Introdução ao LaTeX

NÍCKOLAS DE AGUIAR ALVES
http://fma.if.usp.br/~nickolas
Versão de 13 de novembro de 2020

RESUMO: Estas notas de aula foram desenvolvidas para o minicurso "Introdução ao LaTeX", ofertado nos dias 10 e 12 de novembro de 2020 como parte da XV Semana Temática de Oceanografia. No curso, pretendeu-se apresentar a linguagem LaTeX para a escrita de documentos a partir de seus princípios básicos e mostrar o processo de escrita de um artigo científico em LaTeX, contando com exemplos de uso de pacotes e práticas úteis para profissionais das áreas das Ciências Exatas. Não foram assumidos conhecimentos prévios e o único requisito técnico é um computador com acesso a internet.

Sumário

1 Introdução

> O Coelho Branco pôs seus óculos. "Onde devo começar, por favor, Vossa Majestade?" ele perguntou.
>
> "Comece pelo começo," o Rei disse, muito gravemente, "e prossiga até chegar ao fim: então pare."
>
> Lewis Carroll[6].

Editores de texto "convencionais", como os disponíveis em pacotes de softwares de escritório, são fundamentados na filosofia *WYSIWYG* — *What You See Is What You Get*. Isto significa que ao escrever um documento usando o Microsoft Word, por exemplo, vê-se o resultado final em tempo real e a edição é feita por cima do documento resultante.

Esta filosofia pode soar cômoda à primeira vista, mas possui uma série de pontos negativos. Por completeza, podemos mencionar alguns exemplos

i. adicionar uma imagem ao documento por vezes pode modificar a diagramação do texto como um todo;

ii. a inserção de fórmulas matemáticas é frequentemente pouco prática;

iii. a diagramação do documento é posta inteiramente nas mãos do autor, que frequentemente não possui conhecimentos sobre como diagramar um documento apropriadamente.

O LaTeX é uma linguagem de marcação, semelhante ao HTML*, que permite a escrita de documentos sob a filosofia *WYSIWYM — What You See Is What You Mean*. Ao invés de escrever o documento observando sua aparência final, observa-se sua estrutura. Enquanto num editor *WYSIWYG* o autor trabalha com a noção de "Este é o título de um capítulo, então o coloco em negrito", num editor *WYSIWYM* o autor tem uma postura semelhante a "Este é o título de um capítulo, então eu escrevo `\chapter{Título}`". O LaTeX terá a tarefa de tomar o significado das palavras do autor e atribuir a aparência correta.

Muito mais pode ser dito sobre a filosofia e a história do LaTeX e do TeX, mas não nos ateremos a estes detalhes neste curso introdutório. Para uma visão mais aprofundada, veja [30].

Por fim, estas notas não visam ao objetivo de ensinar como compilar um arquivo em LaTeX, e assume-se que o leitor já tem algum meio de fazer isso. Um modo simples de se iniciar sem grandes preocupações com detalhes técnicos é por meio de editores online, como o Overleaf[32].

2 Classes de Documentos

Diversos tipos de documentos podem ser escritos digitalmente hoje em dia, e diferentes documentos possuem diferentes necessidades. A estruturação utilizada para a escrita de um artigo científico é distinta da necessária para uma apresentação de slides, que é distinta da estruturação de uma tese, e assim por diante.

Por esta razão, o LaTeX conta com diversas classes de documentos, especializadas em diferentes tipos de layout. As três principais são `article`, `report` e `book`. A primeira é, como o nome indica, própria para a escrita de artigos e outros documentos curtos. As que a seguem visam à escrita de documentos mais longos que necessitam, por vezes, ser divididos em partes e mesmo em diversos arquivos `.tex`.

Outros exemplos comuns de classes são a `beamer`[49], para apresentações de slides, ou a `memoir`[54], que objetiva a ser versátil o bastante para reduzir ao máximo a necessidade de pacotes externos (e é a utilizada para a escrita deste documento).

