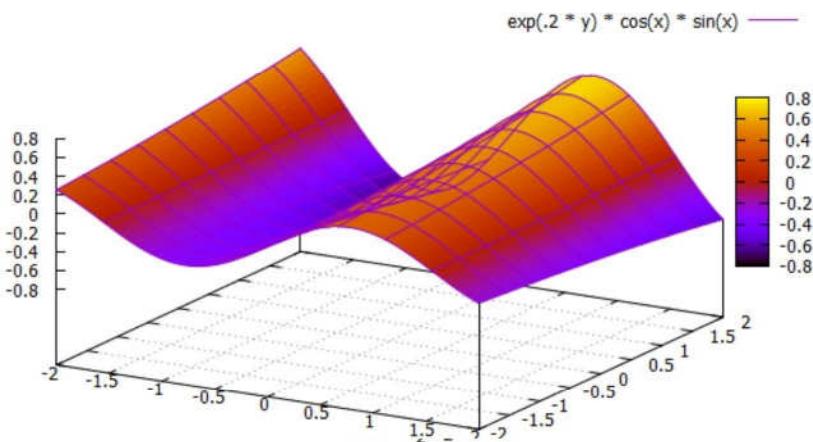


GNUPLOT 5

INTRODUÇÃO E APLICAÇÃO

GRÁFICOS INTERMEDIADOS POR COMPUTADOR



José Augusto N. G. Manzano

ATENÇÃO

Caso deseje obter uma cópia impressa deste material em formato livro poderá fazê-lo junto as plataformas de publicação:

Clube de autores: <https://clubedeautores.com.br/>

AgBook: <https://www.agbook.com.br>

Em ambas as plataformas efetue busca por "augusto manzano" e escolha o livro desejado.

José Augusto N. G. Manzano

GNUPLOT 5

INTRODUÇÃO E APLICAÇÃO

1^a Edição

São Paulo
2019 – Propes Vivens

Copyright © 2019 de José Augusto Navarro Garcia Manzano.

Todos os direitos reservados. Proibida e vedada a reprodução, memorização e/ou recuperação parcial ou total, por qualquer meio ou processo existente ou que venha a existir e inclusão de qualquer parte desta obra em qualquer programa juscibernético. A violação dos direitos autorais é crime (art. 184 e parágrafos, do Código Penal, conforme Lei nº 10.695, de 07/01/2003) com pena de reclusão, de dois a quatro anos, e multa, conjuntamente com busca e apreensão e indenizações diversas (artigos 102, 103 parágrafo único, 104, 105, 106 e 107 itens 1, 2 e 3 da Lei nº 9.610, de 19/06/1998, Lei dos Direitos Autorais).

O Autor responsável pelo conteúdo, proprietário do Direito Autoral acredita que as informações aqui apresentadas são corretas e podem ser utilizadas para qualquer finalidade legal. No entanto, não há qualquer garantia, explícita ou implícita, de que o uso de tais informações conduzirá ao resultado desejado.

**Dados Internacionais de Catalogação na Publicação (CIP)
(Ficha Catalográfica Confeccionada pelo Autor)**

M296i	Manzano, José Augusto Navarro Garcia. Gnuplot 5: introdução e aplicação / José Augusto Navarro Garcia Manzano. – 1. ed. – São Paulo: Propes Vivens, formato 14,8 x 21 cm, 2019. 204 p.
	Bibliografia. ISBN: 978-85-923720-7-1
	1. Estatística – processamento de dados 2. Gnuplot (programa de computador) 3. Teoria dos grafos – processamento de dados I. Título.
13-02651	CDD-519.5028553

Índices para catálogo sistemático:

1. Gnuplot: Programa de computador: recursos explorados por meio da apresentação de gráficos gerados com base em funções matemáticas: Introdução e aplicação
519.5028553

Dedicatória

Dedico este trabalho a minha esposa Sandra e minha filha Audrey. Sem a presença de vocês, minha vida seria monocromática.

A todos que me disseram “não”. Recebemos um “não” quando estamos parados, calados e sem ação. Quando pedimos algo, podemos ouvir sim ou ouvir não. O não ouvido pode ser um balde de água fria sobre a cabeça, a salvação quando se está prestes a fazer algo inadequado, mas pode, também, ser a válvula propulsora que nos faz ir além.

Aos amigos, alunos e leitores, cujas dúvidas e perguntas me incentivam, fazendo com que eu me aprimore cada vez mais, e, assim, continue com o trabalho de ensinar.

Um grande abraço a todos.

Agradecimentos

Ao Pai Celestial, que designou em sua infinita sabedoria a direção profissional de minha vida, fazendo-me descobrir em mim o ser professor. Espero sinceramente estar sendo digno em cumprir seu propósito e missão.

A matemática é o alfabeto com que Deus escreveu o universo.
Galileu Galilei

Sumário

Capítulo 1 – Introdução

1.1 O programa gnuplot	13
1.2 Aquisição e instalação	14
1.3 Comandos operacionais	17
1.4 Notação algébrica e computacional	20
1.5 Operações interativas	21
1.6 Glossário matemático	25

Capítulo 2 – Gráficos Bidimensionais

2.1 Gráfico de função de primeiro grau	33
2.2 Gráfico de função de segundo grau	39
2.3 Gráfico de função exponencial	44
2.4 Gráfico de função trigonométrica	46
2.5 Gráfico de função trigonométrica inversa	50
2.6 Mais de um gráfico	56
2.7 Variáveis e gráficos	62
2.8 Gráficos de função de terceiro e quarto graus	63
2.9 Domínio e contradomínio especificados ou não	66

Capítulo 3 – Gráficos Tridimensionais

3.1 Característica estrutural	71
3.2 Criação de gráficos 3D	72
3.3 Gravação de gráficos em mídia	78
3.4 Tipos de linha	80
3.5 Planos, superfícies e paleta de cores	83
3.6 Comando plot/splot	86

Capítulo 4 – Funções

4.1 Funções internas	91
4.1.1 Funções matemáticas	93
4.1.2 Funções cadeia	104
4.1.3 Funções diversas	108
4.2 Funções definidas pelo usuário	112

Capítulo 5 – Programação

5.1 Arquivos externos	121
5.2 Arquivos de script	143
5.3 Técnicas básicas para programação	145
5.3.1 Tomada de decisão	146
5.3.2 Iterações	151
5.4 Scripts programados	155
5.4.1 Operador ternário	158
5.4.2 Operação de somatório	160
5.5 Novas funções definidas pelo usuário	163
5.6 Salvar o ambiente	163
5.7 Integração com linguagens de programação	164

Capítulo 6 – Outros Recursos

6.1 Comandos set / unset / reset / show	169
6.2 Outros eixos: X2 e Y2	175
6.3 Gráfico de equação paramétrica	180
6.4 Gráfico polar	184
6.5 Gráfico histograma	187
6.6 Gráfico com rótulo	189
6.7 Ajuste de Curvas	191
6.8 Gráfico senoidal com superfície	196
6.9 Gráfico em escala logarítmica	198

Prefácio

Esta obra apresenta de maneira introdutória, prática e dirigida o uso do programa **gnuplot**. São explorados diversos recursos a partir da apresentação de gráficos gerados com base em funções matemáticas e de recursos diretamente do programa e de arquivos de dados.

Este livro é material ideal para reforçar o estudo de funções a estudantes de segundo grau e para programadores de computador que necessitam implementar o uso de gráficos em seus sistemas.

Não é objetivo deste trabalho transmitir aos leitores conceitos matemáticos e estatísticos, além do que se apresenta no glossário do capítulo 1, que é fornecido a título de revisão. Parte-se do pressuposto de que os leitores deste material já possuam tais conceitos. Caso não seja este seu perfil, sugere-se proceder com o estudo desses temas. Neste caso, estude conjuntos, funções, gráficos, trigonometria, logaritmos e um pouco sobre estatística.

No capítulo 1 realiza-se a introdução ao tema. Mostra como é feita a aquisição e a instalação do programa **gnuplot**, descreve rapidamente os comandos operacionais existentes e discute sobre o uso de notação algébrica e orienta como usá-la na forma computacional. Apresenta o uso do programa em algumas operações interativas. A título de orientação e revisão matemática este capítulo possui um glossário com breve descrição de diversos conceitos matemáticos.

O capítulo 2 orienta sobre o uso de gráficos bidimensionais. Apresenta a criação de gráficos a partir de funções de primeiro e segundo graus, funções exponenciais, funções trigonométricas, funções trigonométricas inversas e apresenta como fazer a sobreposição de gráficos.

O estudo do capítulo 3 mostra como usar gráficos tridimensionais. Neste capítulo o leitor tem a oportunidade de conhecer como gravar o gráfico desenvolvido em um arquivo de imagem. Apresenta-se como alterar o tipo das linhas de desenho usadas nos traços das curvas. Este capítulo descreve detalhadamente a estrutura de uso dos comandos *plot* e *splot*.

No capítulo 4 é realizada a apresentação do recurso mais importante do programa **gnuplot**, as funções. Faz-se a descrição e quando possível a demonstração de uso de todas as funções suportadas pelo programa. Um ponto importante deste capítulo é o desenvolvimento das próprias funções e criação de uma biblioteca externa para

uso posterior com as funções criadas. As funções matemáticas e as personalizadas, que foram criadas manualmente, além de comentadas são usadas para indicar os gráficos que elas geram.

O capítulo 5 orienta sobre o uso de arquivos externos com dados a serem usados para a definição de gráficos. Mostra como criar arquivos de dados a partir do editor de texto Bloco de notas e da planilha eletrônica Excel. Apresenta como fazer uso de arquivos de *script* e de programas para gerenciar ações do programa. Este capítulo mostra como integrar o programa **gnuplot** com as linguagens de programação C e Lua.

O capítulo 6 mostra como são criados gráficos do tipo histograma, polar, senoidal e logaritmo. Este capítulo mostra como fazer uso de coordenadas auxiliares X2 e Y2. Apresenta o uso de equações paramétricas e ajustes de curvas.

Os arquivos de auxílio do livro podem ser obtidos a partir do endereço de acesso <http://novo.manzano.pro.br/livros/gnuplot5brinde.zip>.

Espero que este pequeno trabalho possa ser útil ao leitor. A todos um grande abraço.

Augusto Manzano

Sobre o Autor

José Augusto Navarro Garcia Manzano é brasileiro, nascido no Estado de São Paulo, capital, em 26 de abril de 1965, é professor e mestre com licenciatura em matemática. Atua na área de Tecnologia da Informação (desenvolvimento, ensino e treinamento) desde 1986. Participou do desenvolvimento de aplicações computacionais para áreas de telecomunicações e comércio. Na carreira docente iniciou sua atividade em cursos livres, trabalhando posteriormente em empresas de treinamento e atuando nos ensinos técnico, superior e tecnológico. Trabalhou em empresas na área de T. I. como: ABAK, SERVIMEC, CEBEL, SPCI, BEPE, ORIGIN, OpenClass, entre outras.

Atualmente é professor com dedicação exclusiva no IFSP (Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, antiga Escola Técnica Federal). Em sua carreira docente, possui condições de ministrar componentes curriculares de Lógica de Programação (Algoritmos), Estrutura de Dados, Microinformática, Informática, Linguagens de Programação Estruturada, Linguagens de Programação Orientada a Objetos, Engenharia de Software, Tópicos Avançados em Processamento de Dados, Sistemas de Informação, Engenharia da Informação, Arquitetura de Computadores e Tecnologias Web. Possui conhecimento de uso e aplicação das linguagens de programação BASIC Classic, COMAL Assembly, LOGO, PASCAL, FORTRAN, C, C++, D, JAVA puro e com JSP, MODULA-2, Structured BASIC, C#, Lua, HTML, XHTML, JavaScript, PHP, VBA, ADA, Python, Rust, Hope, Haskell, Elixir, OCaml, Groovy e Julia. Possui mais de uma centena de obras publicadas, além de artigos publicados no Brasil e no exterior.

1

Introdução

Este capítulo apresenta aos leitores o programa *gnuplot*, sua aquisição, instalação e uso inicial junto ao sistema operacional Windows. Além dessas informações este capítulo mostra como realizar a tradução da notação algébrica matemática para a forma usada no programa e indica uma série de ações interativas que podem ser realizadas. No final do capítulo é indicado um glossário matemático com a descrição de diversos conceitos necessários ao uso da ferramenta *gnuplot*.

1.1 O programa *gnuplot*

O programa **gnuplot**¹ é uma ferramenta para a criação interativa de gráficos a partir da indicação de funções polinomiais, exponenciais e trigonométricas fornecidas via linha de comando ou a partir de arquivos de dados. Os gráficos produzidos podem ser definidos em 2D (duas dimensões) ou 3D (três dimensões com gráficos de superfície). Assim, pode-se afirmar que este programa se insere na categoria operacional de *software científico*. No entanto, não deve ser comparado com outros programas, tais como: *Mathematica* da Wolfram², *MATLAB* da MathWorks³, *Maxima* (*Macsyma*)⁴ do *Massachusetts Institute of Technology* e *University of Texas System*, *Maple 16*⁵ da Maplesoft, entre outros que possuem funcionalidades e finalidades não existentes no *gnuplot*.

A versão utilizada para a elaboração deste livro é a versão 5.2 lançada em outubro de 2018. A identificação **gnu** no nome do programa não possui nenhuma relação

¹ O nome do programa deve ser referenciado sempre em caracteres minúsculos.

² <http://www.wolfram.com/mathematica/>

³ <http://www.mathworks.com/products/matlab/>

⁴ <http://maxima.sourceforge.net/>

⁵ <http://www.maplesoft.com/products/maple/>

com o projeto **GNU** de Richard Stallman, daí a identificação do nome ser grafado em letras minúsculas. No entanto o programa pode ser usado de forma livre, existindo restrições em relação a sua distribuição quando seu código fonte é alterado.

O uso do programa **gnuplot** se estende a diversas áreas que tenham necessidade de operar com gráficos, destacando-se as áreas de matemática, estatística, física, engenharia nas suas várias vertentes. Segundo o que consta na documentação, o **gnuplot** foi desenvolvido para auxiliar cientistas e alunos a visualizarem gráficos a partir de funções matemática.

O programa pode ser executado em diversas plataformas computacionais, como: Linux, UNIX, Windows, OS/2, VMS, entre outras.

A saída gerada pelo programa pode ser em tela gráfica ou podem ser geradas imagens em diversos formatos gráficos, como: PNG, GIF, JPEG, além de formatos como PDF, EPS ou LaTeX, entre outras possibilidades.

1.2 Aquisição e instalação

A aquisição do programa **gnuplot** deve ser realizada a partir de seu sítio oficial acessado a partir do endereço URL: <http://www.gnuplot.info>, como apresenta a Figura 1.1.

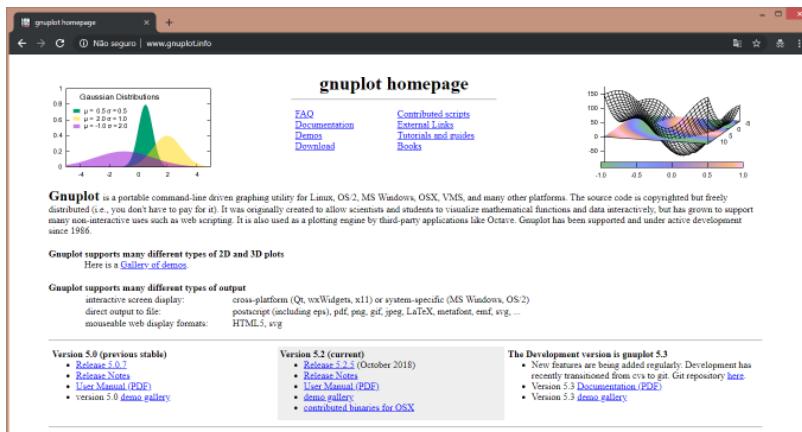


Figura 1.1 – Sítio para aquisição do programa **gnuplot**.

A partir da página apresentada, selecione o link **Download** no topo da página que apresentará a página **gnuplot download** indicada na Figura 1.2.

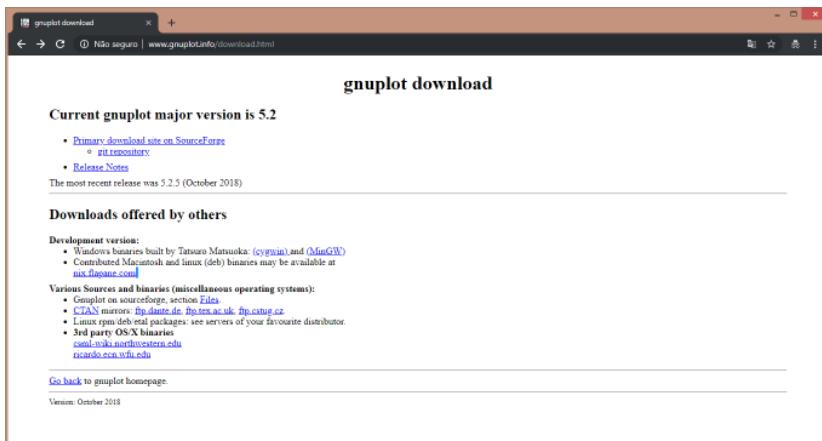


Figura 1.2 – Página de download.

Na sequência selecione em **Current gnuplot major version is 5.2** o *link Primary download site on SourceForge* que apresentará a página do sitio **SourceForge** como indica a Figura 1.3.

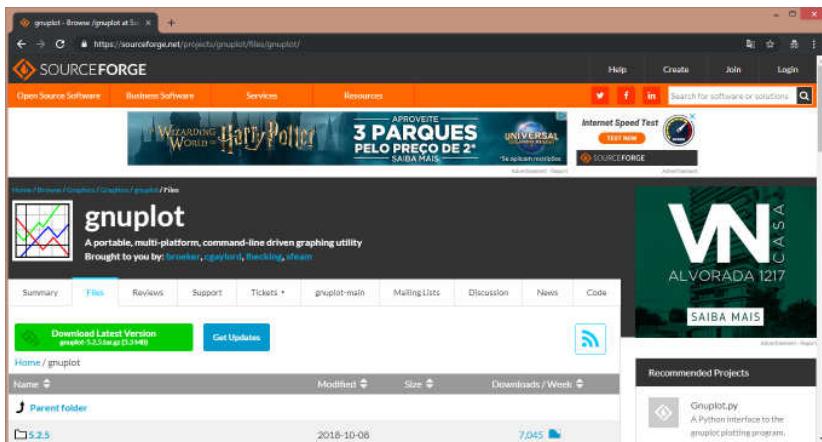


Figura 1.3 – Sítio do serviço SourceForge.

A partir da página apresentada, role-a um pouco para baixo e selecione o *link* da pasta **5.2.5**. Na sequência role a página um pouco para baixo e selecione um dos programas de instalação: **gp525-win64-mingw.exe** para Windows 64 bits ou para o Windows de 32 bits selecione **gp525-win32-mingw.exe**.

Para a demonstração desta obra será considerado o uso do programa de instalação **gp525-win64-mingw.exe**.

Assim que o arquivo de programa **gp525-win64-mingw.exe** é copiada, selecione-o com um duplo clique e siga as instruções de instalação apresentadas mantendo o idioma inglês.

Terminado a instalação para fazer a execução do programa **gnuplot** basta executar a seguinte sequencia de comandos a partir do Windows 8.1:

Botão: Iniciar

Botão: Seta para baixo

Em gnuplot selecione o ícone gnuplot 5.2 patchlevel 5a

Ou selecione no *desktop* do sistema operacional o ícone com o nome **gnuplot 5.2 patchlevel 5a**. Independentemente da forma usada será apresentada a tela de acesso ao programa como apresenta a Figura 1.4.

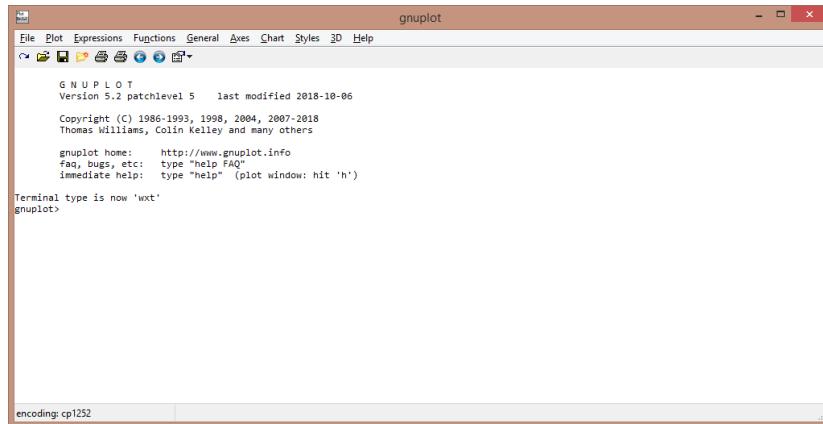


Figura 1.4 – Tela do programa *gnuplot*.

Para obter ajuda basta executar na linha de comando as instruções: **help**, **h**, **?** ou **help FAQ** e açãoar em seguida a tecla **<Enter>**. Para saber sobre um comando específico basta fazer uso do comando **help** mais o comando a ser consultado, por exemplo, **help plot**.

Para encerrar a execução do programa basta escrever na sua linha de comando a instrução **exit** ou a instrução **quit** e açãoar em seguida a tecla **<Enter>**.

Além da interface em tela gráfica, como mostra a Figura 1.4 é possível acessar o programa em uma janela com tela texto convencional. Assim sendo, basta executar o programa **gnuplot.exe** que se encontra no diretório **\Program Files\gnuplot\bin**. A Figura 1.5 mostra a tela do programa em modo texto.

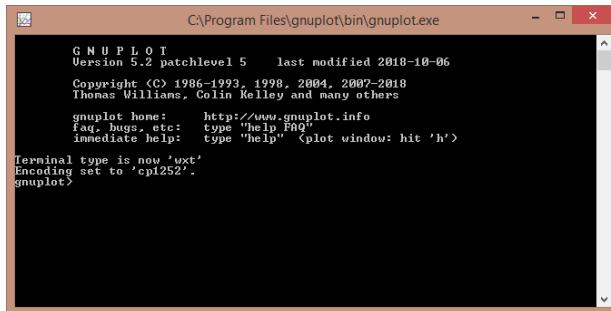


Figura 1.5 – Tela do programa **gnuplot** em modo texto.

Um detalhe a ser considerado é não julgar a potencialidade do programa a partir de sua interface de acesso. A diferença entre as telas gráfica e texto é a barra de menu e barra de ferramentas existentes na janela gráfica. No mais as duas maneiras de uso do programa são semelhantes.

O visual do programa é extremamente simples, mas o resultado final de seu uso pode ser uma experiência excepcional. Aliás não importa se está sendo executada o programa **gnuplot.exe** (apresenta interface texto) ou **wgnuplot.exe** (apresenta a interface gráfica), o resultado final apresentado será sempre o mesmo.

Nesta obra serão apresentados comandos que possam ser usados em qualquer um dos modos de apresentação (gráfico ou texto) do programa **gnuplot**, uma vez que os comandos das barras de menu e ferramenta podem ser executados a partir do *prompt* do programa.

Os comandos existentes nas barras de menu e ferramentas podem ser reproduzidos de maneira interativa no *prompt* de comando do programa, sendo este o foco deste trabalho. Assim sendo, as ações descritas no menu não serão aqui reproduzidas.

1.3 Comandos operacionais

O programa **gnuplot** para ser utilizado necessita que sejam informados comandos que produzam ações em seu ambiente. Os comandos a serem efetivados podem ser formados por uma palavra reservada, por um conjunto de palavras reservadas

ou agrupados em arquivos chamados *scripts*⁶, além de parâmetros para a efetivação dos mesmos.

Veja a seguir uma lista comentada dos comandos encontrados no **gnuplot**:

- **cd:** usado para mudar o diretório de trabalho;
- **call:** usado para chamar um arquivo externo com até 10 parâmetros opcionais;
- **clear:** usado para limpar a tela de apresentação de gráficos;
- **do:** usado para executar uma sequência de comandos mais de uma vez;
- **eval:** usado para executar uma sequência de comandos definida em um *string*;
- **exit:** usado para encerrar a execução do programa, dos fluxos de saída, da sessão do terminal e do arquivo de entradas;
- **fit:** usado para ajustar uma expressão fornecida para um conjunto de dados (x, z) ou (x, y, z), por meio de uma aplicação dos mínimos quadrados não lineares;
- **help:** usado para apresentar o modo de ajuda do programa;
- **history/his/hi:** usado para listar ou salvar as entradas de comandos efetivadas ou para executar uma entrada indicada;
- **if:** usado para efetivar uma ação de tomada de decisão;
- **iteration:** usado para determinar a quantidade de iterações executadas com os comandos **plot**, **splot**, **set** e **unset**;
- **load:** usado para chamar um arquivo externo sem uso de parâmetros;
- **lower:** usado para determinar a posição da janela na pilha de janelas abaixo;
- **pause:** usado para determinar uma pausa na operação de desenho de um gráfico;
- **plot:** usado para desenhar um gráfico bidimensional na tela gráfica;
- **print:** usado para apresentar o resultado de uma expressão fornecida;
- **pwd:** usado para apresentar o nome do diretório de trabalho em uso;
- **quit:** usado para encerrar a execução do programa e efetuar a limpeza da tela de apresentação gráfica antes de sair;
- **raise:** usado para determinar a posição da janela na pilha de janelas acima;

⁶ Um script é uma espécie de programa escrito para realizar certa tarefa no gnuplot.

- **refresh:** usado para redesenhar e redefinir o gráfico com base nos dados fornecidos;
- **replot:** usado para redesenhar um gráfico de acordo com a execução do último **plot** ou **splot**;
- **reread:** usado para reler o conteúdo de carga como se o programa **gnuplot** tivesse acabado de ser carregado;
- **reset:** usado para redefinir os valores padrão após o uso do comando **set**;
- **save:** usado para salvar funções definidas pelo usuário, variáveis, opções definidas e a última execução dos comandos **plot** ou **splot**.
- **set:** usado para definir a configuração de certo parâmetro do ambiente fornecido, como *angles*, *bars*, *ydate*, entre outros de um extenso conjunto de parâmetros;
- **show:** usado para apresentar a configuração de certo parâmetro do ambiente fornecido, como *angles*, *bars*, *ydate*, entre outros de um extenso conjunto de parâmetros;
- **shell:** usado para apresentar uma tela da linha de comando do sistema operacional em uso;
- **splot:** usado para desenhar um gráfico tridimensional na tela gráfica;
- **stats:** usado para gerar um arquivo com o resumo estatístico dos dados usados em uma ou duas colunas;
- **system:** usado para executar um comando externo do sistema operacional;
- **test:** usado para testar a capacidade gráfica do terminal e a paleta de cores;
- **undefine:** usado para limpar uma ou mais variáveis previamente definidas;
- **unset:** usado para redefinir o estado alterado pelo comando **set**, por meio da definição de um parâmetro do ambiente fornecido, como *angles*, *bars*, *ydate*, entre outros de um extenso conjunto de parâmetros;
- **update:** usado para escrever os valores atuais dos parâmetros de ajuste para o determinado arquivo, sendo útil para salvar valores para uso posterior;
- **while:** usado para repetir um bloco de comando certo número de vezes.

Nem todos os comandos citados anteriormente serão apresentados nesta obra, apenas os recursos essenciais à aprendizagem para uso da ferramenta; aqueles que nem forem abordados aqui, são descritos no manual do programa.

1.4 Notação algébrica e computacional

O programa **gnuplot** é uma ferramenta matemática para criação e uso de gráficos. No entanto, apesar dos computadores serem ferramentas auxiliares para a ciência matemática, estes não utilizam para sua operação a notação algébrica, possuindo uma notação própria de trabalho.

A diferença no uso entre notação computacional e matemática pode ser um ponto problemático para alguns usuários inicialmente. Desta forma, qualquer equação ou fórmula matemática necessitam ser convertidas em expressões aritméticas.

A expressão aritmética será a maneira pela qual certa operação matemática será executada em um computador. Tome como exemplo a necessidade de se fazer uso da fórmula de Bhaskara em um computador.

A forma algébrica da fórmula de Bhaskara, advinda do uso da fórmula matemática da equação de segundo grau $x = ax^2 + bx + c$ corresponde a:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

$$\Delta = b^2 - 4ac$$

O uso da fórmula de Bhaskara no programa **gnuplot** deve ser procedido a partir das expressões aritméticas em seguida.

```
delta = b ** 2 - 4 * a * c
x1 = (-b + delta ** (1.0 / 2)) / (2 * a)
x2 = (-b - delta ** (1.0 / 2)) / (2 * a)
```

Observe também as situações exemplificadas a seguir, as quais demonstram diversas representações matemáticas e sua equivalência no programa **gnuplot**.

$x = \frac{a}{b}$	use $x = a / b$	x^y	use $x^{**} y$
$x = a + \frac{b}{c}$	use $x = a + b / c$	e^x	use $\exp(x)$
$x = \frac{a+b}{c}$	use $x = (a + b) / c$	$\log x$	use $\log(x)$

$x = ab$	use <code>x = a * b</code>	$\sin x$	use <code>sin(x)</code>
$a \neq b$	use <code>a <> b</code>	$\cos x$	use <code>cos(x)</code>
$a \leq b$	use <code>a <= b</code>	$\tan x$	use <code>tan(x)</code>
$a \geq b$	use <code>a >= b</code>	$\arcsin x$	use <code>asin(x)</code>
$ x $	use <code>abs(x)</code>	$\arccos x$	use <code>acos(x)</code>
\sqrt{x}	use <code>sqrt(x)</code>	$\arctan x$	use <code>atan(x)</code>
$\sqrt[n]{x}$	use <code>sqrt(x) ** (1.0/n)</code>		

É importante estar atento para não ocorrer erros na representação computacional a partir da representação algébrica.

1.5 Operações interativas

O programa **gnuplot** é baseado em ações interativas, fornecidas em sua linha de comando. A título de apresentação, algumas operações aritméticas básicas serão executadas em seguida. No entanto, para usar tais recursos é necessário conceituar alguns elementos importantes para as operações, sendo variáveis, constantes e operadores aritméticos.

- **Variáveis**

As variáveis são em um computador uma região de memória usada para armazenar certo valor por certo espaço de tempo que serão utilizados para a realização de alguma ação computacional.

Na ciência matemática variável (particularmente chamada de incógnita) pode ser entendida como sendo uma entidade que possui a capacidade de ser a representação real de um valor incógnito que pode ser substituído por uma série de diversos valores fornecidos. No programa **gnuplot** os dois conceitos se misturam, o que é de certa forma comum nas ferramentas computacionais, mas prevalece em média o conceito matemático. Assim sendo, as variáveis

serão representadas com caracteres de letras minúsculas e terão no programa o mesmo significado usado na ciência matemática.

- **Constantes**

Um valor constante é a definição de um valor fixo, inalterado ao longo de certo tempo. Uma constante pode ser previamente definida, com seu valor conhecido ou não.

Uma constante definida, é por exemplo o π (3,14159265358979). Este valor jamais é alterado independentemente do que venha a ser efetuado.

Uma constante sem valor definido (indefinida) é usada na programação de computadores quando se associa um valor a um rótulo que será usado para alguma operação computacional. Por exemplo, $a = 1$, onde a será usado para uma operação e não terá seu valor alterado até o final desta operação.

- **Operadores Aritméticos**

O programa funciona por meio de operadores aritméticos, que têm por objetivo permitir a execução de diversas operações matemáticas, inclusive a definição das equações que geram a apresentação de gráficos. A seguir, apresenta-se uma tabela com os operadores aritméticos usados pelo programa **gnuplot**.

Operador	Operação
=	Atribui um valor a uma variável
+	Manutenção de sinal
-	Inversão de sinal
**	Exponenciação
sqrt(x) ou x ** 0.5	Raiz quadrada
x ** (1.0/n)	Raiz de índice qualquer: $n\sqrt{x}$
-	Subtração
/	Divisão quociente inteiro ou real
*	Multiplicação
+	Adição
%	Resto de uma divisão de inteiros
n!	Fatorial de n
.	Concatenação de cadeias

A partir do exposto é possível executar por meio do comando **print** (que efetua a apresentação de diversas ações) algumas operações aritméticas, tais como:

```
print 2 + 3
print 2 - 3
print 2 * 3
print 2 / 3
print 2.0 / 3.0
print 2 ** 3
print 2 + 3 * 4
print (2 + 3) * 4
print "Alo, ."."Mundo!"
```

Ao ser executado os comandos anteriores no ambiente do programa ocorrerá a apresentação dos respectivos resultados como: **5, -1, 6, 0, 0.6666666666666667, 8, 14, 20 e Alo, Mundo!**. Observe que o uso do comando **print 2/3** apresentou como valor o número **0** e o comando **print 2.0/3.0** apresentou seu resultado como sendo o valor **0.6666666666666667**. Note que no primeiro caso o programa **gnuplot** entende o fornecimento de valores inteiros e no segundo caso é entendido o uso de valores reais. A operação **print 2 ** 3** efetua o cálculo do valor 2 elevado a potência 3. Outro detalhe é o uso dos símbolos de parênteses que permitiram a mudança no nível da prioridade de cálculo matemático e por fim a instrução **print "Alo, ."."Mundo!"** faz a apresentação das duas cadeias de caracteres como se fosse uma única mensagem por meio da operação de concatenação representada pelo operador **.** (ponto).

Observe em seguida o uso de constantes. Assim sendo, forneça no ambiente as seguintes linhas de comandos:

```
a = 5
print a - 3
print pi
print 5!
```

O comando **print a - 3** mostra o resultado obtido a partir do uso de uma constante não especificada na linha. A constante **a** foi definida por meio da instrução **a = 5** na linha anterior e mesmo assim esta variável é considerada não especificada de acordo com as regras da ciéncia matemática. O resultado de **a - 3** é mostrado como **2**. Uma constante não especificada é aquela cujo valor será fornecido no uso desta constante.

O comando **print pi** apresenta como resultado o valor numérico **3.14159265358979**, sendo este um exemplo de uso de constante especificada. Uma constante que seja especificada é aquela que representa um valor matemático definido, no sentido, de não haver a necessidade de escrever o valor em si e tão somente fazer uso do rótulo que identifica tal valor.

A instrução **print 5!**, mostra o resultado **120.0** que é a factorial de 5 ($5 \times 4 \times 3 \times 2 \times 1$).

De acordo com a tabela de operadores aritméticos apresentados procure proceder a execução de outras operações aritméticas.

O desenho de um gráfico no programa **gnuplot** é realizado a partir da entrada no *prompt* de comandos do programa de alguma função matemática. Os passos para esta ação serão indicados no próximos capítulos.

Experimente proceder com o cálculo de uma equação de segundo grau a partir da equação $x^2 - 2x - 3 = 0$.

```
a = 1
b = -2
c = -3
delta = b ** 2 - 4 * a * c
x1 = (-b + delta ** (1.0 / 2)) / (2 * a)
x2 = (-b - delta ** (1.0 / 2)) / (2 * a)
print a, b, c, delta, x1, x2
```

É possível usar no programa números inteiros, reais e complexos. Números inteiros são definidos na forma **1**, **13**, **25**, entre outros. Números reais são definidos com o uso de ponto decimal na forma **1.0**, **0.5**, **9.67**, entre outros. Números complexos são definidos na forma de um par, sendo o primeiro número a parte real do valor e o segundo número a parte imaginária do valor entre chaves na forma **{1, 2}**, **{7.0, 5}**, entre outros.

Os valores reais operam no programa **gnuplot** com até 16 casas decimais.

Para ver a versão do programa podem ser executada a instrução **show version** forma reduzida ou **show version long** para forma completa.

Todos os comandos fornecidos ao *prompt* de comando do programa **gnuplot** deverão ser escritos com caracteres minúsculos.

1.6 Glossário matemático

Neste tópico apresenta-se como glossário matemático a explanação do conjunto de alguns termos utilizados na obra que abrangem conceitos da ferramenta **gnuplot** ou relacionados com os próprios termos utilizados no glossário. Nessas explanações é usado um discurso simplista sem se ater ao rigor ou ditames matemáticos clássicos. Operações matemáticas representadas neste glossário são expressas na forma computacional, baseada em expressões aritméticas.

- **Abscissa:** eixo **x** de uma coordenada cartesiana.
- **Ângulo:** também chamado de canto ou esquina, é a medida da inclinação relativa de duas semirretas de mesma origem (ângulo plano), mensurada em graus ou radianos, representada pelo símbolo θ (letra grega teta). O ângulo de 0° é nulo, ângulos entre 0° e 89° são agudos, de 90° é reto ($\frac{1}{4}$ volta), entre 91° e 179° são obtusos, de 180° é raso ($\frac{1}{2}$ volta) e de 360° é completo (uma volta). Os ângulos podem ser medidos com um instrumento chamado *transferidor*.
- **Concavidade:** é a abertura de uma parábola em um gráfico com base em uma função de segundo grau ou quadrática.
- **Contradomínio:** em uma relação entre dois conjuntos, por exemplo, o contradomínio de uma função é formado pelos elementos pertencentes ao segundo conjunto.
- **Convolução:** tipo de operador matemático que, com base em duas funções fornecidas, produz uma terceira.
- **Coordenadas retangulares:** é o mesmo que plano cartesiano tridimensional, usado na apresentação de um gráfico tridimensional.
- **Corda:** é o segmento de reta determinado por dois pontos de uma curva ou de uma superfície.
- **Cota:** eixo **z** de uma coordenada cartesiana.
- **Curva:** segmento de reta apresentado em um gráfico, não importando se ela é linear (reta) ou parabólica.
- **Diâmetro:** é a distância entre dois pontos de um subconjunto de um espaço métrico. Do ponto de vista geométrico, pode ser a reta determinada pelos pontos médios das cordas paralelas de uma cônica ou qualquer corda que passa pelo centro de uma circunferência.

- **Domínio:** em uma relação entre dois conjuntos, por exemplo, o domínio de uma função é composto pelos elementos do primeiro conjunto.
- **Eixo de coordenadas cartesianas:** ver *Gráfico bidimensional*.
- **Equação paramétrica:** forma de representar retas por meio de um único parâmetro, que estabelece a ligação de duas equações que pertencem a uma mesma reta.
- **Função arco cossecante (arccosec):** é representada pela função $y = \text{cosec}^{-1} x$ ou $y = \text{arccosec } x$ quando $\text{cosec } x$ é bijetora, com base na definição do domínio $D(f) = [-\infty, -1] \cup [1, \infty]$ e da imagem $\text{Im}(f) = [-\pi/2, 0] \cup [0, \pi/2]$ em radianos.
- **Função arco cosseno (arccos):** lê-se *arco cujo cosseno*. É representada pela função $y = \cos^{-1} x$ ou $y = \arccos x$ quando $\cos x$ é bijetora, com base na definição do domínio $D(f) = [-1, 1]$ e da imagem $\text{Im}(f) = [0, \pi]$ em radianos.
- **Função arco cotangente (arccot):** é representada pela função $y = \cot^{-1} x$ ou $y = \text{arccot } x$ quando $\cot x$ é bijetora, com base na definição do domínio $D(f) = \mathbb{R}$ e da imagem $\text{Im}(f) = [0, \pi]$ em radianos.
- **Função arco secante (arcsec):** é representada pela função $y = \sec^{-1} x$ ou $y = \text{arcsec } x$ quando $\sec x$ é bijetora, com base na definição do domínio $D(f) = [-\infty, -1] \cup [1, \infty]$ e da imagem $\text{Im}(f) = [0, \pi/2] \cup [\pi/2, \pi]$ em radianos.
- **Função arco seno (arcsen):** lê-se *arco cujo seno*. É representada pela função $y = \sin^{-1} x$ ou $y = \arcsen x$ quando $\sin x$ é bijetora, com base na definição do domínio $D(f) = [-1, 1]$ e da imagem $\text{Im}(f) = [-\pi/2, \pi/2]$ em radianos.
- **Função arco tangente (arctg):** é representada pela função $y = \tg^{-1} x$ ou $y = \text{arctg } x$ quando $\tg x$ é bijetora, com base na definição do domínio $D(f) = \mathbb{R}$ e da imagem $\text{Im}(f) = [-\pi/2, \pi/2]$ em radianos.
- **Função bijetora:** é a função obtida na relação entre conjuntos que suportem os conceitos de função injetora e subjetora simultaneamente, ou seja, é a função que é injetora e bijetora ao mesmo tempo.
- **Função continua:** caracteriza-se por apresentar o desenho da curva (reta) de forma continua sem que haja espaços entre os trechos do plano do gráfico.
- **Função cossecante (cosec):** é a função inversa à função seno, representada pela função $y = f(x) = 1 / \sin x$ ($f(x) = \text{cosec } x$), com domínio $D(f) = \{x \in \mathbb{R} \mid x \neq k \cdot \pi, k \in \mathbb{Z}\}$ e imagem $\text{Im}(f) = \{y \in \mathbb{R} \mid y \leq -1 \text{ ou } y \geq 1\}$, com período em 2π , resultando no valor máximo de $3\pi/2$ e no valor mínimo de $\pi/2$, sendo

crescente quando $\text{sen } x$ é decrescente e decrescente quando $\text{sen } x$ é crescente. É uma função ímpar.

