de contradomínio, é $\mathbb{R}$. Essa função associa cada elemento de $(0, +\infty)$ ao seu inverso multiplicativo $\dfrac{1}{x}$. Os requisitos que definem uma função são satisfeitos nesse caso pois todos os elementos de $(0, +\infty)$ possuem um inverso e este é único. Essa função pode ser denotada da seguinte maneira:*

$$\begin{aligned} f : (0, +\infty) &\to \mathbb{R} \\ x &\mapsto f(x) \end{aligned}$$

Definição 19 *Uma função $f : X \to \mathbb{R}$ é contínua em um ponto $x_0 \in X$ se dado $\varepsilon > 0$,*

existe $\delta > 0$ *tal que*

$$x \in X, |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon. \tag{4.1}$$

A função f *é dita contínua se for contínua em todo* $x_0 \in X$*.*

Exemplo 11 *Dados* $X, Y \subset \mathbb{R}$ *e um valor fixo* $c \in Y$*, a função constante* c *de* X *em* Y *é a função* $f : X \to Y$ *tal que* $f(x) = c$ *para todo* $x \in X$*. Qualquer função constante é contínua.*

Demonstração: Sejam $c \in Y$ um valor fixo e $f : X \to Y$ a função constante dada por $f(x) = c$. Note que a segunda desigualdade de 4.1 é sempre verdadeira, pois $|f(x) - f(x_0)| = |c - c| = 0 < \varepsilon$. Portanto, tomando qualquer δ, tem-se que, se $x \in X$,

$$|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| = |c - c| = 0 < \varepsilon.$$

■

Exemplo 12 *Uma função afim é uma função* $f : \mathbb{R} \to \mathbb{R}$ *tal que* $f(x) = ax + b$ *para todo* x *real, onde* a *e* b *são números reais dados, com* $a \neq 0$*. As funções afins são contínuas.*

Demonstração: Se $f : \mathbb{R} \to \mathbb{R}$ é uma função afim, então $f(x) = ax + b$, para $a, b \in \mathbb{R}$ números reais dados com $a \neq 0$. Fixado $x_0 \in \mathbb{R}$, se $|x - x_0| < \delta$, então

$$|f(x) - f(x_0)| = |(ax + b) - (ax_0 + b)| = |a| \, |x - x_0| < |a| \, \delta.$$

Sendo assim, dado $\varepsilon > 0$, tomando $0 < \delta < \frac{\varepsilon}{|a|}$, tem-se:

$$|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| = |a| \, |x - x_0| < |a| \, \delta < |a| \, \tfrac{\varepsilon}{|a|} = \varepsilon.$$

Portanto, a função afim é contínua. ■

Exemplo 13 *A função real dada por* $f(x) = |x|$ *é contínua.*

Demonstração: Dado $x_0 \in \mathbb{R}$, se $|x - x_0| < \delta$, tem-se que

$$|f(x) - f(x_0)| = ||x| - |x_0|| \leqslant |x - x_0| < \delta.$$

Sendo assim, tomando $0 < \delta \leqslant \varepsilon$, pelas desigualdades anteriores segue que

$$|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon.$$

Portanto, f é contínua. ∎

Exemplo 14 *As funções $f, g : \mathbb{R} \to \mathbb{R}$, definidas por $f(x) = \operatorname{sen} x$ e $g(x) = \cos x$ são contínuas.*

Para demonstrar este resultado, será necessário um resultado que será enunciado e demonstrado a seguir.

Lema 1 *Para todo $x \in \mathbb{R}$, a desigualdade $|\operatorname{sen} x| \leqslant |x|$ é válida.*

Demonstração: Inicialmente, será mostrado que $\operatorname{sen} x \leqslant x$, para $0 \leqslant x \leqslant \dfrac{\pi}{2}$. Se $x = 0$, a desigualdade é óbvia, pois $\operatorname{sen} 0 = 0 \leqslant 0$. Sendo assim, para $0 < x \leqslant \dfrac{\pi}{2}$, considere na Figura 4.1 o ponto P marcado no primeiro quadrante do ciclo trigonométrico de forma que $\ell\left(\overset{\frown}{\mathrm{AP}}\right) = x$.

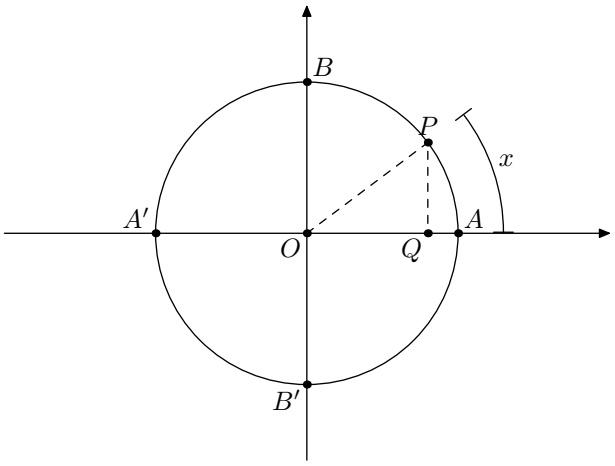


Figura 4.1: Ponto P no primeiro quadrante do ciclo trigonométrico

Considerando o ponto Q, pé da perpendicular baixada de P à reta $\overleftrightarrow{AA'}$, tem-se que

$$\operatorname{sen} x = \overline{PQ} < \ell\left(\overleftrightarrow{AP}\right) = x.$$

A função seno é ímpar, isto é, $\operatorname{sen}(-x) = -\operatorname{sen} x$. Assim, é imediato que $|\operatorname{sen} x| \leq |x|$ para $|x| \leq \frac{\pi}{2}$. Considerando agora o caso em que $|x| > \frac{\pi}{2}$, tem-se que

$$|\operatorname{sen} x| \leq 1 < \frac{\pi}{2} < |x|.$$

Portanto, $|\operatorname{sen} x| \leqslant |x|$ para todo $x \in \mathbb{R}$. ∎

Agora será demonstrado que as funções $f(x) = \operatorname{sen} x$ e $g(x) = \cos x$ são contínuas.

Demonstração: Seja $x_0 \in \mathbb{R}$ um valor fixo. Dado $\varepsilon > 0$, tome $0 < \delta < \varepsilon$ e veja que, pelas fórmulas de transformação em produto, tem-se

$$|\cos x - \cos x_0| = \left| -2 \cdot \operatorname{sen}\left(\frac{x - x_0}{2}\right) \cdot \operatorname{sen}\left(\frac{x + x_0}{2}\right) \right|,$$

donde obtém-se

$$\begin{aligned} |\cos x - \cos x_0| &= 2 \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \cdot \left|\operatorname{sen}\left(\frac{x + x_0}{2}\right)\right| \leq 2 \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \\ |\cos x - \cos x_0| &\leq 2 \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \end{aligned}$$

Pelo Lema 1, é possível concluir que $\left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \leq \left|\frac{x - x_0}{2}\right|$. Sendo assim,

$$\begin{aligned} |\cos x - \cos x_0| &\leq 2 \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \leq 2 \cdot \left|\frac{x - x_0}{2}\right| \\ |\cos x - \cos x_0| &\leq 2 \cdot \left|\frac{x - x_0}{2}\right| \\ |\cos x - \cos x_0| &\leq |x - x_0| \end{aligned}$$