Estes pacotes supracitados servem o propósito de estender as funcionalidades do LaTeX, fornecendo novas customizações, fontes, caracteres, etc. Trataremos de alguns deles com um pouco mais de detalhe ao longo do texto.

Neste documento, nos atentaremos principalmente a documentos escritos com a classe `article`. Sua simplicidade nos permite explorar os fundamentos do LaTeX com

*Na verdade, sendo o LaTeX Turing completo[29], não seria injusto dizer que é uma linguagem de programação.

mais facilidade. Aos interessados, é possível explorar as demais classes em [7] e [8], por exemplo.

3 Esqueleto de um Artigo

Para começarmos a escrever um documento em LaTeX, devemos primeiramente indicar ao LaTeX qual o tipo de documento que desejamos escrever. Não se esqueça: escrevemos o que queremos dizer, não a aparência que desejamos.

Para nosso exemplo, utilizaremos a classe `article`. Devemos informar ao LaTeX que a utilizaremos, e portanto começamos o código em um arquivo `.tex` com o comando `\documentclass{article}`. Todo comando em LaTeX se inicia com um backslash. Os comandos podem receber argumentos, que na maioria dos casos* são fornecidos utilizando chaves (`{}`) ou colchetes (`[]`). As chaves indicam que o argumento é obrigatório — tentar utilizar o comando `\documentclass` sem fornecer o argumento retornará um erro.

Podemos querer escolher uma formatação específica para o artigo. Por exemplo, qual o tamanho do papel que gostaríamos de utilizar? Qual o tamanho da fonte? Estas informações podem ser fornecidas ao LaTeX por meio de um argumento opcional, indicado entre colchetes. Para fonte com tamanho 12pt e papel A4, indicamos `\documentclass[12pt, a4paper]{article}`.

Isto ainda não é tudo, pois estamos escrevendo em português. Diferentemente do inglês, podemos precisar escrever frases cheias de acentuações, como "À noite, vovô Kowalsky vê o ímã cair no pé do pinguim queixoso e vovó põe açúcar no chá de tâmaras do jabuti feliz." Para assegurar que o LaTeX entenda que estamos inserindo texto formatado em UTF-8 e não tenhamos preocupações com a compreensão de acentos e sinais gráficos, carregamos o pacote `inputenc`[22] com a opção `utf8`. Isto é feito por meio do comando `\usepackage[utf8]{inputenc}`, que adicionamos logo abaixo da linha de código anterior. No momento, nosso arquivo `.tex` se assemelha a

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
```

Note que o uso do `inputenc` não é restrito apenas ao porutuguês. Sempre que se desejar usar caracteres UTF-8 é recomendado utilizá-lo[16].

Este é um bom momento para comentar sobre o uso do comando `\usepackage`. Ele permite que o usuário carregue pacotes do LaTeX, estendendo sua funcionalidade básica. A gama de pacotes do LaTeX é extremamente vasta. Por exemplo, há pacotes para

i. incrementar as capacidades de inserção de fórmulas, como o `amsmath`[23];

ii. desenhar diagramas de circuitos elétricos com facilidade, como o `circuitikz`[39];

iii. imprimir, criar e mesmo resolver sudokus, como o `sudokubundle`[53];

*Pacotes, por exemplo, podem utilizar construções diferentes das mais convencionais

iv. descrever movimentos em um tabuleiro de xadrez a partir de notação algébrica, como o `skak`[38];

v. adicionar manchas de café ao documento automaticamente (para que você não o precise fazer manualmente mais tarde), como o `coffee4`[46];

vi. projetar experimentos de Óptica, calculando as trajetórias reais que os raios de luz fariam ao longo do experimento, como o `pst-optexp`[3].

Dentre estes pacotes, exploraremos apenas o `amsmath`, na Seção 9 na página 22, mas também utilizaremos diversos outros pacotes que ainda não mencionamos.

Voltando à escrita de documentos de português, a configuração do documento não acaba com o `inputenc`. Veja o sumário no início neste documento, por exemplo, ou as referências ao final. Estas estruturas são geradas internamente pelo LaTeX, e isto inclui os seus títulos. Para que o resultado final seja "Sumário" ao invés de "Contents", foi preciso informar ao LaTeX que o documento está sendo escrito em português brasileiro. Isto é feito por meio do pacote `babel` com a opção `brazilian`. Assim, nosso código se torna

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
```

Há um último passo na configuração do documento. Para garantir que o resultado que vamos obter esteja formatado corretamente, precisamos garantir que o LaTeX utilize as fontes tipográficas de maneira adequada. Ignorar este passo pode resultar em um PDF com uma boa aparência, mas que possui problemas ao tentarmos copiar o texto, por exemplo.