- **Função cosseno (cos):** é representada pela função $y = f(x) = \cos x$, com domínio $D(f) = \mathbb{R}$ e imagem $Im(f) = [-1, 1]$, com período em 2π , resultando no valor máximo de 1, em $x = 0$ ou $x = 2\pi$, e valor mínimo de -1 em $x = \pi$, sendo crescente no intervalo $[\pi, \pi]$ e decrescente no intervalo $[0, \pi]$. É uma função par.
- **Função cotangente (cot):** é a função inversa à função tangente $y = f(x) = 1 / \operatorname{tg} x$ ($f(x) = \cot x$), com domínio $D(f) = \{x \in \mathbb{R} \mid x \neq \pi / 2 + k \cdot \pi, k \in \mathbb{Z}\}$ e imagem $Im(f) = \mathbb{R}$, com período em π , não possuindo valores extremos, sendo decrescente em todos os pontos do domínio. É uma função ímpar.
- **Função de arco:** ver funções trigonométricas inversas (*função arco cosseno*, *função arco cotangente*, *função arco seno* e *função arco tangente*).
- **Função descontinua:** caracteriza-se por apresentar o desenho da curva (reta) de forma não contínua em determinado plano do gráfico.
- **Função exponencial:** é a função inversa à função logarítmica natural com base na função f de \mathbb{R}^7 , em que \mathbb{R} é definido pela função $f(x) = a^x$, em que a (base) $\in \mathbb{R}$ e x (expoente) com $1 \neq a > 0$. É crescente quando $a > 1$ e decrescente se a é um valor numérico entre 0 e 1 ($0 < a < 1$). Uma função exponencial pode ser expressa como $f(x) = e^x$, em que e representa a constante matemática neperiana e x real resulta em uma função crescente e positiva, que é uma função exponencial natural.
- **Função gnuplot:** caracteriza-se por ser a citação e/ou o uso de alguma função interna do programa. Para mais detalhes, consulte o Capítulo 4.
- **Função hiperbólica:** função que gera uma hipérbole, que é o conjunto dos pontos de um plano cartesiano, cujo módulo da diferença das distâncias entre dois pontos fixos é constante e menor que a distância entre eles.
- **Função injetora:** em uma relação entre dois conjuntos, por exemplo, a função injetora é aquela em que os elementos pertencentes ao primeiro conjunto possuem uma relação com um único elemento do segundo conjunto, ou seja, quando os elementos distintos do domínio possuem imagens distintas.
- **Função matemática:** caracterizada por ser a correspondência relacional entre os elementos de dois ou mais conjunto de valores.

⁷ \mathbb{R} representa o conjunto dos números reais.

- **Função modular:** caracterizada por ser uma função que associa elementos de um conjunto em módulos, ou seja, para cada valor positivo de um conjunto ter-se-á um valor negativo associado.
- **Função polinomial de primeiro grau:** qualquer função f de \mathbb{R} em que \mathbb{R} é dado pela função $f(x) = ax + b$, em que a (coeficiente de x) e b (constante) são números reais dados com $a \neq 0$. Se o valor da variável a for positivo a função é crescente, se o valor da variável a for negativo a função é decrescente. Esta função apresenta como curva do gráfico a imagem de uma reta.
- **Função polinomial de segundo grau:** qualquer função f de \mathbb{R} , em que \mathbb{R} é dado pela função $f(x) = ax^2 + bx + c$ e a (coeficiente de x em segundo grau), b (coeficiente de x em primeiro grau) e c (constante) são números reais dados com $a \neq 0$. Se o valor da variável a for positivo, a concavidade do gráfico é voltada para cima; se o valor da variável a for negativo a concavidade do gráfico é voltada para baixo. Em uma parábola, tem-se uma metade crescente e outra metade decrescente. Se a concavidade estiver voltada para cima, é decrescente do infinito negativo ao vértice e crescente do vértice ao infinito positivo. Se a concavidade estiver voltada para baixo, é crescente do infinito negativo ao vértice e decrescente do vértice ao infinito positivo. Essa função exibe a imagem de uma parábola como curva do gráfico.
- **Função quadrática:** ver *Função polinomial de segundo grau*.
- **Função secante (sec):** é a função inversa à função cosseno $y = f(x) = 1 / \cos x$ ($f(x) = \sec x$), com domínio $D(f) = \{x \in \mathbb{R} \mid x \neq \pi/2 + k\pi, k \in \mathbb{Z}\}$ e imagem $Im(f) = \{y \in \mathbb{R} \mid y \leq -1 \text{ ou } y \geq 1\}$, com período em 2π , resultando no valor máximo em $x = \pi$ e o mínimo em $x = 0$. É crescente quando $\cos x$ é decrescente e decrescente quando $\cos x$ é crescente. É uma função par.
- **Função seno (sen):** representada pela função $y = f(x) = \sin x$, com domínio $D(f) = \mathbb{R}$ e imagem $Im(f) = [-1, 1]$, com período em 2π , resultando no valor máximo de 1 em $x = \pi/2$ e no valor mínimo de -1 em $x = 3\pi/2$. É crescente no intervalo $[0, \pi/2]$ e $[3\pi/2, 2\pi]$ e decrescente no intervalo $[\pi/2, 3\pi/2]$. É uma função ímpar.
- **Função sinal:** caracteriza-se por indicar o sinal de um valor numérico diferente de zero. Apresenta 1 quando valor positivo e -1 quando valor negativo.
- **Função sobrejetora:** em uma relação entre dois conjuntos, por exemplo, a função sobrejetora é aquela em que mais de um elemento pertencente ao primeiro conjunto possui relação com um único elemento do segundo conjunto, ou seja, o conjunto imagem é igual ao contradomínio.

- **Função tangente (tg):** representada pela função $y = f(x) = \operatorname{tg} x$, com domínio $D(f) = \{x \in \mathbb{R} \mid \{x \neq \pi / 2 + k \cdot \pi, k \in \mathbb{Z}\}^8\}$, considerando-se exceções os valores que zeram o valor do cosseno, e imagem $\operatorname{Im}(f) = [-\infty, \infty]$, com período em π . Não possui valores extremos, sendo crescente em todos os pontos de seu domínio. É uma função ímpar.
- **Função trigonométrica:** é usada na medição dos triângulos, com base nas funções trigonométricas seno, cosseno, tangente, secante, cossecante, e cotangente.
- **Gráfico bidimensional:** caracteriza-se por fazer uso de duas coordenadas cartesianas, que são o eixo X⁹ (abscissa), na posição horizontal, e o eixo Y (ordenada), na posição vertical, as quais formam o plano cartesiano¹⁰ ou o eixo de coordenadas cartesianas.
- **Gráfico de duas dimensões:** ver *Gráfico bidimensional*.
- **Gráfico de três dimensões:** ver *Gráfico tridimensional*.
- **Gráfico tridimensional:** caracteriza-se por fazer uso de três coordenadas cartesianas, que são o eixo X (abscissa), na posição horizontal, o eixo Y (ordenada), na posição vertical, e o eixo Z (cota), na posição ortogonal ao plano cartesiano.
- **Grau:** unidade de medida de um ângulo plano ou de uma circunferência (arco de círculo), representada pelo símbolo ° (letra o sobreescrita), a qual representa uma parcela de 1 em 360 graus. O ângulo reto é equivalente a 1/90 do grau e a circunferência é equivalente a 1/360 do grau. A subdivisão de um grau é feita por minutos e segundos, em que $1^\circ = 60'$ (um grau é igual a sessenta minutos), $1' = 60''$ (um minuto é igual a sessenta segundos).
- **Graus para radianos** – use a expressão aritmética: $\operatorname{rad} = (\operatorname{graus} * \pi)/180$.
- **Imagen:** em uma relação entre dois conjuntos, por exemplo, a imagem de uma função é composta pelos elementos do segundo conjunto que possuem relação com os elementos do primeiro conjunto.

⁸ Z representa o conjunto dos números inteiros.

⁹ Nesta obra, as referências às coordenadas de eixos de um gráfico são feitas com caracteres maiúsculos.

¹⁰ Conceito estabelecido por René Descartes ao criar a representação gráfica para apresentar graficamente o resultado do uso de expressões algébricas.

- **Minuto:** unidade de medida que subdivide o ângulo ou arco, igual a $1/60$ do grau, representada pelo símbolo ' (aspas simples).
- **Número complexo:** valor numérico pertencente ao conjunto **C**, formado com base em um par de valores **a** e **b**. É escrito na forma $z = a + bi$ ou $z = (a, b)$, em que **z** é o número complexo, **a** e **b** são valores reais e **i** é a definição da unidade imaginária, com base na propriedade de que $i^2 = -1$. A variável **a** representa a parte real do número, sendo $a = \text{Re}(z)$ e a variável **b** representa a parte imaginária do número, sendo $b = \text{Im}(z)$.
- **Número imaginário:** é um número complexo com parte real igual a zero, representado por **bi**, em que **i** é a representação da unidade imaginária.
- **Número inteiro:** valor numérico pertencente ao conjunto **Z**, formado pelos números naturais, incluindo o zero e todos os números negativos que sejam simétricos aos números naturais.
- **Número natural:** valor numérico pertencente ao conjunto **N**. É formado pelos números inteiros positivos, excluindo-se o número zero.
- **Número real:** valor numérico pertencente ao conjunto **R**. É formado pelos números racionais, que englobam números inteiros, números fracionários e números irracionais.
- **Ordenada:** eixo **y** de uma coordenada cartesiana.
- **Paralelas:** são retas que, estando no mesmo plano e sendo prolongadas de maneira infinita em cada um dos lados, não se encontram em momento algum (EUCLIDES, 2009, p.98).
- **Pi (π):** letra grega usada para representar a divisão do valor do perímetro de uma circunferência em relação a seu diâmetro, que resulta em um número irracional constante, sendo ele uma dízima infinita não periódica com valor aproximado de $3,1415926535\dots$
- **Quadrante:** é uma das quatro regiões de localização de pontos em uma parte ($1/4$) de um círculo trigonométrico. O primeiro quadrante (lado direito superior do círculo) possui abscissa positiva e ordenada positiva ($0^\circ < \text{ângulo} < 90^\circ$). O segundo quadrante (lado esquerdo superior do círculo) possui abscissa negativa e ordenada positiva ($90^\circ < \text{ângulo} < 180^\circ$). O terceiro quadrante (lado esquerdo inferior do círculo) possui abscissa negativa e ordenada negativa ($180^\circ < \text{ângulo} < 270^\circ$). O quarto quadrante (lado direito inferior do círculo) possui abscissa positiva e ordenada negativa ($270^\circ < \text{ângulo} < 360^\circ$).

- **Radiano:** unidade de medida usada para mensurar um trecho de pontos em uma circunferência (um arco) que tenha comprimento igual ao tamanho do raio da circunferência. É representada pelos símbolos *rad* ou c (letra c sobrescrita). Uma circunferência (ângulo completo) equivale a $2\pi \text{ rad}$ ($360^\circ = 2\pi \text{ rad}$). Assim, 1 *rad* é igual a 360° divididos por 2π , resultando, aproximadamente, em $57,29577951^\circ$ ou, ainda, 1° é igual a 2π divididos por 360° , resultando, aproximadamente, em $0,01745329 \text{ rad}$.
- **Radianos para graus** – use a expressão aritmética: **graus = (rad * 180)/π.**
- **Raio:** é o segmento de reta que liga o ponto central de uma circunferência ou esfera a um ponto qualquer dessa circunferência ou esfera, tendo como limite o extremo da circunferência ou esfera. É a medida do comprimento (extensão) do segmento, sendo representada pelo símbolo *r* (letra erre).
- **Teta (θ):** letra grega usada para representar a medida de um ângulo.
- **Variável:** neste trabalho, uma variável pode ser entendida sob duas óticas: a computacional ou a matemática. Na ótica computacional, variável é uma região da memória de trabalho de um computador em que certo valor é armazenado, podendo ser utilizado ou alterado por certo espaço de tempo. Na ótica matemática, variável pode ser entendida como a representação de um valor desconhecido (incógnito) que será substituído em uma fórmula matemática. Na ótica matemática, as variáveis são representadas por letras isoladas ou símbolos. Na ótica computacional, as variáveis são representadas por letras, letras combinadas com números ou palavras, sempre sem espaços em branco.
- **Variável dependente:** em uma função $f(x) = x$, a incógnita $f(x)$, (entendida como *y*) é a variável dependente da função, cujo valor é alterado conforme a manipulação da função.
- **Variável independente:** em uma função $f(x) = x$, a incógnita *x* é a variável independente da função, podendo ser manipulada diretamente.
- **Vértice de uma parábola:** corresponde ao ponto mais extremo da parábola, no qual a reta muda o sentido da curva de forma inversa ao sentido atual.

* * * Anotações * * *

2

Gráficos bidimensionais

Este capítulo são apresentados o uso de gráficos gerados a partir de funções de primeiro grau, de segundo grau, exponenciais e trigonométricas. À medida que os gráficos são apresentados, definem-se alguns recursos de ajustes do programa *gnuplot*, como *set xtics*, *set ytics*, *set mxtics*, *set mytics*, *set grid*, *set xrange*, *set yrange*, *set title*, *set ylabel*, *set xlabel*, *set zeroaxis*, *set size*, *set origin*, *set multiplot*, *reset* e *plot*.

2.1 Gráfico de função de primeiro grau

A geração de gráficos a partir de funções de primeiro grau (funções lineares) caracteriza-se em obter-se a imagem de uma reta a partir do uso da função em sua forma geral:

$$f(x) = ax + b$$

Onde o coeficiente **a** da variável **x** representa a coordenada das abscissas (eixo **x**) e **f(x)** representa a coordena das ordenadas (eixo **y**). A partir da função $f(x) = ax + b$ é possível fazer uso de algumas variações da mesma, tais como:

$$f(x) = x$$

$$f(x) = ax$$

$$f(x) = x + b$$

Se o valor de **a** que é o coeficiente angular da função for menor que zero a função é decrescente, se maior que zero a função é crescente e se igual a zero a função é constante no eixo **y**, mantendo-se paralela ao eixo **x**.

Para fazer a entrada da função no programa **gnuplot** basta substituir **f(x)** pelo comando **plot**, omitir o símbolo de atribuição **=**, informar a equação definida para a função e acionar a tecla **<Enter>** a cada entrada Desta maneira, informe para a linha de *prompt* do programa **gnuplot** a instrução:

```
plot x
```

Observe na imagem do gráfico apresentado junto Figura 2.1 com uma função crescente.

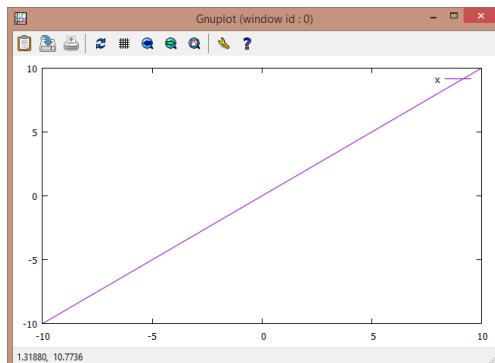


Figura 2.1 – Gráfico gerado a partir da função $f(x) = x$.

Note junto a Figura 2.1 a apresentação do gráfico a partir da função **f(x) = x** por meio do uso da instrução **plot x** que tem por finalidade fazer a apresentação de um gráfico de duas dimensões (2D). O argumento **x** do comando **plot** é uma variável interna definida e reservada no programa para a operação sobre os gráficos.

Observe na Figura 2.1 a distribuição automática dos valores das coordenadas **x** e **y** de cinco em cinco, sendo que esta escala pode ser alterada. Para tanto, basta fazer uso do comando **set** com os complementos **ytics** (mudança da escala do eixo das ordenadas) e **xtics** (mudança da escala do eixo das abscissas). Assim sendo, execute as seguintes linhas de comandos.

```
set xtics 3
set ytics 4
plot x
```

A sequência de comandos acima estabelece que a escala da coordenada x será grafada de três em três e a escala da coordenada Y será grafada de quatro em quatro, como mostra a Figura 2.2.

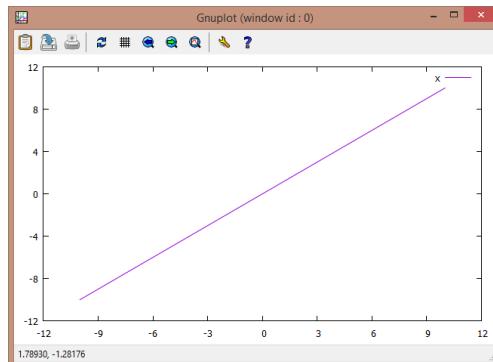


Figura 2.2 – Gráfico da função $f(x) = x$ com escala das coordenadas alteradas.

Quando se usa o comando **set** com algum complemento é preciso ficar atento para o fato das alterações estabelecidas ficarem ativas para qualquer gráfico. Caso queira retornar as configurações ao modo padrão do programa é necessário fazer uso do comando **reset**.

Os gráficos apresentados se caracterizam por serem gráficos gerados a partir de uma função crescente, pois a variável **a** (coeficiente angular de x) quando omitida tem seu valor definido automaticamente como **1**, ou seja, um valor positivo.

Execute a seguir os comandos indicados para a apresentação de um gráfico a partir de uma função decrescente, como mostra a Figura 2.3.

```
reset  
plot -x
```

A Figura 2.3 mostra um gráfico com reta decrescente, uma vez que para cada valor negativo do eixo x se obtém na mesma proporção um valor negativo para o eixo y.

Compare as imagens das Figuras 2.1 e 2.3, observe a diferença entre uma função crescente de uma decrescente. Note que, na Figura 2.1, a reta do gráfico está no sentido crescente, pois à medida que se altera o valor positivo do eixo x, um valor positivo é definido para o eixo y, em igual proporção.

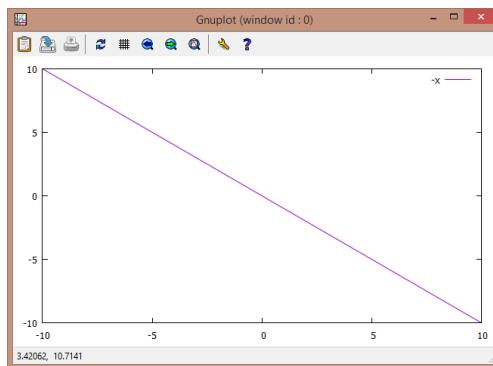


Figura 2.3 – Gráfico decrescente a partir da função $f(x) = -x$.

A título de exemplificação considere no programa **gnuplot** a execução das funções $f(x) = ax$ e $f(x) = x + b$.

Para a função $f(x) = ax$ execute no *prompt* do programa **gnuplot** as instruções:

```
a = 1
plot a * x
```

A execução da ação para a função $f(x) = ax$ apresenta um gráfico como indicado na Figura 2.4, a partir dos comandos seguintes.

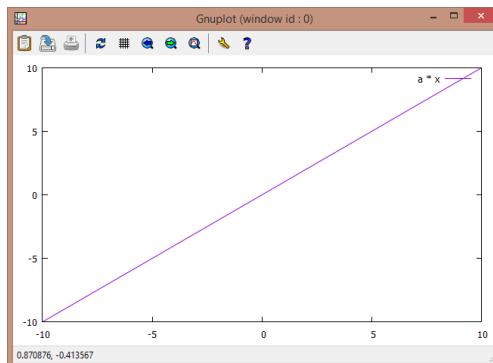


Figura 2.4 – Gráfico gerado a partir da função $f(x) = ax$.

O coeficiente **a** definido como **a = 1** da função $f(x) = ax$ pode ser substituído por um valor constante numérico específico diretamente na instrução de definição. Observe o uso do valor **5** no lugar do coeficiente **a** tendo-se, neste caso, a função $f(x) = 5x$ indicada como:

```
set grid  
plot 5 * x
```

Note o uso do complemento **grid** junto ao comando **set** com a finalidade de mostrar uma grade no plano cartesiano, ou seja, apresentar a retícula das escalas entre os eixos **x** e **y**. Note que a apresentação da grade auxilia a leitura dos valores no gráfico. Considerando-se para **x** o valor **5** ter-se-á para **y** (ou seja, $f(x)$) o valor **25** como pode ser constatado junto a Figura 2.5.

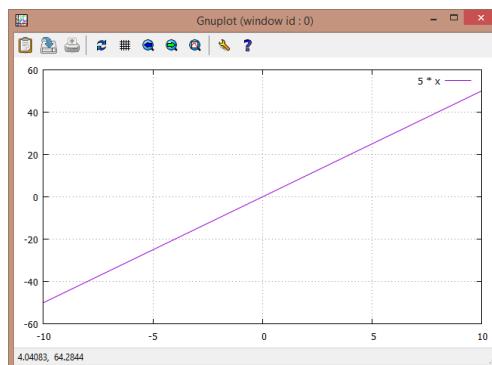


Figura 2.5 – Gráfico gerado a partir da função $f(x) = 5x$.

Para a função $f(x) = x + b$ execute no *prompt* do programa **gnuplot** as instruções:

```
reset  
b = 1  
plot x + b
```

A execução da ação para a função $f(x) = x + b$ com a incógnita **b** definida com o valor **1** por meio de **b = 1** apresenta um gráfico indicado na Figura 2.6.

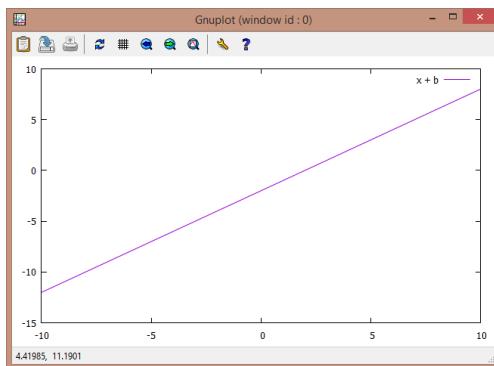


Figura 2.6 – Gráfico gerado a partir da função $f(x) = x + b$.

A função $f(x) = x + b$ pode ser usada com a substituição do coeficiente b por certo valor numérico. Considere o uso do valor **6** a parit das instruções:

```
set grid
plot x + 6
```

Que apresentará o gráfico indicado na Figura 2.7.

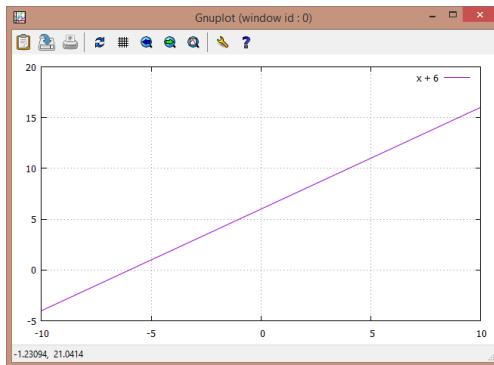


Figura 2.7 – Gráfico gerado a partir da função $f(x) = x + 6$.

Considerando-se para x o valor **5** ter-se-á para y , ou seja, $f(x)$, ou melhor $f(5)$ o valor **11**, se considerado para x o valor **0** ter-se-á para $f(0)$ o valor **6** como pode ser constatado junto a Figura 2.7.

A título de ilustração, e considerando representação de uma função de primeiro grau mais completa o comando seguinte representa a ação para a função $f(x) = ax + b$ e seu gráfico correspondente é apresentado na Figura 2.8.

```
plot 5 * x + 6
```

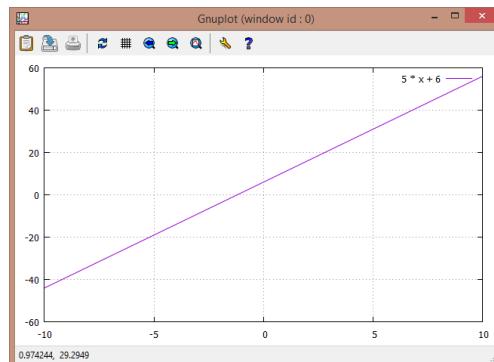


Figura 2.8 – Gráfico da função $f(x) = ax + b$.

Como não foi executado anteriormente antes de apresentar o gráfico o comando **reset** o último **set** com o complemento **grid** ficou ativo para o gráfico apresentado na Figura 2.8.

Não importam os valores definidos para os coeficientes **a** e **b**, o gráfico sempre é apresentado de maneira idêntica, seja ela crescente ou decrescente. A diferença na apresentação do gráfico é a escala da coordenada **y**.

2.2 Gráfico de função de segundo grau

A geração de gráficos a partir de funções de segundo grau caracteriza-se em obter-se a imagem de uma parábola a partir do uso da função na forma geral:

$$f(x) = ax^2 + bx + c$$

Onde o coeficiente **a** da variável **x** é o termo de segundo grau, o coeficiente **b** da variável **x** é o termo de primeiro grau os quais representam a coordenada das abscissas (eixo X) e **f(x)** que representa a coordena das ordenadas (eixo Y). A partir da função dada é possível fazer uso de algumas variações da mesma, tais como:

$$f(x) = x^2$$

$$f(x) = ax^2$$

$$f(x) = ax^2 + bx$$

$$f(x) = ax^2 + c$$

Se o valor de a que é o coeficiente angular da função for menor que zero a função terá sua parábola com concavidade voltada para baixo, se maior que zero a função terá sua parábola com concavidade voltada para cima e se igual a zero a função obtida não é de segundo grau, com constante nula.

Ao fazer uso da função $f(x) = x^2$ ter-se-á como imagem gráfica o que é apresentado na Figura 2.9. Execute para tanto o seguinte comando, observe que os valores do eixo x são mostrados em escala positiva e negativa e os valores do eixo y são mostrados em escala positiva.

```
plot x ** 2
```

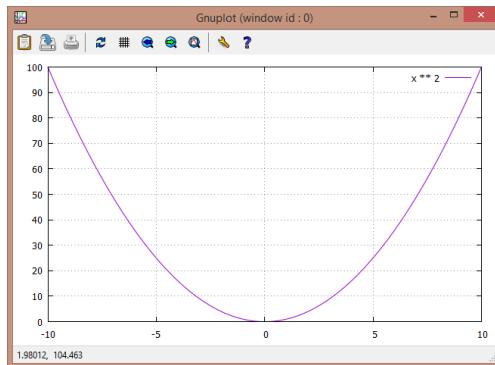


Figura 2.9 – Gráfico gerado a partir da função $f(x) = x^2$ (concavidade para cima).

Para fazer um teste de apresentação com concavidade para baixo, utilize a função $f(x) = -x^2$ a partir da instrução:

```
plot -x ** 2
```

Que apresentará a imagem gráfica da Figura 2.10.

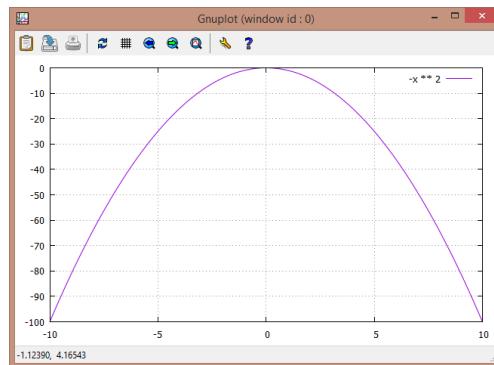


Figura 2.10 – Gráfico gerado a partir da função $f(x) = -x^2$ (concavidade para baixo).

Observe que os valores do eixo x são apresentados nas escalas positiva e negativa, ao passo que os valores do eixo y são exibidos em escala negativa.

Para o próximo gráfico será definido além do uso da função $f(x) = ax^2$, um título de identificação para o mesmo. Assim sendo, execute os comandos seguintes e veja o resultado de acordo com a Figura 2.11.

```
set title "Gráfico com Parábola"  
plot 5 * x ** 2
```

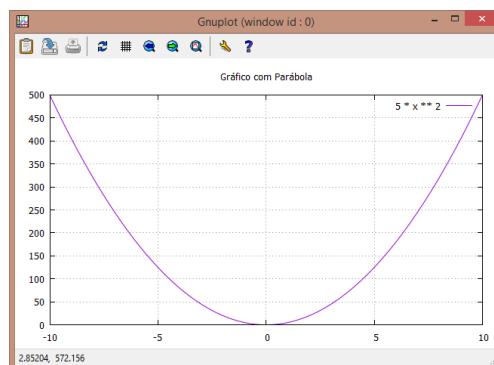


Figura 2.11 – Gráfico gerado a partir da função $f(x) = ax^2$.

Neste exemplo está sendo utilizado o complemento **title** que permitiu a definição de um título de identificação para o gráfico a partir do comando **set**. Lembre-se que quando se utiliza o comando **set** a configuração definida fica ativada, devendo ser desabilitada pelo comando **reset**.

O próximo gráfico será definido a partir da função $f(x) = ax^2 + c$. Assim sendo, execute os comandos seguintes e veja o resultado de acordo com a Figura 2.12.

```
reset
set grid
set title "Parábola"
plot 5 * x ** 2 + 6
```

Observe na Figura 2.12 o comportamento da coordenada y . Note a diferença de valores na escala apresentada em relação aos demais gráficos deste tópico.

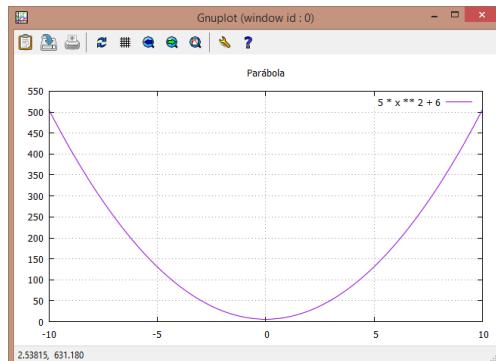


Figura 2.12 – Gráfico gerado a partir da função $f(x) = ax^2 + b$.

A função $f(x) = ax^2 + bx$ apresenta outra variação de gráfico de segundo grau. Note a imagem apresentada na Figura 2.13 a partir dos seguintes comandos.

```
reset
set grid
set title "Gráfico de Segundo Grau"
plot 5 * x ** 2 + 6 * x
```

Outra maneira é o uso da função em sua forma completa $f(x) = ax^2 + bx + c$. A Figura 2.14 mostra o gráfico gerado a partir dos seguintes comandos.

```
reset
set grid
set title "Função Completa de Segundo Grau"
plot 5 * x ** 2 + 6 * x + .5
```

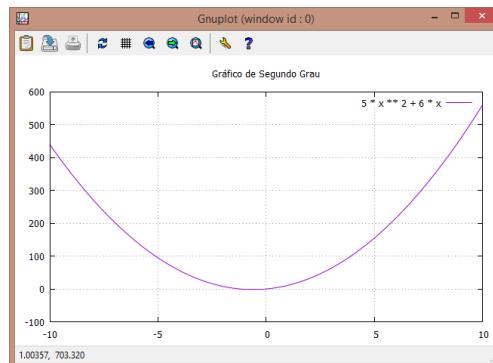


Figura 2.13 – Gráfico gerado a partir da função $f(x) = ax^2$.

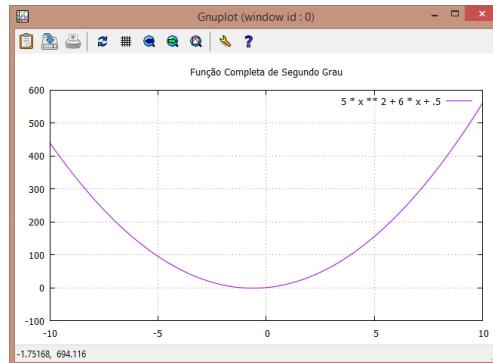


Figura 2.14 – Gráfico gerado a partir da função $f(x) = ax^2 + bx + c$.

2.3 Gráfico de função exponencial

A geração de gráficos a partir de funções exponenciais ocorre a partir de funções do tipo:

$$f(x) = a^x$$

$$f(x) = e^x$$

Na primeira função, a variável a é a base a ser elevada e o coeficiente de x é o expoente usado para elevar a base. Uma função exponencial é crescente quando o valor da variável é $a > 1$ e decrescente se a variável a é um valor numérico entre 0 e 1 ($0 < a < 1$).

A segunda função utiliza a constante e eleva a x .

Ao fazer uso da função $f(x) = a^x$ ter-se-á como imagem gráfica o visual apresentado na Figura 2.15 com uma parábola crescente. Execute as instruções:

```
reset
set grid
plot 2 ** x
```

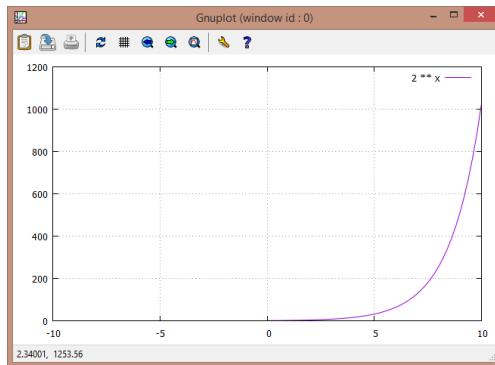


Figura 2.15 – Gráfico gerado a partir da função $f(x) = a^x$ (crescente).

Para a apresentação de uma parábola decrescente em um gráfico exponencial, é preciso utilizar a função $f(x) = a^x$, com os comandos a seguir:

```
reset  
set grid  
set xrange [-8:0]  
set yrange [0:200]  
plot .5 ** x
```

A Figura 2.16 gerada a partir da função $f(x) = a^x$, utiliza os complementos `xrange` e `yrange` com o comando `set`, a fim de modificar, respectivamente, o domínio (eixo x) e o contradomínio (eixo y) da função. Caso queira habilitar a escala automática, use `set autoscale x` para o eixo x ou `set autoscale y` para o eixo y. Ao usar apenas `set autoscale`, ambos os eixos são habilitados. Para desabilitar um deles, ou ambos, use `set noautoscale x`, `set noautoscale y` ou `unset autoscale`.

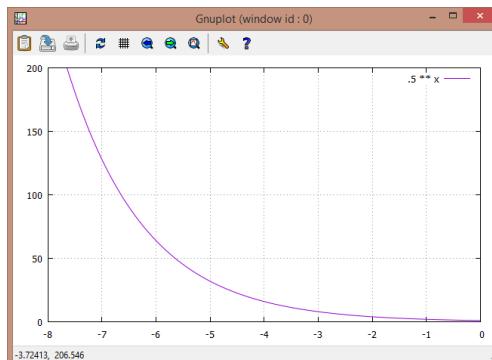


Figura 2.16 – Gráfico gerado a partir da função $f(x) = a^x$ (decrescente).

Atente para a mudança dos valores da escala na apresentação das coordenadas x e y. Note que as instruções `set xrange` e `set yrange` podem fazer uso de valores positivos ou negativos. O uso de tais valores irá depender do quadrante do gráfico a ser utilizado. O programa `gnuplot` sem o uso das instruções `set xrange` e `set yrange` desenha o gráfico utilizando-se para a ordenada um valor automático de escala e para a abscissa valores no limite de -10 até 10.

A função $f(x) = e^x$ gera a imagem do gráfico indicado na Figura 2.17 a partir das instruções:

```
reset
set grid
set xrange [-5:5]
set yrange [0:15]
plot exp(x)
```

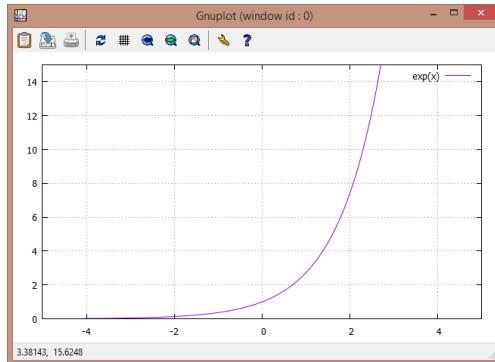


Figura 2.17 – Gráfico gerado a partir da função $f(x) = e^x$.

Em seguida experimente fazer uso da função $f(x) = e^{-x}$ por meio da instrução:

```
plot exp(-x)
```

2.4 Gráfico de função trigonométrica

A geração de gráficos a partir de funções trigonométricas ocorre a partir do uso das funções trigonométricas: cossecante (**cosec**), cosseno (**cos**), cotangente (**cot**), secante (**sec**), seno (**sen**) e tangente (**tg**) definidas como:

$$f(x) = \text{função } x$$

Onde **função** é a definição de uma função trigonométrica matemática a ser utilizada e **x** sendo o valor usado para a coordenada **x**.

Para fazer este tipo de operação o programa **gnuplot** possui um extenso conjunto de funções internas (*função gnuplot*), destacando-se entre elas as funções:

Função Trigonométrica (Matemática)	Função Trigonométrica (gnuplot)
cosec x	$1 / \sin(x)$
cos x	$\cos(x)$
cot x	$1 / \tan(x)$ ou $\cos(x) / \sin(x)$
sec x	$1 / \cos(x)$
sen x	$\sin(x)$
tg x	$\tan(x)$ ou $\sin(x) / \cos(x)$

A função $f(x) = \text{cosec } x$ apresenta a imagem do gráfico indicado na Figura 2.18. Execute os seguintes comandos.

```
reset
set grid
set xrange [0:18]
set yrange [0:20]
plot 1 / sin(x)
```

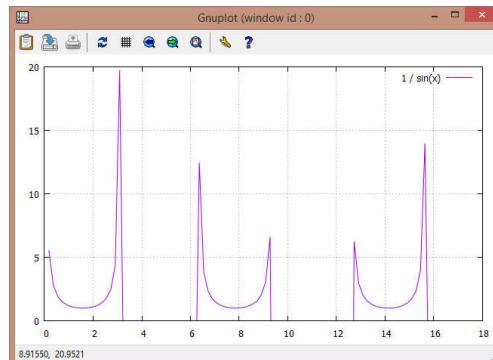


Figura 2.18 – Gráfico gerado a partir da função $f(x) = \text{cosec } x$.

A função $f(x) = \cos x$ apresenta a imagem do gráfico indicado na Figura 2.19. Execute os seguintes comandos.

```
reset
set grid
plot cos(x)
```

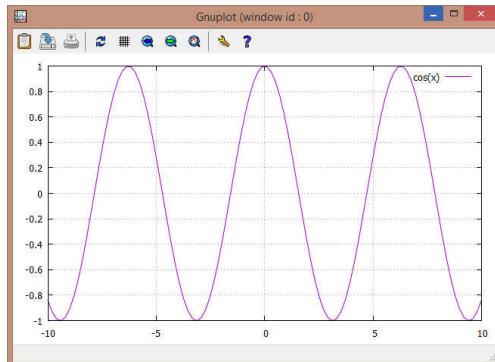


Figura 2.19 – Gráfico gerado a partir da função $f(x) = \cos x$.

A função $f(x) = \cot x$ apresenta a imagem do gráfico indicado na Figura 2.20. Execute os seguintes comandos.

```
reset
set grid
plot 1 / tan(x)
```



Figura 2.20 – Gráfico gerado a partir da função $f(x) = \cot x$.

A função $f(x) = \sec x$ apresenta o gráfico da Figura 2.21 a partir dos comandos.

```
reset  
set grid  
plot 1 / cos(x)
```

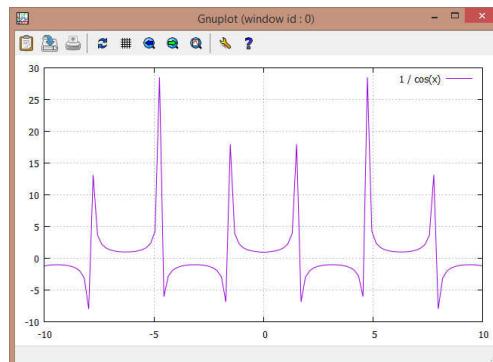


Figura 2.21 – Gráfico gerado a partir da função $f(x) = \sec x$.

A função $f(x) = \operatorname{sen} x$ apresenta o gráfico da Figura 2.22 a partir dos comandos.

```
reset  
set grid  
plot sin(x)
```

A função $f(x) = \operatorname{tg} x$ apresenta o gráfico da Figura 2.23 a partir dos comandos.

```
reset  
set grid  
plot tan(x)
```

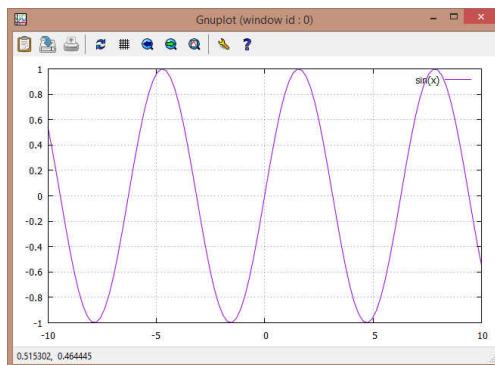


Figura 2.22 – Gráfico gerado a partir da função $f(x) = \sin x$.



Figura 2.23 – Gráfico gerado a partir da função $f(x) = \tan x$.

2.5 Gráfico de função trigonométrica inversa

Uma função trigonométrica normal, possuirá função inversa somente se a função for bijetora. Nem todas as funções trigonométricas possuem funções inversas em seus domínios de definição. No entanto, há a possibilidade de considerar subconjuntos dos domínios para obter-se funções inversas.

A geração de gráficos a partir de funções trigonométrica inversas ocorre a partir do uso das funções: arco cujo cosseno (**arccos**), arco-cotangente (**arccot**), arco cujo seno (**arcsen**) e arco-tangente (**arctg**) e definidas como:

$$y = \text{função } x$$

Onde **função** é a definição de uma função trigonométrica inversa a ser utilizada, sendo **x** o valor usado para a coordenada **x** e **y** o valor da coordenada **y**.