Dessa maneira, se $|x - x_0| < \delta$, segue que $|\cos x - \cos x_0| \leq |x - x_0| < \delta < \varepsilon$. Logo, a função cosseno é contínua. Agora o mesmo será feito para a função seno. Dado $\varepsilon > 0$, tome $0 < \delta < \varepsilon$ e veja que, pelas fórmulas de transformação em produto e pelo Lema 1, tem-se

$$\begin{aligned} |\operatorname{sen} x - \operatorname{sen} x_0| &= \left| 2 \cdot \cos\left(\frac{x + x_0}{2}\right) \cdot \operatorname{sen}\left(\frac{x - x_0}{2}\right) \right| \\ |\operatorname{sen} x - \operatorname{sen} x_0| &= 2 \cdot \left|\cos\left(\frac{x + x_0}{2}\right)\right| \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \leq 2 \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \\ |\operatorname{sen} x - \operatorname{sen} x_0| &\leq 2 \cdot \left|\operatorname{sen}\left(\frac{x - x_0}{2}\right)\right| \leq 2 \cdot \left|\frac{x - x_0}{2}\right| \\ |\operatorname{sen} x - \operatorname{sen} x_0| &\leq 2 \cdot \left|\frac{x - x_0}{2}\right| \\ |\operatorname{sen} x - \operatorname{sen} x_0| &\leq |x - x_0| \end{aligned}$$

Dessa forma, se $|x - x_0| < \delta$, segue que $|\operatorname{sen} x - \operatorname{sen} x_0| \leq |x - x_0| < \delta < \varepsilon$. Logo, a

função seno é contínua.

Os resultados a seguir terão suas demonstrações omitidas deste trabalho mas podem ser conferidas em [17].

Proposição 5 *Considere I um intervalo da reta. Se $f, g : I \to \mathbb{R}$ são funções contínua, então $f \pm g$ e $f \cdot g$ também são contínuas em I.*

Proposição 6 *Considere I um intervalo da reta. Sejam $f, g : I \to \mathbb{R}$ são funções contínuas. Se g não se anula em I, então a função $\dfrac{f}{g}$ é contínua em I.*

Proposição 7 *Sejam I e J intervalos da reta. Se $f : I \to \mathbb{R}$ e $g : J \to \mathbb{R}$, com $f(I) = \{f(x) \mid x \in I\} \subset J$, são funções contínuas, então a função composta $g \circ f$, definida por $(g \circ f)(x) = g(f(x))$ é contínua em I.*

Exemplo 15 *A função $f(x) = \operatorname{tg}(x) = \dfrac{\operatorname{sen} x}{\cos x}$, definida nos intervalos onde a função $\operatorname{tg} x$ não se anula, é contínua em cada um desses intervalos. De fato, $f(x) = \operatorname{tg} x$ é quociente de funções contínuas, portanto contínua.*

Exemplo 16 *Considere um número natural $n > 0$. A função $f : \mathbb{R} \to \mathbb{R}$, definida por $f(x) = x^n$, denominada função potência, é contínua. De fato, $f(x) = x^n = \underbrace{x \cdot x \cdots x}_{n \text{ vezes}}$ é um produto de funções contínuas, portanto contínua.*

Exemplo 17 ***(Função Polinomial)***

Considere números reais $a_0, a_1, \cdots, a_n$ e um número natural n. A função $f : \mathbb{R} \to \mathbb{R}$, definida por $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, denominada função polinomial, é uma função contínua.

De fato, f é uma soma de um número finito de produtos do tipo $a_i \cdot x^i$, produto de uma função constante por uma função potência, logo f é soma finita de funções contínuas, portanto contínua.

Exemplo 18 *A função $f : \mathbb{R} \to \mathbb{R}$, definida por*

$$f(x) = |x^2 + x - 2 - \operatorname{sen}^2(3x^2 - 2x)|$$

é contínua. De fato, f é o resultado de somas e composições de funções contínuas.

4.1 Gráficos com o METAPOST

Neste capítulo são tratados alguns exemplos de funções contínuas. O foco agora será utilizar o METAPOST para traçar gráficos no plano cartesiano.

Definição 20 *Seja $f : A \to B$ uma função real. O gráfico da função f, indicado por G_f, é o subconjunto de $\mathbb{R}^2$ definido por*

$$G_f = \{(x, f(x)) \mid x \in A\}.$$

Para esboçar o gráfico de uma função é necessário marcar os pontos (x, y) que satisfazem $y = f(x)$. Além disso, um plano cartesiano precisa ser desenhado também. O código a seguir mostra como este pode ser desenhado entre os intervalos $[-4, 4]$ em x e em y. Para alterar os intervalos, basta mudar os valores mínimos e máximos de x e y no primeiro bloco de códigos.

```
% limites do desenho
u:=0.8cm; % unidade de medida
minx:=-4; % limite mínimo para o eixo x
maxx:=4; % limite máximo para o eixo x
miny:=-4; % limite mínimo para o eixo y
maxy:=4; % limite máximo para o eixo y

% plano cartesiano
drawarrow ((minx,0)--(maxx,0)) scaled u; % eixo x
drawarrow ((0,miny)--(0,maxy)) scaled u; % eixo y
label.bot (btex $x$ etex, (maxx*u,0)); % rótulo do eixo x
label.llft (btex $y$ etex, (0,maxy*u)); % rótulo do eixo y
```

Nos exemplos que serão mostrados a seguir, este código que desenha o plano cartesiano, bem como os códigos de rotulação e linhas tracejadas, serão omitidos para que se foque apenas no código para os gráficos das funções.

Exemplo 19 *Desenhar o gráfico da função $f(x) = x$ no intervalo $[0, 3]$.*

A Figura 4.2 mostra este gráfico e o código utilizado para gerá-lo. Note que primeiramente é declarada uma variável do tipo `path` para representar a reta do gráfico. O número `n` refere-se a quantidade de subintervalos em que o intervalo $[0, 3]$ será dividido. As variáveis numéricas `a` e `b` se referem ao intervalo em que o gráfico será desenhado. A variável `h` refere-se ao passo dado a partir de `a` para calcular os outros pontos do gráfico.

Definidas as variáveis a serem utilizadas no gráfico é necessário definir as coordenadas dos pontos que o METAPOST calculará para fazer o desenho. Os `n+1` pontos a serem calculados terão, para `i=0, 1,..., n`, abscissa `x[i]:=a+i*h` e ordenada `y[i]:=x[i]`, pois a função a ser esboçada é $y = x$. Para definir a curva do gráfico, `p[1]`, é interessante analisar a linha de comandos

```
p[1]:=z[0] for j=1 upto n:  ..z[j] endfor;
```

utilizada para isso. Esta linha é uma forma mais rápida e prática de escrever

```
p[1]:=z[0]..z[1]..z[2]..z[3]..z[4]..z[5]..z[6]..z[7]..z[8];.
```

Esse laço é prático pois para valores altos de `n` é inviável escrever ponto por ponto. Com este caminho definido, basta um comando `draw` para desenhar o gráfico.

```
path p[];
n:=8;
a:=0; b:=3;
h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=x[i]; endfor
p[1]:=z[0] for j=1 upto n:..z[j] endfor;
pickup pencircle scaled 1bp;
draw p[1] scaled u withcolor red;
```

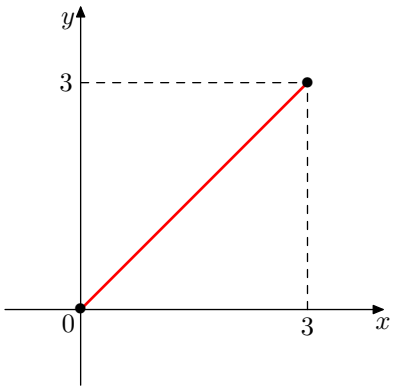


Figura 4.2: Gráfico da função $f(x) = x$.