Tomemos um exemplo consistente. Mais cedo mencionamos a importância do pacote `inputenc` para digitar textos como "À noite, vovô Kowalsky vê o ímã cair no pé do pinguim queixoso e vovó põe açúcar no chá de tâmaras do jabuti feliz." Digamos que tenhamos tomado o cuidado de utilizar o `inputenc`, mas tenhamos esquecido de configurar as fontes apropriadamente. Ao abrir o PDF resultante, encontraremos o texto "À noite, vovô Kowalsky vê o ímã cair no pé do pinguim queixoso e vovó põe açúcar no chá de tâmaras do jabuti feliz" impresso, como esperado. Porém, se o selecionarmos com o mouse e o copiarmos e colarmos em outro lugar, o que obtemos é "A noite, vov^o Kowalsky v^e o ím ã cair no p é do pinguim queixoso e vov ó p õea ç úcar no ch á de t^amaras do jabuti feliz."

O problema que está ocorrendo é que, embora o LaTeX compreenda que digitamos `vovô`, ele não sabe o que fazer com esta informação. Sua configuração padrão permite que o resultado pareça correto, mas ao copiar o resultado do PDF percebemos uma formatação inadequada. Para resolver este problema, utilizamos o pacote `fontenc`[20] com a opção `T1`, que configura as fontes como Type 1. Não vamos entrar em detalhes sobre codificação de fontes. O importante é que isto nos permite usar uma tecnologia de fontes mais robusta e adequada para lidar com caracteres diversos. Mais detalhes sobre a diferença entre o papel do `inputenc` e do `fontenc` podem ser encontrados em [55].

Nosso código agora é

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}
```

e agora podemos começar a inserir informações sobre nosso documento em particular.

As três primeiras informações que devemos fornecer ao documento são seu título, autoria e data. Isto é feito de modo simples. Basta adicionar algumas linhas ao código:

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}

\title{Olá, Mundo!}
\author{Fulano de Tal}
\date{\today}
```

O comando `title` recebe como argumento obrigatório o título do documento, o comando `author` recebe o(s) nome(s) que aparecem como autores e o comando `date` recebe a data. No caso, fornecemos o argumento `\today` ao comando `date`. Poderíamos ter digitado um texto em específico que deve aparecer no lugar da data, como "07 de Fevereiro de 1999"*, mas o `\today` imprime a data em que o documento está sendo compilado, de modo que não é preciso atualizar a data ao atualizar um documento antigo.

Isto conclui o nosso prêambulo — o trecho do documento que antecede a escrita do texto em si. Até agora estávamos preparando o palco para agora podermos começar a peça propriamente dita. Para começarmos a inserir texto, iniciamos o ambiente `document` escrevendo

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}

\title{Olá, Mundo!}
\author{Fulano de Tal}
\date{\today}

\begin{document}

\end{document}
```

*Ou "boliche". O LaTeX não confere se os dados inseridos fazem sentido naquele campo. Porém, utilizar isso contraria a filosofia de *WYSIWYM*, e via de regra desrespeitar esta filosofia leva a documentos mal formatados, por mais que possam parecer atraentes à primeira vista.

Desta vez, temos um comando de abertura e um de encerramento. Isto é o que configura um ambiente. Enquanto comandos são chamados e realizam sua tarefa imediatamente, ambientes são longos. Todo o código digitado entre os comandos de `begin` e `end` será formatado segundo o ambiente sendo utilizado. No caso do ambiente `document`, o texto é formatado como texto a ser impresso no PDF. Para que as informações de título, autoria e data sejam inseridas no PDF, usamos o comando `\maketitle`, e temos

Esqueleto básico de um artigo

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}

\title{Olá, Mundo!}
\author{Fulano de Tal}
\date{\today}

\begin{document}
    \maketitle

\end{document}
```

Este é o esqueleto básico de um artigo. Ao compilar este código, teremos um PDF com título, autoria e data apropriadamente formatados pelo LaTeX, e podemos seguir inserindo texto.