Para fazer este tipo de operação o programa **gnuplot** possui um extenso conjunto de funções internas (*função gnuplot*), destacando-se entre elas as funções:

Função Trigonométrica Inversa (Matemática)	Função Trigonométrica Inversa (gnuplot)
arccos x	acos(x)
arccosec x	atan(1 / sqrt(1 - x ** 2)) + (sgn(x) - 1) * pi / 2
arccot x	-atan(x) + pi / 2
arcSEN x	asin(x) ou atan(x / sqrt(1 - x ** 2))
arcsec x	atan(x / sqrt(1 - x ** 2)) + (sgn(x) - 1) * pi / 2
arctg x	atan(x)

A função $f(x) = \text{arccos } x$ apresenta a imagem do gráfico indicado na Figura 2.24. Execute os seguintes comandos.

```
reset
set grid
set xrange [-1:1]
set yrang [0:pi]
plot acos(x)
```

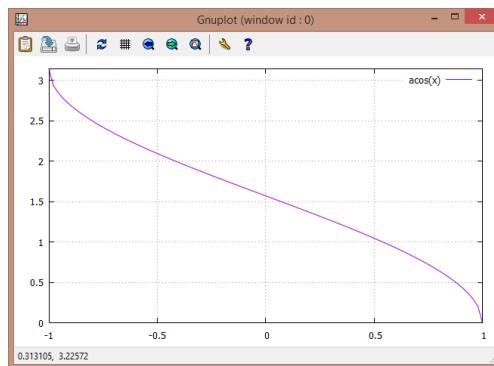


Figura 2.24 – Gráfico gerado a partir da função $f(x) = \arccos x$.

A função $f(x) = \text{arccosec } x$ apresenta a imagem do gráfico indicado na Figura 2.25. Execute os seguintes comandos.

```
reset
set grid
set ylabel "Título eixo Y"
set xlabel "Título eixo X"
set yrange [0:pi/2]
plot atan(1 / sqrt(1 - x ** 2)) + (sgn(x) - 1) * pi / 2
```

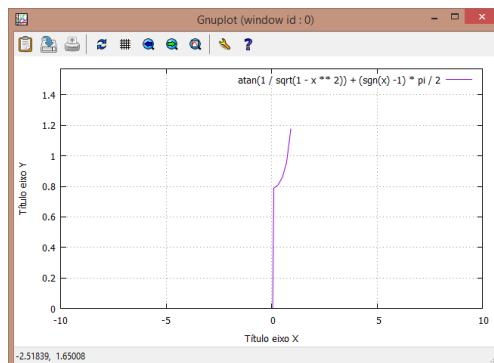


Figura 2.25 – Gráfico gerado a partir da função $f(x) = \text{arccosec } x$.

Note o uso dos complementos **set ylabel** e **set xlabel** com o comando **set** para o estabelecimento dos títulos dos eixos **x** (**xlabel**) e **y** (**ylabel**). Neste exemplo está sendo definida apenas a faixa **yrange**, a faixa **xlabel** está sendo definida de forma automática.

A função **gnuplot sgn(x)** é uma função matemática especial que fornece os valores **-1**, **0** ou **1** como resposta dependendo do valor avaliado representado por **x**. Se o valor de **x** for menor que **0** é retornado **-1**, se o valor de **x** for igual a **0** é retornado **0** e se o valor de **x** for menor que **0** será retornado o valor **1**.

A função **f(x) = arccot x** apresenta a imagem do gráfico indicado na Figura 2.26. Execute os seguintes comandos.

```
reset
set grid
set ylabel "Título eixo Y"
set xlabel "Título eixo X"
set yrange [0:pi]
plot -atan(x) + pi / 2
```

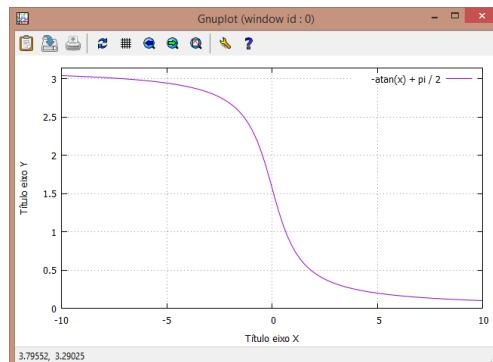


Figura 2.26 – Gráfico gerado a partir da função $f(x) = \text{arccot } x$.

A função **f(x) = arcsen x** apresenta a imagem do gráfico indicado na Figura 2.27. Execute os seguintes comandos.

```
reset
set grid
set ylabel "Título eixo Y"
set xlabel "Título eixo X"
set xrange [-1:1]
set yrange [-pi/2:pi/2]
plot asin(x)
```

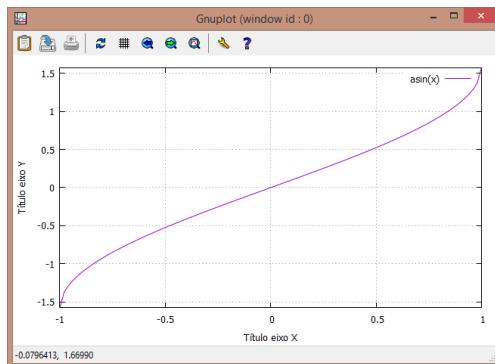


Figura 2.27 – Gráfico gerado a partir da função $f(x) = \arcsen x$.

A função $f(x) = \text{arcsec } x$ apresenta a imagem do gráfico indicado na Figura 2.28. Execute os seguintes comandos.

```
reset
set grid
set yrange [0/pi]
plot atan(x / sqrt(1 - x ** 2)) + (sgn(x) - 1)* pi / 2
```

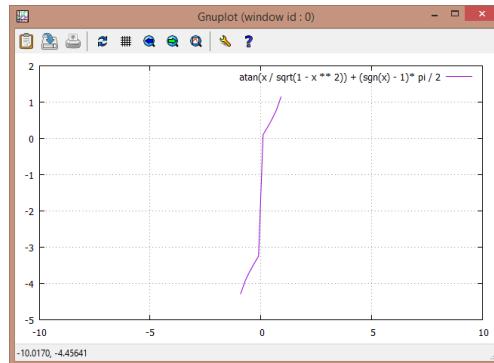


Figura 2.28 – Gráfico gerado a partir da função $f(x) = \text{arcsec } x$.

A função $f(x) = \arctan x$ apresenta a imagem do gráfico indicado na Figura 2.29 com eixo x com auto escala e eixo y fixado. Execute os seguintes comandos.

```
reset
set grid
set autoscale x
set yrang [-pi/2:pi/2]
plot atan(x)
```

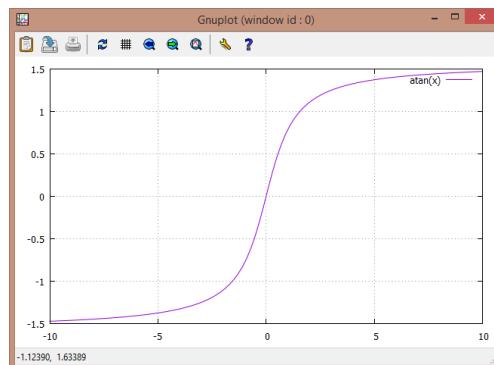


Figura 2.29 – Gráfico gerado a partir da função $f(x) = \arctan x$.

2.6 Mais de um gráfico

O programa **gnuplot** permite que mais de um gráfico seja apresentado no mesmo plano cartesiano de modo a sobrepor gráficos para uma eventual comparação entre esses gráficos.

O exemplo a seguir fará uso de todos os recursos apresentados neste capítulo, a fim de que o leitor tenha noção do conjunto mínimo de comandos do **gnuplot** a serem utilizados na elaboração de gráficos.

Observe atentamente o conjunto de funções indicadas a seguir. Execute no *prompt* de comando os comandos indicados e observe junto a Figura 2.30 as imagens dos gráficos sobrepostos.

A instrução responsável por estabelecer a sobreposição dos gráficos é o próprio comando **plot**, mas com separação das funções por vírgulas, como indicado a seguir (x , $x^{**} 2$, $-x$).

```
reset
set xtics 1
set ytics 1
set xrange [-2:8]
set yrange [-1:10]
set title "Gráficos Sobrepostos"
set ylabel "Título Y"
set xlabel "Título X"
set zeroaxis
plot x, x ** 2, -x
```

Observe que para cada reta é eleita pelo programa uma cor diferente para sua representação. As cores dessas retas podem ser escolhidas pelo usuário, sendo este tema tratado no próximo capítulo.

Note que neste exemplo não fora usado o comando **set grid**, em seu lugar foi usado o complemento **zeroaxis** do comando **set** para traçar apenas as linhas que marcam os eixos **x** e **y** no ponto em que se encontram, ou seja, na coordenada **0,0**. Caso

queira a apresentação apenas da coordenada zero **x** ou **y** use as instruções **set xzeroaxis** ou **set yzeroaxis**.

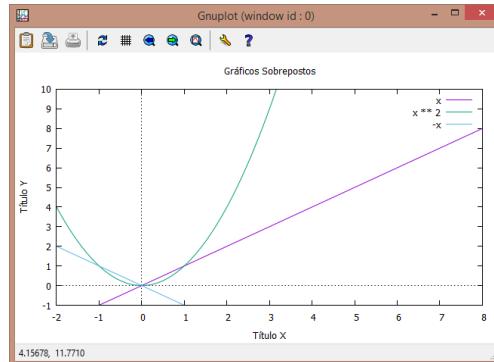


Figura 2.30 – Sobreposição de gráficos.

Um detalhe complementar é o uso aprimorado dos marcadores de escala do gráfico. Já foram apresentadas as instruções **set xtics** e **set ytics**, que permitem ajustar a escala das coordenadas. No entanto, há mais duas instruções que podem auxiliar na demarcação de um gráfico, que são **set mxtics** e **set mytics**. Elas permitem marcar os *tics* entre **xtics** e **ytics**.

Para que seja percebido o efeito de uso dos ajustes dos *tics* será apresentado o mesmo gráfico desenhado a partir da função **int(x)** que é apresentada no capítulo 4. Na sequencia são executadas três etapas de ajustes no gráfico.

A primeira etapa mostra o gráfico com *tics* padrão, como indica a Figura 2.31. Para tanto execute as instruções a seguir.

```
reset
set xrange [0:8]
set yrange [0:6]
plot int(x)
```

A segunda etapa exibe o gráfico com *tics* padrão, ajustados com **xtics** e **ytics**, como indica a Figura 2.32. Para tanto, execute as instruções a seguir:

```
reset  
set xrange [0:8]  
set yrange [0:6]  
set xtics 2  
set ytics 2  
plot int(x)
```

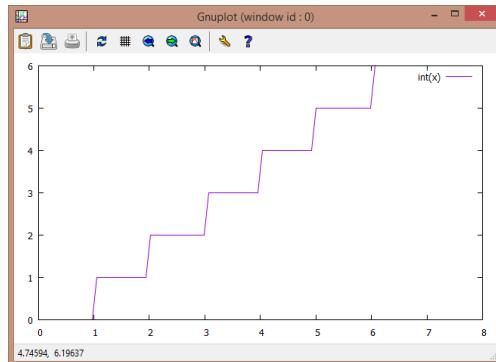


Figura 2.31 – Gráfico com tics padrão.

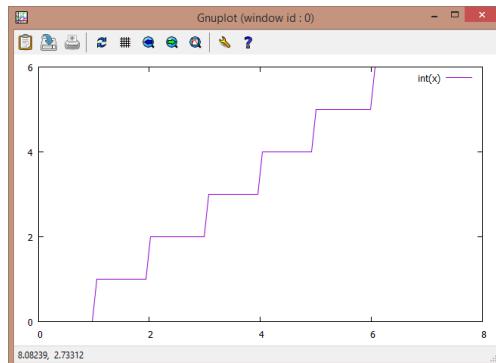


Figura 2.32 – Gráfico com tics padrão ajustados de 2 em 2.

Em relação as Figuras 2.31 e 2.32 note que nas coordenadas da Figura 2.32 não são apresentados os números 1, 3, 5 e 7.

Terceira etapa mostra o gráfico com *tics* padrão ajustados com **xtics** e **ytics** e definidos os *tics* intermediários com **mxtics** e **mytics**, como mostra a Figura 2.33. Para tanto execute as instruções a seguir.

```
reset  
set noautoscale x  
set noautoscale y  
set xrange [0:8]  
set yrange [0:6]  
set xtics 2  
set ytics 2  
set mxtics 2  
set mytics 2  
plot int(x)
```

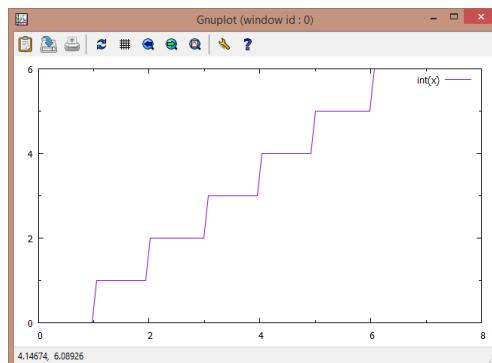


Figura 2.33 – Gráfico com tics intermediários ao tics padrão.

Observe na Figura 2.33 a apresentação dos *tics* nas coordenadas exatamente onde haviam na Figura 2.31 os números 1, 3, 5 e 7. Este é um efeito que visa melhorar a apresentação da imagem do gráfico.

Anteriormente, ensinou-se a desenhar dois gráficos sobrepostos, o que é muito útil para fazer análises e comparações. No entanto, há outra forma de fazer isso, com a qual é possível apresentar mais de um gráfico na mesma tela, mas de forma

separada e sem sobreposição, por meio da instrução **set multiplot**. Desta forma, considere as instruções a seguir e veja o resultado da ação na Figura 2.34.

```
reset
set multiplot
set size 1, 0.5
set origin 0, 0.5; plot sin(x)
set origin 0, 0; plot tan(x)
unset multiplot
```

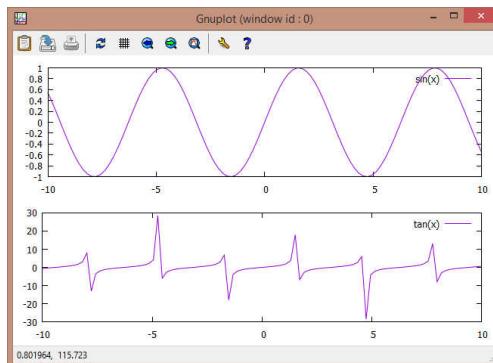


Figura 2.34 – Mais de um gráfico na mesma tela gráfica.

Para o efeito da apresentação de dois ou mais gráficos na mesma tela gráfica é necessário ativar o complemento **multiplot** do comando **set** e desativá-lo com o comando **unset**.

O complemento **size** do comando **set** é usado para dividir a tela em duas áreas. A medida padrão do complemento **size** é **1,1**, em que o primeiro **1** refere-se ao tamanho da tela e o segundo **1**, à escala que determina a relação com o tamanho da tela. A instrução **set size 1.0, 0.5** mantém, respectivamente, o tamanho da tela em **1** e a área de desenho com a proporção da metade da tela. Dessa maneira, diminui-se a área útil de uso.

O tamanho padrão da tela vai de **0.0** a **1.1**, sendo a posição **0.0** o lado esquerdo inferior; a posição **1.1**, o lado direito superior da tela; a posição **0.1**, o lado esquerdo superior; e **1.0**, o lado direito inferior.

Em seguida as instruções **set origin** definem o local da tela onde os gráficos serão desenhados. O argumento com a indicação **set origin 0, 0.5** é usado para acessar a metade superior da tela gráfica e o argumento com a indicação **set origin 0, 0** é usado para acessar a metade inferior da tela.

Em conjunto com a instrução **set origin** está sendo usada a instrução **plot** na mesma linha. Observe que entre as instruções está sendo utilizado um símbolo de ponto e vírgula para separar essas instruções. Quando desejar escrever mais uma instrução em uma mesma linha use o ponto e vírgula para separá-las.

Como mais um exemplo deste recurso considere apresentar em uma mesma tela gráfica quatro diferentes gráficos. Assim sendo, execute as instruções a seguir e observe na Figura 2.35 a apresentação deste efeito.

```
reset
set grid
set mxtics 10
set mytics 5
set size 1,1
set origin 0,0
set multiplot
set size 0.5,0.5; set origin 0, 0.5; plot cos(x)
set size 0.5,0.5; set origin 0, 0; plot 1/cos(x)
set size 0.5,0.5; set origin 0.5, 0.5; plot sin(x)
set size 0.5,0.5; set origin 0.5, 0; plot 1/sin(x)
unset multiplot
```

Para este exemplo, deixa-se o tamanho da tela em seu modo padrão, por meio da instrução **set size 1, 1**, e definem-se quatro áreas de desenho com as instruções **set size 0.5, 0.5**. Cada área desenha um gráfico, elencado de acordo com as instruções que determinam sua posição, como segue:

- **set origin 0, 0.5** – determina a primeira área (lado esquerdo superior);
- **set origin 0, 0** – determina a segunda área (lado direito superior);
- **set origin 0.5, 0.5** – determina a terceira área (lado esquerdo inferior);

- **set origin 0.5, 0** – determina a quarta área (lado direito inferior).

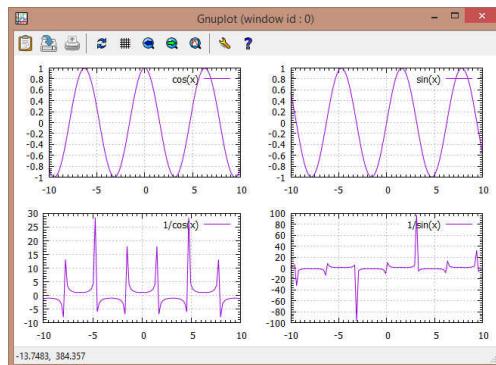


Figura 2.35 – Quatro gráficos na mesma tela gráfica.

2.7 Variáveis e gráficos

Os gráficos apresentados anteriormente atuaram apenas sobre a variável interna x , mas é possível fazer uso de gráficos a partir de uso de variáveis definidas pelo próprio usuário.

Considere a função $f(x) = ax + b$ definida de maneira um pouco diferente do que fora feito, sendo $f(x) = 3 * x + 7$ com a definição dos coeficientes a e b definidos como variáveis atribuídas as valores específicos. Por exemplo, execute a instrução seguinte.

```
reset
set grid
plot a = 3, b = 7, a * x + b
```

A Figura 2.36 apresenta o efeito de uso da função $f(x) = a * x + b$ com a atribuído ao valor 3 e b atribuído ao valor 7.

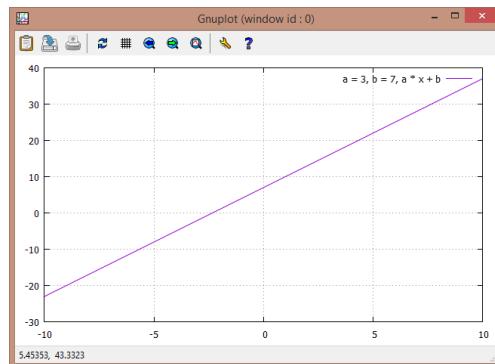


Figura 2.36 – Gráfico gerado a partir da função $f(x) = a * x + b$ com $a = 3$ e $b = 7$.

Note que as variáveis **a** e **b** receberam, respectivamente, os valores **3** e **7** no momento da chamada do comando **plot**. A definição dos valores é necessária uma vez que **a** e **b** são variáveis definidas pelo usuário e não no sistema, como ocorre com a variável **x** usada com o comando **plot**.

Desta maneira, é necessário associar a qualquer variável que não seja a variável do sistema valores para que sejam utilizadas pelo programa, pois ao contrário será apresenta mensagem de erro informando que certa variável em uso não se encontra definida no ambiente operacional do programa.

2.8 Gráficos de função de terceiro e quarto graus

A geração de gráficos a partir de funções de terceiro grau ou funções cúbicas caracteriza-se em obter-se a imagem de uma parábola a partir do uso da função na forma geral:

$$f(x) = ax^3 + bx^2 + cx + d$$

Onde o coeficiente **a** da variável **x** é o termo de terceiro grau, o coeficiente **b** da variável **x** é o termo de segundo grau, o coeficiente **c** da variável **x** é o termo de primeiro grau os quais representam a coordenada das abscissas (eixo **x**) e **f(x)** que representa a coordena das ordenadas (eixo **y**). A partir da função dada é possível fazer uso de algumas variações da mesma de maneira semelhante as variações apresentadas nas funções de primeiro e segundo graus.

Se o valor de **a** que é o coeficiente angular da função for menor que zero a função terá sua curva decrescente, se maior que zero a função terá sua curva crescente e se igual a zero a função será de segundo grau.

Ao fazer uso da função $f(x) = 2x^3 + 3x^2 + 4x + 5$ ter-se-á como imagem gráfica o que é apresentado na Figura 2.37. Execute para tanto o seguinte comando.

```
reset
set grid
plot 2 * x ** 3 + 3 * x ** 2 + 4 * x + 5
```

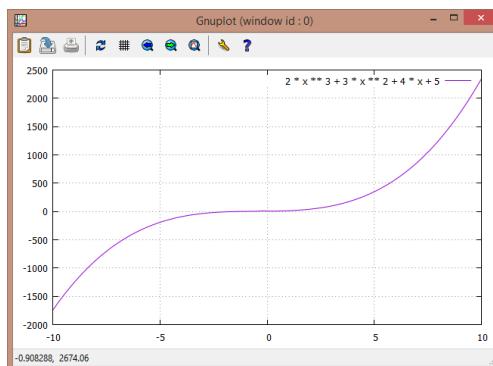


Figura 2.37 – Gráfico gerado a partir da função $f(x) = 2x^3 + 3x^2 + 4x + 5$.

Ao fazer uso da função $f(x) = -2x^3 + 3x^2 + 4x + 5$ ter-se-á como imagem gráfica o que é apresentado na Figura 2.38. Execute para tanto o seguinte comando.

```
reset
set grid
plot -2 * x ** 3 + 3 * x ** 2 + 4 * x + 5
```

A geração de gráficos a partir de funções de quarto grau caracteriza-se em obter-se a imagem de uma parábola a partir do uso da função na forma geral:

$$f(x) = ax^4 + bx^3 + cx^2 + dx + e$$

Onde o coeficiente **a** da variável **x** é o termo de quarto grau, o coeficiente **b** da variável **x** é o termo de terceiro grau, o coeficiente **c** da variável **x** é o termo de

segundo grau, o coeficiente **d** é coeficiente da variável x de primeiro grau os quais representam a coordenada das abscissas (eixo X) e $f(x)$ que representa a coordena das ordenadas (eixo Y). A partir da função dada é possível fazer uso de algumas variações da mesma de maneira semelhante as variações apresentadas nas funções de primeiro e segundo graus.

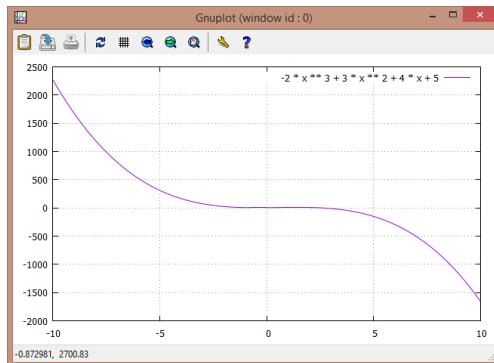


Figura 2.38 – Gráfico gerado a partir da função $f(x) = -2x^3 + 3x^2 + 4x + 5$.

Se o valor de **a** que é o coeficiente angular da função for menor que zero a função terá sua curva voltada para baixo, se maior que zero a função terá sua curva voltada para cima e se igual a zero a função será de terceiro grau.

Ao fazer uso da função $f(x) = 2x^4 + 3x^3 + 4x^2 + 5x - 6$ ter-se-á como imagem gráfica o que é apresentado na Figura 2.39. Execute para tanto o seguinte comando.

```
reset
set grid
plot 2 * x ** 4 + 3 * x ** 3 + 4 * x ** 2 + 5 * x - 6
```

Ao fazer uso da função $f(x) = -2x^4 + 3x^3 + 4x^2 + 5x - 6$ ter-se-á como imagem gráfica o que é apresentado na Figura 2.40. Execute para tanto o seguinte comando.

```
reset
set grid
plot -2 * x ** 4 + 3 * x ** 3 + 4 * x ** 2 + 5 * x - 6
```

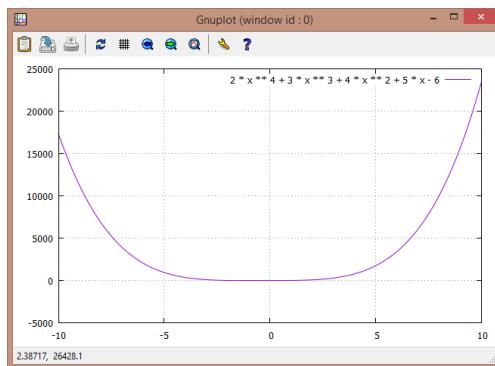


Figura 2.39 – Gráfico gerado a partir da função $f(x) = 2x^4 + 3x^3 + 4x^2 + 5x - 6$.

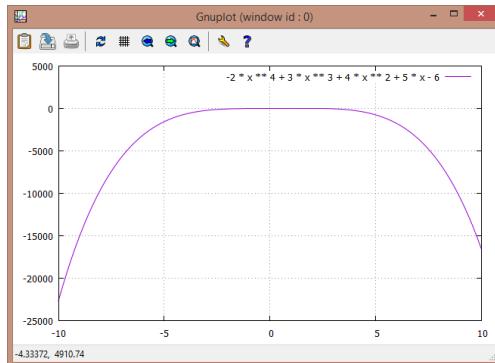


Figura 2.40 – Gráfico gerado a partir da função $f(x) = -2x^4 + 3x^3 + 4x^2 + 5x - 6$.

Há a possibilidade de se operar com o programa **gnuplot** gráficos desenhados a partir de equações do quinto grau no formato $f(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$, caso se tenha esta necessidade ou interesse.

2.9 Domínio e contradomínio especificados ou não

O programa **gnuplot** possibilita a definição fixada de domínio e contradomínio para o uso com funções. Neste sentido é possível definir gráficos de funções a partir de domínios e contradomínios fixados e livres separadamente ou em conjunto.

Para demonstrar a geração de um gráfico sobre duas funções com a definição de domínio fixado sobre o intervalo de valores $-\pi$ e π considere as funções $f(x) = x$ e $f(x) = x - x^3 / 5$ a partir das instruções:

```
reset
plot [-pi:pi] x, x - x ** 3 / 5
```

Observe a imagem do gráfico apresentado, como mostra a Figura 2.41.

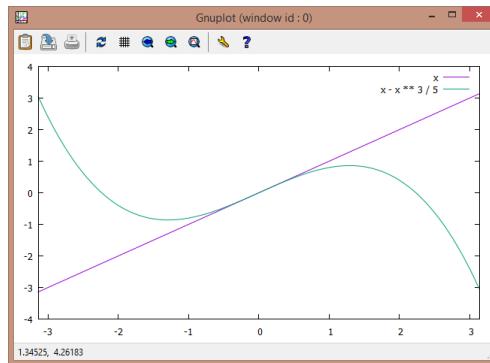


Figura 2.41 – Gráfico das funções $f(x) = x$ e $f(x) = x - x^3 / 5$ com domínio $-\pi$ e π .

Observe que a Figura 2.41 mostra o ponto de domínio entre a faixa de valores $-\pi$ e π onde as duas funções se encontram. Para a definição de valores de domínio usa-se a especificação desses valores após o comando **plot** e antes da definição das equações que geram os gráficos.

Além da definição de valores para a faixa de domínio é possível especificar valores para a faixa de contradomínio entre funções.

Para demonstrar a geração de um gráfico sobre duas funções com a definição de domínio e contradomínio sobre o intervalo de domínio $-\pi$ e π e contradomínio -2 e 2 considere as funções $f(x) = x$ e $f(x) = x - x^3 / 5$ a partir das instruções:

```
reset
plot [-pi:pi][-2:2] x, x - x ** 3 / 5
```

Observe a imagem do gráfico apresentado, como mostra a Figura 2.42.

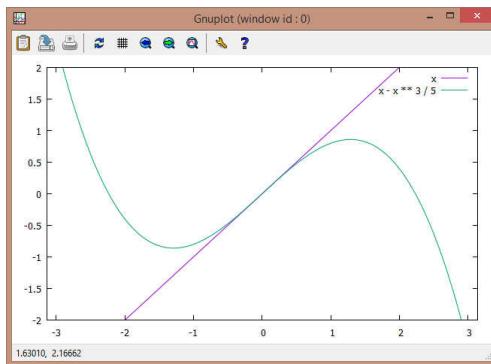


Figura 2.42 – Gráfico das funções $f(x) = x$ e $f(x) = x - x^3 / 5$ com domínio $-\pi$ e π e contradomínio -2 e 2 .

Note que a Figura 2.42 mostra o ponto de domínio e contradomínio respectivamente definidos entre as faixas de valores $-\pi$ e π com -2 e 2 onde as duas funções se encontram. Para a definição de valores de contradomínio usa-se a especificação desses valores após a definição dos valores de domínio após a uso do comando **plot**.

Os valores de domínio e contradomínio podem ser definidos de forma livre após o uso do comando **plot** e antes da definição das funções em uso. Basta manter os símbolos de colchetes sem a definição de valores internos. No caso específico de uso de valores para contradomínio livre, basta omitir os símbolos de colchetes como ocorreu com o gráfico indicado na Figura 2.41.

As instruções a seguir demonstram a geração de gráfico a partir das funções $f(x) = x$ e $f(x) = x - x^3 / 5$ com domínio fixado e contradomínio livre. Para tanto, execute as instruções:

```
reset
plot [-pi:pi][] x, x - x ** 3 / 5
```

Observe a imagem do gráfico apresentado, como mostra a Figura 2.43, sendo este idêntico a Figura 2.41.

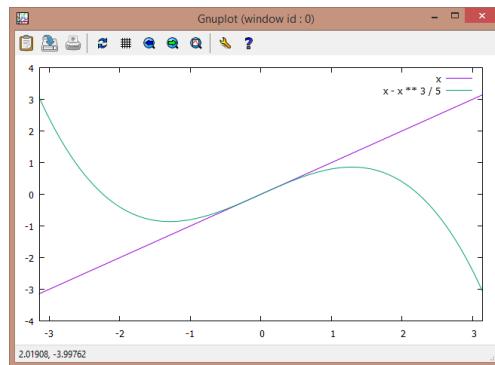


Figura 2.43 – Gráfico das funções $f(x) = x$ e $f(x) = x - x^3 / 5$ com domínio definido e contradomínio livre.

As instruções a seguir demonstram a geração de gráfico a partir das funções $f(x) = x$ e $f(x) = x - x^3 / 5$ com domínio livre e contradomínio fixado. Para tanto, execute as instruções:

```
reset
plot [][-2:2] x, x - x ** 3 / 5
```

Observe a imagem do gráfico apresentado, como mostra a Figura 2.44.

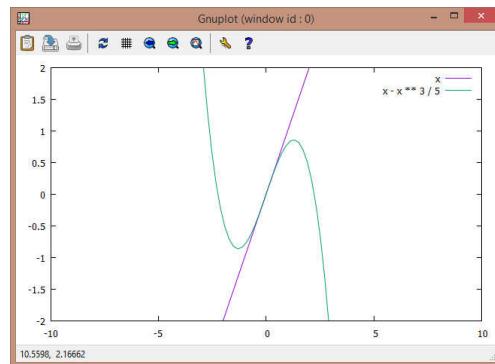


Figura 2.44 – Gráfico das funções $f(x) = x$ e $f(x) = x - x^3 / 5$ com domínio livre e contradomínio definido.

A omissão padrão dos valores de domínio e contradomínio são indicados com o uso de colchetes em branco antes do comando **plot** ou mesmo com a omissão dos

colchetes. Os gráficos mostrados neste capítulo, exceto os gráficos deste tópico são definidos com domínio e contradomínio livres.

A definição explícita de domínio e contradomínio pode ser usada para a geração de gráficos baseados em função modular $f(x) = |x|$ e função sinal $g(x) = \text{sgn}(x)$ com domínio fixado entre os valores -3 e 3 e contradomínio fixado entre os valores -2 e 2 a partir das instruções:

```
reset
plot [-3:3][-2:2] abs(x), sgn(x)
```

Observe a imagem do gráfico apresentado, como mostra a Figura 2.45.

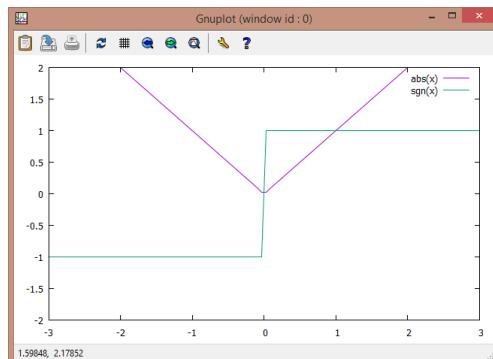


Figura 2.45 – Gráfico das funções $f(x) = |x|$ e $g(x) = \text{sgn}(x)$ com domínio livre e contradomínio definido.

Os gráficos apresentados na Figura 2.45 são produzidos a partir do uso da função interna **abs()** que gera os valores módulos (valores positivos) de um conjunto de valores negativos e a função **sgn()** que indica o valor -1 para uma série de valores negativos e 1 para uma série de valores positivos.

A definição de valores para domínio podem ser definidas com o uso da instrução **set xrange** e os valores de contradomínio podem ser definidos com o uso da instrução **set yrange**.

3

Gráficos tridimensionais

Este capítulo apresenta o uso de gráficos gerados no plano tridimensional. Na medida em que os gráficos são apresentados, são apresentados alguns recursos de operação do gnuplot, como `splot`, `set xlabel`, `set ztics`, `set xrange`, `set log`, `unset log`, `with`, `linetype`, `linecolor`, `set hidden3d`, `set terminal`, `set xyplane`, `set output` e `set pm3d`, entre outras.

3.1 Característica estrutural

Um gráfico tridimensional opera em um plano cartesiano com os eixos **x** (abscissas), **y** (ordenadas) e **z** (cotas). No gráfico 3D, o ponto de uma coordenada pode ser posicionado de acordo com sua altura no plano cartesiano, que, para um gráfico tridimensional, contém três planos simultaneamente perpendiculares. Eles são interceptados em um ponto comum a três retas simultaneamente perpendiculares. Esse plano cartesiano também pode ser referenciado como *coordenadas retangulares*.

Os eixos **x** e **y** são usados normalmente no plano cartesiano como são usados para um gráfico bidimensional. A diferença está no acréscimo da coordenada **z** que permite definir a altura que o elemento do gráfico terá no plano cartesiano. Os eixos **x** e **y** definem um plano horizontal para o gráfico e o eixo **z** é ortogonal a este plano (SILVA, 2013).

Para desenhar um gráfico bidimensional usa-se uma função com duas variáveis, sendo $f(x) = x$, onde $f(x)$ representa a coordenada **y** (ordenadas) e **x** representa o elemento da coordenada **x** (abscissas) a ser desenhado com o comando `plot`.

Um gráfico tridimensional faz uso de três variáveis, sendo $f(x,y) = x + y$, onde $f(x,y)$ representa a coordenada **z** (cotas), **x** representa a coordena **x** (abscissas) e **y**

representa a coordenada y (ordenadas) a ser desenhado com o comando **splot** com o objetivo de representar pontos de funções de primeiro grau, de segundo grau, exponenciais, trigonométricas, entre outras categorias de funções permitindo uma apresentação da curva em superfície.

3.2 Criação de gráficos 3D

Como abordado anteriormente, a geração de um gráfico 3D é realizada por meio do comando **splot**. Todas as funções e recursos apresentados no capítulo anterior podem ser usados com o **splot**. No entanto, o efeito tridimensional apresentado não é o melhor possível.

Para fazer a entrada de uma função que apresente um gráfico 3D informe no *prompt* de comando do programa **gnuplot** a linha de instrução:

```
splot x + y
```

Observe a imagem do gráfico apresentado, como mostra a Figura 3.1.

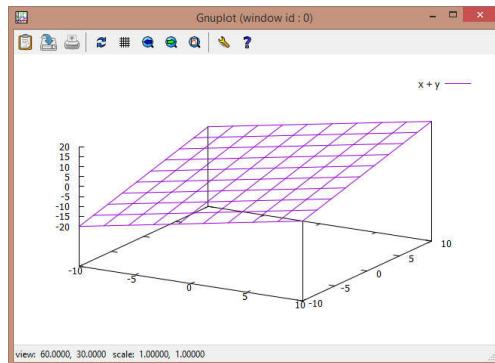


Figura 3.1 – Gráfico gerado a partir da função $f(x,y) = x + y$.

Os gráficos tridimensionais possuem uma característica operacional que permite mudar seu eixo de apresentação. Para isso, basta clicar dentro da área de desenho do gráfico, manter o botão de ação pressionado (normalmente o botão esquerdo) e arrastá-lo a fim de ver o gráfico de outros ângulos.

Os argumentos **x** e **y** do comando **splot** são variáveis internas definidas e reservadas no programa para a operação sobre os gráficos tridimensionais.

A próxima sequência de comandos visa deixar o gráfico com uma apresentação mais aprimorada. A Figura 3.2 mostra o resultado desta ação, após um pequeno ajuste com o ponteiro do *mouse*.

```
reset
set xtics 3
set ytics 3
set ztics 9
set title "Gráfico Tridimensional"
set ylabel "Título Y"
set xlabel "Título X"
set zlabel "Título Z"
set zeroaxis
splot x + y
```

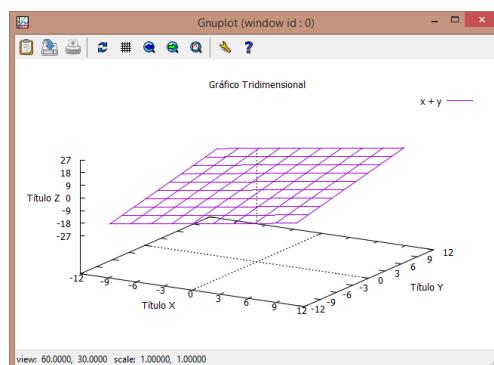


Figura 3.2 – Gráfico com parâmetros ajustados.

Note o uso dos complementos **ztics** e **zlabel** do comando **set** para estabelecer respectivamente o intervalo do eixo **z** e o título deste eixo. Perceba que esses dois complementos são similares ao conhecimento passado no capítulo anterior. Assim sendo pode-se mudar o limite do eixo **x** por meio do complemento **xrange**.

O próximo gráfico apresenta o resultado de uma função de segundo grau em um gráfico tridimensional na cor laranja, diferentemente da cor padrão vermelho. Para tanto é utilizada a função $f(x,y) = x^2 + y^2$. A Figura 3.3 mostra o resultado após a execução dos comandos a seguir.

```
reset
set title "Gráfico Tridimensional"
set ylabel "Y"
set xlabel "X"
set zlabel "Z"
set xrange [-15:15]
set yrange [-15:15]
set zrange [0:100]
set zeroaxis
splot x ** 2 + y ** 2 linetype 8
```

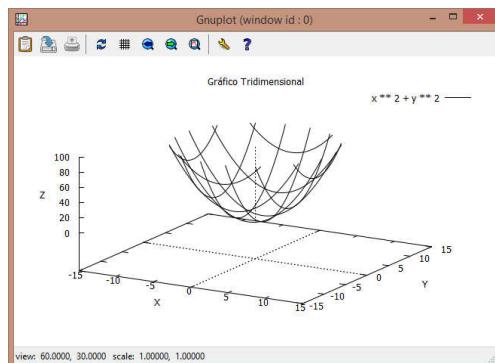


Figura 3.3 – Gráfico da função $f(x,y) = x^2 + y^2$.

Neste exemplo perceba o uso do comando **linetype** em conjunto com o comando **splot** com a definição do valor **8**. Este comando (que pode ser usado com **plot**) é responsável pela mudança da cor do gráfico, mas em especial ele muda o tipo do ponto usado para representar a curva do gráfico. Para ver, exceto quando está em uso **with lines**. Neste caso, o comando apenas muda a cor, pois prevalece o que está definido em **with lines**. Para ver a quantidade de pontos de apresentação no gráfico execute no *prompt* do programa **gnuplot** o comando **test** e note que existem

21 pontos diferentes de apresentação da curva do gráfico definido com nove cores possíveis.

Os usuários do **gnuplot** costumam usar, para a mudança de cor, o complemento **linetype**. No entanto, se o objetivo é apenas mudar a cor, procure usar o complemento **linecolor** que, além das nove cores básicas, oferece muitas outras opções.

O complemento **linetype** permite o uso de apenas nove cores predefinidas numa tabela de 23 possibilidades (23 tipos diferentes de pontos para apresentação da curva do gráfico) numeradas de **-1** a **21**. Se for usar este complemento para mudar cores são então possíveis de uso em relação a cor os valores indicados em seguida.