Exemplo 20 *Desenhar o gráfico da função* $f(x) = x^2$ *no intervalo* $[-2, 2]$.

```
path p[];
n:=80;
a:=-2; b:=2;
h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=x[i]**2; endfor
p[1]:=z[0] for j=1 upto n:..z[j] endfor;
pickup pencircle scaled 1bp;
draw p[1] scaled u withcolor red;
```

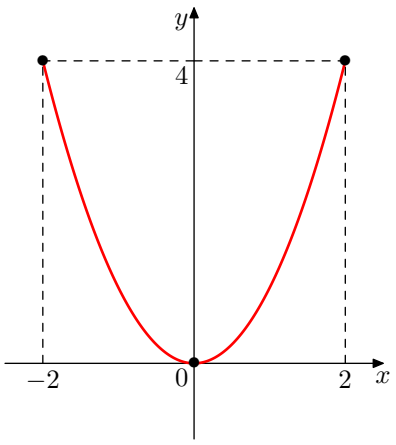


Figura 4.3: Gráfico da função $f(x) = x^2$.

4.2 Limitando eixos

Exemplo 21 *Desenhar o gráfico da função* $f(x) = x^3$.

Quando o gráfico de uma função extrapola o limite imposto ao traçar os eixos do plano cartesiano, se você não utilizar um comando que corte parte da curva, o METAPOST traçará o gráfico inteiro e o resultado não será o desejado. Uma maneira de evitar essas extrapolação é utilizar os comandos `cutbefore` e `cutafter`.

```
path p[], q;
q:=((minx,miny)--(maxx,miny)--
(maxx,maxy)--(minx,maxy)--cycle) scaled 0.95;
n:=100;
a:=-2; b:=2;
h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=x[i]**3; endfor
p[1]:=z[0] for j=1 upto n:..z[j] endfor;
draw q scaled u withcolor 0.5white dashed evenly;
pickup pencircle scaled 1bp;
draw p[1] scaled u withcolor red;
```

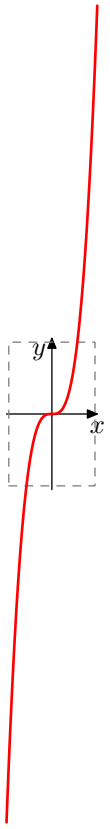


Figura 4.4: Função $f(x) = x^3$ traçada com os limites dos eixos extrapolados

Na Figura 4.4, o retângulo cinza tracejado é o caminho `q` do código. Com o comando `p[1] cutbefore q`, o METAPOST retira de `p[1]` o que está para fora desse caminho do lado esquerdo. O comando `p[1] cutafter q` faz o mesmo para o excedente à direita. Assim, o resultado de se usar ambos os comandos pode ser visto na Figura 4.5.

```
path p[], q;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)
--(minx,maxy)--cycle) scaled 0.95;
n:=100;
a:=-2; b:=2;
h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=x[i]**3; endfor
p[1]:=z[0] for j=1 upto n:..z[j] endfor;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q)
scaled u withcolor red;
```

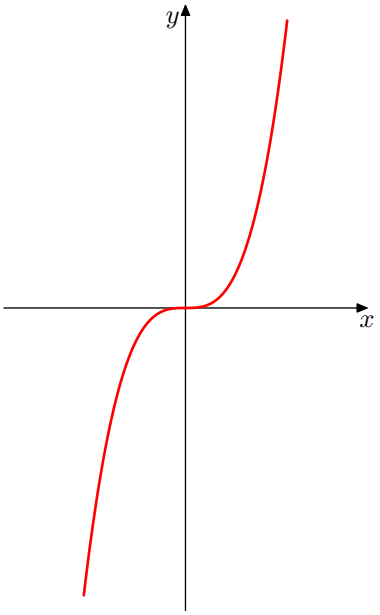


Figura 4.5: Função $f(x) = x^3$

Exemplo 22 *Desenhar o gráfico da função* $f(x) = x^7 - 2x^5 - 3x^4 + 4x^2 + 1$.

```
path p[], q;
q:=((minx,miny)--(maxx,miny)--
(maxx,maxy)--(minx,maxy)--cycle)
scaled 0.95;
n:=100;
a:=-2; b:=2;
h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=x[i]**7-2*(x[i]**5)
-3*(x[i]**4)+4*(x[i]**2)+1;
endfor
p[1]:= z[0] for j=1 upto n:..z[j]
endfor;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore
q) scaled u withcolor red;
```

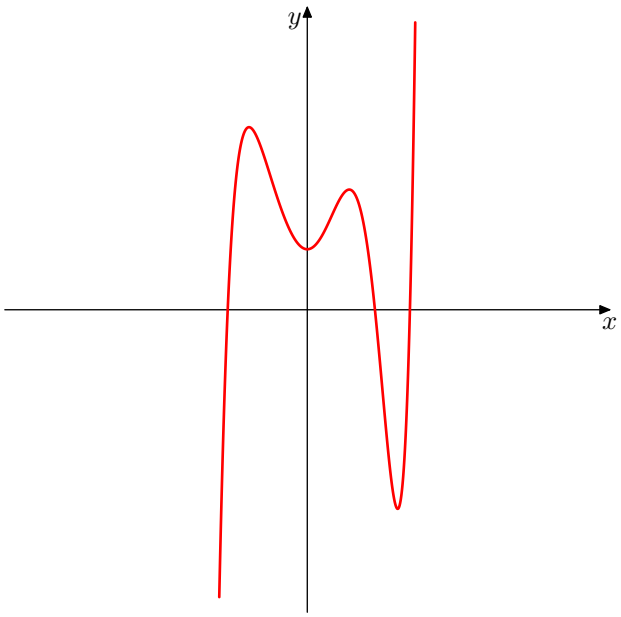


Figura 4.6: Função $f(x) = x^7 - 2x^5 - 3x^4 + 4x^2 + 1$

Funções com módulo podem ser escritas com o operador `abs`, como mostra o exemplo a seguir.

Exemplo 23 *Desenhar o gráfico da função* $f(x) = |x^2 - 4|$.

```
path p[], q;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)
--(minx,maxy)--cycle) scaled 0.95;
n:=100;
a:=-2; b:=2;
h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=abs(x[i]**2-4); endfor
p[1]:= z[0] for j=1 upto n:..z[j] endfor;
draw q withcolor 0.5white dashed evenly;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q)
scaled u withcolor red;
```

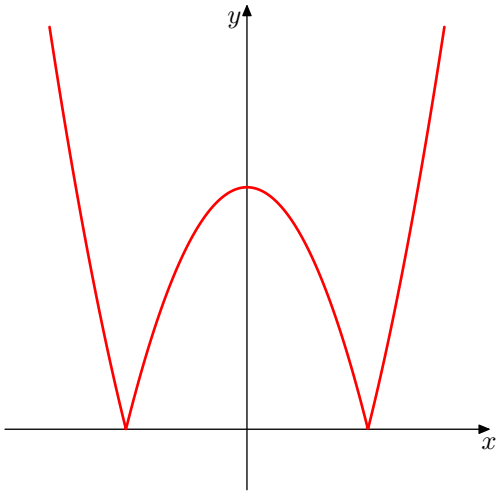


Figura 4.7: Função $f(x) = |x^2 - 4|$

Quando o gráfico de uma função precisa ser separado em partes para ser desenhado, basta gerar um caminho `p` para cada parte. Veja o exemplo a seguir.