O fato do comando `\maketitle` estar indentado (isto é, mais afastado da margem que as demais linhas de código) pode ser curioso para alguns. O LaTeX não distingue indentação, mas, como sempre em programação, o código pode ser mais simples de ler quando indentado apropriadamente. O LaTeX não vai se incomodar se o código estiver difícil de ler, o único afetado por isso é o próprio usuário.

Vale mencionar ainda que espaços múltiplos e tabs são entendidos pelo LaTeXcomo um único espaço, e o mesmo vale para diversas quebras de linha. Para encerrar um parágrafo de texto é preciso que se inclua uma linha vazia no meio do código. Por exemplo,

```
Primeiro parágrafo de texto, com algum conteúdo para ganhar volume.

Segundo parágrafo, iniciado devido à presença de uma linha vazia entre
↪  este bloco de texto e o anterior.
Mesmo tendo quebrado uma linha, esta frase ainda está no segundo
↪  parágrafo, pois não há uma linha vazia entre este bloco de texto e o
↪  anterior.
```

4 Particionamento

Em geral, documentos de texto são divididos em porções menores, como capítulos e seções. O LaTeX fornece uma maneira prática de efetuar isso, com comandos da forma

`\section`. Este comando em específico inicia uma nova seção do documento, que é uma porção menor que um capítulo, mas maior que uma subseção. A Figura 1 ilustra a hierarquia das partições de texto, cujos comandos nada mais são do que a palavra inglesa que denomina a partição (do topo ao fundo da hierarquia, temos `part`, `chapter`, `section`, `section`, `subsection`, `subsubsection`, `paragraph` e `subparagraph`).

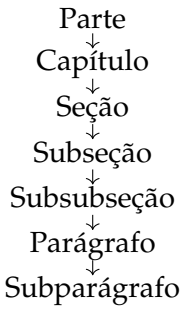

Figura 1: *Hierarquia de Particionamento*

As divisões `part` e `chapter` não estão disponíveis nas classe `article`, que serve ao propósito de escrever documentos relativamente curtos. Pode ser conveniente utilizar uma classe como `report` ou `book` para escrever um documento que tem como capítulos diversos documentos da classe `article`.

Tendo dividido o documento, é interessante que haja um sumário no início do texto para rápida localização dos temas dispostos ao longo do texto. Para a criação de um sumário, basta utilizar o comando `\tableofcontents` dentro do ambiente `document`. No lugar em que o comando for colocado, surgirá um sumário indexando as seções, subseções e etc dispostas ao longo do documento.

Dado que estamos escrevendo um artigo, é importante adicionarmos um resumo do texto. Isso pode ser feito através do uso do ambiente `abstract`, que possui exatamente este propósito. Bastará inserí-lo dentro do ambiente `document` e digitar o conteúdo do resumo no ambiente `abstract` e o próprio LaTeX cuidará da formatação adequada.

É interessante perceber que, graças ao `babel`, o título do sumário é, de fato, *Sumário* e o título do resumo é, de fato, *Resumo*.

Isto é muito abstrato na ausência de exemplos, então vamos complementar nosso código. Para evitar que fiquemos sem ideias de texto, podemos utilizar o pacote `lipsum`[17], que gera parágrafos de *lorem ipsum*, um texto em latim comum para design de documentos, visto que permite ver o resultado da aparência do documento final sem precisarmos nos preocupar com o conteúdo. O pacote fornece o comando `\lipsum`, que recebe um argumento opcional com os parágrafos que gostaríamos que fossem impressos. Por exemplo, `\lipsum[2]` imprime o segundo parágrafo, enquanto `\lipsum[5-9]` imprime do quinto ao nono parágrafo.