Número	Cor
1, 10, 19	Vermelho
2, 11, 29	Verde
3, 12, 21	Azul
4, 13	Magenta
5, 14	Ciano
6, 15	Amarelo
-1, 0, 7, 16	Preto
8, 17	Laranja
9, 18	Cinza

Se fizer uso do complemento **linecolor** o programa permite usar 16.581.375 opções de cores geradas por meio do uso da opção **rgb** ($r = red$, $g = green$, $b = blue$) que permite combinar as cores vermelho, verde e azul a fim de obter todas as cores do espectro de cores. Cada uma das cores **rgb** pode ser variada de 0 a 255. Os valores das cores devem ser informados no formato hexadecimal.

Observe a sequência de instruções a seguir e atente para o uso do complemento **linecolor** para a apresentação da curva na cor azul.

```
reset
set title "Gráfico Tridimensional"
set ylabel "Y"
set xlabel "X"
set zlabel "Z"
set xrange [-15:15]
set yrange [-15:15]
set zrange [0:100]
set zeroaxis
splot x ** 2 + y ** 2 linecolor 3
```

Para fazer uso de outras cores utilize o complemento **linecolor** seguido de **rgb** mais o nome da cor entre aspas inglesas ou o número hexadecimal da cor entre aspas inglesas e precedido do símbolo trailha (#).

Por exemplo, na sequência de instruções anteriores considere para a linha de instrução **plot** o complemento **linecolor rgb "magenta"** ou considere **linecolor rgb "#36A626"** para apresentar a curva em uma tonalidade de verde.

A função $f(x,y) = -y^2$ apresenta a imagem do gráfico indicado na Figura 3.4. Execute os seguintes comandos.

```
reset
set grid
set ztics 50
set zeroaxis linestyle 3
splot -y ** 2
```

Observe que neste exemplo o comando **linestyle** com a cor azul definida está sendo indicado após a instrução **set zeroaxis**, o que permite alterar a cor somente dos pontos de interseção 0, 0 e 0 para as coordenadas x, y e z.

Um detalhe a ser considerado é em relação ao uso da instrução **set grid** que mostra a grade apenas paras as coordenadas x e y.

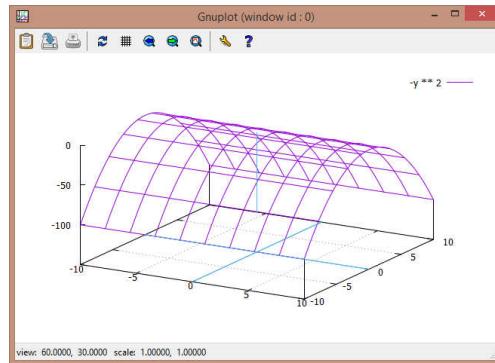


Figura 3.4 – Gráfico gerado a partir da função $f(x,y) = -y^2$.

É possível ocultar as linhas da parte detrás da curva. Para tanto use a instrução **set hidden3d**. Assim, execute os comandos seguinte e note o efeito junto a figura 3.5.

```
reset  
set grid  
set ztics 50  
set hidden3d  
splot -y ** 2
```

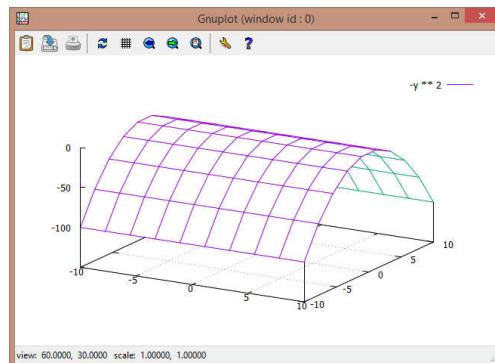


Figura 3.5 – Gráfico 3D com uso de **set hidden3d**.

3.3 Gravação de gráficos em mídia

Os gráficos desenvolvidos no programa podem ser gravados em arquivos do tipo imagem, como por exemplo o formato PNG. Para gravar um gráfico em um arquivo de imagem é necessário direcionar o fluxo de saída para um arquivo gráfico. Para tanto, basta fazer uso dos complementos **terminal** e **output** do comando **set**.

Quando se ativa o **terminal** para saída gráfica, é necessário após seu uso voltar o **terminal** para o modo padrão.

Para salvar uma imagem em formato PNG use a instrução **set terminal png**, para voltar ao modo normal de operação use a instrução **set terminal wxt**. Para ver uma lista completa dos recursos que podem ser ativados e desativados pelo comando **terminal** use a instrução **set terminal** ou **set term**, sem nenhum argumento.

Após executar a instrução **set terminal png** é necessário executar a instrução **set output** com o nome do arquivo mais sua extensão entre aspas inglesas. Por exemplo, desejando gravar o gráfico com o nome **graf01.png** use a instrução **set output "graf01.png"**.

A seguir é criado um gráfico a partir da função $f(x,y) = xy$ mostra junto a Figura 3.6 que será gravado no computador com o nome **graf01.png**.

```
reset
set grid
set terminal png
set output "graf01.png"
splot x * y
set terminal wxt
replot
```

Na sequência de instruções anteriores é efetivada a gravação da imagem do gráfico com o nome **graf01.png**. Observe que após efetivada a saída para o arquivo gráfico o **terminal** volta a ser configurado como **wxt**. O comando **replot** (ou se preferir **rep**) repete o desenho do gráfico e o mostra na tela de saída. Quando a saída está configurada para um arquivo a imagem não é apresentada na tela gráfica.

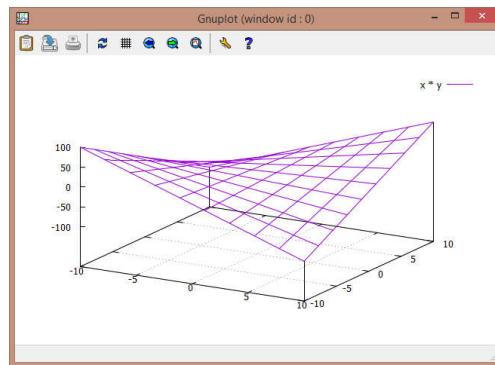


Figura 3.6 – Gráfico gerado a partir da função $f(x,y) = xy$.

O comando **replot** (pode ser referenciado na forma reduzida como **rep**) pode ser usado com a finalidade de sobrepor gráficos. Observe atentamente as seguintes linhas de instruções. A Figura 3.7 mostra a imagem obtida a partir desta ocorrência.

```
reset
splot x ** 2 - .5 * x - 2
replot sin(x)
rep tan(x)
```

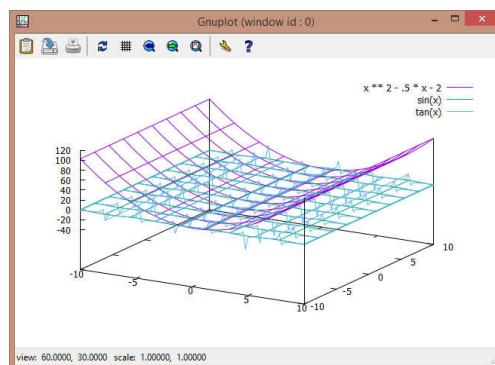


Figura 3.7 – Gráficos sobrepostos com comando **replot**.

3.4 Tipos de linha

Além da cor dos elementos do gráfico é possível alterar o formato das linhas das curvas de um gráfico. O programa **gnuplot** oferece algumas possibilidades. Para a efetivação desta ação deve-se usar o comando **with** seguido do argumento que determina o tipo de linha apresentada. Para tanto, considere a função $f(x,y) = x^2 + -0,5x - 2$ apresenta a imagem do gráfico indicado na Figura 3.8. Execute os seguintes comandos.

```
reset
set grid
set ztics 50
set zeroaxis linestyle 4
splot x ** 2 - .5 * x - 2 with points
```

Observe que a curva do gráfico está representada por pontos formados com o caractere **+**. Além da opção **points** é possível fazer uso dos estilos **lines** (modo padrão), **linespoints**, **impulses**, **dots**, **steps**, **fsteps** e **histeps**.

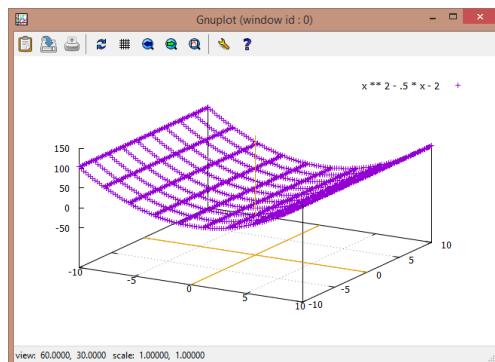


Figura 3.8 – Gráfico gerado a partir da função $f(x,y) = x^2 + -0,5x - 2$.

Há a possibilidade de alterar a forma de apresentação gráfica para uma escala logarítmica ou retornar o modo de ação para escala linear. Basta usar a instrução **set log** para definir a escala logarítmica e **unset log** para retornar para a escala linear. O comando **unset** tem por finalidade desativar a ação do comando **set**. Esta

funcionalidade não é perceptível com qualquer gráfico, apenas com os gráficos que permitem o uso de escala logarítmica.

Os gráficos criados podem ser copiados para a área de *clipboard* do computador com o objetivo de serem colados em outros locais. Para tanto, basta na janela gráfica acionar na barra de ferramentas o primeiro botão à esquerda denominado **Copy the plot to clipboard**.

A manipulação de tipos de linha permite criar efeitos em gráficos como um gráfico com área rachurada. Para tanto, considere um gráfico bidimensional a partir da função **plot airy(x)** que apresentará o gráfico da Figura 3.9, a partir da seguinte sintaxe.

```
reset  
set grid  
plot airy(x) with impulses linetype 1  
replot airy(x) with lines linetype 1
```

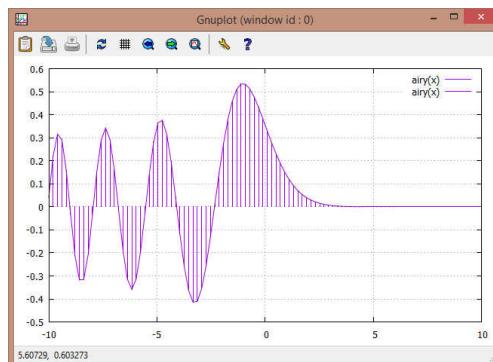


Figura 3.9 – Gráfico com efeito de rachura e linha.

O efeito da rachura é obtido a partir do complemento **impulses** após o comando **with**. Com a técnica de redesenhar o gráfico com **replot** faz-se duas sobreposições do mesmo gráfico.

A instrução **plot airy(x) with impulses linetype 1** faz o desenho do gráfico com rachura na cor vermelha.

Com a instrução **replot airy(x) with lines 1 linetype 1** faz novamente o desenho do gráfico com as linhas da curva na cor 1 (vermelho).

Para a apresentação deste gráfico está sendo utilizado junto com o comando **plot** a função **airy()** que é comentada no próximo capítulo.

Observe que a Figura 3.9 apresenta no gráfico dois títulos de legenda, uma vez que está sendo feito o desenho no mesmo plano cartesiano de dois gráficos. Para deixar apenas um título é necessário informar esta ocorrência em dos comandos **plot** usados. Observe a seguinte sequência e seu resultado junto a Figura 3.10.

```
reset
set grid
plot airy(x) with impulses linetype 1 notitle
replot airy(x) with lines linetype 1
```

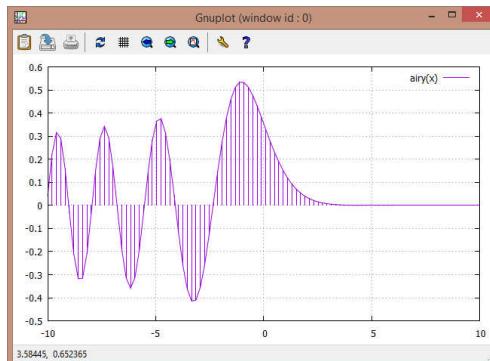


Figura 3.10 – Gráfico sem o título de uma das legendas.

Perceba o uso do complemento **notitle** junto ao primeiro comando **plot**. Este complemento inibe a apresentação do título da legenda quando desejado.

3.5 Planos, superfícies e paleta de cores

Os gráficos tridimensionais permitem alguns ajustes pertinentes, como a escolha do plano cartesiano de apresentação, ou a superfície do gráfico, e a apresentação da paleta de cores com os tons de definição da superfície do gráfico.

Para alterar a posição do plano cartesiano de apresentação de um gráfico 3D usa-se a instrução **set xyplane** que efetiva a mudança da altura de apresentação do gráfico no plano **xy**. Esta instrução pode ser usada de forma absoluta (**set xyplane at**) ou na forma relativa (**set xyplane relative**). Os valores operacionalizados podem ser inteiros ou reais.

```
set xyplane at 1
```

Neste caso a instrução efetua a apresentação do gráfico no plano **xy** no eixo **z** com posição **1**.

```
set xyplane relative 0.5
```

Com esta instrução é realizada a apresentação do gráfico no plano **xy** no eixo **z** na posição **0.5** que é uma fração da amplitude do eixo das cotas. Vale ressaltar que o valor **0.5** é o valor padrão do programa **gnuplot**.

O exemplo seguinte apresenta um gráfico tridimensional com plano **xy** relativo definido na posição **7**. A Figura 3.11 mostra o ocorrido.

```
reset
set grid
set ztics 50
set xyplane relative 7
set hidden3d
splot -y ** 2
```

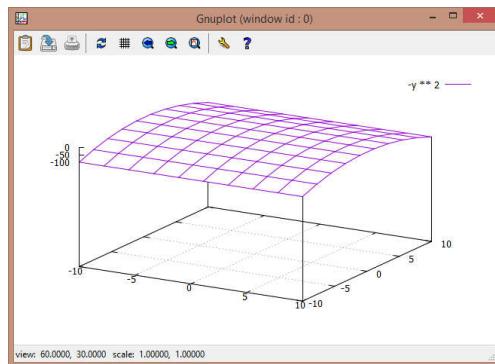


Figura 3.11 – Gráfico com plano xy relativo na posição 7.

O próximo exemplo usa um plano **xy** absoluto na posição **5.5**. Veja a Figura 3.12.

```
reset
set grid
set ztics 50
set xyplane at 5.5
set hidden3d
splot -y ** 2
```

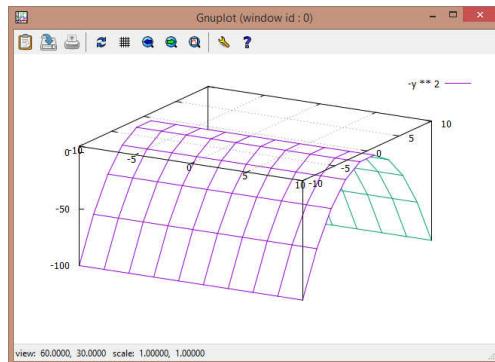


Figura 3.12 – Gráfico com plano xy absoluta na posição 5.5.

Em relação ao uso de paleta de cores basta informar a instrução **set pm3d** (*palette map 3D*) que dá a capacidade do gráfico ser desenhado com uma superfície preenchida. Assim sendo execute as instruções a seguir e veja na Figura 3.13 a apresentação da paleta de cores.

```
reset  
set pm3d  
set grid  
set ztics 50  
set xyplane relative 7  
set hidden3d  
splot -y ** 2
```

A paleta de cores mostra um gradiante de cores, onde cada parcela dos valores usados para apresentar o gráfico são indicadas.

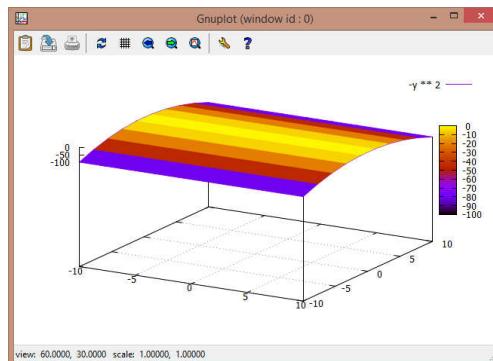


Figura 3.13 – Gráfico com paleta de cores com tonalidades.

O próximo exemplo mostra um gráfico com as cores na superfície definidas de modo mais gradual. Esta forma depende dos ajustes dos eixos e das funções utilizadas. Assim, execute as próximas instruções e veja na Figura 3.14 o efeito apresentado.

```
reset
set pm3d
set grid
set xrange [-2:2]
set yrange [-2:2]
splot exp(.2 * y) * cos(x) * sin(x)
```

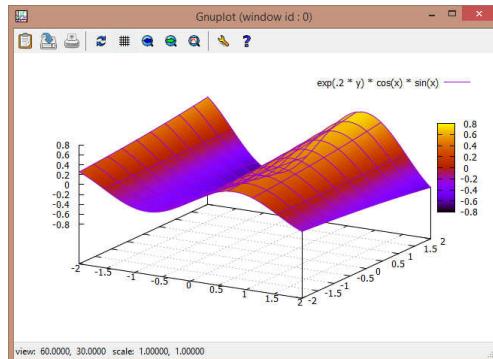


Figura 3.14 – Gráfico a partir da função $\exp(.2 * y) * \cos(x) * \sin(x)$.

3.6 Comando plot/splot

De todos os comandos existentes no programa **gnuplot** os comandos **plot** e **splot** são os mais importantes, pois é partir destes que respectivamente os gráficos 2D e 3D são desenhados¹. De forma mais ampla os comandos **plot** e **splot** possuem uma estrutura sintática maior do que fora apresentada:

¹ Encontra-se também referências aos termos “plotar” e “graficar” como a ação de desenhar um gráfico.

- Comando – **plot** (gráficos bidimensionais – 2D):

```
plot
  {<faixa>}
  {<iteração>}
  {<função> | {"<nome do arquivo>" {modificadores}}}
  {axes <axes>}
  {notile | title <título da legenda>}
  {with <estilo da linha>}
  {, {definições{,} ...}}
```

- Comando – **splot** (gráficos tridimensionais – 3D)

```
splot
  {<faixa>}
  {<iteração>}
  {<função> | "<nome do arquivo>" {modificadores}}
  {notile | title <título da legenda>}
  {with <estilo da linha>}
  {, {definições{,} ...}}
```

Note que estruturalmente a diferença entre os dois comandos é a existência do argumento **axes** junto ao comando **plot**. Segue descrito dos argumentos opcionais usados com o comando **plot** e **splot**.

- O complemento opcional **faixa** é usado para especificar a região do gráfico que será exibida sem a necessidade de usar o comando **set**, tendo como sintaxe:

```
[ {<variável> = } {<mínimo>} : {<máximo>} ]
[ {<mínimo>} : {<máximo>} ]
```

A primeira forma se aplica a variável independente **xrange** ou **trange**², que esteja em modo paramétrico (quando se usa equação paramétrica). A segunda forma se aplica a variável dependente **yrange** (e **xrange**, se estiver o modo paramétrico ativado). **<variável>** é a definição de um nome para a variável independente. As opções padrão podem ser alteradas com **set**. Os parâmetros opcionais **<mínimo>** e **<máximo>** podem ser expressões com constantes ou com *.

² Usado para definir intervalo paramétrico para calcular valores de x e y no modo paramétrica ou modo polar. O complemento **trange** refere-se ao uso da variável t (variável de ângulo theta).

No modo não paramétrico a ordem dos intervalos é estabelecida com **xrange** e **yrange** (forma padrão) e na ordem paramétrica (a ser configurada) a ordem é estabelecida como **trange**, **xrange** e **yrange**.

O exemplo seguinte apresenta um gráfico não paramétrico com o eixo **x** (**xrange**) definido na escala de **-3** até **3** e o eixo **y** (**yrange**) definido na escala de **-5** até **5**.

```
plot [-3:3] [-5:5] tan(x)
```

O exemplo seguinte apresenta um gráfico não paramétrico com apenas o eixo **X** (**xrange**) definido na escala de **-3** até **3**.

```
plot [-3:3] tan(x)
```

O exemplo seguinte apresenta um gráfico não paramétrico com apenas o eixo **Y** (**yrange**) definido na escala de **-5** até **5**.

```
plot [] [-5:5] tan(x)
```

O exemplo seguinte apresenta um gráfico não paramétrico com o eixo **x** (**xrange**) definido com escala mínima de **-3** e o eixo **y** (**yrange**) definido com escala máxima até **5**.

```
plot [-3:] [:5] tan(x)
```

O exemplo seguinte apresenta um gráfico não paramétrico com a definição da variável **valor** usada na definição do cálculo do gráfico.

```
plot [valor = -3:6] tan(valor)
```

Os argumentos para **xrange** e **yrange** podem ser definidos a partir de valores ou de expressões matemáticas, como por exemplo:

```
plot [-2:2] [-3:cos(.5)/22.85] tan(x)
```

O modo paramétrico é ativado com a instrução **set parametric** e desativado com a instrução **unset parametric**. Quando o modo paramétrico é ativado o foco de ação torna-se a variável **t** e não mais a variável **x** e uma mensagem é apresentada na tela do programa. Quando este modo é acionado é necessário fazer uso de duas equações com a mesma variável, variável **t**. As linhas seguintes mostram o uso de gráficos paramétricos.

```
set parametric  
plot sin(t), t ** 2  
unset parametric
```

- O complemento opcional **iteração** é usado para especificar a contagem de uma série a partir da sintaxe:

```
for [<variável> = <início> : <fim> {:<incremento>}]
for [<variável> in "cadeia de série de palavras"]
```

A primeira forma se aplica a definição de uma variável que será usada para iterar o cálculo de um gráfico. A segunda forma é usada para estabelecer uma lista de cadeias que podem ser usadas no gráfico.

O exemplo seguinte apresenta um gráfico que traçará cinco curvas a partir da iteração da variável *i* de 1 até 5 que será multiplicada por *x* para a definição da tangente da curva.

```
plot for [i=1:5] tan(i*x)
```

O exemplo seguinte apresenta um gráfico que traçará sete curvas a partir da iteração da variável *i* de 2 até 20 com incremento 3 que será multiplicada por *x* para a definição da tangente da curva.

```
plot for [i=2:20:3] tan(i*x)
```

O exemplo seguinte apresenta um gráfico do seno de duas curvas obtidas a partir da função **rand()** com legendas **curva1** e **curva2** definidas a partir da ação de iteração para a variável *n*.

```
plot for [n in "curva1 curva2"] sin(rand(0)) title n
```

- O complemento não opcional seletivo **função** (que pode ser usado como alternativa para **nome do arquivo**) é usado para especificar a ação de desenho do gráfico com base na sintaxe:

```
plot função(x)
```

Este modo de uso é a forma mais comum de operação do comando **plot** e é amplamente usado nos exemplos anteriores.

- O complemento não opcional seletivo **nome do arquivo** (que pode ser usado como alternativa para **função**) é usado para especificar a ação de desenho de um gráfico baseado em um arquivo de dados gravado em disco, com a sintaxe:

```
plot "<nome do arquivo>" {modificadores}
```

O complemento **modificadores** estabelece o tipo de arquivo usado que pode ser usado, que pode ser **binary** (para acesso a arquivo binário), **index** (para acesso a um conjunto de dados de um arquivo que contenha vários dados), **every** (para acesso a uma amostragem periódica de um conjunto de dados), **thru** (para estabelecer compatibilidade do programa com versões anteriores do

mesmo), **using** (para estabelecer quais colunas de valores de um arquivo devem ser usadas no desenho do gráfico), **smooth** (recurso que estabelece acesso as rotinas de uso geral para interpolação e aproximação de dados), **volatile** (para indicar que os dados fornecidos não estarão disponíveis para o comando **replot**, serão os dados usados apenas uma vez) e **noautoscale** (para ignorar os pontos que determinam automaticamente os limites das faixas dos eixos cartesianos).

Este complemento será tratado a parte nesta obra.

- O complemento opcional **axes** existente apenas no comando **plot** é usado para selecionar um dos quatro conjuntos de eixos existentes para que seja usado no escalonamento do desenho da linha no gráfico, podendo ser: **x1y1** (refere-se ao eixo inferior esquerdo), **x2y2** (refere-se ao eixo superior direito), **x1y2** (refere-se ao eixo inferior direito) e **x2y1** (refere-se ao eixo superior esquerdo).

O exemplo seguinte apresenta um gráfico de duas curvas. A curva do seno definida em **x1y1** e a curva do logaritmo em **x2y2**. Depois tire os argumentos **x2y1** e **x1y1** e veja como é apresentado o gráfico.

```
plot sin(x) axes x1y1, log(x) axes x2y2
```

- O complemento opcional **title** permite definir o título apresentado no gráfico. O exemplo seguinte apresenta um gráfico do seno de **x** com título definido para a legenda.

```
plot sin(x) title "Resultado do seno de x"
```

- O complemento opcional **notitle** é usado para inibir o título da legenda na apresentação do gráfico.
- O complemento opcional **with** permite modificar o estilo de apresentação dos trechos das curvas, tendo este recurso sido apresentado anteriormente neste capítulo.
- O complemento **definições** refere-se ao estabelecimento de uma lista de ações baseadas nos complementos já descritos separados por vírgula.

Agora, ao executar os comandos **plot** e **splot**, qualquer um dos argumentos apresentados pode ser utilizado.

4

Funções

Este capítulo aborda os recursos ferramentais do programa **gnuplot**. São aqui apresentadas funções diversas, entre as quais estão as funções matemáticas, funções de manipulação de cadeias de caracteres e funções diversas. Aprenda, neste capítulo, a criar uma biblioteca pessoal de funções próprias.

4.1 Funções internas

Além das funções matemáticas conhecidas e algumas já utilizadas nesta obra, o programa **gnuplot** têm um conjunto de funções similares as funções matemáticas e também outras funções específicas.

Matematicamente, função é uma forma de generalizar, em um rótulo de identificação, uma fórmula matemática, que pode ser simples ou complexa. A finalidade de uma função é estabelecer a relação entre dois elementos, em que uma variável dependente é relacionada a uma variável independente. Por exemplo, a notação $f(x) = x$ diz que x é uma variável independente e $f(x)$, no lugar da variável y , é uma variável dependente, pois o valor de $f(x)$ depende do valor de x . Já a notação $f(x,y) = xy$ diz que $f(x,y)$ representa a variável z (noção tridimensional) como dependente e as variáveis x e y como independentes.

Na esfera da programação de computadores, funções são definidas como recursos que sempre devolvem como resposta um valor baseado no fornecimento de um argumento ou parâmetro. Assim, o retorno da função representa uma variável dependente e o argumento, a variável independente.

Neste tópico as funções internas (*função gnuplot*) são comentadas, apresentadas e exemplificadas as funções do programa existentes sob as categorias: matemática, cadeia (*string*). As funções internas do programa estão sendo referenciadas no

contexto deste trabalho como *função gnuplot* como descritas no glossário do capítulo 1 para diferenciar do conceito existente para funções em matemática identificada no glossário como *função matemática*.

Uma *função gnuplot* pode possuir para sua ação uma das seguintes estruturas de escrita:

y = função("ação")

y = função("ação", x)

y = função("ação", x, [, ...])

y = função("ação1", "ação2")

y = função(x)

y = função(x, y)

y = função(x, y, z)

O formato a ser utilizada dependerá da ação a ser desejada. Note que o argumento de uma função para o programa pode variar de zero a uma quantidade indefinida representada por **[, ...]**. Os argumentos indicados entre aspas inglesas identificam o fornecimento de alguma ação predefinida e os argumento sem aspas inglesas indicam o fornecimento de um valor numérico.

As funções suportadas pelo programa **gnuplot** são apresentadas neste tópico em ordem alfabética e divididas sob a categoria que pertencem e poderá ocorrer de algum exemplo fazer uso de uma função que será apresentada na sequência adiante. Neste caso, basta localizar a função usada para saber o que representa e o que efetivamente faz.

Os argumentos **x**, **y** ou **z** podem ser a representação de valores inteiros, reais ou complexos. Valores complexos são definidos como o comprimento do valor indicado no argumento da função. Os números complexos, no programa **gnuplot**, são formados por pares de números, indicados entre chaves, sendo o primeiro valor a parte real do número e o segundo, a parte imaginária do número. Essa estrutura tem como limitação a capacidade de aceitar apenas valores constantes para ambas as partes; não são aceitas variáveis e expressões aritméticas. Um número complexo é definido com base na fórmula matemática $\sqrt{(\text{Re}(x))^2 + \text{Im}(x)^2}$, expressa no **gnuplot**

como `sqrt(real(x) ** 2 + imag(x) ** 2)`, segundo sua documentação. Os demais argumentos são comentados quando necessários.

4.1.1 Funções matemáticas

As funções matemáticas suportadas pelo programa **gnuplot** são as indicadas neste tópico. A título de exemplo, informam-se o nome da função, sua descrição e o retorno de valor. Ao final do tópico, apresentam-se os gráficos básicos gerados pelas funções.

abs(x)

Retorna o valor do módulo de um número, ou seja, retorna sempre o valor informado como positivo. Opera com números inteiros, reais ou complexos. Quando informado um número complexo o retorno será a partir do tamanho do argumento.

```
print abs(-3)           retorna o valor 3
print abs(3)           retorna o valor 3
plot abs(x)
```

acos(x)

Retorna o valor do arco cosseno de um valor fornecido em radianos, sendo este o inverso do cosseno. Opera com números inteiros, reais ou complexos.

```
print acos(0.5)*180.0/pi  retorna o valor 60.0
```

acosh(x)

Retorna o valor do arco cosseno hiperbólico de um valor fornecido em radianos, sendo este o inverso do cosseno. Opera com números inteiros, reais ou complexos.

```
print acosh(60.0*pi/180)  retorna o valor 0.306042106...
```

airy(x)

Retorna o valor de tensão calculado a partir da função matemática **Ai(x)**. Opera com números inteiros, reais ou complexos.

```
print airy(1)           retorna o valor 0.137723025...
```

arg(x)

Retorna o valor do argumento (fase) de um número complexo em radianos ou graus. Opera com números complexos. A mudança do estilo dos ângulos para radianos ou graus é feita com o uso das instrução **set angles radians** (para ativar ângulos em radianos) ou **set angles degrees** (para ativar ângulos em graus).

```
print arg({3,4})           retorna o valor 0.927295218...
```

asin(x)

Retorna o valor do arco seno de um valor fornecido em radianos, sendo este o inverso do seno. Opera com números inteiros, reais ou complexos.

```
print asin(0.5)*180.0/pi  retorna o valor 30.0
```

asinh(x)

Retorna o valor do arco cosseno hiperbólico de um valor fornecido em radianos, sendo este o inverso do cosseno. Opera com números inteiros, reais ou complexos.

```
print asinh(60.0*pi/180)  retorna o valor 0.914356655...
```

atan(x)

Retorna o valor do arco tangente de um valor fornecido em radianos, sendo este o inverso da tangente. Opera com números inteiros, reais ou complexos.

```
print atan(1.0)*180.0/pi  retorna o valor 45.0
```

atan2(y,x)

Retorna o valor do arco tangente de **x/y** em radianos, sendo este o inverso da tangente. Opera com números inteiros e reais.

```
print atan2(10.0,-10.0)*180/pi  retorna o valor 135.0
```

besj0(x)

Retorna o valor da função esférica da equação diferencial de Bessel a partir de um valor fornecido em radianos para j_0 . Opera com números inteiros ou reais.

```
print besj0(0)           retorna o valor 1.0
```

besj1(x)

Retorna o valor da função esférica da equação diferencial de Bessel a partir de um valor fornecido em radianos para j_1 . Opera com números inteiros ou reais.

```
print besj1(1)           retorna o valor 0.440050585...
```

besy0(x)

Retorna o valor da função esférica da equação diferencial de Bessel, com base em um valor fornecido em radianos para y_0 . O número do argumento x deve ser maior que zero. Opera com números inteiros ou reais.

```
print besy0(1)           retorna o valor 0.088256964...
```

besy1(x)

Retorna o valor da função esférica da equação diferencial de Bessel, com base em um valor fornecido em radianos para y_1 . O número do argumento x deve ser maior que zero. Opera com números inteiros ou reais.

```
print besy1(1)           retorna o valor -0.78121282...
```

ceil(x)

Retorna o maior valor inteiro a partir do valor fornecido arredondando o número para o inteiro mais próximo acima. Matematicamente a representação desta ação é feita com o símbolo $\lceil x \rceil$. Opera com números inteiros, reais ou complexos.

```
print ceil(1.1)          retorna o valor 2
```

cos(x)

Retorna o valor do cosseno de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print cos(60*pi/180)      retorna o valor 0.5
```

cosh(x)

Retorna o valor do cosseno hiperbólico de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print cosh(log(2.0))      retorna o valor 1.25
```

EllipticK(x)

Retorna o valor da integral elíptica completa de primeira espécie de $K(x)$, em que x é um valor real entre -1 e 1 .

```
print EllipticK(0.5)      retorna o valor 1.685750354...
```

EllipticE(x)

Retorna o valor da integral elíptica completa de segunda espécie de $E(x)$, em que x é um valor real entre -1 e 1 .

```
print EllipticE(0.5)      retorna o valor 1.467462209...
```

EllipticPi(x,y)

Retorna o valor da integral elíptica completa de terceira espécie de $\Pi(x,y)$, em que x deve ser um valor real menor que 1 e y deve ser um valor real maior que -1 e menor que 1 .

```
print EllipticPi(.5, 0)    retorna o valor 2.221441469...
```

erf(x)

Retorna a parte real do argumento **x** da função erro como a aproximação de integrais em série. Se o argumento for um valor complexo, o componente imaginário será ignorado. Opera com números inteiros, reais ou complexos.

```
print erf(1)           retorna o valor 0.842700792...
```

erfc(x)

Retorna como resulta o valor de **1 – erf(x)**. Se o argumento for um valor complexo, o componente imaginário será ignorado. Opera com números inteiros, reais ou complexos.

```
print erfc(1)          retorna o valor 0.157299207...
```

exp(x)

Retorna o exponencial da base **e** elevado ao expoente **x**. Opera com números inteiros, reais ou complexos.

```
print exp(5)           retorna o valor 148.4131591...
```

expint(y,x)

Retorna a integral exponencial da parte real **do** argumento. Opera com números inteiros, reais ou complexos. O argumento **y** deve ser um número inteiro não negativo e maior que 1; o argumento **x** deve ser maior ou igual a 0.

```
print expint(1,2)      retorna o valor 0.048900510...
```

floor(x)

Retorna o menor valor inteiro a partir do valor fornecido arredondando o número para o inteiro mais próximo abaixo. Matematicamente a representação desta ação é feita com o símbolo **[x]**. Opera com números inteiros, reais ou complexos.

```
print floor(1.9)       retorna o valor 1
```

gamma(x)

Retorna a parte real do argumento da função gama de um argumento **x** real. Opera com números inteiros, reais ou complexos. Se o argumento for um valor inteiro, retorna-se o valor da fatorial do número fornecido, no caso, -1. Se o argumento for um número complexo, a parte imaginária do número é ignorada.

```
print gamma(6)           retorna o valor 120
```

ibeta(p, q, x)

Retorna o valor incompleto da função. O argumento **q** deve ser maior que **0** e o argumento **x** deve estar na faixa de **0** até **1**. Se os argumentos são números complexos, os componentes imaginários são ignorados.

```
print ibeta(1, .5, .7)    retorna o valor 0.452277442...
```

igamma(a,x)

Retorna a função gama incompleta das partes real de seus argumentos. O argumento **a** deve ser um valor maior que **0** e o argumento **x** deve ser um valor maior ou igual a **0**. Se os argumentos são complexos, os componentes imaginários são ignorados. Opera com números inteiros, reais ou complexos.

```
print igamma(.5, 2)       retorna o valor 0.954499725...
```

imag(x)

Retorna a parte imaginária de um número real definido no argumento **x**. Esta função somente opera com números complexos.

```
print imag({3,4})        retorna o valor 4.0
```

int(x)

Retorna a parte inteira do valor fornecido no argumento **x** truncando a mantissa do valor. Opera com números inteiros ou reais.

```
print int(1.9)           retorna o valor 1
```

inverf(x)

Retorna o valor inverso da função erro de **real(x)**, onde o argumento **x** deve ser um valor maior ou igual a **0** e menor que **1**. Opera com números inteiros, reais ou complexos.

```
print inverf(.5)           retorna o valor 0.476936276...
```

invnorm(x)

Retorna o valor inverso da distribuição normal da função **real(x)**, o qual deve estar na faixa de valores entre maior que **0** e menor que **1**. Opera com números inteiros, reais ou complexos.

```
print invnorm(0.1)         retorna o valor -1.28155156...
```

lambertw(x)

Retorna o valor da função de Lambert W. Opera com números reais.

```
print lambertw(1.9)        retorna o valor 0.829176330...
```

lgamma(x)

Retorna o logaritmo natural da função **lgamma** como argumento **x**. Opera com números inteiros ou reais.

```
print lgamma(1.9)          retorna o valor -0.03898427...
```

log(x)

Retorna o logaritmo natural de base **e** do argumento **x**. Opera com números inteiros ou reais.

```
print log(1.5)             retorna o valor 0.405465108...
```

log10(x)

Retorna o logaritmo de base **10** do argumento **x**. Opera com números inteiros ou reais.

```
print log10(1.5)            retorna o valor 0.176091259...
```

norm(x)

Retorna o valor da distribuição normal da função **real(x)**. Opera com números inteiros, reais ou complexos.

```
print norm(0.1)           retorna o valor 0.539827837...
```

rand(x)

Retorna um valor pseudo aleatório do argumento **x** utilizado como semente entre 0 e 1. Opera com números inteiros. Se desejar, por exemplo, gerar números entre 1 e 5 use **int(rand(0)*5)+1** ou **ceil(rand(0)*5)**.

```
print rand(0)           retorna um valor entre 0 e 1
```

real(x)

Retorna o valor do argumento **x** como sendo um número real. Opera com números inteiros, reais ou complexos.

```
print real(3)           retorna o valor 3.0
```

sgn(x)

Retorna o valor 1 se o argumento **x** for maior que 0, -1 se o argumento **x** menor que 0 e 0 se o argumento **x** for igual a 0. Opera com números inteiros, reais ou complexos.

```
print sgn(3)           retorna o valor 1
print sgn(0)           retorna o valor 0
print sgn(-3)          retorna o valor -1
```

sin(x)

Retorna o valor do seno de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print sin(30*pi/180)    retorna o valor 0.5
```

sinh(x)

Retorna o valor do seno hiperbólico de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print sinh(log(2.0))           retorna o valor 0.75
```

sqrt(x)

Retorna o valor da raiz quadrada de um valor fornecido como argumento x. Opera com números inteiros, reais ou complexos.

```
print sqrt(25)                 retorna o valor 5
```

tan(x)

Retorna o valor da tangente de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print tan(45*pi/180)          retorna o valor 1.0
```

tanh(x)

Retorna o valor da tangente hiperbólica de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

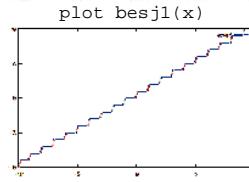
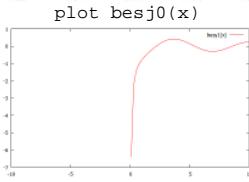
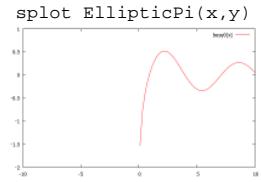
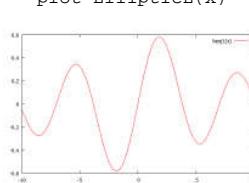
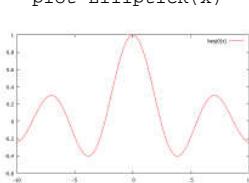
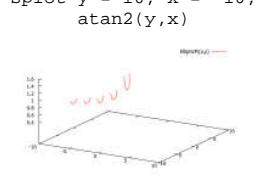
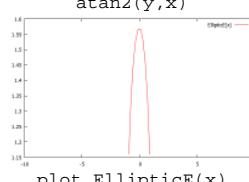
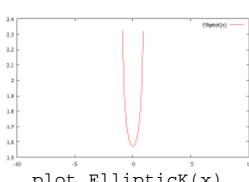
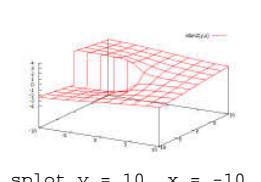
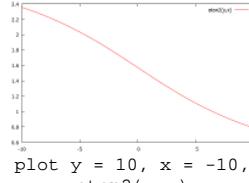
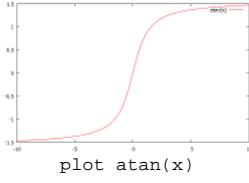
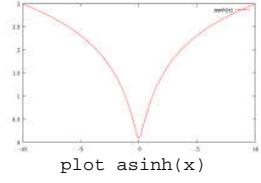
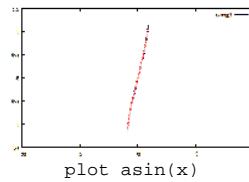
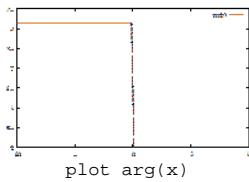
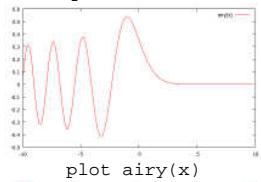
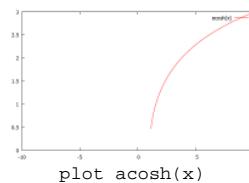
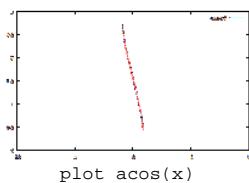
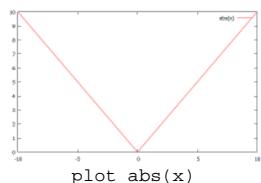
```
print tanh(log(2.0))         retorna o valor 0.6
```

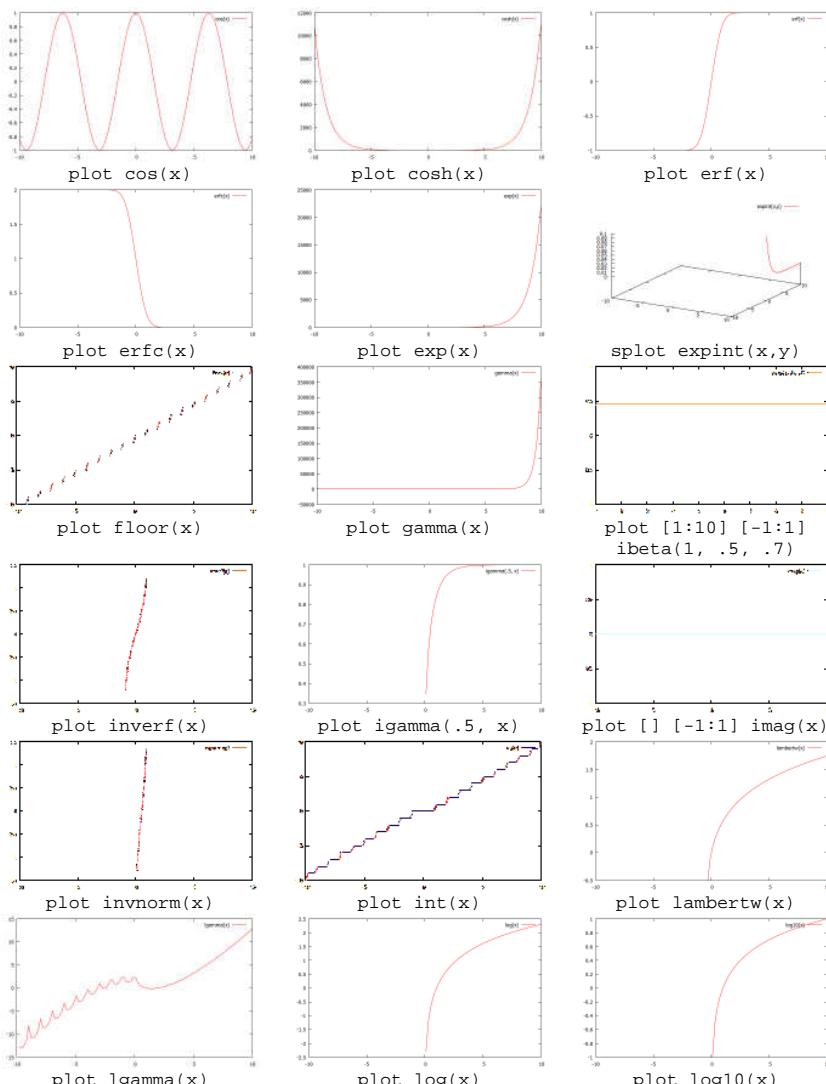
voigt(x,y)

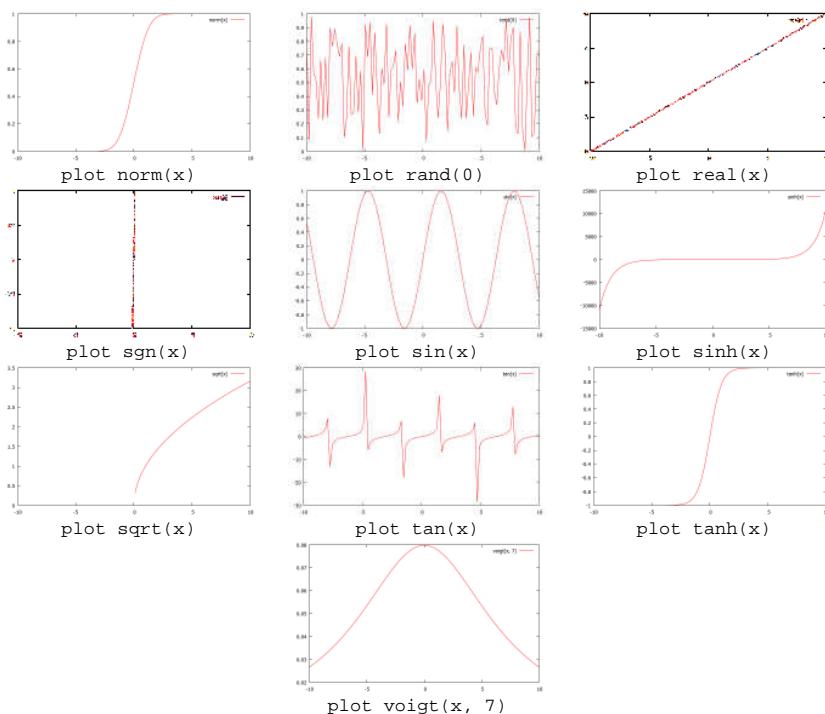
Retorna o valor da aproximação da função de Voigt/Faddeeva, utilizada na análise espectral, em 10^4 , com convolução de Gauss e Lorentz. Opera com números inteiros, reais ou complexos.