Exemplo 24 *Desenhar o gráfico da função* $f(x) = \dfrac{1}{x}$.

Essa função não é definida em $x = 0$. Portanto, é preciso traçar a porção do gráfico em que $x < 0$ e a outra em que $x > 0$. Para a primeira curva `p[1]` do gráfico, o intervalo utilizado foi `a:=minx` (que é definido inicialmente como o valor mínimo de x no plano que está sendo utilizado) e `b:=-0.1`. Esse valor para `b` não pode ser 0 porque o METAPOST não seria capaz de calcular o último ponto da curva. A segunda curva, `p[2]`, foi definida com os valores de `a:=0.1` e `b:=maxx`. O código e o desenho resultante estão mostrados na Figura 4.8.

```
path p[], q;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)
--(minx,maxy)--cycle) scaled 0.95;
n:=1000;
a:=minx; b:=-0.1; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=1/x[i]; endfor;
p[1]=z[0] for j=1 upto n: ..z[j] endfor;
a:=0.1; b:=maxx; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=1/x[i]; endfor;
p[2]=z[0] for j=1 upto n: ..z[j] endfor;
draw q withcolor 0.5white dashed evenly;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q)
scaled u withcolor red;
```

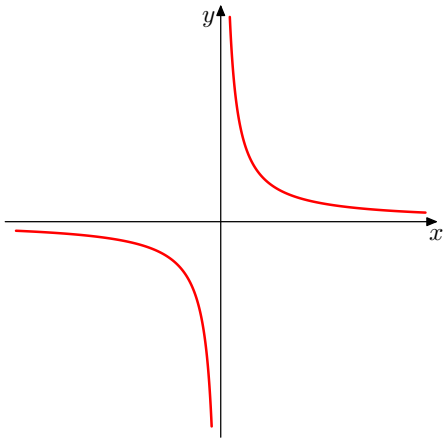


Figura 4.8: Função $f(x) = \dfrac{1}{x}$

4.3 Macros

Os recursos do METAPOST descritos até então fazem parte de um pacote de macros simples predefinido do METAPOST chamado Plain. O objetivo desta seção é explicar como escrever macros para criar outras operações e números úteis para o esboço de gráficos de funções reais.

Uma definição macro do tipo

`def` ⟨token simbólico⟩ `=` ⟨texto de substituição⟩ `enddef;`

faz com que o ⟨token simbólico⟩ vire uma abreviação para o ⟨texto de substituição⟩, onde o ⟨texto de substituição⟩ pode ser virtualmente qualquer sequência de comandos, definições, etc. Por exemplo, o número π não é predefinido no METAPOST. Porém, é possível com uma macro definir que toda vez que "`pi`" for escrito, ele interprete como um valor aproximado de π. Isso é prático pois se o usuário fosse definir `pi` como uma variável numérica, seria necessário defini-la novamente a cada figura de um documento. No entanto, utilizar uma macro como

```
def pi = 3.1415926536 enddef;
```

já deixa definido para todas as figuras do documento em questão que "`pi`" é um valor aproximado de π.

O METAPOST possui o comando `arclength` que calcula o comprimento de um caminho. Como $\pi = \dfrac{C}{d}$, onde C é o comprimento de um círculo qualquer e d o seu diâmetro, em vez de se definir π por meio de uma aproximação, é possível defini-lo como o comprimento do círculo predefinido `fullcircle`, uma vez que seu diâmetro é unitário. A sintaxe usada para isso é

```
def pi = arclength fullcircle enddef;
```

As macros com argumentos são semelhantes, com a diferença de terem parâmetros formais que informam como usar os argumentos no ⟨texto de substituição⟩. Por exemplo, na Seção 1.6 foi citado que, apesar de o METAPOST ter em seu pacote simples de macros os operadores `sind` e `cosd` para calcular, respectivamente, o seno e cosseno de ângulos em graus, é possível criar uma macro para calcular essas razões trigonométricas em radianos.

Como π rad equivalem a 180°, uma macro que defina um comando novo `sin(a)`, em que `a` é um número real (em radianos), pode ser escrita assim:

```
def sin(expr a) = sind(a*pi/180) enddef;
```

A palavra `expr` nesta definição significa que o parâmetro `a` pode ser qualquer expressão arbitrária. Essas definições de macros são práticas para criar comandos que não estão predefinidos no METAPOST.

Para definir novas funções e operações matemáticas, é recomendável utilizar macros do tipo `vardef`, que trabalha diretamenta com valores numéricos. A lista de funções matemáticas a seguir foi adaptada da criada por Heck [7].

```
vardef sqr primary x = (x*x) enddef;
vardef log primary x = (if x=0: 0 else: mlog(x)/mlog(10) fi) enddef;
vardef ln primary x = (if x=0: 0 else: mlog(x)/256 fi) enddef;
vardef exp primary x = ((mexp 256)**x) enddef;
vardef inv primary x = (if x=0: 0 else: x**-1 fi) enddef;
vardef pow (expr x,p) = (x**p) enddef;

% funções trigonométricas
numeric pi; pi := 3.1415926;
numeric radian; radian := 180/pi;
vardef tand primary x = (sind(x)/cosd(x)) enddef;
vardef cotd primary x = (cosd(x)/sind(x)) enddef;
vardef sen primary x = (sind(x*radian)) enddef;
vardef cos primary x = (cosd(x*radian)) enddef;
vardef tan primary x = (sen(x)/cos(x)) enddef;
vardef cot primary x = (cos(x)/sen(x)) enddef;

% funções hiperbólicas
vardef senh primary x = save xx; xx = exp xx; (xx-1/xx)/2 enddef;
vardef cosh primary x = save xx; xx = exp xx; (xx+1/xx)/2 enddef;
vardef tanh primary x = (senh(x)/cosh(x)) enddef;
vardef coth primary x = (cosh(x)/senh(x)) enddef;

% inversas das funções trigonométricas e hiperbólicas
vardef arcsend primary x = angle((1+-+x,x)) enddef;
vardef arccosd primary x = angle((x,1+-+x)) enddef;
vardef arcsen primary x = ((arcsend(x))/radian) enddef;
vardef arccos primary x = ((arccosd(x))/radian) enddef;
vardef arccosh primary x = ln(x+(x+-+1)) enddef;
vardef arcsenh primary x = ln(x+(x++1)) enddef;
```

Os exemplos que seguem utilizam essas macros.