Um documento exemplo pode ser gerado então com

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
```

```
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}
\usepackage{lipsum}

\title{Olá, Mundo!}
\author{Fulano de Tal}
\date{\today}

\begin{document}
    \maketitle
    \begin{abstract}
        \lipsum[1]
    \end{abstract}

    \section{Quid est veritas?}
        \lipsum[2]

        \subsection{Est vir qui adest}
            \lipsum[3-4]

\end{document}
```

Notamos novamente que a indentação não é necessária, mas é uma boa prática para a organização do documento.

5 Figuras e Tabelas

> "[E] qual o uso de um livro," pensou Alice, "sem gravuras ou diálogos?"
>
> Lewis Carroll[6].

Naturalmente, desejamos que o editor de texto seja capaz de inserir figuras e tabelas no documento. Tratemos primeiro das figuras, e pensemos nas tabelas em um instante.

O LaTeX não possui suporte nativo para a inclusão de figuras, mas isto pode ser facilmente remediado com o uso de pacotes. No nosso caso, utilizaremos o pacote `graphicx`[5], que nos fornece o comando `\includegraphics`. O comando recebe um argumento obrigatório e diversos argumentos opcionais ativados por palavras–chaves. O argumento obrigatório é o caminho para o arquivo de imagem que desejamos inserir, começando a partir da pasta onde o arquivo `.tex` sendo compilado encontra. As extensões suportadas dependem de detalhes do processo de compilação, mas como via de regra é possível usar, entre outros, arquivos `.png`, `.jpeg` e `.pdf`[4].

Como foi dito, o comando também recebe argumentos opcionais. Eles permitem configurar o tamanho da figura. Por exemplo, podemos fornecer o argumento opcional `scale=3` para triplicar o tamanho da imagem ou o argumento `width=5cm` para que a

figura tenha 5 cm de largura. Há também outros argumentos, como `height`, e podem ser usadas outras unidades de medida comuns em tipografia, como `pt`, `em`, `ex`, `bp`, etc. Tipicamente, é interessante configurarmos o tamanho da figura com base no tamanho do bloco de texto. Este tamanho pode ser acessado por meio do comando `\textwidth`. Assim, uma figura chamada `car-garage.pdf` armazenada na pasta `images` pode ser inserida no documento com 50% da largura do bloco de texto utilizando o comando

```
\includegraphics[width=0.5\textwidth]{images/car-garage.pdf}
```

É pouco conveniente digitar o nome do diretório `images` sempre que inserirmos uma imagem, mas deixar todas as figuras soltas na mesma pasta que o `.tex` principal pode ser pouco organizado. Este problema pode ser contornado inserindo o comando `\graphicspath{{images/}}` no preâmbulo do documento, que informa ao pacote `graphicx` que ele deve buscar pelas figuras que vamos inserir no diretório `images`.

Nosso código se torna

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}
\usepackage{lipsum}
\usepackage{graphicx}
    \graphicspath{{images/}}

\title{Olá, Mundo!}
\author{Fulano de Tal}
\date{\today}

\begin{document}
    \maketitle
    \begin{abstract}
        \lipsum[1]
    \end{abstract}

    \section{Quid est veritas?}
        \lipsum[2]

        \subsection{Est vir qui adest}
            \lipsum[3-4]

            \includegraphics[width=0.5\textwidth]{car-garage.pdf}
\end{document}
```

e devemos nos lembrar que há uma figura chamada `car-garage.pdf` no diretório `images`, que por sua vez está no mesmo diretório que o arquivo `.tex` que estamos escrevendo.