```
print voigt(2.3, 7)          retorna o valor 0.072336358...
```

Gráficos das Funções Matemáticas







4.1.2 Funções de cadeia

As funções de cadeia fornecem recursos para a manipulação de cadeias de caracteres (*strings*).

`gprintf("formato", x)`

Usada para apresentar um *string* com elemento de formatação a ser informado dentro das aspas inglesas. A formatação é definida a partir de símbolos especiais, tais como:

Formato	Descrição
%f	Notação de ponto flutuante
%e / %E	Notação exponencial
%g / %G	Notação exponencial curta
%x / %X	Notação hexadecimal
%o / %O	Notação octal
%t	Mantissa de base 10
%l	Mantissa com base em escala logarítmica
%s	Mantissa com base em escala logarítmica ou científica exponencial
%T	Potência na base 10
%L	Potência na base atual logarítmica
%S	Potência científica
%c	Substituição de caracteres para potência científica
%b	Mantissa em notação ISO/IEC 80000
%B	Prefixo em notação ISO/IEC 80000
%P	Múltiplo da constante pi

```

print gprintf("X = %g", 1)           mostra X = 1
print gprintf("X = %2.2f", pi)        mostra X = 3.14
print gprintf("X = %e", pi)           mostra X = 3.141593e+000
print gprintf("X = %x", 11)          mostra X = b
print gprintf("X = %o", 11)          mostra X = 13
print gprintf("X = %t", pi)          mostra X = 3.141593
print gprintf("X = %l", pi)          mostra X = 3.141593
print gprintf("X = %s", pi)          mostra X = 3.141593
print gprintf("X = %T", pi)          mostra X = 0
print gprintf("X = %L", pi)          mostra X = 0
print gprintf("X = %S", pi)          mostra X = 0
print gprintf("X = %c", pi)          mostra X =
print gprintf("X = %b", pi)          mostra X = 3.141593
print gprintf("X = %B", pi)          mostra X = 3.141593
print gprintf("X = %P", pi)          mostra X = 1.000000

```

 sprintf("formato", x, ...)

Usada para apresentar um *string* com mais de um elemento de formatação a ser informado dentro das aspas inglesas. A formatação é definida a partir dos símbolos especiais indicados na função **gprintf()**.

```
print sprintf("X = %g", 1)           mostra X = 1
print sprintf("X = %2.2f", pi)        mostra X = 3.14
print sprintf("%g, %g e %g", 1, 2, 3) mostra 1, 2 e 3
```

 strftime("formato do tempo", "cadeia com data e hora")

Usado para aplicar especificadores de formatação a informação de tempo medido em segundos desde o ano de 2000. O argumento **cadeia com data e hora** deve ser uma cadeia contendo as informações da data e da hora a serem tratadas. A formatação é definida a partir de símbolos especiais, tais como:

Formato	Descrição
%a	Apresenta nome do dia da semana abreviado
%A	Apresenta nome completo do dia da semana
%b / %h	Abreviatura do nome do mês
%B	Apresenta nome completo do mês
%d	Apresenta dia do mês de 1 a 31
%D	Apresentação de data abreviada para saída como %m/%d/%y
%F	Apresentação de data abreviada para saída como %Y-%m-%d
%k	Apresentação da hora com um ou dois dígitos de 0 a 23
%H	Apresentação da hora com dois dígitos de 00 a 23
%l	Apresentação da hora com um ou dois dígitos de 1 a 12
%I	Apresentação da hora com dois dígitos de 00 a 12
%j	Apresentação do dia do ano de 1 a 366
%m	Apresentação do mês de 1 a 12
%M	Apresentação do minuto de 0 a 60
%p	Apresentação de "am" ou "pm"
%r	Apresentação de hora abreviada como %l:%M:%S
%R	Apresentação de hora abreviada como %H:%M

Formato	Descrição
%S	Apresentação dos segundos de 0 a 60
%s	Apresenta número de segundos desde o início do ano de 2000
%T	Apresenta hora abreviada para %H:%M:%S
%U	Apresenta a semana do ano (semana começa no domingo)
%w	Apresenta o valor do dia de uma semana (domingo = 0)
%W	Apresenta a semana do ano (semana começa na segunda-feira)
%Y	Apresenta ano de 0 até 99 na faixa 1969-2068
%Y	Apresenta o ano com 4 dígitos

```
formato = "%Y-%m-%d %H:%M:%S"
tempo = "2013-01-21 17:35:12"
conta = strftime(formato, strptime(tempo, tempo) + 1.)
print "Tempo = ", conta
```

strlen("cadeia")

Usada para apresentar o tamanho em caracteres da cadeia informada entre aspas inglesas.

```
print strlen("Casa")           mostra 4
print strlen("Vila Velha")     mostra 10
```

strptime("formado do tempo", "cadeia com data e hora")

Usado para ler a partir do argumento **cadeia com data e hora** a informação de tempo especificada de acordo com a formatação de tempo desde o ano de 2000. O padrão de formatação desta função é o mesmo definido na função **strftime()**.

```
formato = "%Y-%m-%d %H:%M:%S"
tempo = "2013-01-21 17:35:12"
segundos = strptime(formato, tempo)
print "Data/Hora = ", tempo
print "Segundos = ", segundos
```

strstr("cadeia", "chave")

Usada para indicar a posição que uma chave tem dentro da cadeia informada.

```
print strstr("Casa", "C")           mostra 1  
print strstr("Vila Velha", "h")     mostra 9
```

substr("cadeia", início, fim)

Usada para apresentar o trecho de caracteres que seja citado entre as posições de **início** e **fim**.

```
print substr("Casa", 2, 4)          mostra asa  
print substr("Vila Velha", 6, 10)    mostra Velha
```

system("comando")

Usada para executar de dentro do programa **gnuplot** um comando externo do sistema operacional.

```
print system("pause")      mostra prompt em pause
```

word("cadeia", n)

Retorna a subcadeia indicada no argumento **n** da **cadeia**, sendo uma lista.

```
print word("um dois três", 2)    mostra dois
```

words("cadeia")

Retorna a quantidade de sub cadeia indicada no argumento **cadeia**.

```
print words("um dois três")      mostra 3
```

4.1.3 Funções diversas

Além das funções matemáticas e das funções de cadeia há um outro conjunto classificado como funções diversas que efetuam outras operações importantes, sendo apresentada a seguir.

As funções dessa categoria, em sua maioria, não são aqui contextualizadas e exemplificadas, pois exigem outros conhecimentos que ainda serão apresentados e comentados.

column(x)

Retorna o valor numérico inteiro de identificação de uma coluna estabelecida em um arquivo de dados no argumento **x** e aceita a informação retornada como entrada de dados no ambiente. Uma forma simplificada de uso da função é **\$x**. O argumento **x** é utilizado para indicar o valor na coluna. Se usada **column(1)**, a referência é o primeiro ponto de leitura da coluna; caso se use **column(2)**, a segunda coluna é a referência; e assim por diante.

columnhead(x)

Retorna o nome da coluna que contém a primeira entrada como argumento **x**, a partir de um arquivo de dados.

exists("variável") – similar a antiga função **defined(variável)** descontinuada

Função que retorna o valor 1 se o nome de uma variável existir na memória e retorna 0 caso a variável não exista na memória.

```
print exists( "x" )           mostra 1 se existir  
print exists( "x" )           mostra 0 se não existir
```

stringcolumn(x)

Retorna o conteúdo de uma coluna indicada no argumento **x** inteiro como uma variável de cadeia de caracteres. Esta função é usada em expressões para definir o desenho dos pontos gráficos a partir do acesso a um arquivo de dados.

time(x)

Retorna a hora do sistema atual. Este valor pode ser convertido para uma cadeia de data com a função **strftime()** ou pode ser utilizado em conjunto com **timecolumn()** para gerar o tempo relativo / parcelas de data. O tipo do argumento determina o que é retornada. Se o argumento for um número inteiro será dado retorno ao tempo atual em segundos a partir de 1 de janeiro de 2000. Se o argumento é real (ou complexo), o resultado será verdadeiro. Se o argumento for uma cadeia, será considerado uma sequência de formato passada para a função **strftime()** para indicar o uso de uma sequência formatada de tempo.

```
print time(0)                      mostra 412121016  
  
formato = "%a, %b, %d/%Y %Hh%Mmin"  
tempo = time(0)  
conta = strftime(formato, tempo)  
print "Data/Hora atual = ", conta
```

Será apresentada a informação **Data/Hora atual = Mon, 21/2013 22h19min.**

timecolumn(x)

É usada em expressões para a manipulação de arquivos de dados. Faz a leitura dos dados com base em uma coluna, como um valor de hora ou data, e retorna um valor de acordo com o formato definido pelo programa **gnuplot**.

tm_hour(x)

Interpreta o argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, e retorna o valor da hora, no intervalo de 0 a 23, como um número real.

```
print tm_hour(1234567890)          mostra 23.0
```

tm_mday(x)

Retorna o valor do argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, como o dia do mês, no intervalo de 1 a 31, como um número real.

```
print tm_mday(1234567890)         mostra 13.0
```

tm_min(x)

Interpreta o argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, e retorna o valor do argumento como minutos, no intervalo de 0 a 59, como um número real.

```
print tm_min(1234567890)           mostra 31.0
```

tm_mon(x)

Retorna o valor do argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, como mês, no intervalo de 0 a 11, como um número real.

```
print tm_mon(1234567890)           mostra 1.0
```

tm_sec(x)

Interpreta o argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, e retorna o valor do argumento como segundos, no intervalo de 0 a 59, como um número real.

```
print tm_sec(1234567890)           mostra 30.0
```

tm_wday(x)

Retorna o valor do argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, como o número do dia da semana, no intervalo de 0 a 6, como um número real.

```
print tm_wday(1234567890)          mostra 0.0
```

tm_yday(x)

Retorna o valor do argumento **x** como uma informação de tempo, em segundos, de 1 de janeiro de 2000 em diante, como o número do dia do ano, no intervalo de 1 a 366, como um número real.

```
print tm_yday(1234567890)          mostra 43.0
```

tm_year(x)

Retorna o valor do argumento x como uma informação de tempo, em segundos, a partir de 1 de janeiro de 2000, sendo este valor referente ao ano informado e devolvido como um número real.

```
print tm_year(1234567890)           mostra 2039.0
```

valid(x)

Testa a validade da coluna indicada no argumento x a partir de arquivo de dados.

value(x)

Essa função permite estabelecer a atribuição do valor de uma variável sobre outras. Por exemplo, se definido **b = value(a)**, a variável **b** passa a ter o valor da variável **a**, ou seja, utilizar essa função é o mesmo que usar a atribuição direta **b = a**. Essa função permite efetuar a leitura do nome de uma variável com base em um arquivo de dados. Se o argumento **x** for uma expressão numérica, retorna-se o valor da expressão. Se o argumento for uma cadeia que não corresponde a uma variável definida, retorna-se o valor **NaN**.

```
a = 5  
b = value(a)  
print a, b
```

xtic(x) / ytic(x) / ztic(x)

Estas funções determinam que o conteúdo de uma coluna de um arquivo de dados representada pelo argumento **x** seja usada como rótulo do eixo indicado (**x**, **y** ou **z**).

4.2 Funções definidas pelo usuário

Até o momento, foram apresentadas diversas funções do programa **gnuplot**; porém, apesar desse extenso conjunto, muitas outras funções não estão disponíveis. No entanto, é possível criar funções das quais se necessita.

A definição de funções pelo usuário pode ser feita com o estilo de instrução:

função(argumento) = ação

E a partir desta definição, basta executar a instrução como se segue:

plot função(argumento)

Para demonstrar essa ocorrência, considere a criação das funções dispostas na tabela a seguir, que não existem no programa **gnuplot**.

Função	Operação
cot(x)	Cotangente
sccs(x)	Cosecante
sec(x)	Secante
coth(x)	Cotangente hiperbólica
scsh(x)	Cosecante hiperbólica
sech(x)	Secante hiperbólica
acot(x)	Arco cotangente
acsc(x)	Arco cosecante
asec(x)	Arco secante
acoth(x)	Arco cotangente hiperbólico
ascsh(x)	Arco cosecante hiperbólico
asech(x)	Arco secante hiperbólico
nrt(b.n)	Raiz de índice n

Basta informar as linhas de instruções a seguir com os algoritmos para cada função de modo que estejam disponíveis para uso.

```

cot(x) = 1/tan(x)
csc(x) = 1/sin(x)
sec(x) = 1/cos(x)
coth(x) = exp(-x)/(exp(x)-exp(-x))*2+1
csch(x) = 2/(exp(x)-exp(-x))
sech(x) = 2/(exp(x)+exp(-x))

```

```
acot(x) = -atan(x)+pi/2
acsc(x) = atan(1/sqrt(1-x**2))+(sgn(x)-1)*pi/2
asec(x) = atan(x/sqrt(1-x**2))+(sgn(x)-1)*pi/2
acoth(x) = log((x+1)/(x-1))/2
acsch(x) = log((sgn(x)*sqrt(x**2+1)+1)/x)
asech(x) = log((sqrt(1-x**2)+1)/x)
nrt(b,n) = b**(1.0/n)
```

Apesar de resolvido este problema apresenta-se outra questão. Imagine que haja a necessidade de sempre estar se utilizando essas “novas” funções. É inconveniente ter que ficar escrevendo essas funções todas as vezes que inicializa o programa **gnuplot**.

Uma alternativa é criar um arquivo contendo essas funções e constantes que seja automaticamente executado pelo programa **gnuplot** quando este é executado. O arquivo em questão deverá chamar-se **.gnuplot** se estiver em uso um sistema operacionais padrão Unix (Linux/Mac) e **gnuplot.ini** se estiver em uso o sistema operacional Windows ou OS/2.

No caso do sistema operacional Windows é necessário configurar uma variável de ambiente chamada **GNUPLOT** que faça acesso ao diretório onde o programa está instalado. Neste caso, diretório **C:\Program Files\gnuplot\bin**. Para tanto, abra a janela **Propriedades do Sistema**, que no Windows 7, está disponível para acesso em **Botão: Iniciar\Painel de Controle\Sistema e Segurança\Sistema**, selecione **Configurações Avançadas do Sistema** e na caixa de diálogo apresentada acione o botão **Variáveis de Ambiente**. Quando apresenta a caixa de diálogo **Variáveis de Ambiente** selecione no quadro **Variáveis de Ambiente** o botão **Novo**. No campo **Nome da variável** informe **GNUPLOT** e no campo **Valor da variável** informe o local **C:\Program Files\gnuplot\bin**. Acione o botão **OK** para sair da caixa de diálogo **Nova Variável de Sistema**. Em seguida acione novamente o botão **OK** para sair da caixa de diálogo **Variáveis de Ambiente** e por último acione o botão **OK** mais uma vez para sair da caixa de diálogo **Propriedades do Sistema**.

No caso do sistema operacional Unix (Linux ou Mac) grave o arquivo no diretório **/home/<usuário>/lib/gnuplot**, onde **<usuário>** é nome do usuário do computador e acrescente esta informação no arquivo **.gnuplot** com a instrução **set**.

Além das funções é pertinente definir algumas constantes matemáticas. Para tanto, as constantes definidas pelo usuário serão criadas com um prefixo de identificação

c_ (ação opcional) para evitar conflitos com uma variável preexistente que leve o mesmo nome, não inviabilizando o uso da variável original. Assim sendo, considere as constantes indicadas com a, mantissa de 16 casas. Internamente o programa assume apenas 14 casas:

```
c_e      = 2.7182818284590452354
c_euler = 0.5772156649015328606
c_phi   = 1.6180339887498948482
```

Veja a seguir a estrutura do arquivo de inicialização. O código apresentado é o mesmo tanto para o sistemas operacionais Unix (Linux/Mac) e Windows.

```
# Carrega biblioteca pessoal
#####
# Definição de constantes
#####
c_e      = 2.7182818284590452354
c_euler = 0.5772156649015328606
c_phi   = 1.6180339887498948482

# Definição de funções
#####
cot(x)  = 1/tan(x)
csc(x)  = 1/sin(x)
sec(x)  = 1/cos(x)
coth(x) = exp(-x)/(exp(x)-exp(-x))*2+1
csch(x) = 2/(exp(x)-exp(-x))
sech(x) = 2/(exp(x)+exp(-x))
acot(x) = -atan(x)+pi/2
acsc(x) = atan(1/sqrt(1-x**2))+(sgn(x)-1)*pi/2
```

```
asec(x) = atan(x/sqrt(1-x**2))+(sgn(x)-1)*pi/2  
acoth(x) = log((x+1)/(x-1))/2  
acsch(x) = log((sgn(x)*sqrt(x**2+1)+1)/x)  
asech(x) = log((sqrt(1-x**2)+1)/x)  
nrt(b,n) = b**(1.0/n)
```

Salve o arquivo de configuração, feche o programa **gnuplot**. Em seguida faça sua chamada e execute o comando **print** com as constantes definidas para ver os resultados apresentados.

A parte de conteúdo do arquivo de inicialização, que possui o símbolo trilha (#), representação matemática de número, é usado para definir uma linha de comentário. Ela não é executada no momento de inicialização do programa **gnuplot**.

Segue no estilo deste capítulo a documentação e exemplos de uso das funções definidas pelo usuário.

cot(x)

Retorna o valor da tangente de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print cot(45*pi/180)      retorna o valor 1.6
```

scc(x)

Retorna o valor do cosecante de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print csc(30*pi/180)      retorna o valor 2.0
```

sec(x)

Retorna o valor do secante de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print sec(60*pi/180)      retorna o valor 2.0
```

coth(x)

Retorna o valor do cotangente hiperbólica de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print coth(log(2.0))      retorna o valor 1.666666666...
```

csch(x)

Retorna o valor do cosecante hiperbólica de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print csch(log(2.0))      retorna o valor 1.333333333...
```

sech(x)

Retorna o valor do secante hiperbólica de um valor fornecido em radianos. Opera com números inteiros, reais ou complexos.

```
print sech(log(2.0))      retorna o valor 0.8
```

acot(x)

Retorna o valor do arco cotangente de um valor fornecido em radianos, sendo este o inverso da tangente. Opera com números inteiros, reais ou complexos.

```
print acot(1.0)*180.0/pi  retorna o valor 45.0
```

acsc(x)

Retorna o valor do arco cossecante de um valor fornecido em radianos, sendo este o inverso da tangente. O argumento **x** deve ser expresso com um valor maior ou igual a **0** e menor que **1**. Opera com números inteiros, reais ou complexos.

```
print acsc(0)*180.0/pi   retorna o valor -45.0
```

asec(x)

Retorna o valor do arco secante de um valor fornecido em radianos, sendo este o inverso da tangente. Opera com números inteiros, reais ou complexos.

```
print asec(0.5)*180.0/pi  retorna o valor 30.0
```

acoth(x)

Retorna o valor do arco cotangente hiperbólico de um valor fornecido em radianos, sendo este o inverso do cosseno. Opera com números inteiros, reais ou complexos.

```
print acoth(60.0*pi/180)    retorna o valor 1.884942539...
```

acsch(x)

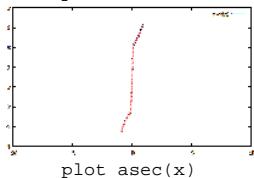
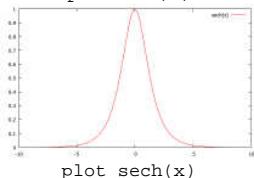
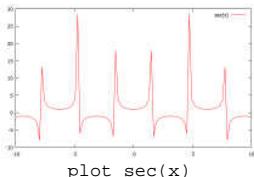
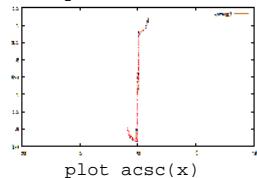
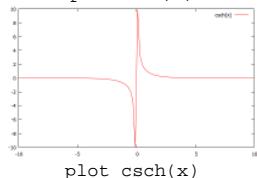
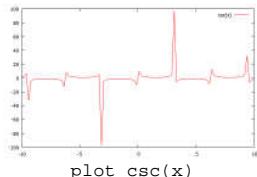
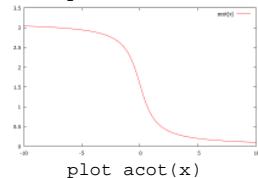
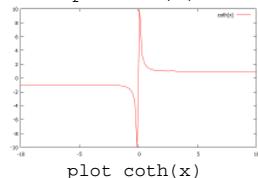
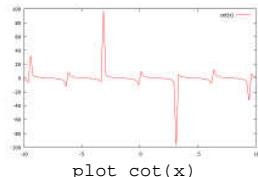
Retorna o valor do arco cossecante hiperbólico de um valor fornecido em radianos, sendo este o inverso do cosseno. Opera com números inteiros, reais ou complexos.

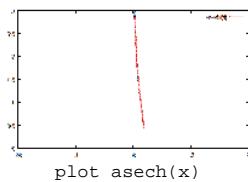
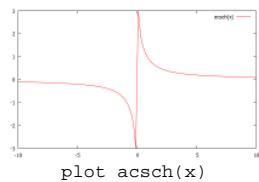
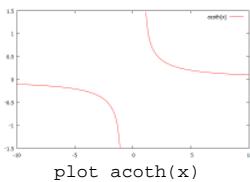
```
print acsch(60.0*pi/180)    retorna o valor 0.849142301...
```

asech(x)

Retorna o valor do arco secante hiperbólico de um valor fornecido em radianos, sendo este o inverso do cosseno. Opera com números inteiros, reais ou complexos.

```
print asech(1.0*pi/180)    retorna o valor 4.741297982...
```

Gráficos das Funções Definidas pelo Usuário



* * * Anotações * * *

5

Programação

Este capítulo mostra detalhes relacionados a programação do programa **gnuplot**. São apresentadas as etapas de trabalho com arquivos externos com dados usados na criação de gráficos desenvolvidos nos programas **Bloco de notas**, **Excel** ou **Calc**; uso de arquivos com scripts (programas escritos na linguagem do gnuplot) e técnicas básicas de programação. São apresentados novos comandos e/ou recursos como: código de controle **\n**, **index**, **linecolor**, **set key**, **set xtics**, **smooth**, **rotate**, **set view**, **using**, **set xdata**, **set timefmt**, **set bars**, **set terminal**, **size**, **set size**, **set rmargin**, **set lmargin**, **every**, **xtic**, **set datafile missing**, **set bmargin**, **set tmargin**, **set datafile separator**, entre outros.

5.1 Arquivos externos

Até o presente momento foram criados gráficos a partir de funções, onde a própria função efetuou automaticamente a entrada dos valores para o desenho do gráfico. No entanto, este tipo de ação é de certa maneira limitada, pois há situações onde os dados devem ser informados para o programa **gnuplot** a partir de outras fontes.

Os arquivos externos para serem utilizados deverão ser gravados em formato texto puro (apenas códigos ASCII¹), normalmente com extensão **.txt**, mas há o的习惯 desses arquivos para uso no programa **gnuplot** serem gravados com a extensão **.dat**, a propósito pode ser usada qualquer extensão, desde que o arquivo esteja gravado no formato ASCII puro. Nesta obra estará sendo usada a extensão **.txt**.

Os dados no arquivo deverão estar baseados em colunas, podendo ter quantas colunas forem necessárias, mas lembrando de que se o gráfico é 2D duas colunas serão

¹ American Standard Code for Information Interchange (Código Americano Padrão para Intercâmbio de Informações), pronuncia-se *asqui* e não *asqui dois*, como algumas pessoas inadvertidamente falam.

usadas, se for um gráfico 3D três colunas serão usadas. Não poderá jamais ser usado para um gráfico 3D um arquivo que contenha apenas duas colunas. A menos que seja feita uma ação de cálculo com as duas colunas existentes para gerar o conteúdo de uma terceira coluna, como será mostrado neste tópico.

Arquivos texto são gerados em programas de edição de textos como o **Bloco de notas** do Windows ou podem ser gerados a partir de uma planilha eletrônica como o programa **Excel** ou **Calc**. O detalhe a ser considerado é que os números do arquivo devem estar separados com o uso do caractere de controle gerado pelo uso da tecla **<Tab>**. É claro que se pode usar outra forma de delimitação para separar valores, mas deverá esta forma ser configurada com a instrução **set datafile separator**.

O arquivo de dados externo deve ser gravado preferencialmente no diretório de trabalho do usuário. Para preparar o arquivo de dados considere inicialmente duas colunas contendo os dados para o eixo das abscissas referentes aos dias da semana e para o eixo das ordenadas os dados referentes ao número de clientes atendidos. A tabela em questão considera cinco dias da semana de segunda-feira a sexta-feira numerados de 1 a 5. A Figura 5.1 mostra a imagem do arquivo produzido em um editor de textos. Grave o arquivo com o nome **clientes.txt**.

The screenshot shows a Windows Notepad window with the title bar 'clientes.txt - Bloco de notas'. The menu bar includes 'Arquivo', 'Editar', 'Formatar', 'Egíbir', and 'Ajuda'. The main content area contains the following text:

```
# Primeira coluna - dias da semana
# 1 = segunda-feira
# 2 = terça-feira
# 3 = quarta-feira
# 4 = quinta-feira
# 5 = sexta-feira
# Segunda coluna - número de clientes atendidos

1      23
2      17
3      42
4      26
5      75
```

Figura 5.1 – Arquivo **clientes.txt**.

As linhas indicadas com o símbolo tralha (#) são ignoradas pelo programa **gnuplot**. Efetivada a criação do arquivo execute no programa **gnuplot** a instrução:

```
plot "clientes.txt" with lines
```

Será apresentado um gráfico de linhas com os dados lidos a partir do arquivo dispostos respectivamente nas coordenadas **x** com os dias da semana e **y** com o total de clientes atendidos, como mostra a Figura 5.2.

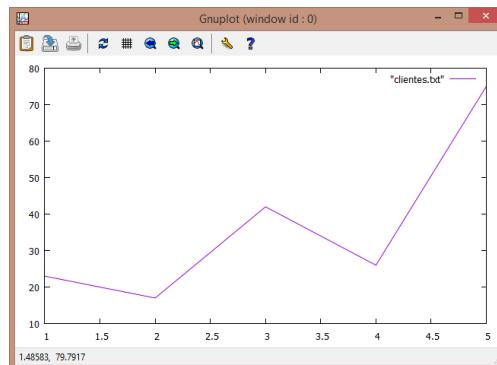


Figura 5.2 – Gráfico a partir do arquivo clientes.txt.

Se ao executar o comando para desenhar o gráfico fosse omitido o complemento **with lines** o gráfico seria mostrado apenas pontos e não em linhas como feito. Note que há uma diferença padrão na execução de um gráfico a partir de uma função e a execução de um gráfico a partir de um arquivo esterno.

Note que a apresentação do gráfico é realizada a partir de uma ação padrão, sendo que há alguns detalhes a serem configurados, como **xrange**, **yrange**, **title**, **ytics**, **ylabel**, **xlabel**, **set key right bottom** e **mytics**. Assim, use os comandos a seguir e note o resultado junto a Figura 5.3.

```
reset
set xrange [0.5:5.5]
set yrange [0:90]
set mytics 10
set ytics 20
set title "Relação \n Dias da semana X Clientes"
set xlabel "Dias da semana"
set ylabel "Cliente atendidos"
set key right bottom
plot "clientes.txt" with lines title "Atendimentos"
```

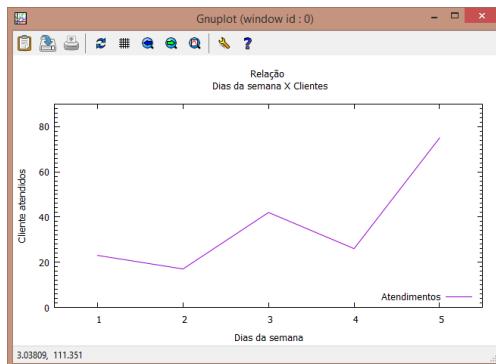


Figura 5.3 – Gráfico com ajustes.

Os detalhes da sequência de instruções indicadas já são conhecidas, exceto ao uso do código de controle `\n` na cadeia "**Relação \n Dias da semana X Clientes**", que faz com que haja a divisão do texto em duas linhas no ponto onde o código é indicado, e ao uso da instrução `set key right bottom` que fez com que a legenda do gráfico fosse posicionada no canto inferior direito. Veja a legenda **Atendimentos** definida com o complemento `titles` do comando `plot`.

A mudança da posição da legenda de um gráfico é feita com a instrução `set key` com o uso dos seguintes argumentos:

- **left bottom** – canto inferior esquerdo;
- **right bottom** – canto inferior direito;
- **left top** – canto superior esquerdo;
- **right top** – canto superior direito;
- **outside** – legenda fora da área do gráfico;
- **bellow** – legenda fora e abaixo da área do gráfico;
- **box** – legenda com moldura;
- **at x:y** – define a posição da legenda na coordenada informada.

Para desfazer uma configuração estabelecida com **set key** usa-se **unset key** ou **reset** para desabilitar todos os **set**. Para ver as definições do complemento **key** use a instrução **show key**. Na sequência será usado um arquivo externo com três colunas, sendo o eixo das abscissas com os dias da semana, o eixo das ordenadas com o número de clientes atendidos e o eixo das cotas com o total da venda do dia.

Para esta segunda proposta será criada uma planilha no programa **Excel** ou **Calc** contendo três colunas: a primeira coluna refere-se ao dia da semana (mesma regra do arquivo anterior), a segunda coluna ao número de clientes atendidos e a terceira coluna ao valor de vendas efetuado no dia. A Figura 5.4 mostra como deverá ser definido os dados junto a planilha. Atente para uso do símbolo trilha a frente dos nomes das colunas, bastaria apenas colocar o símbolo a frente da primeira coluna.

The figure displays two screenshots of spreadsheet software. On the left is Microsoft Excel with a window titled 'Pasta1 - Excel'. It shows a table with three columns: '#Dia', '#Cliente', and '#Venda'. The data rows are as follows:

#Dia	#Cliente	#Venda
1	23	154
2	17	234
3	42	564
4	26	389
5	75	342

On the right is LibreOffice Calc with a window titled 'Sem título 1 - LibreOffice Calc'. It also shows a table with three columns: '#Dia', '#Cliente', and '#Venda'. The data rows are identical to the Excel table:

#Dia	#Cliente	#Venda
1	23	154
2	17	234
3	42	564
4	26	389
5	75	342

Figura 5.4 – Planilha Excel com dados do arquivo vendas.txt.

Um detalhe a ser considerado na planilha é a colocação do símbolo trilha a frente do nome da primeira coluna para não ocorrer erro na leitura do arquivo. Nas demais colunas o símbolo foi colocado apenas por uma questão de homogeneidade.

Para gravar o arquivo em formato texto no programa **Excel** basta na caixa de diálogo **Salvar como** selecionar no campo **Tipo** a opção **Texto (separado por tabulações)**, no campo **Nome do arquivo** indique o nome **vendas**, mantendo o diretório de trabalho **Documentos** selecionado. Basta para finalizar acionar o botão **Salvar**. Será apresentada mensagens de advertência. Basta apenas acionar o botão **Sim**.

Para gravar o arquivo em formato texto no programa **Calc** basta na caixa de diálogo **Salvar como** selecionar no campo **Tipo** a opção **Texto CSV (.csv)**, no campo **Nome do arquivo** indique o nome **vendas.txt**, mantendo o diretório de trabalho **Documentos** selecionado. Basta para finalizar acionar o botão **Salvar**. Será

apresentada mensagens de advertência. Basta apenas acionar o botão **Utilize o formato texto CVS**. Ao ser apresentada a caixa de diálogo **Exportar arquivo de texto**, selecione para o campo **Delimitador de campo** a opção **Tabulação**. Desmarque a opção **Salvar o conteúdo da célula como mostrado** e acione o botão **OK**. Quando apresentada uma caixa de advertência informando sobre a gravação, acione o botão **OK**.

No programa **gnuplot** execute a instrução a seguir e observe na Figura 5.5 o gráfico desenhado.

```
reset
set grid
set xrange [0.5:5.5]
set yrange [0:90]
set mytics 10
set ytics 20
set ztics 200
set title "Relação \n Dias da semana X Clientes"
set xlabel "Dias da semana"
set ylabel "Cliente atendidos"
set zlabel "Vendas efetuadas" rotate by 90
set key left top
splot "vendas.txt" with lines title "Volume de vendas"
```

O código de instruções anterior também é conhecido. Destacando-se o uso de **ztics** para estabelecer os **tics** do eixo de cotas, o complemento **rotate by** com valor **90** junto ao comando **zlabel** para direcionar em noventa graus o ângulo de giro da apresentação da legenda do eixo. Note o uso da instrução **set key left top** para a colocação da legenda do gráfico na parte superior esquerda.

É sabido que se posicionar o ponteiro do *mouse* dentro da janela gráfica e arrastá-lo o gráfico muda de posição. Quando esta ação é efetuada ocorrem mudanças nos valores da barra de *status* da janela. Observe na Figura 5.5 que a barra de *status* apresenta as informações padrão **view: 60.0000, 30.000 scale: 1.00000 1.00000**.

Quando o gráfico é mudado de posição com o *mouse* esses valores se alterar. O trecho **view** se refere ao controle do ângulo de visão do gráfico, sendo o primeiro

valor o indica controle de giro no eixo **x** e o segundo valor indica o controle de giro no eixo **z**. O trecho **scale** se refere a escala de tamanho do gráfico dentro da tela gráfica, o primeiro indica o aumento na escala total do gráfico e o segundo valor indica o aumento na escala do eixo **z**. Com o uso do ponteiro do *mouse* apenas o **view** e ajustado, a **scale** não pode ser alterada a não ser via *prompt* de comando.

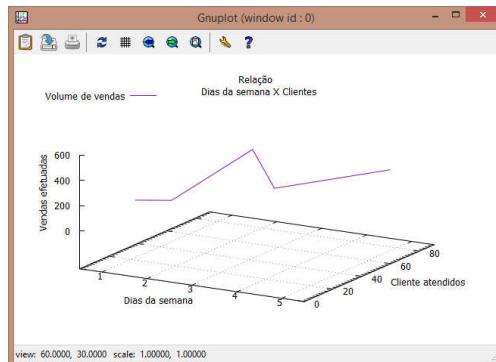


Figura 5.5 – Gráfico a partir do arquivo vendas.txt.

A mudança da rotação do gráfico pode se efetuar interativamente. Por meio de uso da instrução a seguir que apresentará o gráfico como indica a Figura 5.6.

```
set view 53, 134, 1, 1.5  
replot
```

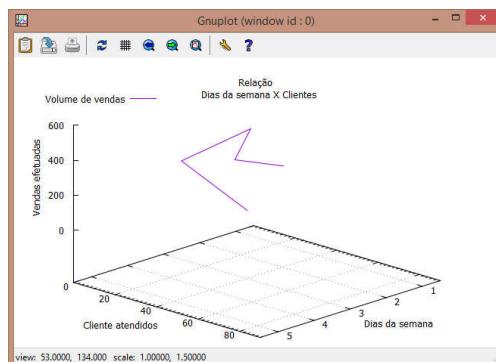


Figura 5.6 – Gráfico com ângulo ajustado.

Para o gráfico ser apresentado na sua forma padrão basta executar a instrução:

```
set view 60, 30, 1, 1  
replot
```

A Figura 5.7 mostra o arquivo **vendas.txt** criado na planilha eletrônica **Excel** aberto no programa **Bloco de notas**.

Um arquivo de dados pode conter várias colunas. Por exemplo, um arquivo com três colunas que será usado para a apresentação de dados em um gráfico bidimensional. A Figura 5.8 mostra o conteúdo dos campos **Código** (coluna 1), **Quant.** (coluna 2) e **Valor** (coluna 3) do arquivo **estoque.txt**.

vendas.txt - Bloco de notas		
#Dia	#Cliente	#Venda
1	23	154
2	17	234
3	42	564
4	26	389
5	75	342

Figura 5.7 – Arquivo vendas.txt aberto no Bloco de notas.

estoque.txt - Bloco de notas		
#Código	Quant.	Valor
1	2	800
2	3	330
3	9	150
4	5	350
5	8	55

Figura 5.8 – Arquivo de dados estoque.txt.