Exemplo 25 *Desenhar o gráfico da função* $f(x) = \operatorname{sen} x$ *no intervalo* $[0, 2\pi]$.

```
def pi = arclength fullcircle enddef;
numeric radian; radian := 180/pi;
vardef sen primary x =
(sind(x*radian)) enddef;
path p[], q;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)
--(minx,maxy)--cycle) scaled 0.95;
n:=1000;
a:=0; b:=2*pi; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=sen x[i]; endfor;
p[1]=z[0] for j=1 upto n: ..z[j] endfor;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q)
scaled u withcolor red;
```

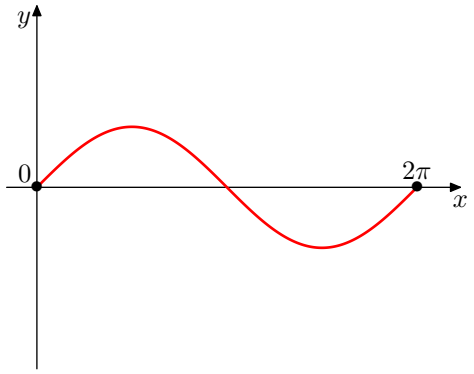


Figura 4.9: Função $f(x) = \operatorname{sen} x$

Exemplo 26 *Desenhar o gráfico da função* $f(x) = \dfrac{\cos 10x}{x}$.

```
def pi = arclength fullcircle enddef;
numeric radian; radian := 180/pi;
vardef cos primary x =
(cosd(x*radian)) enddef;
path p[], q;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)
--(minx,maxy)--cycle) scaled 0.95;
n:=1000;
a:=0.1; b:=maxx; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=(cos (10*x[i]))/x[i]; endfor;
p[1]=z[0] for j=1 upto n: ..z[j] endfor;
a:=minx; b:=-0.1; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=(cos (10*x[i]))/x[i]; endfor;
p[2]=z[0] for j=1 upto n: ..z[j] endfor;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q)
scaled u withcolor red;
draw ((p[2] cutafter q) cutbefore q)
scaled u withcolor red;
```

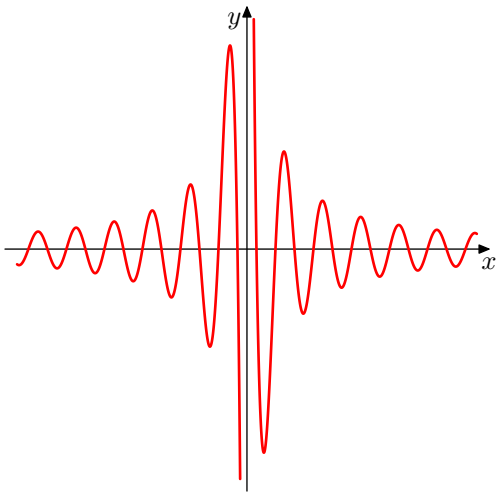


Figura 4.10: Função $f(x) = \dfrac{\cos 10x}{x}$

Exemplo 27 *Desenhar, em um mesmo plano cartesiano, os gráficos das funções* $f(x) = 1 + \sqrt{1 - |x-1|^2}$ *e* $g(x) = 1 + \arccos(1 - |x|) - \pi$ *no intervalo* $[-2, 2]$.

```
def pi = arclength fullcircle enddef;
numeric radian; radian := 180/pi;
vardef arccosd primary
x = angle((x,1+-+x)) enddef;
vardef arccos primary
x = ((arccosd(x))/radian) enddef;
path p[], q;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)
--(minx,maxy)--cycle) scaled 0.95;
n:=1000;
a:=-2; b:=2; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=1+sqrt(1-(abs(x[i])-1)**2); endfor;
p[1]=z[0] for j=1 upto n: ..z[j] endfor;
for i=0 upto n: x[i]:=a+i*h;
y[i]:=1+arccos(1-abs(x[i]))-pi; endfor;
p[2]=z[0] for j=1 upto n: ..z[j] endfor;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q)
scaled u withcolor red;
draw ((p[2] cutafter q) cutbefore q)
scaled u withcolor 0.6red;
```

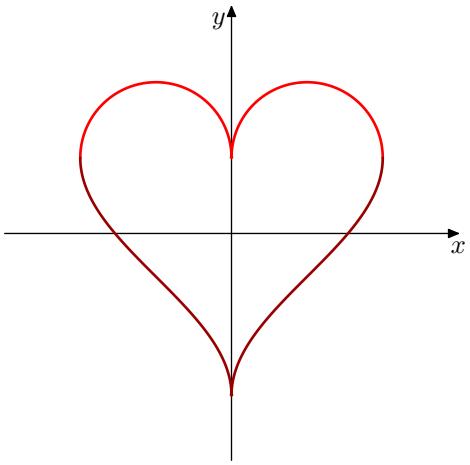


Figura 4.11: Funções $f(x) = 1 + \sqrt{1 - |x-1|^2}$ e $g(x) = 1 + \arccos(1 - |x|) - \pi$

Este capítulo será finalizado com algumas figuras apresentadas sem discussão das ferramentas diferentes utilizadas, visando apenas mostrar a pontecialidade do METAPOST quando se trata de gráficos de funções.

```
% variáveis e macros
vardef ln primary x = (if x=0: 0 else: mlog(x)/256 fi) enddef;
numeric u, minx, maxx, miny, maxy; path q;
numeric h,n; path p[ ];
path hline[], vline[], int;
% limites gráficos
u:=1.5cm; minx:=-1; maxx:=6; miny:=-1; maxy:=3;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)--(minx,maxy)--cycle) scaled 0.95;
% gráfico
n:=1000; a:=0.1; b:=maxx; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h; y[i]:=1+ln(x[i]); endfor;
p[1]=z[0] for j=1 upto n: ..z[j] endfor;
% integral
hline0 = (0,0)--(maxx,0);
vline0 = (0,0)--(0,maxy);
vline0.5 = (1,0)--(1,maxy);
vline4 = (5,0)--(5,maxx);
int = buildcycle(hline0, vline0.5, p[1], vline4);
% desenhos
fill int scaled u withcolor 0.8[blue,white];
draw ((1,0)--vline0.5 intersectionpoint p[1]) scaled u;
draw ((5,0)--vline4 intersectionpoint p[1]) scaled u;
drawarrow ((minx,0)--(maxx,0)) scaled u;
drawarrow ((0,miny)--(0,maxy)) scaled u;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q) scaled u withcolor red;
% rótulos
label.bot (btex $x$ etex, (maxx*u,0));
label.llft (btex $y$ etex, (0,maxy*u));
label.ulft(btex $f$ etex, z[1000]*u);
label(btex $\int\limits_a^b{f(x)}\;dx$ etex, center(int scaled u));
label.bot(btex $a$ etex, (u,0));
label.bot(btex $b$ etex, (5u,0));
```