Contudo, percebemos que o resultado da inserção da figura não é exatamente o que gostaríamos. Apenas inserimos a figura, sem nenhuma maneira de colocar uma legenda, centralizá-la ou referenciá-la. Para podermos fazer isso, utilizaremos o ambiente `figure`. Sua estrutura geral é da forma*

```
\begin{figure}[!htpb]
        \centering
        \includegraphics[width= 0.5\textwidth]{car-garage.pdf}
        \caption{Diagrama de espaço--tempo para o paradoxo do carro e da
        ↪  garagem.}
        \label{fig: car-garage}
\end{figure}
```

sendo que este código gera a Figura 2.

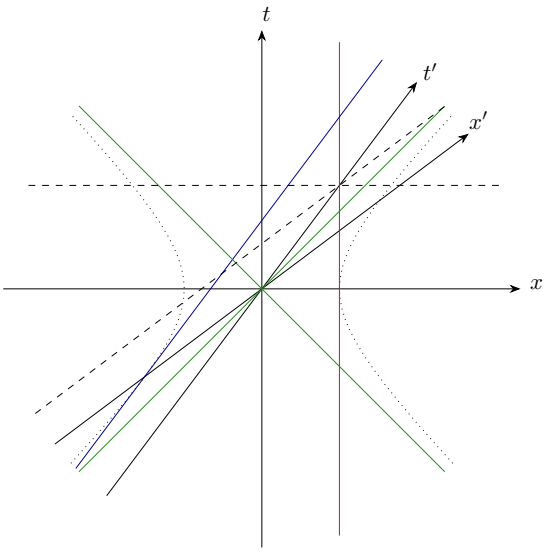

Figura 2: *Diagrama de espaço–tempo para o paradoxo do carro e da garagem.*

Vamos a cada trecho do código. O ambiente `figure` permite que a imagem seja tratada como um float: um elemento "flutuante" do texto que será posicionado pelo LaTeX no local que fornecer a melhor diagramação.

O argumento opcional `htbp!` explicita alguns critérios que o usuário gostaria de pedir ao LaTeX que leve em conta enquanto escolhe o local com a melhor diagramação. `h` representa *here* (aqui), `t` *top* (em cima), `b` *bottom* (embaixo), `p` *page* (página). A ordem é irrelevante. Caso `h` seja especificado, o LaTeX tentará deixar a figura no mesmo local onde o código foi inserido. Caso não seja possível e `t` tenha sido fornecido, o LaTeX a colocará no topo de uma página. Caso não seja possível e `b` tenha sido fornecido, o mesmo ocorre

*Note que nosso código está se tornando razoavelmente grande. Por isso, não o reproduziremos por completo nos próximos exemplos.

para o fundo de uma página. Por fim, caso nenhuma dessas opções tenha sido possível, a figura é adicionada a uma fila de espera e posicionada em uma página exclusiva para floats. O ponto de exclamação `!` indica um pedido para que o LaTeX ignore algumas restrições (como número máximo de floats em uma página), mas isso não significa que ele necessariamente colocará o float em uma das posições especificadas. Perceba que a especificação `htbp` permite que o LaTeX tenha diversas possibilidades de posicionamento antes de tentar colocar o float em uma página independente, o que significa que em geral os floats estarão mais próximos de onde seu código foi digitado. Mais detalhes acerca do posicionamento de floats podem ser encontrados em [25].

A seguir, temos o comando `\centering`, que assegura a centralização da figura. Logo após temos `\caption`, que recebe um argumento obrigatório. Seu papel é fornecer uma legenda ao float, sendo a legenda seu argumento. À legenda é então atribuído, pelo comando `\label`, um rótulo, que é seu argumento. Note que o rótulo é atribuído à legenda, não ao float. Logo, o comando `\label` deve vir imediatamente após o comando `\caption`. O rótulo nos permite então utilizar referências cruzadas: com o comando `\ref` podemos obter o número atribuído à figura. Para escrever "Figura 2", digitamos `Figura \ref{fig: car-garage}`. O rótulo `fig: car-garage` armazena o número da figura e nos permite referenciá-la.

A existência de referências cruzadas significa que é irrelevante a figura estar no local onde seu código foi digitado. Ela sempre pode ser encontrada com base em referências cruzadas.