Para a apresentação de gráficos bidimensionais basta que se use duas colunas da quantidade de colunas existentes. Automaticamente o gráfico a ser criado mostrará os dados das colunas **Código** e **Quant.**, a partir da execução da instrução a seguir que mostrará o gráfico indicado na Figura 5.9.

```
reset
set grid
set title "Produtos em Estoque"
set xlabel "Código do Produto"
set ylabel "Quantidade em estoque"
set xrange [1:5]
set xtics 1
set yrange [1:9]
set key left top
plot "estoque.txt" with lines
```

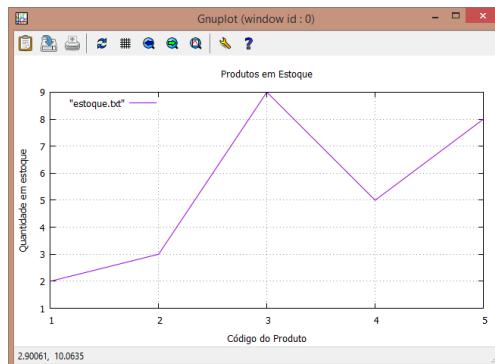


Figura 5.9 – Gráfico estoque.txt com coluna 1 em x e coluna 2 em y.

Como o arquivo de dados possui mais de duas colunas é possível selecionar, por exemplo, o eixo **x** com o **Código** e o eixo **y** com o **Valor**. Para tanto, execute as instruções a seguir e veja na Figura 5.10 o resultado desta ação.

```
reset  
set grid  
set title "Produtos em Estoque"  
set xlabel "Código do Produto"  
set ylabel "Valor unitário em R$"  
set xrange [1:5]  
set xtics 1  
set yrange [0:800]  
set key right bottom  
plot "estoque.txt" using 1:3 with lines
```

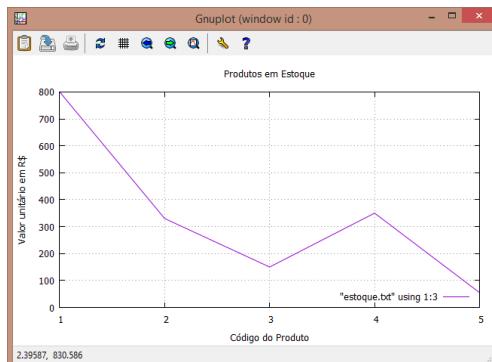


Figura 5.10 – Gráfico estoque.txt com coluna 1 em x e coluna 3 em y.

Observe o uso do complemento **using** com o parâmetro **1:3** indicando o uso da coluna 1 (eixo x) e coluna 3 (eixo y) antes do complemento **with**. Se colocado este complemento em outra posição ocorrerá um erro de execução do comando.

Outra possibilidade é apresentar um gráfico a partir do arquivo externo **vendas.txt** que mostre os códigos do produto no eixo x e no eixo y indique a multiplicação do valor da quantidade em estoque com o valor unitário. Para tanto, execute as seguintes instruções e veja na Figura 5.11 o resultado desta ação.

```
reset
set grid
set title "Produtos em Estoque"
set xlabel "Código do Produto"
set ylabel "Valor total em R$ \n no estoque"
set xrange [1:5]
set xtics 1
set yrange [1:2000]
set key left bottom
plot "estoque.txt" using ($1):($3*$2) with lines
```

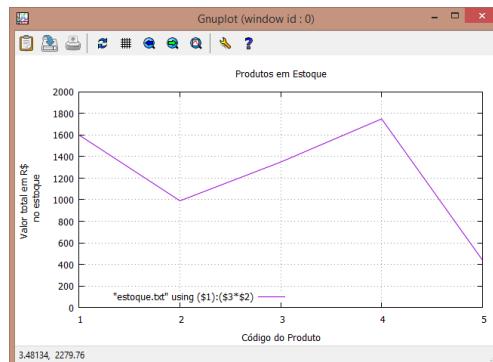


Figura 5.11 – Gráfico com coluna 1 em x e colunas 2 multiplicada por 3 em y.

Outra possibilidade em um gráfico é mudar de posição os conteúdos do eixo x em relação ao eixo y. Os eixos permanecem na posição padrão o que se muda é o conteúdo apresentado. Para tanto, a partir do arquivo de dados **clientes.txt** far-se-á um gráfico que apresentará no eixo x as informações da segunda coluna e no eixo y as informações da primeira coluna. A Figura 5.12 mostra esta ocorrência.

```
reset
plot "clientes.txt" using 2:1 with lines
```

Compare a Figura 5.12 com a Figura 5.2 e veja a numeração dos eixos. Para apresentar a forma padrão não é necessário usar o complemento **using** com **1:2**.

A partir do conhecimento apresentado o próximo arquivo de dados terá armazenada as informações de uma semana do movimento de ações da empresa na bolsa de valores. Assim sendo, considere o arquivo de dados, da Figura 5.13, a ser definido com o nome **acoes.txt**.

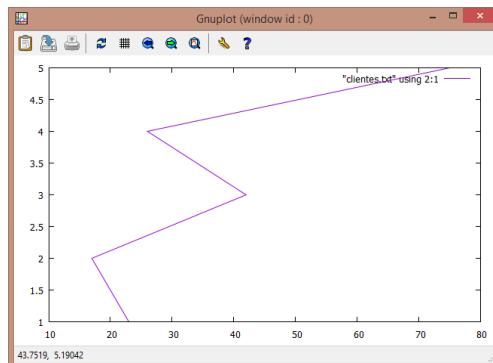


Figura 5.12 – Gráfico do arquivo *clientes.txt* com eixos trocados.

A coluna **Abre** mostra o valor da ação na abertura da bolsa de valores, a coluna **Alta** indica o valor máximo atingido no pregão (período de operação da bolsa), **Baixa** apresenta o menor valor que a ação atingiu e **Fecha** mostra o valor da ação no momento do fechamento das operações da bolsa de valores.

Data	Abre	Alta	Baixa	Fecha
07-Jan-13	79.85	88.55	54.81	75.12
08-Jan-13	78.25	93.23	77.35	80.15
09-Jan-13	76.32	78.90	60.61	77.54
10-Jan-13	75.83	87.25	75.25	85.24
11-Jan-13	90.23	99.67	75.42	98.34

Figura 5.13 – Arquivo de dados *acoes.txt*.

Em seguida execute as instruções seguintes e confira na Figura 5.14 o resultado do gráfico apresentado.

```
reset
set xdata time
set timefmt "%d-%b-%y"
set format x "%d-%b"
set bars 8
set yrang [50:110]
set xrange ["06-Jan-13":"12-Jan-13"]
set xtics rotate by -45
set terminal wxt size 640, 480
set rmargin 5
set bmargin 3
plot "acoes.txt" using 1:2:3:4:5 with financebars
```

A instrução **set xdata** está determinando que os elementos apresentados no eixo x são referentes a informação de tempo (data calendário/hora). Em **set timefmt** está sendo definido o formato da data usado no arquivo DIA-MÊS-ANO (%d-\$b-%y).

A instrução **set format x** estabelece a formatação que os dados do eixo x deverão possuir. Com a instrução **set bars** se defini o tamanho que as barras horizontais vão ser apresentadas.

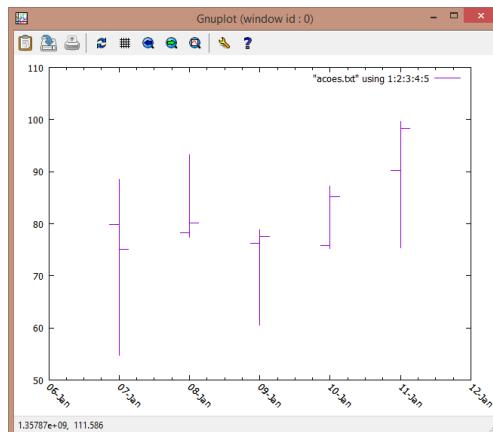


Figura 5.13 – Gráfico do arquivo acoes.txt.

Na linha de código **set xrange ["06-Jan-13":"12-Jan-13"]** faz a definição da faixa de abrangência que o eixo **x** irá usar para apresentar os pontos do gráfico. Como neste exemplo está se trabalhando com informação de tempo, faz-se a definição de uma faixa de calendário.

Com a instrução **set xtics rotate by -45** está se estabelecendo que o eixo **x** terá sua legenda escrita de forma inclinada da esquerda para a direita. O valor **-45** indica um giro de menos quarenta e cinco graus.

A instrução **set terminal wxt size 640, 480** configura o tamanho (**size**) da tela de apresentação do gráfico será de 640 por 480 pontos, no **terminal** em modo **wxt** que é o terminal padrão do programa.

As instruções **set rmargin** e **set bmargin** definem o tamanho da distância da borda da tela em relação a borda do gráfico. Neste caso estão sendo usada para permitir melhor visualização do gráfico na tela. Os complementos **rmargin** e **bmargin** alteram respectivamente as margens direita e inferior da tela gráfica. É possível usar também os complementos **lmargin** e **tmargin** para manipulação das margens esquerda e superior.

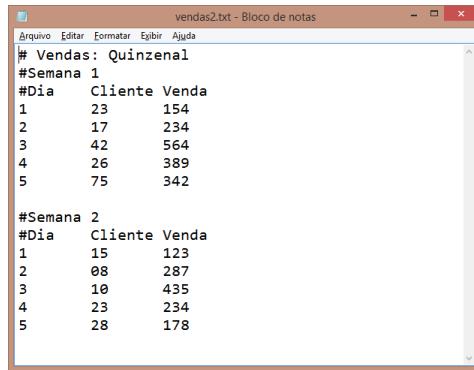
Por último o parâmetro **financebars** do complemento **with** do comando **plot** permite apresentar o gráfico em estilo financeiro.

Alguns comandos usados podem ser simplificados, tais como: **replot** como **rep**, **with** como **w**, **using** como **u**, **title** como **t**, **linespoints** como **linep** ou **lp**, **points** como **p**, **lines** como **l**, entre outros. Experimente modificar os comandos indicados e veja o resultado apresentado.

Os arquivos de dados externos poderão ter dados dispostos de várias maneiras. Imagine um arquivo de dados **vendas2.txt** contendo as informações do movimento quinzenal de vendas. A Figura 5.14 mostra a estrutura deste arquivo.

Note que o arquivo é dividido por dois lotes semanais separados por linhas em branco, como se fosse a junção de vários outros arquivos em um mesmo arquivo. As linhas com tralha, como já comentado, são ignoradas, mas uma linha em branco possui para o programa **gnuplot** o significado de descontinuidade dos dados.

Os dados existentes acima e abaixo da linha em branco serão tratados como pertencentes ao mesmo conjunto, tanto que serão apresentados na mesma cor de linha. No entanto, serão apresentadas duas linhas distintas, pois a linha em branco existente cria o efeito de quebra na disposição dos dados (JANERT, 2010).



```
# Vendas: Quinzenal
#Semana 1
#Dia Cliente Venda
1 23 154
2 17 234
3 42 564
4 26 389
5 75 342

#Semana 2
#Dia Cliente Venda
1 15 123
2 08 287
3 10 435
4 23 234
5 28 178
```

Figura 5.14 – Arquivo de dados vendas2.txt.

No entanto mostra a ocorrência de cada semana num quadro quinzenal de dados. Para ver o gráfico da Figura 5.15 execute os comandos a seguir.

```
reset
plot "vendas2.txt" with lines
```

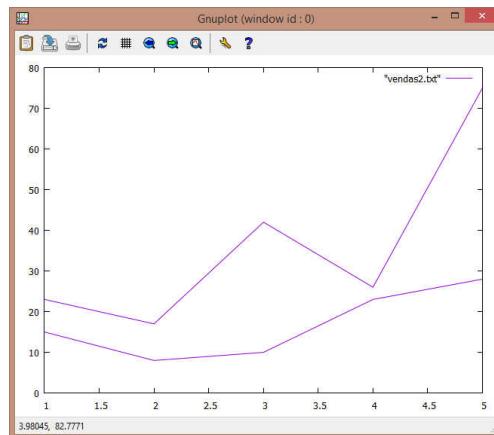


Figura 5.15 – Gráfico do arquivo vendas2.txt.

A partir do arquivo **vendas2.txt** crie o arquivo **vendas3.txt** onde há entre a primeira e segunda semanas a divisão com duas linhas em branco.

A separação de dados com duas linhas em branco permite selecionar a faixa de dados de forma distinta. Assim é possível escolher qual semana será desenhada no gráfico por meio do complemento **índex** (pode ser usado *i*). A Figura 5.16 apresenta a disposição dos dados do arquivo **vendas3.txt**.

```

vendas3.txt - Bloco de notas
Arquivo Editar Formatar Egíbir Ajuda
# Vendas: Quinzenal
#Semana 1
#Dia Cliente Venda
1 23 154
2 17 234
3 42 564
4 26 389
5 75 342

#Semana 2
#Dia Cliente Venda
1 15 123
2 08 287
3 10 435
4 23 234
5 28 178

```

Figura 5.16 – Arquivo de dados vendas3.txt.

Atente junto a Figura 5.16 o fato de haver entre os dados da primeira e segunda semanas duas linhas em branco. Se for executada a instrução **plot "vendas3.txt" with lines** o efeito apresentado será o mesmo da Figura 5.15, mas se for executada a instrução a seguir o efeito será diferente, como pode ser constatado junto a Figura 5.17.

```

reset
plot "vendas3.txt" index 0 with lines

```

O complemento **índex** com argumento **0** definido apresenta os dados apenas do primeiro grupo referente a primeira semana. Caso queira ver o gráfico da segunda semana use **índex 1** como indica as instruções a seguir e a Figura 5.18.

```

reset
plot "vendas3.txt" index 1 with lines

```

Atente para o fato do argumento do complemento **índex** ser utilizado no sentido cardinal. Este complemento permite selecionar o conjunto de dados específico do arquivo de dados que será desenhado a partir de certo grupo de dados.

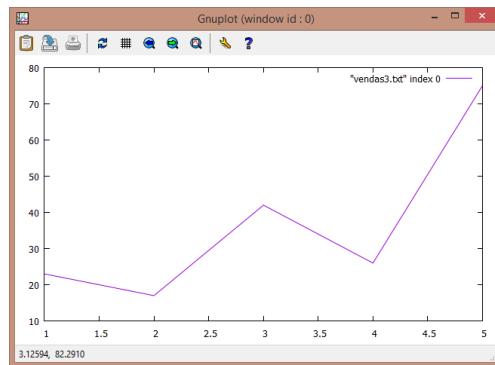


Figura 5.17 – Gráfico do arquivo vendas3.txt com a primeira semana.

Outra maneira de selecionar os dados de um grupo de dados é fazer uso da linha de comentário que antecede os dados fornecendo como nome de identificação da faixa de dados. As instruções a seguir desenham um gráfico a partir do nome do grupo de dados **Semana 2**.

```
reset  
plot "vendas3.txt" index "Semana 2" with lines
```

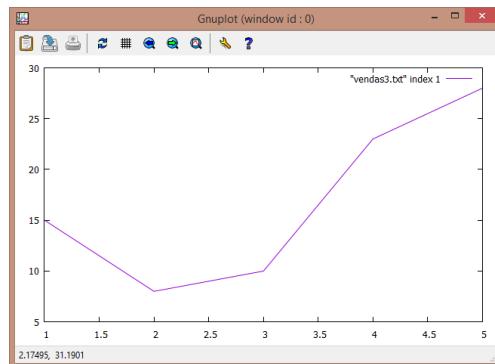


Figura 5.18 – Gráfico do arquivo vendas3.txt com a segunda semana.

Outro recurso é a possibilidade de apresentar os dados do mesmo arquivo com linhas de cores diferentes para cada semana. Para tanto, execute as instruções seguintes.

```
reset  
plot "vendas3.txt" using 1:2:-2 \  
with lines linecolor variable
```

Atente inicialmente para o símbolo barra invertida (\) usado quando se deseja definir uma quebra de linha de maneira que o programa **gnuplot** saiba que a linha seguinte é continuação da linha anterior. Este recurso é conveniente quando se possui uma linha de instrução muito extensa.

Note o complemento **using** com a indicação **1:2:-2**, onde **1** indica é o primeiro bloco de dados do arquivo a ser desenhado, **2** é o segundo bloco de dados do arquivo a ser desenhado. **-2** indica a contagem de retorno em dois blocos como se cada grupo de dados estivesse fisicamente separados. Por exemplo, imagine um arquivo com sete blocos de dados, o primeiro bloco é **0** e o sétimo bloco é **6** e informasse para **using** o argumento **1:4** indicando que o gráfico a ser desenhada usará os dados do bloco **1** até o bloco **4**, mas se for indicado **using 2:5:2** isto dirá que deverão ser usados os blocos de **2** até **5** de **2** em **2** a partir de **2**, ou seja, serão considerado os blocos **2** e **4**.

O complemento **linecolor** é usado para mudar a cor da linha de curva apresentada em um gráfico com argumento **variable** (seleciona cores variavelmente) que tenham sido obtidos a partir de um arquivo de dados. Enquanto que **linestyle** é usado para alterar a cor das linhas de gráficos via **plot** e **splot**.

Outra situação a ser considerada é o uso de arquivo de dados que contenha seus dados dispostos de forma intercalada. Para tanto, considere o arquivo **vendas4.txt** disposto como mostra a Figura 5.19.

Note que a coluna **Dia** apresenta pares de informação que vinculados a coluna **Ocorrência** que associa o número de cliente atendidos e as vendas realizadas. Quando se possui esta estrutura de arquivo o tratamento para sua apresentação é feito de maneira diferenciada. Execute as instruções a seguir e observe junto a Figura 5.20 o gráfico apresentado.

```
reset  
plot "vendas4.txt" every 2 using 1:2 with lines
```

#Dia	Ocorrência
1	17 # Clientes no dia
1	123 # Vendas efetuadas
2	12 # Clientes no dia
2	157 # Vendas efetuadas
3	09 # Clientes no dia
3	247 # Vendas efetuadas
4	25 # Clientes no dia
4	432 # Vendas efetuadas
5	30 # Clientes no dia
5	278 # Vendas efetuadas

Figura 5.19 – Arquivo de dados vendas4.txt.

O complemento **every** (pode ser usado **ev**) permite definir a amostragem de dados que será desenhada no gráfico. Neste caso informa-se o valor **2** que é o número de linhas do arquivo de dados para cada par de informação. Se o arquivo contivesse os dados agrupados de três em três linhas seria usado o valor **3**.

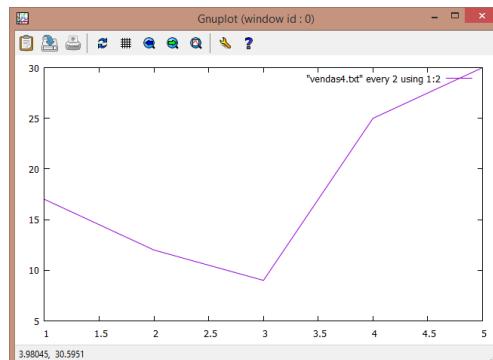
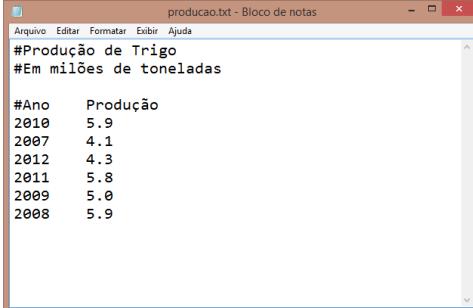


Figura 5.20 – Gráfico do arquivo vendas4.txt.

Nem sempre um arquivo de dados está com os dados dispostos de forma ordenada, o que afeta a apresentação do gráfico. Para demonstrar esta situação considere o próximo arquivo chamado **produção.txt** com os dados de produção de trigo no

Brasil entre 2008 e 2012 (ZAFALON, 2013) como mostra a Figura 5.21. Observe que aos anos estão dispostos de forma desordenada.



```
#Produção de Trigo
#Em milhões de toneladas

#Ano      Produção
2010      5.9
2007      4.1
2012      4.3
2011      5.8
2009      5.0
2008      5.9
```

Figura 5.21 – Arquivo de dados producao.txt.

Execute as instruções a seguir e note na Figura 5.22 a apresentação do gráfico a partir da ordem dos dados existente no arquivo. Note que isto ocasiona a apresentação confusa dos dados e dependendo do tamanho do arquivo não é possível fazer a ordenação manual dos mesmos. É neste momento, que se pode lançar Mao do complemento **smooth**.

```
reset
plot "producao.txt" with lines
```

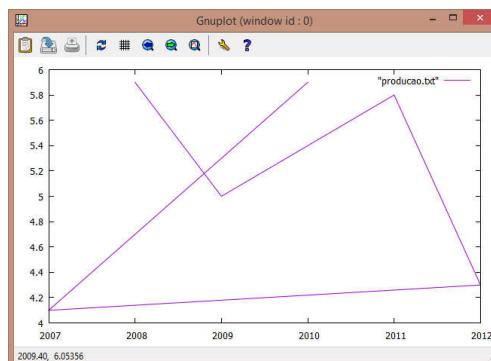


Figura 5.22 – Gráfico do arquivo producao.txt com dados desordenados.

Veja na Figura 5.22 que o eixo das abscissas apresenta os anos em ordem crescente, mas os pontos ficam posicionados na ordem que estão definidos no arquivo de dados. Na sequência execute as instruções a seguir e observe a forma de apresentação no gráfico indicado pela Figura 5.23.

```
reset  
plot "producao.txt" smooth frequency with lines
```

O resultado apresentado na Figura 5.23 é bem mais agradável. O complemento **smooth** foi usado com o argumento **frequency**. O complemento **smooth** possui como argumentos alguns valores, sendo os mais importantes:

- **frequency** – torna cada um dos dados monotônicos do eixo **x** associados a cada um dos dados do eixo **y**, aos quais é somados. Os pontos resultantes são, então, ligados por segmentos de reta.

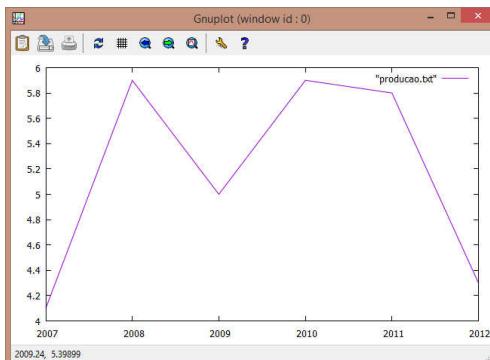


Figura 5.23 – Gráfico do arquivo producao.txt com dados ordenados.

- **unique** – torna cada um dos dados monotônicos do eixo **x** associados a cada um dos dados do eixo **y**, aos quais são a média de **y**. Os pontos resultantes são, então, ligados por segmentos de reta.
- **cumulative** – torna cada um dos dados monotônicos do eixo **x** que possuam o mesmo valor associados a cada um dos dados do eixo **y** por meio de um único ponto, que contém a soma cumulativa dos valores de **y**.
- **cnormal** – torna cada um dos dados monotônicos do eixo **x** normalizados ao valores do eixo **y** no intervalo de valores entre **0** e **1**. Pontos, com o mesmo

valor de **x** são substituídos por um único ponto, contendo a soma cumulativa de valores de **y** dividido pela soma total de todos os valores existentes em **y**.

Todos os exemplos de dados apresentados neste tópico utilizam apenas valores. E se alguma coluna contiver texto, como se faz? Assim sendo, considere o próximo exemplo de arquivo de dados. Grave-o com o nome **produtos.txt**. Observe a Figura 5.24 com os dados do arquivo.

#Produto	Quant.
Computador	2
Impressora	3
"Super Modem"	9
"Mesa padrão"	5
"Cadeira V6"	8

Figura 5.24 – Arquivo de dados **produtos.txt**.

Observe na Figura 5.24 que as palavras compostas estão delimitadas entre aspas inglesas e as palavras simples não. Este é um cuidado que se deve ter com este estilo de arquivo.

Em seguida para apresentar um gráfico, cujo eixo das abscissas apresente os nomes dos produtos basta executar as instruções seguinte e verificar junto a Figura 5.25 esta ocorrência.

```
reset
set grid
set xtics rotate by -65
set rmargin 7
set bmargin 6
plot "produtos.txt" using 0:2:xtic(1) with line
```

A obtenção dos textos para o eixo das abscissas é conseguido com o complemento **using** com o argumento **0:2:xtic(1)**, onde a função **xtic()** seleciona os valores da primeira coluna como etiquetas (**tics**) para o eixo **x**. O valor **0** representa os dados

da primeira coluna, como sendo uma pseudo-coluna, ou seja, representa os dados da primeira coluna como se esses fosse dados numéricos e não uma sequencia de caracteres. O valor **2** é usado para acesso aos dados do eixo das ordenadas.

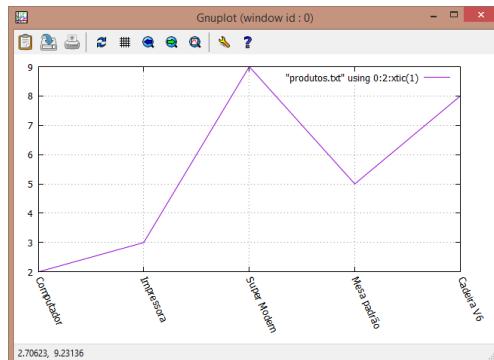


Figura 5.25 – Gráfico com eixo das abscissas com texto.

A forma anterior permite que colunas formadas por sequencia de caracteres e textos sejam apresentadas nos eixos cartesianos. Se fosse usado diretamente a instrução `plot "produtos.txt" with line` ou `plot "produtos.txt" using 1: 2 with line` ter-se-ia a apresentação da mensagem de erro **Bad data on line 3**, informando que os dados da coluna são incompatíveis ou da mensagem de erro **warning: Skipping data file with no valid points x range is invalid** que o ponto para o eixo x é inválido.

5.2 Arquivos de script

Na medida do avanço do estudo deste trabalho pode-se perceber o aumento no número de linhas das instruções executadas no programa **gnuplot**. A quantidade de linhas irá depender do que se deseja colocar em um gráfico podendo chegar a centenas de linhas, senão milhares. Uma forma de tornar o trabalho mais cômodo é criar arquivos que guardem as instruções que podem ser carregados e executados.

Os arquivos de *script* programados, segundo indicado na documentação, devem ter como forma de identificação a extensão **.gp** (*gnuplot program*), caracterizando-se por ser um arquivo de comandos externos com instruções de nível lógico embutidas dentro de um único lote com objetivo de executar ações lógicas. É comum encontrar arquivos de *script* com a extensão **.plt** (*plot*) que possuem apenas os comandos

para a ação de desenhar os gráficos, tais como: **plot**, **splot**, **set**, **unset** e **reset**. No transcorrer das explicações deste e do próximo tópico ficará claro a diferença existente entre esses dois tipos de arquivo.

Neste tópico serão descritas as etapas para a criação de *script* simples, ou seja, *scripts* em arquivos com extensão **.plt**. No próximo tópico serão tratadas as etapas para o uso de arquivos de *scripts* com extensão **.gp**, usado na identificação de *scripts* que executem ações programadas.

A criação dos arquivos de *script* de comandos são feitas com o uso de um editor de textos e gravados no formato texto puro com a extensão **.plt**. Veja a seguir um exemplo de arquivo de *script*.

```
# Script de Demonstração
reset
clear
set mxtics 10
set mytics 5
set size 1,1
set origin 0,0
set multiplot
    set size 0.5,0.5; set origin 0, 0.5
    set title "Gráfico 1"
    set ylabel "Título Y.1"
    set xlabel "Título X.1"
    plot cos(x) title "Legenda 1"
    set size 0.5,0.5; set origin 0, 0
    set title "Gráfico 2"
    set ylabel "Título Y.2"
    set xlabel "Título X.2"
    plot 1/cos(x) title " Legenda 2"
    set size 0.5,0.5; set origin 0.5, 0.5
    set title "Gráfico 3"
    set ylabel "Título Y.3"
```

```
set xlabel "Título X.3"
plot sin(x) title " Legenda 3"
set size 0.5,0.5; set origin 0.5, 0
set title "Gráfico 4"
set ylabel "Título Y.4"
set xlabel "Título X.4"
plot 1/sin(x) title " Legenda 4"
unset multiplot
```

Num editor de texto escreva o código anterior e grave-o com o nome **script01.plt** no seu diretório padrão de trabalho. Note que o conjunto de instruções escritas entre as instruções **set multiplot** e **unset multiplot** foi estabelecido com um deslocamento de dois espaços em branco. Esta estratégia de escrita visa deixar claro no código quais instruções estão dentro de determinado bloco de ações. Isto facilita a leitura de códigos, preferencialmente longos. Esta técnica de escrita chama-se *indentação*.

Atenção especial ao comando **clear** que tem por finalidade efetuar a limpeza da tela de apresentação dos gráficos.

A execução de um arquivo de *script* se faz por meio da instrução:

```
load "script01.plt"
```

Que apresentará imediatamente a tela gráfica com a imagem do gráfico definido no *script*. A chamada de arquivos com o comando **load** pode ser definida entre aspas simples ou entre aspas inglesas.

5.3 Técnicas básicas para programação

A partir dos detalhes já apresentados é hora de tratar o desenvolvimento de *scripts* que efetuam diversas ações com controle operacional e lógico. Neste contexto já foi apresentado alguns detalhes como variáveis, constantes, operadores aritméticos, expressões aritméticas, além de outros detalhes apresentados.

Para o uso da parte de programação mais avançada necessita-se ter conhecimento coadjuvante desta ação e neste sentido serão mostrados alguns detalhes técnicos básicos de programação.

Já fora visto que a entrada de dados no programa **gnuplot** acontece praticamente de maneira indireta, ou por meio de uso de alguma função ou por meio de algum arquivo de dados externo. Pouco é feito por meio de algum valor fornecido como parâmetro para algum comando do programa.

Fica a atenção maior para a efetivação do processamento do programa, que em sua maior parte concentra o processamento matemático. No entanto, é possível efetuar ações de processamento lógico como tomadas de decisão e execução de laços.

Antes de demonstrar o uso dos comandos apresentados neste tópico no contexto de programação serão esses apresentados no contexto interativo de uso no *prompt* do programa **gnuplot**.

5.3.1 Tomada de decisão

Para a efetivação da tomada de decisão tem-se a instrução **if** para uma decisão simples ou **if...else** para decisão composta que poderá ser usada a partir de uma das sintaxes seguintes.

Decisão simples

```
if (<condição>) {  
    <comando1>  
    <comando2>  
    <comandoN>  
}
```

Decisão composta

```
if (<condição>) {  
    <comando1>  
    <comando2>  
    <comandoN>  
} else {  
    <comando1>  
    <comando2>  
    <comandoN>  
}
```

Numa ação de decisão simples o bloco de comandos delimitado entre as chaves somente será executado se a condição que estiver a frente de **if** (se) for verdadeira, caso a condição seja falsa o bloco de comandos é desconsiderado e fluxo de execução lógico do *script* é desviado automaticamente para a primeira instrução após este bloco.

Numa ação de decisão composta o bloco de comandos delimitado entre as chaves somente será executado se a condição que estiver a frente de **if** for verdadeira, caso a condição seja falsa será executado o bloco constante após **else** (senão).

Para a efetivação do processamento de tomada de decisão é necessário o uso de uma condição definida entre parênteses, que do ponto de vista da programação de computadores é a relação lógica entre elementos constituído de uma variável versus outra variável, de uma variável versus uma constante, ou de uma constante versus uma variável.

Para o estabelecimento da condição, ou seja, da relação lógica entre os elementos avaliados é necessário fazer uso de operadores que auxiliam este tipo de operação, chamados *operadores relacionais*, sendo:

Operador	Operação	Ação representativa		
==	Igual a	var == var	var == const	const == var
!=	Diferente de	var != var	var != const	const != var
>	Maior que	var > var	var > const	const > var
<	Menor que	var < var	var < const	const < var
>=	Maior ou igual a	var >= var	var >= const	const >= var
<=	Menor ou igual a	var <= var	var <= const	const <= var

Na tabela anterior na coluna **Ação representativa** é demonstrado os tipos de relações lógicas possíveis de uso entre variáveis (var) e constantes (const).

A partir dessas informações é possível demonstrar de forma interativa as instruções de tomada de decisão.

O conjunto de instruções a seguir demonstram a tomada de decisão simples. O pequeno programa informa se o valor da variável **i** é *par*, caso não seja nada será apresentado. Assim sendo, escreva as seguintes instruções:

```
i=0
i=i+1; if (i%2==0) {print i," par"}
```

Ao fazer a execução das linhas anteriores nada é apresentado, pois a variável **i** é inicializada com valor **0** e na linha seguinte a instrução **i=i+1** efetua a soma de **+1** a variável **i**, que passa a ter o valor **1**. Na sequência a instrução **if** verifica se o resto da divisão do valor de **i** por **2** é igual a **0**, se esta condição for verdadeira é apresentada a mensagem apresentado na tela a mensagem informando que o valor é *par*.

Para verificar essas ocorrências, basta acionar as teclas <Seta para cima> para que a instrução `i=i+1; if (i%2==0) print i, " é par"` seja reescrita e <Enter> para que a instrução seja reexecutada. Ao fazer isso sucessivamente algumas vezes verá a alternância de apresentação ou não da mensagem quando o valor da variável `i` possui um valor par.

O conjunto de instruções a seguir demonstram a tomada de decisão composta. O pequeno programa informa se o valor da variável `i` é *par* ou *ímpar* dependendo do valor da variável em certo momento. Assim sendo, escreva as seguintes instruções:

```
i=0
i=i+1; if (i%2==0) {print i, " par"} else {print i, " ímpar"}
```

Para ver a alternância das mensagens basta como anteriormente acionar as teclas <Seta para cima> e <Enter>.

A condição definida para o comando `if` pode ser formada por uma ou mais relações lógicas. Quando houver a existência de mais de uma relação lógica para uma mesma condição torna-se necessário fazer uso de operadores lógicos de conjunção (`&&`) e disjunção (`||`) que possuem as seguintes estruturas operacionais a partir da análise de duas relações lógicas para a mesma condição, podendo-se operar com mais de duas relações lógicas.

Operador lógico de conjunção (&&)

Condição 1	Condição 2	Resultado
Verdadeira	Verdadeira	Verdadeiro
Verdadeira	Falsa	Falso
Falsa	Verdadeira	Falso
Falsa	Falsa	Falso

Observe que o resultado lógico da condição é verdadeiro quando todas as relações lógicas são verdadeiras. Basta que uma relação lógica possua valor falso para que o resultado da condição não seja verdadeiro.

As instruções a seguir demonstram a tomada de decisão utilizando-se uma condição com duas relações lógicas por meio do operador lógico de conjunção.

O conjunto de instruções propostos faz a criação e inicialização das variáveis `i` e `j`, ambas com valor inicial 1. Na sequência as instruções seguintes acrescentam em `i`

o valor 1 e em j o valor de i+1. Será apresentada a mensagem **verdadeiros** quando o valor de i for par e simultaneamente o valor de j for ímpar. Assim sendo, escreva as seguintes instruções:

```
i=1; j=1
i=i+1; j=j+i; if (i%2==0 && j%2!=0) {print "verdadeiros"}
```

Na primeira execução é apresentada a mensagem **verdadeiros**, pois i possui o valor 2 e j possui o valor 3. Repetindo-se as instruções por três vezes nada é mostrado, somente na quarta execução volta a ser apresentada a mensagem para a condição verdadeira.

A execução sucessiva das instruções `i=i+1; j=j+i; if (i%2==0 && j%2!=0) {print "verdadeiros"}` por cinco vezes faz as seguintes operações na memória sobre as variáveis de acordo com a tabela a seguir.

i=1		j=1		Tabela verdade operador lógico &&		
i=i+1	i	j=j+i	j	Resultado	i%2==0	j%2!=0
i=1+1	2	j=1+2	3	Verdadeiro	Verdadeiro	Verdadeiro
i=2+1	3	j=3+3	6	Falso	Falso	Falso
i=3+1	4	j=6+4	10	Falso	Verdadeiro	Falso
i=4+1	5	j=10+5	15	Falso	Falso	Verdadeiro
i=5+1	6	j=15+6	21	Verdadeiro	Verdadeiro	Verdadeiro

Operador lógico de disjunção (||)

Condição 1	Condição 2	Resultado
Verdadeira	Verdadeira	Verdadeiro
Verdadeira	Falsa	Verdadeiro
Falsa	Verdadeira	Verdadeiro
Falsa	Falsa	Falso

Observe que o resultado lógico da condição é verdadeiro quando pelo menos uma das relações lógicas é verdadeira. O resultado lógico da condição será falso apenas quando todas as relações lógicas forem falsas.

As instruções a seguir demonstram a tomada de decisão utilizando-se uma condição com duas relações lógicas por meio do operador lógico de disjunção.

O conjunto de instruções propostos é semelhante ao anterior tendo como diferencial o uso do operador lógico de disjunção para a verificação da condição. Assim sendo, escreva as seguintes instruções:

```
i=1; j=1
i=i+1; j=j+i; if (i%2==0 || j%2!=0) {print "verdadeiros"}
```

Na primeira execução é apresentada a mensagem **verdadeiros**, pois **i** possui o valor **2** e **j** possui o valor **3**. Repetindo-se as instruções ocorrerá na segunda execução a não apresentação da mensagem, porém nas próximas três execuções a mensagem é apresentada.

A execução sucessiva das instruções **i=i+1; j=j+i; if (i%2==0 || j%2!=0) {print "verdadeiros"}** por cinco vezes faz as seguintes operações na memória sobre as variáveis de acordo com a tabela a seguir.

i=1		j=1		Tabela verdade operador lógico		
i=i+1	i	j=j+i	j	Resultado	i%2==0	j%2!=0
i=1+1	2	j=1+2	3	Verdadeiro	Verdadeiro	Verdadeiro
i=2+1	3	j=3+3	6	Falso	Falso	Falso
i=3+1	4	j=6+4	10	Falso	Verdadeiro	Falso
i=4+1	5	j=10+5	15	Falso	Falso	Verdadeiro
i=5+1	6	j=15+6	21	Verdadeiro	Verdadeiro	Verdadeiro

A verificação de uma condição pode ser negada com o operador lógico de negação (!) que deve para sua ação se posicionado a frente da condição a ser negada.

Operador lógico de negação (!)

Condição	Resultado
Verdadeira	Falsa
Falsa	Verdadeiro

As instruções a seguir demonstram a tomada de decisão utilizando-se uma condição com intervenção do operador lógico de negação.

O conjunto de instruções a seguir apresentará a mensagem **não é par** quando a condição for **não igual**. Observe que não se deseja usar a condição **diferente de** (**!=**),

deseja-se obter o resultado de *diferente de* por meio do uso do operador relacional *igual a*. Assim sendo, escreva as seguintes instruções:

```
i=0  
i=i+1; r= i%2; if (!r==0) {print i, " não é par"}
```

Observe que para esta situação não é possível negar diretamente a expressão **i%2==0**. Foi necessário colocar esta ação por meio da instrução **r=i%2** fora da condição para em seguida por meio de **!r==0** efetivar a negação da condição. O operador de negação opera diretamente sobre a variável a sua frente.

Para ver a alternância na apresentação da mensagem **não é par** basta acionar as teclas <Seta para cima> e <Enter>

5.3.2 Iterações

A ação de iteração se caracteriza por realizar repetições de certas ações. Esse efeito foi usado quando do trabalho dos comandos **plot**, **splot**, **set** e **unset** quando da definição dos intervalos de apresentação dos gráficos.

O programa **gnuplot** opera com dois tipos de iterações, uma de estilo arbitrário e outra no estilo condicional respectivamente com os comandos **for** e **while**, os quais possuem como estrutura sintáticas as forma seguintes.

Iteração arbitrária

```
do|plot for (<iteração>) {  
    <comando1>  
    <comando2>  
    <comandoN>  
}
```

Iteração condicional

```
while (<condição>)  
{  
    <comando1>  
    <comando2>  
    <comandoN>  
}
```

Para o comando **for** a **iteração**, não importando se o prefixo que o antecede é **do** ou **plot**, se refere a definição de um contador para uma variável inteira com valor de início, fim e incremento opcional quando seu valor for maior que 1. A **iteração** pode estar relacionada ao uso de uma variável do tipo **cadeia** onde a lista de palavras passada será iterada sobre a variável indicada. Assim sendo, há duas maneiras básicas de fazer uso de **for**, como segue:

```
for [variável = início:fim:{incremento}] {ação}
for [variável in "Palavra1 Palavra2 Palavra3 etc..."] {ação}
```

A título de conhecimento da instrução **{do|plot}** for serão apresentados alguns exemplos interativos de uso.