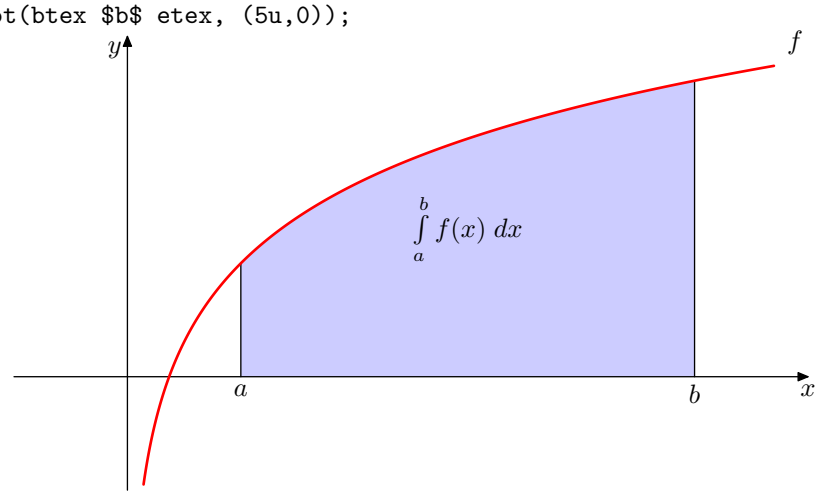


Figura 4.12: Representação gráfica de $\int_a^b f(x)\,dx$ com $f(x) = 1 + \ln x$

```
% variáveis e macros
numeric u, minx, maxx, miny, maxy; path q, c[];
numeric h,n; path p[ ];
path hline[], vline[], int; picture lab[];
vardef ln primary x = (if x=0: 0 else: mlog(x)/256 fi) enddef;
% limites gráficos
u:=1.5cm; minx:=-1; maxx:=6; miny:=-1; maxy:=3;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)--(minx,maxy)--cycle) scaled 0.95;
% gráfico
n:=1000; a:=0.1; b:=maxx; h:=(b-a)/n;
for i=0 upto n: x[i]:=a+i*h; y[i]:=1+ln(x[i]); endfor;
p[1]=z[0] for j=1 upto n: ..z[j] endfor;
% integral
c0 = -500*dir(50)--500*dir(50);
for i=0 upto 300: draw c0 shifted (0,5*i);
draw c0 shifted (0,-5*i); endfor;
hline0 = (0,0)--(maxx,0);
vline0 = (0,0)--(0,maxy);
vline0.5 = (1,0)--(1,maxy);
vline4 = (5,0)--(5,maxx);
int = buildcycle(hline0, vline0.5, p[1], vline4);
clip currentpicture to int scaled u;
% desenhos
draw ((1,0)--vline0.5 intersectionpoint p[1]) scaled u;
draw ((5,0)--vline4 intersectionpoint p[1]) scaled u;
drawarrow ((minx,0)--(maxx,0)) scaled u;
drawarrow ((0,miny)--(0,maxy)) scaled u;
pickup pencircle scaled 1bp;
draw ((p[1] cutafter q) cutbefore q) scaled u withcolor red;
% rótulos
label.bot (btex $x$ etex, (maxx*u,0));
label.llft (btex $y$ etex, (0,maxy*u));
label.ulft(btex $f$ etex, z[1000]*u);
label.bot(btex $a$ etex, (u,0));
label.bot(btex $b$ etex, (5u,0));
lab0:=thelabel(btex $\int\limits_a^b{f(x)}\;dx$ etex, center(int scaled u));
unfill bbox lab0; draw lab0;
```

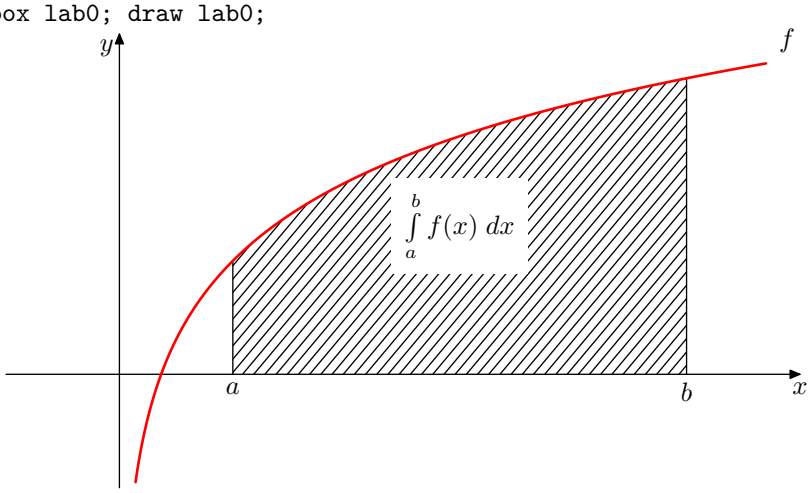


Figura 4.13: Outra representação gráfica de $\int_a^b f(x)\,dx$ com $f(x) = 1 + \ln x$

```
% variáveis e macros
numeric u, minx, maxx, miny, maxy; path p[];
% limites gráficos
u:=1.5cm; minx:=-1; maxx:=6; miny:=-1; maxy:=3;
% gráfico
p[1]:=(-1u,-u){right}..{dir 60}(2u,.5u)..(4u,1u){right}..(6u,3u);
drawarrow ((minx,0)--(maxx,0)) scaled u;
drawarrow ((0,miny)--(0,maxy)) scaled u;
draw p[1] withpen pencircle scaled 1bp withcolor red;
% método
x0 = 5.8*u; numeric t[];
for i=0 upto 2: (t[i],whatever) = p[1] intersectiontimes ((x[i],-infinity)--(x[i],infinity));
z[i] = point t[i] of p[1];
(x[i+1],0) = z[i] + whatever*direction t[i] of p[1];
draw (x[i],0)--z[i]--(x[i+1],0);
fill fullcircle scaled 3.6pt shifted z[i];
endfor;
% rótulos
label.urt (btex $x$ etex, (maxx*u,0));
label.llft (btex $y$ etex, (0,maxy*u));
label.bot(btex $x_0$ etex, (x0,0));
label.bot(btex $x_1$ etex, (x1,0));
label.bot(btex $x_2$ etex, (x2,0));
label.bot(btex $x_3$ etex, (x3,0));
```