O conjunto de instruções a seguir apresenta os valores inteiros de 1 a 10 de 1 em 1 com base no uso do prefixo **do**.

```
do for [i=1:10] {print i}
```

O conjunto de instruções a seguir apresenta os valores inteiros de 1 a 10 de 2 em 2 com base no uso do prefixo **do**.

```
do for [i=1:10:2] {print i}
```

O conjunto de instruções a seguir apresenta os valores inteiros das tangentes de 1 a 10 de 1 em 1 com base no uso do prefixo **do**.

```
do for [i=1:10:1] {print tan(i)}
```

O conjunto de instruções a seguir apresenta separadamente as palavras da lista de palavras indicada com base no uso do prefixo **do**.

```
do for [texto in "Carro Avião Ônibus Barco"] {print texto}
```

O conjunto de instruções a seguir apresenta em um gráfico os valores da tangente de **x** multiplicado por **i** variando de 1 a 7 de 3 em 3 com base no uso do prefixo **plot**. Note que com o uso de **plot** não se faz uso das chaves delimitando a ação.

```
plot for [i=1:7:3] cos(i*x)
```

Para o próximo exemplo devem ser criados dois arquivos de dados com os nomes **arq1.txt** (movimento de compra de produtos) e **arq2.txt** (movimento de venda dos produtos). A Figura 5.26 apresenta o conteúdo dos arquivos.

The figure consists of two side-by-side screenshots of Microsoft Notepad windows. The left window, titled 'arq1.txt - Bloco de notas', contains the following text:
#arq1.txt - Compras
Computador 1100
Impressora 250
Mesa 50

The right window, titled 'arq2.txt - Bloco de notas', contains the following text:
#arq2.txt - Vendas
Computador 1500
Impressora 300
Mesa 80

Figura 5.26 – Arquivos de dados arq1.txt (esquerdo) e arq2.txt (direito).

O conjunto de instruções a seguir apresenta, separadamente, as linhas do gráfico baseado na lista de palavras indicadas com os nomes dos arquivos, advindos de **arq1.txt** e **arq2.txt**, por meio do prefixo **plot**:

```
plot for [n in "arq1 arq2"] n.".txt" using 0:2:xtic(1) with  
lines
```

Escreva toda a instrução em uma única linha. Note que a variável **n** assume os nomes **arq1** e **arq2**, faz a concatenação por meio de **n.".txt"**, estabelece o eixo **x** a partir de uma lista de palavras (cadeias) e traça o gráfico no formato **lines**.

É possível fazer uso de outra maneira de acesso aos arquivos de dados que tenham nomes seriados, como **arq1.txt**, **arq2.txt** e assim por diante. As instruções a seguir apresentam as linhas dos arquivos dados **arq1.txt** e **arq2.txt** a partir da localização do arquivo no computador com base no uso do prefixo **plot**.

```
arquivo(n) = sprintf("arq%d.txt",n)  
plot for [i=1:2] arquivo(i) using 0:2:xtic(1) with lines
```

Note a definição da função **arquivo(n)** atribuída a função **sprintf("arq%d.txt",n)**, onde a indicação **arq%d.txt** representada por **%d** indica o valor a ser substituído da variável **i**. Desta forma, estará se associando o valor **i** com o nome arquivo que contém **arq** e tenha como extensão a terminação **.txt** atribuindo o nome do arquivo a variável **n**. O símbolo **%d** está sendo utilizado como curinga que representa os números **1** e **2** para pegar os arquivos **arq1.txt** e de **arq2.txt**. O símbolo **%d** está sendo usado numa operação de concatenação.

A instrução **plot for [i=1:2] arquivo(i) using 0:2:xtic(1) with lines** apresenta um gráfico a partir da iteração de **1** até **2** abrindo os arquivos representados pela função definida **arquivo(i)**. O gráfico apresentado é idêntico ao mostrado na Figura 5.29.

A função **sprintf()** pode ser usada para gerar os nomes das legendas de um gráfico. Para tanto, execute as seguintes instruções:

```
plot for [n=1:4] -x+n+tan(x) title sprintf("Legenda %d",n)
```

Note que a variável **n** que variará de **1** até **4** é usada na função **sprintf()** para concatenar o valor numérico ao texto **legenda**.

Para o comando **while** é necessário considerar o uso de condição para a ação de sua iteração. Nesta estrutura serão executados as instruções fornecidas enquanto a

condição permanece verdadeira, pois no momento em que a condição torna-se falsa a ação de iteração é finalizada.

Aparentemente **while** e **for** são parecidos, pois ambos realizam, em tese, o mesmo tipo de ação. No entanto, a uma diferença, **for** executa uma iteração de forma determinística (opera com valores inteiros nas iterações) e **while** pode executar uma iteração de forma indeterminista (opera com valores reais).

O conjunto de instruções a seguir apresentam os valores de 1 até 10 de 0.5 em 0.5.

```
i=1  
while (i<=10) {print i; i=i+.5}
```

Note que a variável **I** é iniciada com valor **1**. A iteração **while** verifica se a variável **i** é menor ou igual a **10**, sendo esta condição verdadeira é realizada a execução das instruções dentro do bloco **while**. Observe que após escrever o valor da variável **i**, ela é acrescida de **+1**. Isto fará que em certo momento **i** tenha um valor maior que **10** o que ocasionará seu encerramento de maneira automática.

Agora, usar a instrução **do for [i=0:9:0.5] {print i}** só conseguirá a apresentação de um único valor **0**.

As instruções seguinte apresentam os valores inteiros de 1 a 10 de 2 em 2.

```
i=1  
while (i<=10) {print i; i=i+2}
```

O conjunto de instruções a seguir apresenta em um gráfico os valores da seno de **x** multiplicado por **i** variando de 1 a 7 de 3 em 3.

```
i = 0  
while (i <= 10) {plot cos(x+i); i = i+.1}
```

A instrução **while** não permite fazer uso dos recursos apresentados com **for** para a abertura de arquivos de dados.

5.4 Scripts programados

Neste tópico será utilizado o conceito de *script* programado que difere dos arquivos de *script* por uma questão operacional. Nos arquivos de *script* existem os comandos de apresentação e configuração de gráficos como: **plot**, **splot**, **set**, **unset** e **reset**, sendo esses armazenados em um arquivo com extensão **.plt**. Já os arquivos com *scripts* programados usam os comandos anteriores, acrescidos dos comandos para execução de ações lógicas, como: **if**, **if...else**, **do for**, **plot for** e **while**, sendo esses armazenados em um arquivo com extensão **.gp**.

Para um teste em relação ao uso de *scripts programados* considere o arquivo de programa **prog01.gp** com as instruções indicadas a seguir, que fará a apresentação dos nomes existente em uma pequena lista de palavras. Quando a primeira palavra for capturada da lista o programa indicará que a palavra é a **1a. palavra da lista**. Para as demais palavras nada será mostrado a não ser o nome localizado na lista.

```
# prog01.gp
# Listagem de nome de uma lista

reset
lista = "Carro Ônibus Avião Trem Charrete Bicicleta"
conteudo(n) = word(lista, n)
do for [i = 1:words(lista)] {
    if (i==1) {
        print conteudo(i)." é a ".i."a. palavra da lista"
    } else {
        print conteudo(i)
    }
}
```

O *script programado* com nome **prog01.gp** faz uso de alguns recursos que podem ser úteis, tanto na ação interativa, como na ação programada do programa **gnuplot**.

A instrução **lista = "Carro Ônibus Avião Trem Charrete Bicicleta"** cria a variável **lista**, uma matriz de cadeia de caracteres com palavras separadas por espaço em branco, representando os nomes de alguns meios de transporte.

A linha **conteudo(n) = word(lista, n)** está definido para a matriz **conteudo(n)** um valor inteiro para cada componente da variável **lista** retornado pela função **word()** que ao localizar cada palavra da lista separada por espaço em branco atribui a esta palavra um valor inteiro da **lista**. Assim sendo, **Carro** é associado como conteúdo 1 da variável **conteudo**, **Ônibus** como **conteudo(2)** e assim por diante.

Na sequência a linha de instrução com **do for [i = 1:words(lista)]** faz as iterações da variável **i** 1 até o valor máximo detectado pela função **words()**. A função **words()** retorna a quantidade de palavras separadas por espaço em branco em uma cadeia de caracteres.

O trecho com a instrução **if...else** detecta se a variável **i** está com valor **1** e sendo esta condição verdadeira escreve a primeira palavra e junto desta a mensagem **é a 1a. palavra da lista**. Observe que para a apresentação da mensagem está sendo usada uma concatenação do valor da variável **i** juntamente com a mensagem a ser impressa. Após o comando **else** encontra-se a instrução que apresentará as demais palavras da lista.

O próximo *script* de programa faz uma ação semelhante a anterior utilizando-se o comando **while**. Grave este *script* com o nome **prog02.gp**.

```
# prog02.gp
# Listagem de nome de uma lista

reset
lista = "Carro Ônibus Avião Trem Charrete Bicicleta"
conteudo(n) = word(lista, n)
i=1
while (i <= words(lista)) {
    if (i==1) {
        print conteudo(i)." é a ".i.".a. palavra da lista"
    } else {
        print conteudo(i)
    }
    i=i+1
}
```

Observe no código anterior os detalhes usados. Atente em especial a instrução **i=i+1** responsável por fazer os acréscimos de valores na variável **i** para que seu valor torne-se falso para a condição da iteração em execução.

Um recurso existente é a capacidade de um *script* efetuar a chamada de outro *script*. Assim sendo, considere um *script* de programa chamado **prog03.gp** que fará a chamada do *script* de programa chamado **script02.plt** que tem por finalidade executar a apresentação dos resultados de uma tabuada, onde o valor da tabuada é definido no arquivo **script02.plt** e a parte lógica da operação definida no arquivo **prog03.gp**.

As instruções seguintes se referem ao arquivo **script02.plt**.

```
# script02.plt
# Valor para o calculo da tabuada
n = 5
```

As próximas instruções se referem ao arquivo **prog03.gp**.

```
# prog03.gp
# Calculo da tabuada

load "script02.plt"
do for [i = 1:10] {
    r = n*i
    n1 = sprintf("%g",n)
    i1 = sprintf("%g",i)
    r1 = sprintf("%g",r)
    print n1." x ".i1." = ".r1
}
```

Grave os programas com seus respectivos nome e em seguida execute o comando:

```
load "prog03.gp"
```

Ao ser executado o arquivo de *script* de programa é apresentada a tabuada do valor 5. O valor 5 é estabelecido no arquivo **script02.plt** e a tabuada é executada no arquivo **prog03.gp**

O arquivo **prog03.gp** executa a instrução **load "script02.plt"** que cria na memória a variável **n** com valor **5**. Na iteração **while** é realizado o uso da função **sprintf()** que transforma o conteúdo das variáveis de tipo inteiro, sinalizadas com **%g** em seu tipo equivalente em formato cadeia, pois de outra maneira não é possível apresentar os resultados da tabuada.

5.4.1 Operador ternário

O programa **gnuplot** suporta operadores unários, binários e ternários. Boa parte desses operadores foram já apresentados, por exemplo, usar o sinal de negativo a frente de uma variável ou número mostra o uso de um operador do tipo unário, se estiver sendo feita uma operação aritmética está se utilizando um operador do tipo binário. São vários os operadores dos tipos unários e binários. No entanto, o tipo ternário só há um no programa **gnuplot** que é usado para efetivar tomadas de decisão, a partir da seguinte estrutura sintática:

```
(condição)?[<ação: cond. verdadeira>]:[<ação: cond. falsa>]
```

Numa ação ternária se a condição é verdadeira se se executa a primeira parcela do operador após a condição, depois do símbolo **?** (interrogação). Se for a condição falsa executa-se a segunda parcela após a condição, depois do símbolo **:** (dois pontos). Este operador substitui em algumas ocasiões o uso da instrução **if...else**.

O exemplo a seguir mostra a definição de duas funções chamadas **min(a,b)** e **max(a,b)** que retornam respectivamente o menor e o maior valor entre dois valores fornecidos.

```
min(a,b) = (a<b)?a:b  
print min(1,2) # mostra 1 menor valor em "a"  
print min(9,2) # mostra 2 menor valor em "b"  
print min(5,5) # mostra 1 menor valor em "a"  
  
max(a,b) = (a>b)?a:b  
print max(2,1) # mostra 2 menor valor em "a"  
print max(2,9) # mostra 9 menor valor em "b"
```

O próximo exemplo utiliza o operador ternário para selecionar a coluna do arquivo de dados a ser apresenta. Para tanto, crie um arquivo de dados com o nome

valor.txt que contenha cinco colunas, sendo a primeira coluna os valores do eixo x e as demais colunas os valores do eixo y. A Figura 5.27 mostra o conteúdo do arquivo de dados **valor.txt**.

##	v1	v2	v3	v4
01	10	12	14	16
02	12	14	16	18
03	14	16	18	20
04	16	18	20	22
05	18	20	22	24
06	20	22	24	26

Figura 5.27 – Arquivos de dados **valor.txt**.

O objetivo é traçar um gráfico que apresente no eixo y os valores da coluna 4 ou da coluna 2 mais a coluna 3, desde que os valores da coluna 3 sejam menores que 20. Assim sendo, observe a sequência de instrução seguinte e observe na Figura 5.28 a apresentação do gráfico obtido.

```
plot "valor.txt" using 1:(($3<20?$4:($2+$3)) with lines
```

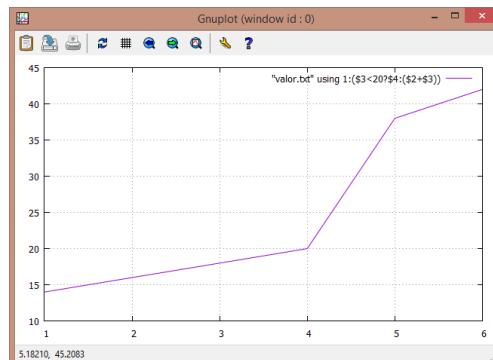


Figura 5.28 – Gráfico obtido a partir do arquivo de dados **valor.txt**.

O uso de um operador ternário pode ser aplicado a outra situação matemática. Considere para tanto, a seguinte função:

$$f(x) = \begin{cases} 3, & \text{para } x \geq -6 \text{ e } x \leq 6 \\ 1 + x^3, & \text{para } x < -6 \text{ e } x > 6 \end{cases}$$

A função anterior será $f(x) = 3$ somente quando x estiver entre os valores de -6 e 6, fora do intervalo definido a função será $f(x) = 1+x^3$. Para apresentar o gráfico da função no programa **gnuplot** basta definir a seguinte sequência de instruções e observar a imagem da Figura 5.29.

```
f1(x) = 3
f2(x) = 1 + x ** 3
f(x) = (x >= -6 && x <= 6) ? f1(x) : f2(x)
plot f(x)
```

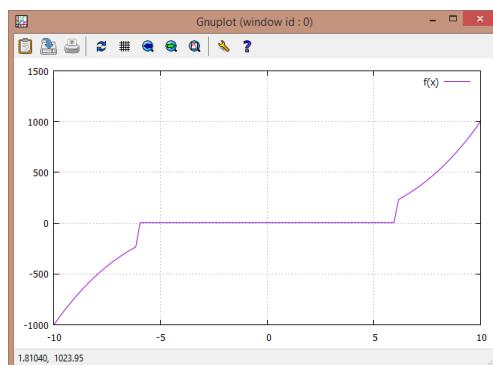


Figura 5.29 – Gráfico obtido a partir da função $f(x)$ com operador ternário.

5.4.2 Operação de somatório

Para auxiliar algumas operações interativas ou programadas o programa **gnuplot** possui um comando de ação iterativa que efetua o somatório da faixa de valores fornecido. Trata-se do comando **sum** que possui a seguinte sintaxe:

```
sum [variável = <início>:<fim>] <expressão>
```

Neste comando **variável** opera apenas com valores inteiros definidos entre **início** e **fim**. A indicação **expressão** se refere a indicação da operação pretendida com o comando. Assim sendo, execute a instrução seguinte.

```
print sum [s=0:100] s
```

É mostrado como resultado o valor **5050.0** que é o somatório dos números de 1 até **100**, onde a variável **s** dentro de **[]** faz a contagem cumulativa e a variável **s** fora de **[]** é a expressão com o total da soma.

O próximo exemplo apresenta um gráfico em formato **boxes** a partir do arquivo de dados **valor.txt**, com a soma dos valores de todas as colunas do eixo **y**, ou seja, as colunas de **2 a 4**, como pode ser constatado na Figura 5.30.

```
plot "valor.txt" using 1:(sum [c=2:4] column(c)) with boxes
```

O efeito obtido é o mesmo que se obteria usado a instrução **plot "valor.txt" using 1:(\$2+\$3+\$4) with boxes**.

Há de se tomar um cuidado no uso da definição das colunas. Já é sabido que, por exemplo, **\$2** e **column(2)** representam o número da coluna e podem ser usados um ou outro. No entanto, isto é válido quando se usa **\$2** e se deseja usar **column(2)**, mas pode não ser válido o oposto.

Por exemplo, é válido executar:

```
plot "valor.txt" using 1:(column(2)+ column(3)+ column(4))  
with boxes
```

Mas não será válido, o uso das forma reduzida junto ao comando **sum**:

```
plot "valor.txt" using 1:(sum [c=2:4] $(c)) with boxes
```

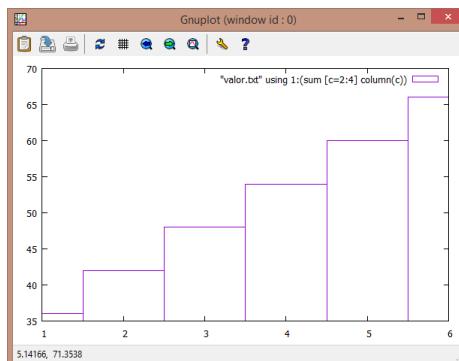


Figura 5.30 – Gráfico com a soma das colunas do arquivo de dados **valor.txt**.

A título de experimentação de uso do comando **sum** em arquivos de *script* em modo programado há no manual do programa **gnuplot** um exemplo pertinente ao uso do comando **sum** (adaptado de WILLIAMS & KELLEY, 2011).

```
# prog04.gp
# Função de Fourier

reset
set grid
set multiplot layout 2, 2
fourier(k, x) = sin(3./2*k)/k*2./3*cos(k*x)
do for [potencia = 0:3] {
    TERMOS = 10**potencia
    set title sprintf("Termo %g de Fourier na série.", TERMOS)
    plot 0.5 + sum [k=1: TERMOS] fourier(k,x) notitle
}
unset multiplot
```

Grave o arquivo de *script* de programas com o nome **prog04.gp** e execute-o com **load "prog04.gp"**. Note que neste exemplo está sendo usada a variável **TERMOS** grafada com caracteres maiúsculos, sendo possível o uso de tais caracteres apenas na definição de variáveis, os comandos do programa são sempre escritos com caracteres minúsculos.

Um detalhe neste *script* é o uso da instrução **set multiplot** para determinar que mais de um gráfico seja apresentado, neste caso quatro. Observe o uso de **layout 2, 2** como argumento de **set multiplot** que divide a tela em dois quadrantes horizontais e dentro destes em mais dois quadrantes verticais, sendo esta outra maneira de apresentar mais de um gráfico na tela gráfica, como foi demonstrado no final do capítulo2.

A parte mais importante do programa é a definição da função de **Fourier** por meio da instrução **fourier(k, x) = sin(3./2*k)/k*2./3*cos(k*x)**.

Dentro da iteração **do for** encontra-se definido o comando **sum** que efetua a soma dos termos operados sobre a função **Fourier()** que são somados a **0.5** para a apresentação do gráfico em execução. O complemento **notitle** é usado para inibir a apresentação do título da legenda dentro do gráfico.

5.5 Novas funções definidas pelo usuário

Na medida em que se usa o programa **gnuplot** pode ocorrer a necessidade de se ir aumentando a coleção de funções definidas pelo próprio usuário. Esta temática foi apresentada no estudo do capítulo anterior, já servindo de base para o estudo deste capítulo.

Assim sendo cabe acrescentar, a título de exemplo, mais três funções ao arquivo de inicialização do programa. Assim sendo acrescente no arquivo existente as funções a seguir:

```
fourier(k,x) = sin(3./2*k)/k*2./3*cos(k*x)
gauss(x,m,s) = exp(-0.5*((x-m)/s)**2 )/sqrt(2*pi*s**2)
binom(n,k) = n!/(k!*(n-k)!)
```

Abra o arquivo de configuração, acrescente essas funções após a existente, salve o arquivo, feche o programa **gnuplot** e carregue-o novamente para que as mudanças sejam aceitas.

Desta maneira o programa **gnuplot** vai sendo personalizado de modo a atender eventualmente necessidades particulares.

5.6 Salvar o ambiente

O programa **gnuplot** possui um recurso que permite salvar o ambiente de trabalho. Pode-se salvar funções definidas pelo usuário, variáveis definidas e todas as opções definidas no ambiente. Para esta ação há o comando **save** que possui a sintaxe:

```
save {<opções>} "nome do arquivo"
```

onde **<opções>** pode ser a definição das palavras chave **function**, **variable**, **set** ou **terminal**. Se nenhuma opção for usada serão gravadas as funções, variáveis, definição de opções do comando **set** e as últimas ações lote de comando **splot**.

Os arquivos salvos com este comando são gravados em formato texto puro e podem ser usados com o comando **load** para serem carregados na memória do programa **gnuplot**.

Veja algumas possibilidades:

```
save "ambiente1.txt"
```

Salva no arquivo **ambiente1.txt** toda a configuração padrão do ambiente se incluído os comandos **set**, o padrão do terminal em uso, as funções definidas pelo usuário e as variáveis de ambiente.

```
save functions "ambiente2.txt"
```

Salva no arquivo **ambiente2.txt** todas as funções definidas pelo usuário.

```
save var "ambiente3.txt"
```

Salva no arquivo **ambiente3.txt** todas as variáveis ativas na memória.

```
save set "ambiente4.txt"
```

Salva no arquivo **ambiente4.txt** todas as configurações do comando **set/unset**.

```
save terminal "ambiente5.txt"
```

Salva no arquivo **ambiente5.txt** a configuração ativa do terminal em uso. Pode-se substituir **terminal** por **term**.

5.7 Integração com linguagens de programação

O programa **gnuplot** além de poder ser usado de modo programado e interativo por meio de seu *prompt* pode também ser usado em modo *backend*, sendo executado a partir da chamada de outro programa. Isto é útil para profissionais e usuários que querem fazer uso dos recursos do programa **gnuplot** a partir de outros programas que tenham sido desenvolvidos com esta finalidade.

Toda linguagem de programação de computadores existente, seja ela qual for, tem a capacidade de criar arquivos, principalmente do tipo texto puro (ASCII). Além de arquivos textos a grande maioria dessas linguagens opera também arquivo binários, sendo que este tipo de arquivo não é usado com o programa **gnuplot**.

Desta forma, pode-se criar nessas linguagens arquivos de dados, arquivos de *script* com instruções **gnuplot**, tanto para o modo comando (**.plt**) quanto para o modo programado (**.gp**).

É importante considerar que para o programa **gnuplot** ser chamado de dentro dos programas criados em linguagens de programação é necessário ter a variável de ambiente **PATH** do sistema operacional configurada para acesso ao diretório **bin** do programa do diretório **\gnuplot**. As exemplificações a seguir consideram que esta variável esteja configurada no ambiente de trabalho do leitor.

Antes de abordar esta temática é interessante aprender a funcionalidade do comando **pause** do programa **gnuplot** que tem por finalidade estabelecer uma pausa no ponto onde o comando é colocado. O comando **pause** opera com teclado e *mouse*. Neste trabalho o foco será apenas para controle do teclado, a partir da sintaxe:

```
pause <tempo> {"<cadeia>"}
```

Onde **tempo** indica uma constante que pode ser um dos valores: **-1** para fazer uma pausa e aguardar até que a tecla **<Enter>** é acionada, **0** para não definir nenhuma pausa e um valor maior que zero para definir o tempo em segundo de espera.

O parâmetro **cadeia** permite a colocação opcional de uma mensagem que será apresentada quando o comando **pause** for executado.

Desta forma, é possível fazer uso das seguintes combinações para o teclado:

```
pause -1 # Aguarda <Enter> ser acionado  
pause 2 # Aguarda 2 segundos  
pause -1 "Tecle <Enter> para continuar."  
pause 5 "Aguarde 5 segundos."
```

Ao executar cada uma da linhas poderá perceber os detalhes de cada uma das variações permitidas para uso.

A demonstração de acesso entre linguagens de programação e o programa **gnuplot** serão executadas a partir de uso das linguagens **C** e **Lua**, será então necessário criar um arquivo de *script* de comando que desenhe um gráfico a partir de uma das funções que se encontra definida dentro do arquivo de inicialização. Assim sendo, grave o próximo conjunto de instruções com o nome **script03.plt**.

```
# script03.plt  
  
reset
```

```
set grid
plot gauss(x,-2,2)
pause -1 "Tecle <Enter> para continuar."
```

Outro exemplo com uso do comando **pause** é o próximo arquivo de programa que simula uma animação num gráfico com a apresentação de uma linha ondulante. Para tanto, crie o arquivo de programa **prog05.pg** a seguir e execute-o com a instrução **load "prog05.gp"**

```
# prog05.gp
# Simulação de animação

reset
set grid
set xrange[0:3]
set yrange[-3:3]
i=0
while ( i <= 20) {
    plot besy0(i+x)
    pause 0.2
    i=i+0.2
}
```

A chamada do programa **gnuplot** a partir das linguagens **C** e **Lua** pode ser feita a partir dos seguintes códigos, cujos nomes respectivamente são **gplt.c** e **gplt.lua**:

Linguagem C

```
// C chama gnuplot

#include <stdlib.h>
int main(void)
{
    system("gnuplot script03.plt");
    return 0;
}
```

Linguagem Lua

```
-- Lua chama gnuplot  
  
os.execute("gnuplot script03.plt");
```

O próximo passo é criar um programa nas linguagens **C** e **Lua** que criem um arquivo texto contendo comandos do programa **gnuplot** e façam e apresentem o resultado existente no arquivo texto criado. O programa em linguagem **C** chama-se **cria.c** e em linguagem **Lua** chama-se **cria.lua**.

Linguagem C

```
// Criação de arquivo texto  
  
#include <stdio.h>  
#include <stdlib.h>  
  
int main(void)  
{  
    FILE *ARQ;  
    ARQ = fopen("teste.plt", "w");  
    fprintf(ARQ,"reset\n");  
    fprintf(ARQ,"set grid\n");  
    fprintf(ARQ,"plot gauss(x,-2,2) title \"Gauss em C\"\n");  
    fprintf(ARQ,"pause-1\n");  
    fclose(ARQ);  
    system("gnuplot teste.plt");  
    return 0;  
}
```

Linguagem Lua

```
-- Criacao de arquivo texto  
  
ARQ = io.open("teste.plt","w")  
ARQ:write("reset\n")
```

```
ARQ:write("set grid\n")
ARQ:write("plot gauss(x,-2,2) title \"Gauss em Lua\"\n")
ARQ:write("pause -1\n")
ARQ:close()
os.execute("gnuplot teste.plt")
```

Os programas deste tópico demonstram o uso do programa **gnuplot** integrado com linguagens de programação, de uma maneira simples, mas que dá ideia de como ampliar este trabalho.

Não é objetivo deste trabalho ensinar programação de computadores ou linguagens de programação. O objetivo é mostrar as potencialidades de trabalho do **gnuplot**.

6

Outros Recursos

Este capítulo mostra alguns detalhes complementares ao conteúdo dos demais capítulos. Neste capítulo são indicados o uso de recursos, como: **set**, **unset**, **reset**, **show**, **set angles**, **fit**, **set format**, **set functions**, **set isosamples**, **show functions**, **show variables**, **style function**, **set terminal**, **set output**, **set zero**, **set xdata**, **set ydata**, **set zdata**, **set x2data**, **set y2data**, **set xdtics**, **set ydtics**, **set zdtics**, **set x2dtics**, **set y2dtics**, **set xlabel**, **set ylabel**, **set zlabel**, **set x2label**, **set y2label**, **set xmtics**, **set ymtics**, **set zmtics**, **set x2mtics**, **set y2mtics**, **set xrange**, **set yrange**, **set zrange**, **set x2range**, **set y2range**, **set xtics**, **set ytics**, **set ztics**, **set x2tics**, **set y2tics**, **set xrange**, **set yrange**, **set zrange**, **set x2range**, **set y2range**, **set xtics**, **set ytics**, **set ztics**, **set x2tics**, **set y2tics**, **set xzeroaxis**, **set yzeroaxis**, **set zzeroaxis**, **set x2zeroaxis**, **set y2zeroaxis**, **set parametric**, **set dummy**, **set polar**, **set style**, **set contour**, **set logscale**, entre outros.

6.1 Comandos Set / Unset / Reset / Show

Assim como os comandos **plot** e **splot**, o comando **set** é muito importante, pois permite definir uma série de configurações para os ajustes do ambiente de operação do programa. Enquanto **set** determina certa configuração, **unset** faz o inverso, desabilitando uma configuração. Já o comando **reset** desabilita, de uma única vez, todas as configurações feitas com **set**, retornando o ambiente do programa ao seu estado padrão, exceto por alguns poucos complementos de **set**.

O comando **set** para ser usado necessita da informação de um complemento que por sua vez pode necessitar de algum argumento de configuração.

O comando **show**, já apresentado com **show version** e **show version long** tem por finalidade mostrar o estado das configurações internas do ambiente. Este comando

para ser usado necessita da informação de complementos, que são os mesmos usados com o comando **set**.

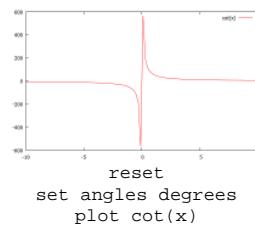
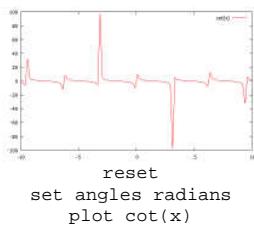
A melhor maneira de conhecer todos os complementos do comando **set** é fazer um estudo das informações apresentadas no **help** do programa.

São agora apresentados mais alguns complementos, em ordem alfabética. Os recursos apresentados são indicados na forma mais simples possível. Lembre-se de que o **help** do programa é a maior fonte de consulta e que alguns dos complementos foram utilizados nos capítulos anteriores.

angles

Complemento usado para alterar o modo de operação do programa que pode ser **degrees** (grau) ou **radians** (radianos). O modo padrão do programa é **radians**. Os exemplos a seguir indicam respectivamente como ativar o modo graus e o modo radianos.

```
set angles degrees
set angles radians
```



format

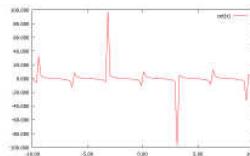
Complemento usado para formatar os rótulos dos dados nos eixos cartesianos. Os exemplos a seguir indicam respectivamente as definições de formatação para os dados do eixo **x** com valor real com duas casas decimais e o eixo **y** com valor real com três casas decimais.

```
set format x "% .2f"
set ytics format "% .3f"
```

```

    reset
    set format x "%2f"
    set ytics format "%3f"
    plot cot(x)

```



functions

Complemento usado para apresentar a informação de data e hora junto à imagem do gráfico. Por padrão a data e a hora são apresentadas no canto inferior direito, mas outra posição pode ser definida.

O exemplo a seguir indica a data e a hora no lado inferior direito escrito com fonte **Times New Roman** com o ano com quatro dígitos.

```
set timestamp "%d/%m/%Y %H:%M" font "Times New Roman"
```

O exemplo a seguir indica a hora e a data com ano com dois dígitos no lado superior esquerdo com fonte padrão do programa.

```
set timestamp "%H:%M %d/%m/%y" top
```

O exemplo a seguir indica a data e a hora no lado superior direito (uso de **offset** que opera o deslocamento na horizontal e vertical) com o ano com quatro dígitos na fonte padrão do programa.

```
set timestamp "%d/%m/%Y %H:%M" top offset 80, -2
```

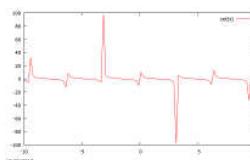
O complemento **offset** opera na posição horizontal com a definição do primeiro valor e opera na posição vertical com a definição do segundo valor (após a vírgula).

O exemplo a seguir indica apenas a data com o ano com quatro dígitos na fonte padrão do programa.

```
set timestamp "%d/%m/%Y"
```

```

    reset
    set timestamp "%d/%m/%Y"
    plot cot(x)
```

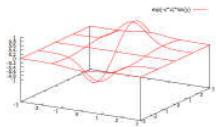


isosamples

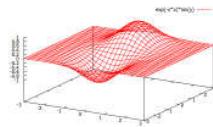
Complemento usado para determinar as isolinhas usadas na definição da superfície do desenho de um gráfico. Isolinha é a curva parametrizada por um argumento que é alterado enquanto outro argumento permanece fixo. Quanto maior o valor dos argumentos, mais precisa será a taxa de amostragem na curva desenhada em um gráfico, porém o tempo para processamento do desenho do gráfico será maior. O valor padrão é 10 (**set isosamples 10, 10**).

Os exemplos a seguir mostram o desenho de um gráfico tridimensional a partir da função $\exp(-x^2)\sin(y)$, com **isosamples 5** e **isosamples 30**.

```
set isosamples 5
set isosamples 30
```



```
reset
set xrange [-3:3]
set yrange [-3:3]
set isosamples 5
splot exp(-x*x)*sin(y)
```



```
reset
set xrange [-3:3]
set yrange [-3:3]
set isosamples 30
splot exp(-x*x)*sin(y)
```

show functions

Instrução usada para mostrar a lista de todas as funções definidas pelo usuário que estão ativas na memória, tanto as definidas em tempo de execução do programa com as definidas no arquivo de inicialização. O uso do comando **reset** não cancela uma função definida.

```
show functions
```

show variables

Instrução usada para mostrar a lista das variáveis definidas na memória incluindo-se constantes definidas para o arquivo de inicialização. É possível fazer uso desta instrução de algumas formas.

O exemplo a seguir mostra todas as variáveis padrão do sistema e as variáveis definidas pelo usuário, como é o caso das constantes armazenadas no arquivo de inicialização.

```
show variables
```

O exemplo a seguir mostra todo o conteúdo da instrução anterior e acrescenta todas as variáveis iniciadas com **GPVAL_**.

```
show variables all
```

O próximo exemplo mostra todas as variáveis iniciadas por **c_** (variáveis definidas como constantes no arquivo de inicialização).

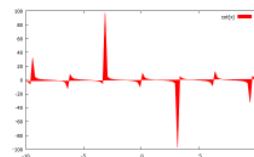
```
show variables c_
```

style function

Complemento usado para alterar o modo padrão de desenho dos gráficos. O modo padrão é **lines**. Pode-se escolher outra forma como: **boxerrorbars**, **boxes**, **boxplot**, **boxxyerrorbars**, **candlesticks**, **circles**, **dots**, **ellipses**, **filledcurves**, **fillsteps**, **financebars**, **fsteps**, **histeps**, **histograms**, **image**, **impulses**, **labels**, **lines**, **linepoints**, **points**, **polar**, **rgbalpha**, **rgbimage**, **steps**, **vectors**, **xerrorbars**, **xerrorlines**, **xyerrorbars**, **xyerrorlines**, **yerrorbars** e **yerrorlines**.

```
set style function dots  
set style function filledcurves
```

```
reset  
set style function filledcurves  
plot cot(x)
```



terminal / output

O complemento **terminal** é usado para definir o dispositivo de saída que deve ser considerado pelo programa **gnuplot**. O dispositivo padrão é **wxt** (janela de terminal multiplataforma wxWidgets). Para ver o terminal ativo execute **show terminal** e para ver os terminais disponíveis execute **set terminal**. A sintaxe padrão é:

```
set terminal {<tipo terminal>} push | pop}
```

Se o argumento <tipo terminal> é omitido é apresentada uma lista dos terminais disponíveis. O argumento **push** é usado para salvar na memória a configuração atual do terminal em uso e **pop** é usado para restaurar o terminal atual. Isto é aplicável quando se deseja trabalhar com mais de um terminal mantendo-se as configurações definidas para certo terminal inalteradas, mesmo usando outro terminal.

O complemento **output** é usado para determinar qual será a saída para o **gnuplot** com a syntaxe:

```
set output {<nome do dispositivo>}
```

O nome do dispositivo deve ser colocado entre aspas inglesas. Se o nome do dispositivo for omitido, qualquer arquivo de saída aberta por uma chamada anterior será fechado e uma nova saída possivelmente será enviada para **STDOUT**. Esteja com a atenção voltada para esta questão, pois alguns terminais ignoram a saída definida.

Caso venha a utilizar os comandos anteriores, use primeiro **set terminal** para então usar **set output**. O uso deste recurso encontra-se demonstrado no capítulo 3. Veja a seguir um exemplo deste uso, a partir da inicialização do programa **gnuplot**.

```
# mostra configuração terminal atual
show terminal

# salva configuração do terminal atual
set terminal push

# nova configuração do terminal
set terminal postscript portrait color

# mostra configuração do terminal modificado
show terminal

# recupera configuração antes da mudança anterior
set terminal pop
```

```
# mostra configuração inicial do terminal  
show terminal
```

zero

Complemento usado para definir o padrão do valor zero no ambiente. Internamente, o programa opera com um valor próximo de zero, sendo 1⁻⁸ (**show zero**). Para torná-lo zero, quando necessário, usa-se:

```
set zero 0
```

É possível definir outros valores como sendo o **set zero** do programa.

6.2 Outros eixos: X2 e Y2

Os gráficos desenvolvidos até este momento fizeram uso dos eixos X e Y quando bidimensionais que ficam respectivamente posicionados na parte inferior e lado esquerdo do plano cartesiano e dos eixos X, Y e Z quando tridimensionais que ficam respectivamente posicionados na parte inferior esquerda, inferior direita e lado extremo esquerdo do plano cartesiano. Em particular, nos gráficos bidimensionais além dos eixos X e Y há os eixos X2 e Y2 que ficam respectivamente na parte superior e lado direito do gráfico que são obtidos a partir das instruções dos complementos indicados nas colunas **Eixo X2** e **Eixo X3**. As demais colunas são apresentadas a título de ilustração, uma vez que os complementos a serem usados são compatíveis para cada coordenada pretendida a partir do uso do comando **set**.

Eixo X	Eixo Y	Eixo Z	Eixo X2	Eixo Y2
xdata	ydata	zdata	x2data	y2data
xdtics	ydtics	zdtics	x2dtics	y2dtics
xlabel	ylabel	zlabel	x2label	y2label
xmtics	ymtics	zmtics	x2mtics	y2mtics
xrange	yrange	zrange	x2range	y2range
xtics	ytics	ztics	x2tics	y2tics
xzeroaxis	yzeroaxis	zzeroaxis	x2zeroaxis	y2zeroaxis

As instruções **set xdata**, **set ydata**, **set zdata**, **set x2data** e **set y2data**, cada qual em sua posição, permitem formatar o eixo correspondente com o formato de data e hora, por meio da definição de formação, que utiliza a instrução **set timefmt** e opera de acordo com a tabela de códigos de formatação a seguir.

Formato	Descrição
%d	Apresenta dia do mês de 1 a 31
%m	Apresentação do mês de 1 a 12
%Y	Apresenta ano de 0 até 99 na faixa 1969-2068
%Y	Apresenta o ano com 4 dígitos
%j	Apresentação do dia do ano de 1 a 366
%H	Apresentação da hora com dois dígitos de 00 a 23
%M	Apresentação do minuto de 0 até 59
%s	Apresenta número de segundos desde o início do ano de 2000
%S	Apresentação dos segundos de 0 a 60
%b	Apresentação do nomes do mês abreviado
%B	Apresentação do nome do mês por extenso

As instruções **set xdtics**, **set ydtics**, **set zdtics**, **set x2dtics** e **set y2dtics** têm por objetivo converter os *ticsl* (rótulos) do eixo indicado em marcas relacionadas ao dia da semana de 0=Sun (domingo) até 6=Sat (sábado), com sobreposição dos valores que excederem sete dias. Para voltar os rótulos a forma padrão basta em contrapartida fazer uso das instruções: **set noxdtics**, **set noydtics**, **set nozdtics**, **set nox2dtics** e **set noy2dtics**.

As instruções **set xlabel**, **set ylabel**, **set zlabel**, **set x2label** e **set y2label** efetuam a colocação de legendas nos eixos a que se referem.

As instruções **set xmtics**, **set ymtics**, **set zmtics**, **set x2mtics** e **set y2mtics** têm por objetivo converter os *ticsl* (rótulos) do eixo indicado em marcas relacionadas ao dia do mês de um ano 1=Jan (janeiro) até 12=Dec (dezembro), com sobreposição dos valores que excederem doze meses. Para voltar os rótulos a forma padrão use antes do complemento o comando **unset**.

As instruções **set xrange**, **set yrange**, **set zrange**, **set x2range** e **set y2range** efetuam a definição dos limites do plano cartesiano para a apresentação do gráfico. O argumento a ser fornecido deve estar no formato **[mínimo, máximo]**.

As instruções **set xtics**, **set ytics**, **set ztics**, **set x2tics** e **set y2tics** são utilizadas para definir rótulos para o eixo a ser operado. O cancelamento deste recurso é efetuado com o comando **unset**.

As instruções **set xzeroaxis**, **set yzeroaxis**, **set zzzeroaxis**, **set x2zeroaxis** e **set y2zeroaxis** efetuam a apresentação de uma linha mais forte demarcando o eixo oposto.