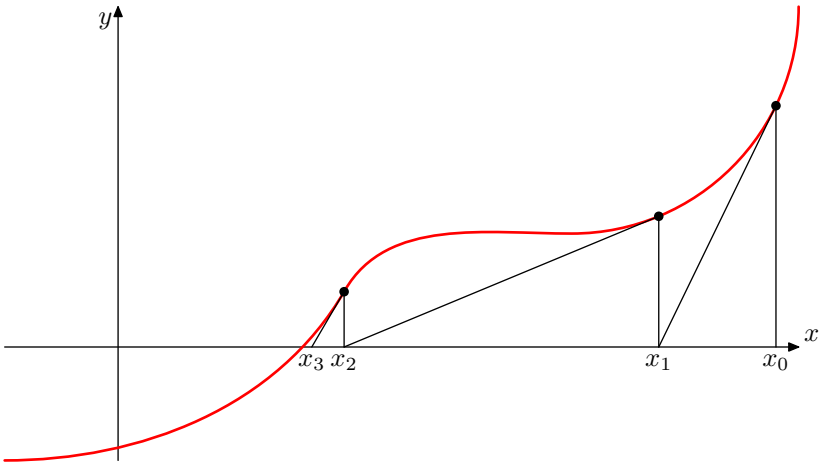


Figura 4.14: Alguns passos do Método Iterativo de Newton para encontrar zeros

```
% variáveis e macros
pair t; path p[], q, r[], s[]; def f(expr x) = (4-x**2) enddef;
% limites gráficos
u:=1cm; minx:=-3; maxx:=3; miny:=-1; maxy:=5;
q:=((minx,miny)--(maxx,miny)--(maxx,maxy)--(minx,maxy)--cycle) scaled 0.95u;
% retângulos
n:= 16; x0:= -2; x1:= 2; inc:= (x1-x0)/n; for i=x0 step inc until x1-inc:
r[i] = (i,0)--(i+inc,0)--(i+inc,max(f(i),f(i+inc)))--(i, max(f(i),f(i+inc)))--cycle;
r[i]:= r[i] scaled u; fill r[i] withcolor 0.8white; draw r[i]; endfor;
% gráfico
inc := 0.01;
p[1]:= (minx*u,f(minx)*u) for x=minx+inc step inc until maxx: ..(x*u,f(x)*u) endfor;
drawarrow ((minx,0)--(maxx,0)) scaled u; drawarrow ((0,miny)--(0,maxy)) scaled u;
draw ((p[1] cutafter q) cutbefore q) withpen pencircle scaled 1bp withcolor red;
% translação
t := (6.5u,0); inc:= (x1-x0)/n; for i=x0 step inc until x1-inc:
s[i] = (i,0)--(i+inc,0)--(i+inc,min(f(i),f(i+inc)))--(i, min(f(i),f(i+inc)))--cycle;
s[i] := s[i] scaled u shifted t; fill s[i] withcolor 0.8white; draw s[i]; endfor;
drawarrow ((minx,0)--(maxx,0)) scaled u shifted t;
drawarrow ((0,miny)--(0,maxy)) scaled u shifted t;
draw ((p[1] cutafter q) cutbefore q) shifted t withpen pencircle scaled 1bp withcolor red;
% rótulos
label.urt (btex $x$ etex, (maxx*u,0)); label.llft (btex $y$ etex, (0,maxy*u));
label.urt (btex $x$ etex, t+(maxx*u,0)); label.llft (btex $y$ etex, t+(0,maxy*u));
```

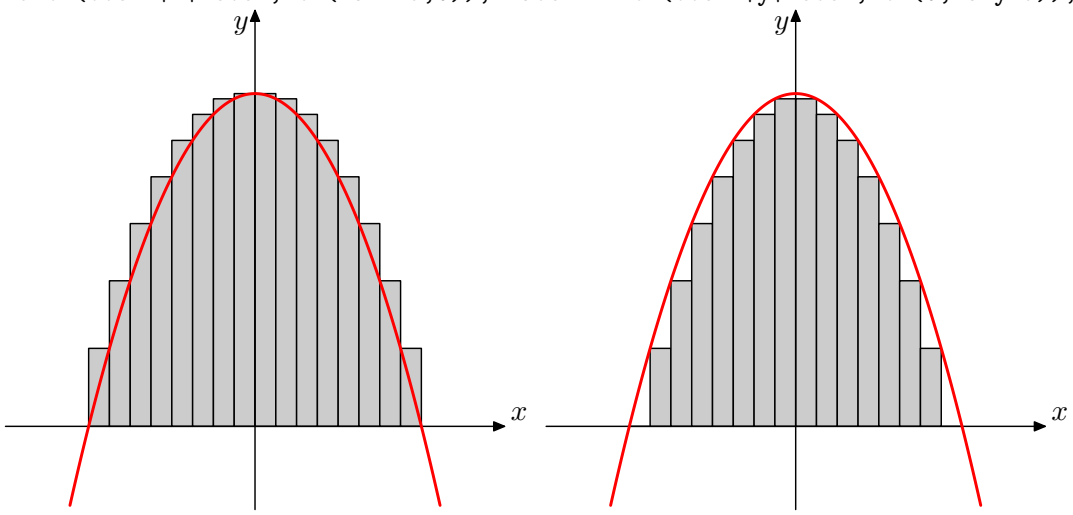


Figura 4.15: Ilustração da Soma de Riemann Superior e Inferior da função $f(x) = 4 - x^2$

Considerações Finais

O METAPOST é uma ferramenta poderosa para se usar conhecimentos Matemáticos na elaboração de figuras. As imagens geradas por meio dessa linguagem têm o nível profissional satisfatório para o que é esperado de um trabalho acadêmico. Pessoas que ensinam e pesquisam Matemática poderão melhorar a apresentação de seus trabalhos e orientar seus alunos na aplicação dos conhecimentos matemáticos adquiridos. Assim, espera-se que esse trabalho possa, de certa forma, ter contribuído para a formação de professores de Matemática e, consequentemente, para a formação de seus alunos.

Essa ferramenta não é muito conhecida em ambientes não acadêmicos pois foi feita, primariamente, para ser utilizada juntamente com o LaTeX. Como o METAPOST é pioneiro na criação de imagens vetoriais, outras linguagens já foram desenvolvidas com algumas semelhanças e mais funcionalidades, como PGF/Ti*k*Z, PSTricks e Asymptote. Mesmo assim, a sua praticidade e simplicidade de comandos não a torna antiquada.

Os itens a seguir podem ser pesquisados para trabalhos futuros.

- Hobby [12] apresenta um pacote de macros utilizados especialmente para diversos gráficos estatísticos.

- Henkel [10] fez um estudo sobre o método numérico utilizado pelo METAPOST para o cálculo do comprimento de curvas cúbicas de Bézier por meio de integrais elípticas e concluiu que o método vale a pena, mas precisa de mais pesquisa matemática para ser mais robusto em alguns casos especiais.

- Roegel [23] estudou as funcionalidades do pacote de macros `3d` para a Geometria Espacial.

- Oswald [20] mostra como fazer diversas figuras com o METAPOST, incluindo parametrização de curvas, caminhos recursivos e gráficos tridimensionais, como os da Figura 4.16.

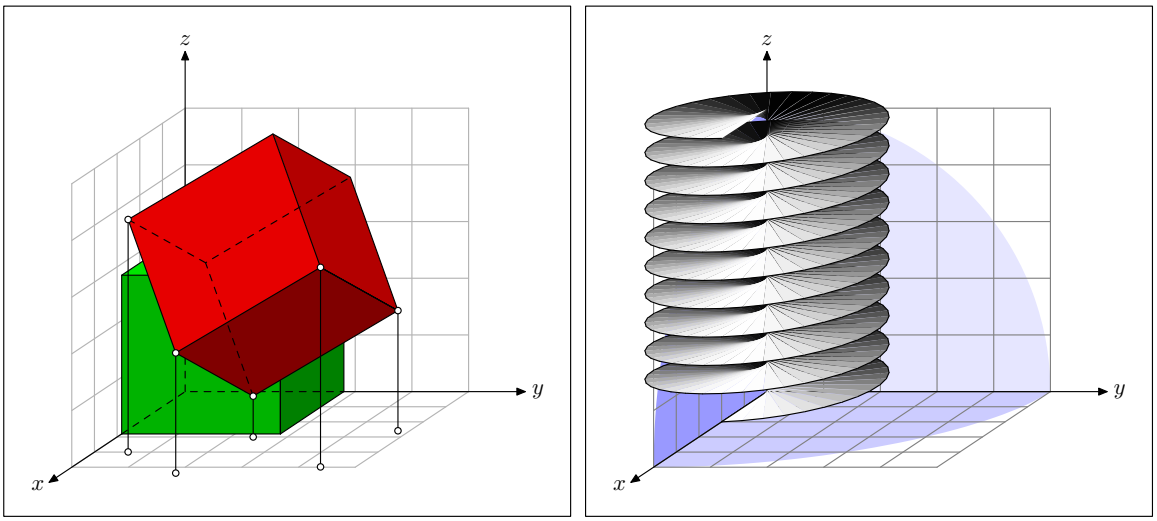


Figura 4.16: Figuras tridimensionais.

- Heck [7] mostra exemplos de campos vetoriais, como mostrado na Figura 4.17.

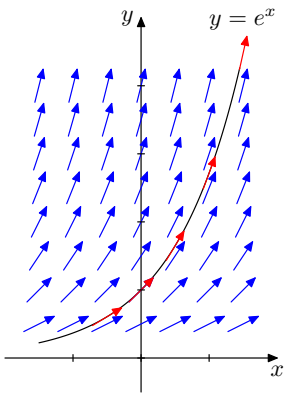


Figura 4.17: Campo direcional correspondente à equação diferencial ordinária $y' = y$

- Hagen [6] utiliza o METAPOST para criar esquematizações de quebra-cabeças (como na Figura 4.18) e busca criar uma biblioteca de macros que permitam desenhar ilustrações matemáticas.

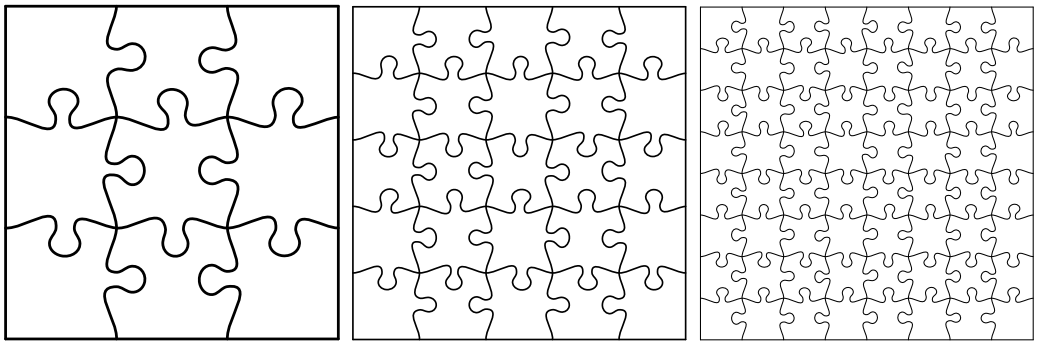


Figura 4.18: Quebra-cabeças.

Referências Bibliográficas

[1] *ALVES, Sergio.* **Elipses inscritas num triângulo**. Revista do Professor de Matemática, v. 96, 2018.

[2] *ANIZ, Claudemir; NASCIMENTO, Luiz H. A. S.* **A elipse de Steiner**. Revista do Professor de Matemática, v. 82, 2013.

[3] *BEEBE, Nelson.* **The design of TEX and METAFONT: A retrospective**. Disponível em `http://www.math.utah.edu/~beebe/talks/2005/tug2005/pt2005.pdf`; acesso em 30/09/2018.

[4] *BRIANCHON, Charles J. PONCELET, Jean-Victor.* **Recherche sur la détermination d'une hyperbole équilatère au moyen de quatre conditions données**. Annales de Mathématiques pures et Appliquées, 1821.

[5] *DELGADO, Jorge.* **Geometria Analítica** - Jorge Delgado, Katia Frensel, Lhaylla Crissaff. 1.ed. Rio de Janeiro: SBM, 2013.

[6] *HAGEN, Hans.* **Puzzling graphics in METAPOST**. PRAGMA Advanced Document Engineering, 1997. Disponível em `http://www.pragma-ade.com/articles/art-puzz.pdf`; acesso em 30/09/2018.

[7] *HECK, André.* **Tutorial in MetaPost**. AMSTEL Institute, 2003, disponível em `http://tex.loria.fr/prod-graph/heck-metapost2003.pdf`; acesso em 30/09/2018.

[8] *HENDERSON, Troy; HENNIG, Stephan.* **A Beginner's Guide to METAPOST for Creating High-Quality Graphics**. 2013. Disponível em `http://www.tug.org/docs/metapost/mpintro.pdf`; acesso em 30/09/2018.

[9] *HENDERSON, Troy.* **MetaPost Previewer**. Disponível em `http://www.tlhiv.org/mppreview/`; acesso em 30/09/2018.

[10] *HENKEL, Harmut.* **Calculating the Cubic Bézier Arc Length by Elliptic Integrals**. Oftersheim, Alemanha. 2014. Disponível em `http://www.circuitwizard.de/metapost/arclength.pdf`; acesso em 30/09/2018.

[11] *HOBBY, John D.* **Digitized Brush Trajectories**. Tese de Doutorado, Department of Computer Science, Stanford University, Stanford, CA, USA, 1986.

[12] *HOBBY, John D.* **Drawing Graphs with METAPOST**. Technical Report 164, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Disponível em `https://www.tug.org/docs/metapost/mpgraph.pdf`; acesso em 30/09/2018.

[13] *HOBBY, John D.* **METAPOST - A USER'S MANUAL**. Technical Report 162, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Disponível em `http://www.tug.org/docs/metapost/mpman.pdf`; acesso em 30/09/2018.

[14] *KNUTH, Donald E.* **Fundamental Algorithms**. Volume 1 of The Art of Computer Programming. Addison-Wesley, Reading, MA, USA, 1968.

[15] *LIANG, Frank.* **Word hy-phen-a-tion by compu-ter**. Tese de Doutorado, Computer Science Department, Stanford University, Stanford, CA, USA, 1984.

[16] *MINDA, David. PHELPS, Steve.* **Triangles, Ellipses, and Cubic Polynomials**. The American Mathematical Monthly, Vol. 115, No. 8, p. 679-689, 2008.

[17] *MUNIZ NETO, Antonio C.* **Fundamentos de Cálculo**. 1.ed. Rio de Janeiro: SBM, 2015.

[18] *MUNIZ NETO, Antonio C.* **Geometria**. 1.ed. Rio de Janeiro: SBM, 2013.

[19] *NASCIMENTO, Luiz H. A. S.* **Elipse Inscrita de Steiner**. Dissertação de Mestrado, PROFMAT, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brasil, 2017.

[20] *OSWALD, Urs.* **MetaPost: A Very Brief Tutorial**. Disponível em `http://www.ursoswald.ch/metapost/tutorial.pdf`; acesso em 30/09/2018.

[21] *PEDROSA, Israel.* **Da Cor à Cor Inexistente**. Rio de Janeiro: Editora Universidade de Brasília, 1982.

[22] *PLASS, Michael.* **Optimal pagination techniques for automatic typesetting systems**. Tese de Doutorado, Computer Science Department, Stanford University, Stanford, CA, USA, 1981.

[23] *ROEGEL, Denis.* **La géométrie dans l'espace avec METAPOST**. Cahiers GUTenberg nº 39–40, pp. 107-138, 2001. Disponível em `https://members.loria.fr/DRoegel/TeX/39-roegel.pdf`; acesso em 30/09/2018.

[24] *RUGGLES, Lynn E.* **Paragon, an interactive, extensible, environment for typeface design**. Tese de Doutorado, University of Massachusetts Amherst, Amherst, MA, USA, 1987.

[25] *SALELLES, Ignacio A. Z.* **Interfacing with graphics objects**. Tese de Doutorado, Department of Computer Science, Stanford University, Stanford, CA, USA, 1982.

[26] *VICENTE, Lucas.* **Escrevendo Imagens com MetaPost**. Disponível em `http://www.ime.usp.br/~alkaid/metapost.pdf`; acesso em 30/09/2018.

[27] *VIETH, Ulrik.* **The `mflogo` package**. Disponível em `http://mirrors.ctan.org/macros/latex/contrib/mflogo/mflogo.pdf`; acesso em 30/09/2018.