O gráfico a seguir apresenta a segunda coordenada em uso. A configuração de alguns complementos da segunda coordenada como **y2tics** e **x2tics** afetam os complementos irmãos da primeira coordenada e vice versa, quando estão em uso. Isto necessita um pouco de atenção. A Figura 6.1 mostra esta ocorrência, observe os detalhes apresentados do lado direito e superior do gráfico.

```
reset
set mytics 10
set title "Exemplos de eixos X/Y e X2/Y2"
set key left top

# Configuração da primeira coordenada
set xlabel "Abscissa X"
set ylabel "Ordenada Y"
set ytics 2
set xtics 1

# Configuração da segunda coordenada
set x2range [-5:5]
set y2range [-5:5]
set x2label "Abscissa X2"
set y2label "Ordenada Y2"
set y2tics 2
set x2tics 1
set y2dtics
set x2mtics
set xzeroaxis
```

```
set yzeroaxis
plot cos(x), sin(x) linetype 3
```

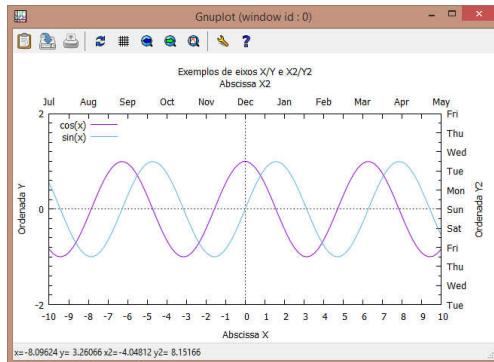


Figura 6.1 – Gráfico com definição da segunda coordenada: X2 e Y2.

Grave a sequência de instruções anteriores em um arquivo de *script* com o nome **script04.plt**, pois desta forma ficará mais fácil proceder algumas mudanças que serão apresentadas.

Os complementos de manipulação dos *labels* indicados neste tópico: **xlabel ylabel**, **zlabel**, **x2label** e **y2label** possuem outras possibilidades de configuração para uso, podendo-se destacar entre elas as apresentadas a seguir

```
set nlabel { "<rótulo>" }
  {offset <deslocamento>}
  {font "<fonte>[, <tamanho>]"}
  {textcolor lt <cor>}
```

Onde **n** de **nlabel** representa uma das possibilidades de uso **x**, **y**, **z**, **x2** e **y2**.

O argumento **rótulo** caracteriza-se pela definição da cadeia de caracteres, do texto a ser apresentado.

O argumento **offset** trata do deslocamento do texto do rótulo na grade do plano cartesiano referente a coordenada em foco. Este argumento aceita duas

informações, sendo uma para a coordenada **x** e outra para a coordenada **y** para posicionar o conteúdo na grade.

O argumento **font** permite indicar o nome da fonte de letras a ser usado e seu tamanho.

O argumento **textcolor lt** permite alterar a cor do rótulo apresentado de acordo com a tabela de cores do programa como se encontra definido no capítulo3.

Observe no conjunto de instruções do **script04.plt** a definição de algumas opções do complemento *label*. Implemente as mudanças e execute o *script* para ver os detalhes no gráfico.

```
reset
set mytics 10
set title "Exemplos de eixos X/Y e X2/Y2"
set key left top

# Configuração da primeira coordenada
set xlabel "Abscissa X"
set ylabel "Ordenada Y" offset 8, 0
set ytics 2
set xtics 1

# Configuração da segunda coordenada
set x2range [-5:5]
set y2range [-5:5]
set x2label "Abscissa X2" font "Helvetica, 12"
set y2label "Ordenada Y2" textcolor lt 3
set y2tics 2
set x2tics 1
set y2dtics
set x2mtics
set xzeroaxis
set yzeroaxis
```

```
plot cos(x), sin(x) linetype 3
```

As opções dos complementos *labels* podem ser usadas de forma separada ou em conjunto; quando juntas, sugere-se manter a ordem da estrutura da instrução apresentada, sendo **offset-font-textcolor**, **offset-font**, **offset-textcolor** ou **font-textcolor**.

6.3 Gráfico de equação paramétrica

O desenvolvimento de gráficos a partir de equações paramétricas é suportado pelo programa **gnuplot**.

Sabe-se que para fazer uso de gráficos a partir de equações paramétricas de uma curva é necessário levar em consideração que certo ponto (**P**) do plano cartesiano, representado por **P(x,y)** é definido genericamente de maneira minimalista a partir das funções **x = f(t)** e **y = g(t)**, onde **x** e **y** são coordenadas a serem consideradas e obtidas a partir das funções **f(t)** e **g(t)** respectivamente, que possuem em comum a variável independente **t**.

Para criar gráficos a partir de funções paramétricas usa-se no programa **gnuplot** a instrução:

```
set parametric
```

que ativa o padrão paramétrico. Quando a instrução **set parametric** é executada é apresentada a mensagem **dummy variable is t for curves, u/v for surfaces** informando que a variável independente **t** está ativa para traçar as curvas de equações paramétricas e as variáveis **u** e **v** estão ativas para traçar gráficos de superfície.

A partir da execução da instrução **set parametric** já é possível fazer uso de gráficos desenhados a partir de equações paramétricas.

O exemplo a seguir, apresenta um gráfico criado a partir de equações paramétricas. Como indica a Figura 6.2.

```
reset  
set parametric  
plot sin(3*t), cos(5*t)
```

Observe após a instrução **set parametric** a definição das equações paramétricas **sin(3*t)** e **cos(5*t)**. Note o uso da variável **t** sendo comum as duas equações. Se indicada apenas uma das equações nada será apresentado.

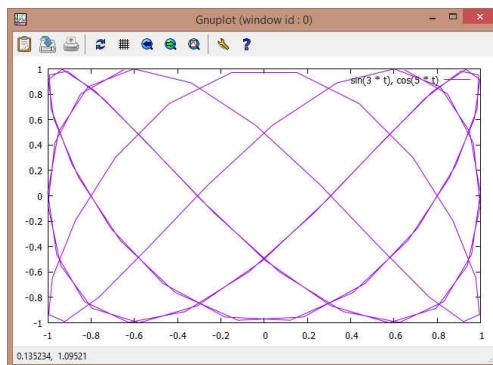


Figura 6.2 – Gráfico a partir das equações paramétricas $\sin(3*t)$ e $\cos(5*t)$.

O próximo exemplo mostra um gráfico tridimensional com a imagem de uma hélice. Para este caso é necessário dizer ao programa **gnuplot** que o gráfico tridimensional a ser usado trabalhará num nível paramétrico maior.

O uso da instrução **set parametric** ativa na memória a variável independente **t** que somente opera com duas referências. Se tentar colocar na instrução **plot** uma ou mais de duas referências de ação o programa apresenta mensagens de erro. Para apresentar a hélice é necessário ativar outra variável independente, diferente da variável **t** e para isso torna-se necessário fazer uso da instrução **set dummy** após o uso da instrução **set parametric**.

A instrução **set dummy** tem por objetivo mudar as variáveis independentes padrão do programa. No modo não paramétrico as variáveis independentes são **x**, **y** e **z** de acordo com uso dos comandos **plot** e **splot**. No modo paramétrico as variáveis independentes são **t**, **v** e **u**.

A instrução **set dummy** é operada a partir da seguinte sintaxe:

```
set dummy {<var. independente>} {,<var. independente>}
```

que pode ser usada com uma das seguintes configurações.

```
set dummy a  
set dummy a, b  
set dummy , b
```

A primeira forma troca a variável independente **x** por **a**, a segunda forma trocas as variáveis independentes **x** e **y** por **a** e **b** e a terceira forma troca apenas a variável **y** por **b** mantendo-se a variável **x** inalterada. Por exemplo, a partir da execução da segunda forma não é mais possível usar as instruções **plot x** ou **plot x+y**, deve em seu lugar ser usada as instruções **plot a** ou **plot a+b**. Para voltar ao estado normal basta fazer uso do comando **reset** ou isoladamente use **unset dummy** e **unset parametric**.

Em relação ao uso de equações paramétricas a variável independente **t** é destinada a gráficos bidimensionais. Para gráficos tridimensionais é necessário definir a variável que será independente. Assim sendo, execute as seguintes instruções e veja o resultado obtido junto a Figura 6.3.

```
reset  
set parametric  
set dummy w  
set urange [-10:10]  
splot cos(w), sin(w), w
```

Atente no programa para o uso da instrução **set urange** usada para definir o conjunto paramétrico das faixas de coordenadas usadas para calcular **x**, **y** e **z**. Este recurso se assemelha a instrução **set xrange**.

Os dois próximos exemplos mostra o que ocorre com a apresentação de um gráfico criado a partir de equações paramétricas. Assim sendo, considere o conjunto de instruções a seguir e o gráfico apresentado junto a Figura 6.4.

```
reset  
set urange [-10:10]  
plot sin(x*1.5)*cos(x/2.5), x/2
```

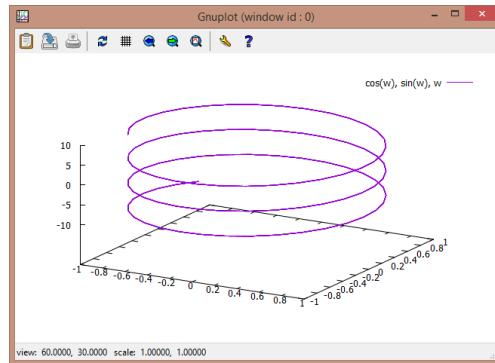


Figura 6.3 – Gráfico de uma hélice.

Note no exemplo anterior que o uso da variável x nas funções da instrução **plot** fez com que fossem criadas duas curvas no gráfico, uma do seno de x multiplicado por 1.5, isto multiplicado pelo cosseno de x dividido por 2.5, mais o gráfico de x dividido por dois.

Ao se definir o ambiente para modo paramétrico e substituir a variável x por t ter-se-á uma relação paramétrica onde as duas antes operacionalizadas em separado passam a ser operadas em conjunto criando outro efeito gráfico.

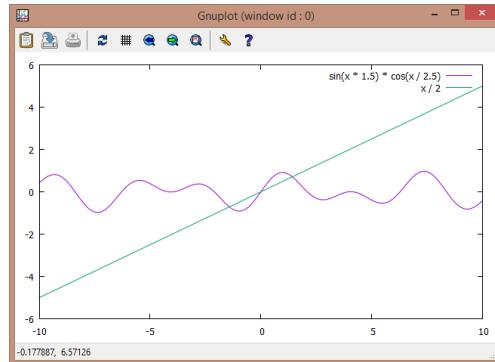


Figura 6.4 – Gráfico sem uso de função paramétrica.

Execute as instruções a seguir e observe junto a Figura 6.5 a apresentação do gráfico a partir de uma definição de ambiente paramétrico.

```
reset
set parametric; set urange [-10:10]
plot sin(t*1.5)*cos(t/2.5), t/2
```

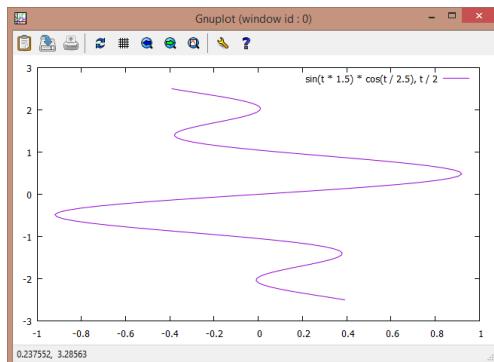


Figura 6.5 – Gráfico com uso de função paramétrica.

6.4 Gráfico polar

O plano cartesiano, baseado nas coordenadas x e y é muito conhecido e bastante utilizado, mas existem outras formas de se representar dados em gráficos e uma delas são as coordenadas polares nos suas coordenadas ao representadas por r (raio) e θ (ângulo). Os gráficos polares são bons para representar dados cílicos, aqueles que ocorrem de acordo com certa periodicidade ou sazonalidade.

Observe junto a Figura 6.6 o gráfico criado a partir da execução da instrução:

```
reset
plot 1 - sin(x)
```

O gráfico apresentado a partir da função $f(x) = 1 - \sin(x)$ é gerado com base em coordenadas cartesianas. Para utilizar, coordenadas polares é preciso fazer uso da instrução **set polar** para ativar este modo de trabalho e **unset polar** para desativar este modo de trabalho.

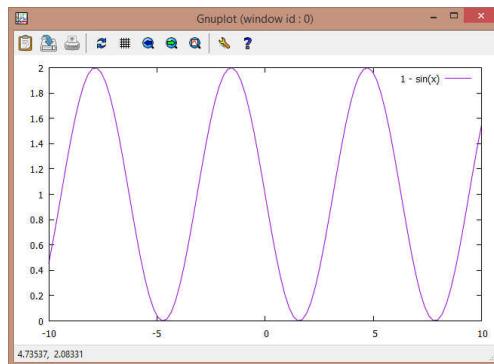


Figura 6.6 – Gráfico da função $f(x) = 1 - \sin(x)$.

Execute as instruções seguintes e observe na Figura 6.7 a apresentação de um gráfico polar. Compare as Figuras 6.6 e 6.7 observando a diferença apresentada em relação ao modo cartesiano e ao modo polar.

```
reset  
set polar  
plot 1-sin(t)
```

O gráfico apresentado na Figura 6.7 caracteriza-se por apresentar as coordenadas polares em um retângulo.

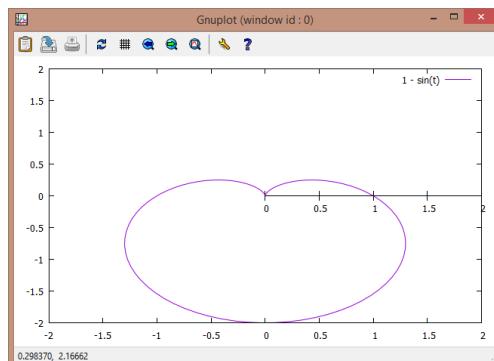


Figura 6.7 – Gráfico da função $f(t) = 1 - \sin(t)$.

O próximo exemplo apresenta um gráfico polar do tipo borboleta. Para tanto, execute as instruções a seguir e veja na Figura 6.8 o resultado apresentado.

```
reset
set polar
plot exp(cos(t))-2*cos(4*t)+sin(t/12)**5
```

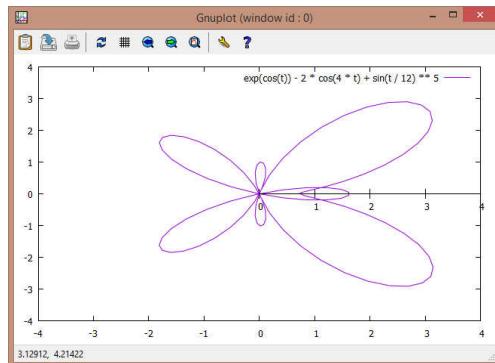


Figura 6.8 – Gráfico da função $f(t) = \exp(\cos(t)) - 2\cos(4t) + \sin(t/12)^5$.

O próximo exemplo mostra outro gráfico no estilo borboleta obtido a partir do modo polar do programa **gnuplot**. Informe as instruções a seguir e observe na Figura 6.9 os detalhes do gráfico.

```
reset
set polar
set trange [-2*pi:2*pi]
plot t*sin(t)
```

Atente no programa para o uso da instrução **set trange** usada para definir o conjunto paramétrico ou polar das faixas de coordenadas usadas para calcular **x** e **y**. Este recurso se assemelha a instrução **set xrange**.

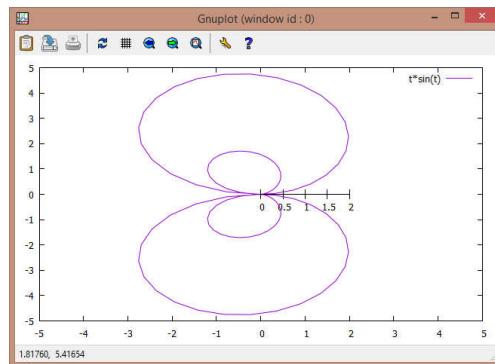


Figura 6.9 – Gráfico da função $f(t) = \exp(\cos(t)) - 2\cos(4t) + \sin(t/12)^5$.

6.5 Gráfico histograma

Histograma é um tipo de gráfico usado para demonstrar uma distribuição de frequência, sendo este o modo mais comum para a apresentação de dados. A criação de um histograma no programa **gnuplot** não é feito a partir de funções matemáticas.

Para a confecção deste tipo de gráfico necessita-se de um arquivo de dados com uma distribuição de dados um pouco diferente do que é apresentado no capítulo 5. Assim sendo, considere a Figura 6.10 com o arquivo **anual.txt** contendo os dados a serem utilizados.

A screenshot of a Windows Notepad window titled "anual.txt - Bloco de notas". The window contains a table of monthly data from January to June for the years 2011, 2012, and 2013. The data is as follows:

Figura 6.10 – Arquivo **anual.txt**.

Note que diferentemente dos outros arquivos de dados já utilizados este possuí a primeira linha com a identificação dos nomes das colunas e as demais linhas com os dados de cada coluna.

As instruções a seguir devem ser gravadas em um arquivo de *script* com o nome **scrip05.plt**, que deverá ser executado por meio da instrução **load "scrip05.plt"**. A Figura 6.11 apresenta o resultado obtido.

```
# scrip05.plt - Gerador de Histograma

reset
set grid
set yr [0:140]
set title "Movimento Anual"
set style data histograms
set style histogram clustered gap 1
set boxwidth 1
set style fill solid 1.0 border lt -1
set bmargin 3.5
set key inside below
plot "anual.txt" using 2:xtic(1) ti col, \
    "" using 3 ti col, \
    "" using 4 ti col
```

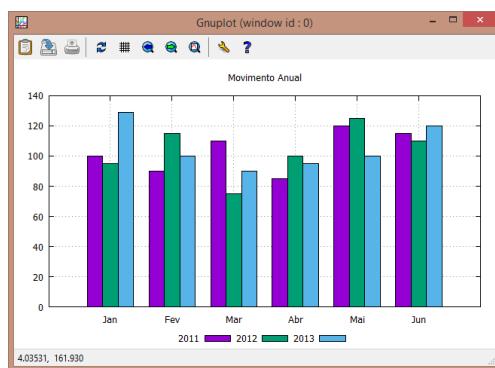


Figura 6.11 – Histograma a partir do arquivo de dados *anual.txt*.

Os passos para a elaboração de um histograma são mais longos do que os passos habituais para gráficos baseados em funções, pois é necessário cumprir alguns passos.

A instrução **set style data histograms** é usada para definir o estilo dos dados em uso para o formato de um gráfico histograma.

A instrução **set style histogram clustered gap 1** define o estilo padrão usado para a apresentação dos dados. Cada conjunto de dados na coluna do arquivo de dados é capturado como um grupo (**clustered**) para o eixo **x** e **gap** determina a distância entre as colunas do mesmo grupo de dados.

Com a instrução **set boxwidth 1** é definida a largura da coluna e a instrução **set style fill solid 1.0 border lt -1** define a margem que contorna cada coluna. O trecho **set style fill** determina o estilo de preenchimento da coluna do gráfico, neste caso como preenchimento **solid** (sólido). A opção **solid** preenche a coluna com uma cor que determina a intensidade do preenchimento. Este valor pode variar de **0.0** até **1.0**, onde **0.0** é sem cor e **1.0** é a intensidade máxima da cor.

A instrução **set key inside below** orienta a apresentação da legenda na parte inferior do gráfico fora das barras do gráfico.

Na instrução **plot**, o complemento **using** indica apenas o uso das colunas **2, 3 e 4**. A coluna **1** é usada para determinar os rótulos do eixo **X**. O trecho **2:xtic(1)** utiliza, em primeiro lugar, a coluna **2**, com os rótulos (**xtic**) da coluna **1**, oriunda do arquivo de dados **anual.txt**, no eixo **X**. Nas demais colunas, antes de **using**, define-se um caractere nulo, pois, de outra forma, gera-se erro. Também se utiliza **ti col**, que é o mesmo que **title column**. O complemento **using** pode ser simplificado apenas com a letra **u**. No caso de uso de **ti col**, solicita-se que a primeira linha da coluna seja utilizada como título para a legenda.

6.6 Gráfico com rótulo

É possível apresentar dentro da grade do gráfico rótulos que fornecem informações do próprio gráfico.

O exemplo seguinte mostra um gráfico de linhas do arquivo de dados **clientes.txt**, contendo a informação dos valores cartesianos que geram o gráfico. Para tanto, escreva a instrução seguinte toda em uma única linha e note o efeito ocasionado junto a Figura 6.12.

```
plot "clientes.txt" using 1:2 with lines, "" using
1:2:(sprintf("[%g,%g]",$1,$2)) with labels
```

Note o uso da função **sprintf()** que formata e faz a apresentação dos dados das colunas 1 (**\$1**) e 2 (**\$2**) como rótulos (**with labels**). Observe que este trecho é separado do primeiro trecho por uma vírgula e uma definição nula entre aspas inglesas, pois não a nada sendo usado por **using** a não ser a sobreposição dos valores como rótulos nos pontos já desenhados no gráfico.

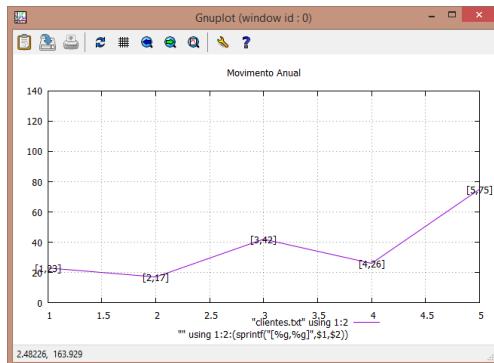


Figura 6.12 – Rótulos nos pontos cartesianos.

Outra possibilidade com este recurso é criar rótulos inteligentes que apresentem mensagens informando se o valor é alto ou baixo a partir da definição de um operador ternário de decisão. Execute a instrução a seguir em uma única linha e acompanhe o gráfico apresentado na Figura 6.13.

```
plot "clientes.txt" using 1:2 with lines, "" using
1:2:((($2>40.0)?"Alto":"Baixo") with labels
```

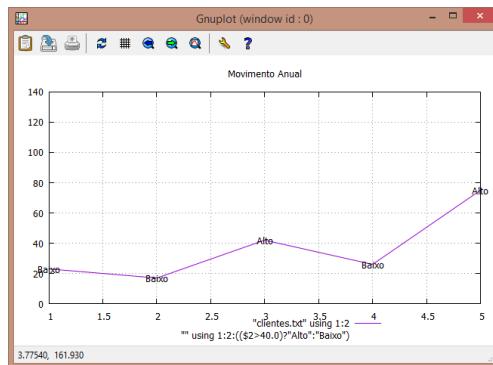


Figura 6.13 – Rótulos inteligentes.

Note que os valores acima de **40** são indicados com o rótulo **Alto** e abaixo de **40** são indicados com o rótulo **Baixo**.

6.7 Ajuste de curvas

O programa **gnuplot** fornece a seu usuário a capacidade de efetuar o ajuste de curvas para funções definidas pelo seu usuário pelo método do *ajuste de curvas dos mínimos quadrados*, sendo conseguido com o uso do comando **fit**.

Esse método de ajuste é usado quando há um conjunto de dados que precisa de uma distribuição de pontos equilibrada, de forma a expressar melhor a curva do gráfico.

Para demonstrar esse recurso, considera-se a função linear $y = a + bx$, em que **a** e **b** são variáveis independentes a serem determinadas. Aqui deve ser minimizada a soma das distâncias de cada ponto (x_i, y_i) em relação a cada ponto $(x_i, a+bx_i)$. O problema nessa questão é que a diferença entre os pontos pode ser positiva ou negativa, o que pode ocasionar uma soma nula das diferenças, mesmo com valores distantes na reta. Os pontos apresentados no gráfico podem ficar dispersos demais.

O comando **fit**, para executar sua ação, utiliza o algoritmo de Levenberg-Marquardt para gerar os parâmetros usados na iteração seguinte.

O comando **fit** para realizar sua ação faz uso do algoritmo de Levenberg-Marquardt para gerar os parâmetros usados na próxima iteração.

A partir da definição da função $y = a + bx$ que é o modo mais simples de representar o relacionamento entre as variáveis x e y tem-se a equação que determina a reta entre os pontos a e b . A partir desta assertiva será determinada a reta dos mínimos quadrados que melhor se ajusta a tabela seguinte definida no arquivo de dados **dadosregr.txt**, como mostra a Figura 6.14.

#X	Y
1.5	2.2
2.4	4.2
3.7	7.8
5.8	3.1
7.3	8.8
8.3	7.2
8.7	4.2
9.1	3.4
9.9	7.8

Figura 6.14 – Arquivo **dadosregr.txt**.

O próximo passo é definir para o programa a função a ser usada para determinar a curva de ajuste a partir dos dados registrados no arquivo de dados **dadosregr.txt** a fim de fornecer para o comando **fit** a função a ser usada para o processamento dos cálculos de ajuste da reta. Assim sendo, entre a instrução:

```
f(x) = a + b * x
```

Em seguida execute a instrução:

```
fit f(x) "dadosregr.txt" using 1:2 via a, b
```

Após a execução do comando **fit** é apresentado um relatório com as iterações efetuadas até encontrar a convergência dos pontos da reta **a** e **b**. Observe os dados do relatório a seguir:

```
iter      chisq      delta/lim   lambda   a           b
 0 1.1344000000e+02  0.00e+00  4.96e+00  1.000000e+00  1.000000e+00
 1 5.2389412482e+01 -1.17e+05  4.96e-01  1.112914e+00  6.332392e-01
 2 4.2547102694e+01 -2.31e+04  4.96e-02  3.260037e+00  3.321199e-01
 3 4.2365284806e+01 -4.29e+02  4.96e-03  3.597342e+00  2.878852e-01
 4 4.2365284359e+01 -1.06e-03  4.96e-04  3.597872e+00  2.878157e-01
iter      chisq      delta/lim   lambda   a           b
After 4 iterations the fit converged.
final sum of squares of residuals : 42.3653
rel. change during last iteration : -1.05668e-08
```

```
degrees of freedom      (FIT_NDF) : 7
rms of residuals       (FIT_STDFIT) = sqrt(WSSR/ndf)   : 2.46012
variance of residuals (reduced chisquare) = WSSR/ndf   : 6.05218

Final set of parameters           Asymptotic Standard Error
==================================  =====
a          = 3.59787            +/- 1.949        (54.18%)
b          = 0.287816           +/- 0.2807       (97.52%)

correlation matrix of the fit parameters:
      a      b
a    1.000
b   -0.907  1.000
```

Veja que o relatório informa que após cinco iterações (de 0 até 4) encontrou a convergência dos ponto **a** e **b** respectivamente com os valores **3.59787** e **0.287816** que são os pontos que definem a reta de ajuste das curvas dos mínimos quadrados.

Para ver apenas o gráfico com a dispersão dos pontos execute a instrução a seguir e acompanhe a Figura 6.15.

```
reset
plot "dadosregr.txt" linecolor rgb "magenta" notitle
```

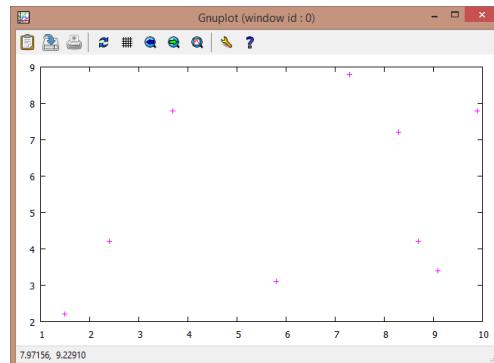


Figura 6.15 – Gráfico com pontos dispersos.

Para ver a reta com o ajuste da curva o gráfico com a dispersão dos pontos execute as instruções seguintes e veja na Figura 6.16 o resultado da operação.

```

reset
set xrange [0:10]
set yrange [0:10]
set xtics 0, 1, 10
set ytics 0, 1, 10
plot f(x) with line notitle
replot "dadosregr.txt" linecolor rgb "magenta" notitle

```

Na sequência anterior de instruções o uso de **set xtics 0, 1, 10** e **set ytics 0, 1, 10** definem que o gráfico apresentado terá suas coordenadas estabelecidas de 0 a 10 com intervalo de 1 em 1.

O comando **plot** é usado para apresentar a reta da função **f(x)** que será indicada automaticamente na cor vermelha e o comando **replot** faz a sobreposição do gráficos dos dados do arquivo de dados na cor magenta.

Para tornar o visual mais limpo, ambos os gráficos usam o complemento **notitle** que impede a apresentação do título da legenda.

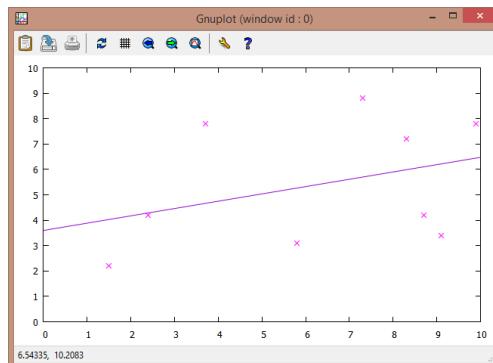


Figura 6.16 – Gráfico com reta com ajuste dos mínimos quadrados.

O comando **fit** para efetivar os ajustes nas curvas dos mínimos quadrados faz uso de diversas variáveis que são iniciadas pelo prefixo “**FIT_**”. Entre as variáveis existentes pode-se destacar algumas que podem ser redefinidas para ação do comando, sendo:

- **FIT_LIMIT** – destinada a alterar o limite do valor *epsilon* padrão (1e-5) para convergência das iterações. Quando a soma de quadrados residuais ocorre entre duas etapas de iteração e o resultado obtido for menor que o fator do valor *epsilon*, o ajuste é considerado "convergente". O valor padrão desta variável está expresso no relatório apresentado como **limit for stopping : 1e-005**.
- **FIT_MAXITER** – destinada a especificar o limite de iterações executadas sem convergência por **FIT_LIMIT**. Se o valor for **0** significa que não há limite.

A definição dessas variáveis são realizadas apenas uma vez e podem ser definidas junto do arquivo de inicialização para que sejam carregadas automaticamente quando da execução do programa.

O exemplo a seguir especifica para a variável **FIT_LIMIT** o valor **1e-8** como limite de convergência. Em seguida execute as instruções a seguir.

```
reset
f(x) = a + b * x
FIT_LIMIT = 1e-8
fit f(x) "dadosregr.txt" using 1:2 via a, b
set xrange [0:10]
set yrange [0:10]
set xtics 0, 1, 10
set ytics 0, 1, 10
plot f(x) with line notitle
replot "dadosregr.txt" linecolor rgb "magenta" notitle
```

Observe após esta nova execução que o relatório apresentado indica a informação **limit for stopping : 1e-008** e não mais **limit for stopping : 1e-005**, como ocorreu anteriormente. Note que o uso da variável **FIT_LIMIT** permitiu esta mudança na execução do comando **fit**.

Outro ponto de destaque a quantidade de iterações efetuadas que passou para seis (definidas de **0** até **5**). Isto faz com que o cálculo da convergência para o ajuste da curva seja mais preciso, se assim for necessário.

6.8 Gráfico senoidal de superfície

O gráfico a seguir (em forma clássica) é desenvolvido em etapas para que outros recursos do programa **gnuplot** sejam apresentados. O gráfico em questão é produzido pela equação $\sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$, em que o seno da raiz quadrada de x ao quadrado mais y ao quadrado são divididos pela raiz e, então, soma-se x ao quadrado e y ao quadrado. Essa equação permite criar um gráfico tridimensional.

As instruções seguintes produzem a imagem indicada na Figura 6.17.

```
reset
splot sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```

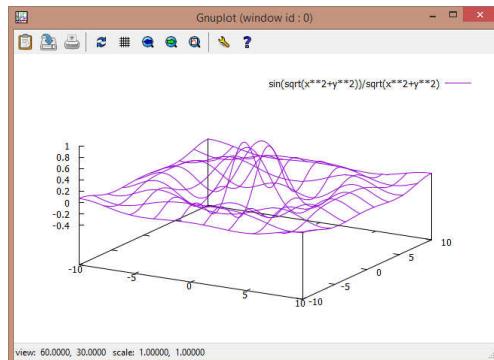


Figura 6.17 – Gráfico tridimensional de superfície.

O próximo passo é fazer o ajuste da quantidade de isolinhas do gráfico. Assim, use as seguintes instruções e acompanhe na Figura 6.18 a ocorrência desses ajustes.

```
reset
set isosamples 30
splot sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```

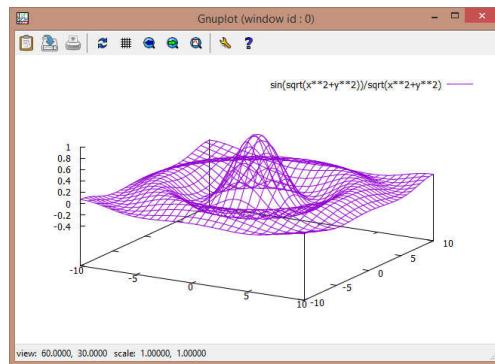


Figura 6.18 – Gráfico com ajuste das isolinhas.

Os próximos passos é esconder a linhas sobrepostas do gráfico com a instrução **set hidden3d** e definir a textura de cores para a superfície com a instrução **set pm3d**. Execute as instruções a seguir e veja o resultado na Figura 6.19.

```
reset
set isosamples 30
set hidden3d
set pm3d
splot sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```

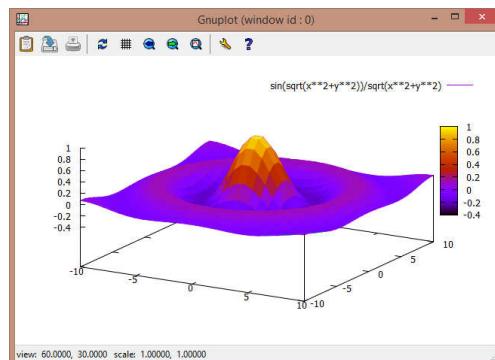


Figura 6.19 – Gráfico com definição acentuada da superfície.

Até a execução desses passos não há muita novidade, mas há ainda a possibilidade em um gráfico 3D em solicitar a apresentação do contorno da superfície de base com o uso da instrução **set contour** que insere no gráfico contornos de superfície na parte inferior do plano (opção **base**), na parte da superfície (opção **surface**) ou em ambas as posições (opção **both**). As instruções seguintes mostram o efeito aplicado na base e na superfície, sem uso do efeito **pm3d** como indica a Figura 6.20 (JONES, 2006).

```
reset
set isosamples 30
set hidden3d
set contour both
splot sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```

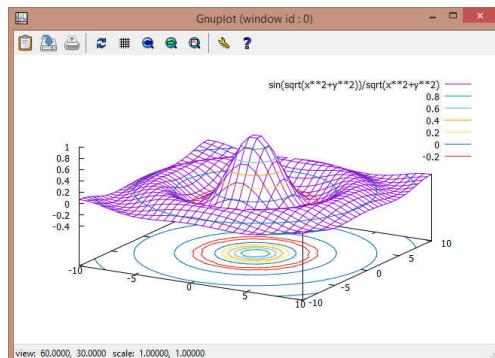


Figura 6.20 – Gráfico com contornos da superfície.

A instrução **set contour both** proporciona a apresentação de linhas de contorno que auxiliam a leitura das faixas de valores do gráfico. Perceba que, para cada linha de contorno, há um valor indicado na legenda do gráfico.

6.7 Gráfico em escala logarítmica

O criação de gráficos pode ser realizada a partir de diversas fontes de dados. Boa parte dos gráficos utilizados até este momento foram gráficos lineares baseados em

funções matemáticas ou arquivos de dados. No entanto, as grandezas usadas para representar um gráfico podem se tornar problemáticas.

O uso de uma grandeza logarítmica visa diminuir a representação de uma escala de modo que torne esta escala mais fácil de ser compreendida.

Para a demonstração de uso de gráfico logarítmico considere um investimento de R\$ 100,00 por 10 meses a uma taxa mensal de 1% de juros. Isto proporcionará um retorno de investimento na casa de R\$ 110,46, ou seja, R10,46 de rendimento sobre o valor investido. Na sequência prepare o arquivo de dados **invest.txt** como mostra a Figura 6.21.

#	Tempo	Valor
0	0	100
1	1	101
2	2	102.01
3	3	103.0301
4	4	104.060491
5	5	105.191005
6	6	106.1520151
7	7	107.2135352
8	8	108.2856796
9	9	109.3685273
10	10	110.4622125

Figura 6.21 – Arquivo de dados *invest.txt*.

Observe na Figura 6.21 a coluna da esquerda, que será a coordenada X do gráfico com valores lineares e a coluna da direita, que será a coordenada Y com valores indicados numa escala logarítmica obtida a partir da fórmula de matemática financeira para cálculo de valor futuro $VF = VP \cdot (1+i)^n$, onde: VP é o valor presente (R\$ 100,00); i é a taxa de juros (1%); n é o valor do tempo de investimento (10) e VF é o valor futuro, neste caso um valor aproximado de R\$ 110,46. Note que o valor do montante cresce no tempo exponencialmente para a formação do valor futuro.

A partir do arquivo de dados **invest.txt** execute as instruções seguintes e veja na Figura 6.22 o resultado do gráfico de evolução financeira.

```
reset  
plot "invest.txt" with lines
```

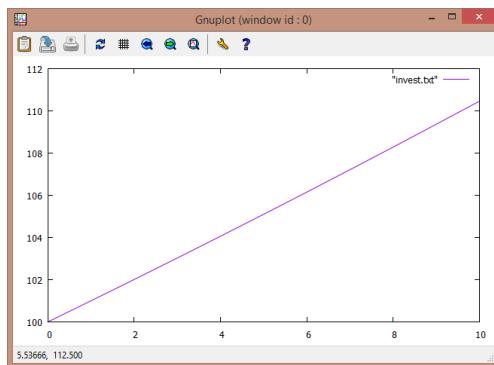


Figura 6.22 – Gráfico gerado a partir do arquivo de dados *invest.txt*.

Note que o gráfico mostra em relação ao tempo (eixo das abscissas) a evolução do investimento (eixo das ordenadas). Até aí tudo, mas este gráfico dá a impressão de que o crescimento do investimento ao longo do período é linear quando na verdade o crescimento do investimento é exponencial em uma escala logarítmica.

As instruções a seguir estabelecem os ajustes para a apresentação de um gráfico em escala logarítmica. Observe a Figura 6.23 com este detalhe.

```
reset
set logscale x
plot "invest.txt" with lines
```

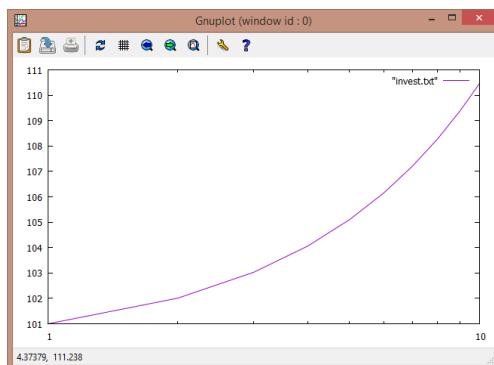


Figura 6.23 – Gráfico com ajuste de escala logarítmica.

Observe que a Figura 6.23 apresenta uma curva de crescimento não linear, o que dá uma visão mais apropriada em relação ao crescimento financeiro ao longo do tempo.

Para definir o modo gráfico em escala logarítmica, utilizou-se a instrução **set logscale**, que permite mudar a forma de apresentação da escala do eixo da coordenada indicada. Pode-se operar sobre os eixos X, Y, X2, Y2, Z, entre outras possibilidades, como pode ser atentado no modo ajuda do programa. É possível escolher a base de escalonamento da escala logarítmica com **set logscale x2**, em que o eixo X é definido com escala logarítmica de base 2.

* * * Anotações * * *



Bibliografia

EUCLIDES. **Os Elementos**. São Paulo: Editora UNESP, 2009.

JANERT, P. K. **Gnuplot in Action: Understanding Data with Graphs**. Greenwich (CT): Manning Publications Co., 2010.

JONES, M. T. **Data visualization tools for Linux: A quick look at six open source graphics utilities**. IBM developerWorks 2006. Disponível em: <<http://www.ibm.com/developerworks/linux/library/l-dataviztools/?ca=dhw-739>>. Acesso em 27 nov. 2018

SILVA, R. de C. C. **Sistema Tridimensional de Coordenadas Cartesianas**, 2013. Disponível em: <<http://www.ritaccs.pro.br/documentos/3-sistematridimensionaldecoordenadas72503.pdf>>. Acesso em: 11 nov. 2012.

WILLIAMS, T. & KELLEY, K. **gnuplot 4.6: An Interactive Plotting Program**. 2012. Disponível em: <http://www.gnuplot.info/docs_4.6/gnuplot.pdf>. Acesso em 15 nov. 2018.

ZAFALON, M. **País tem safra recorde, mas produção de trigo é a menor em cinco anos**. Folha de São Paulo, São Paulo, 10 jan. 2013. Colunistas. Disponível em: <<http://www1.folha.uol.com.br/colunas/vaivem/1212615-pais-tem-safra-recorde-mas-producao-de-trigo-e-a-menor-em-cinco-anos.shtml>>. Acesso em 15 nov. 2018.