Alguém poderia dizer que é um incômodo ter que buscar através de páginas e páginas pelo float em questão. Isso também pode ser remediado com facilidade. O comando `\pageref` funciona como o `\ref`, mas ao invés de retornar o número da figura, retorna a página em que ela se encontra. Para obter "Figura 2 na página 10" podemos digitar `Figura \ref{fig: car-garage} na página \pageref{fig: car-garage}`.

Uma grande parte da sintaxe para inserção de tabelas se assemelha à das figuras, com o código geral sendo semelhante a

```
\begin{table}[!htbp]
        \centering
        \caption{Exemplos de números representados em base binária}
        \label{tab: binary}
        \begin{tabular}{c|c}
                Base Decimal & Base Binária \\ \hline
                1 & 1 \\
                2 & 10 \\
                3 & 11 \\
                4 & 100 \\
                5 & 101
        \end{tabular}
\end{table}
```

que gera a Tabela 1 na próxima página.

Tabela 1: *Exemplos de números representados em base binária*

Base Decimal	Base Binária
1	1
2	10
3	11
4	100
5	101

A colocação dos comandos `\caption` e `\label` acima da tabela se dá por convenção: é tradicional dispor as legendas de figuras abaixo das mesmas e as de tabelas acima. Perceba que, a menos deste detalhe, as únicas novidades no código estão no ambiente `tabular`.

O ambiente `tabular` se propõe a escrever tabelas. Ele recebe um argumento obrigatório que deve descrever a disposição das colunas na tabela. No exemplo, mostrou-se duas colunas com texto centralizado (por isso o `c` presente no código) separadas por uma barra (o que justifica o símbolo `|`). Podemos adicionar mais colunas escrevendo mais letras, separar as colunas com barras duplas (escrevendo `||`) ou mudar o alinhamento — `l`, abreviação de *left*, para alinhar à esquerda e `r` (*right*) para alinhar à direita.

As linhas são adicionadas à tabela uma a uma. A entrada de cada coluna deve ser separada por um *ampersand* (`&`) e indica-se o fim de uma linha com um backslash duplo (`\\`), que representa um fim de linha em geral no LaTeX. O comando `\hline` indica que se deseja colocar uma linha horizontal no local especificado.

6 Expressões Matemáticas

Uma das principais motivações para o uso do LaTeX é o seu poderoso sistema para lidar com expressões matemáticas. Vejamos como isso é feito. Por ora, analisaremos apenas algumas expressões simples e que envolvem Cálculo de modo limitado. A justificativa é que na Seção 9 na página 22 utilizaremos pacotes como o `amsmath`, `physics`[13], `mathtools`[21], entre outros, que permitem um controle mais poderoso das expressões matemáticas e, em particular, fornecem métodos mais simples para lidar com equações comuns nas Ciências Físicas.

Podemos inserir equações no meio de uma frase, como $pV = nk_BT$, ou com mais destaque na página, como

$$\frac{v^2}{2} + \frac{p}{\rho} + gz = k.$$

Para inserir expressões em linha, delimitamos a expressão usando `\(...\)`. Por exemplo, $pV = nk_BT$ é obtido digitando `\(pV = n k_B T\)`. Expressões em destaque são obtidas de modo semelhante, mas os delimitadores são `\[ ... \]`, de forma que

$$\frac{v^2}{2} + \frac{p}{\rho} + gz = k$$

é o resultado do código

```
\[ \frac{v^2}{2} + \frac{p}{\rho} + g z = k \]
```

Por completeza, mencionamos que estas não são as únicas formas de inserir equações em linha e em destaque. `$...$` possui um resultado muito semelhante a `\(...\)` e `$$...$$` possui um resultado próximo a `\[...\]`. A principal diferença é que os comandos `$...$` e `$$...$$` são do TEX puro, que é mais antigo, enquanto `\(...\)` e `\[...\]` são comandos de fato do LATEX, mais moderno. É altamente recomendado que se dê preferência ao uso de `\[...\]` em oposição a `$$...$$`, como pode ser visto em [47], mas há discordâncias sobre se `\(...\)` é ou não preferível a `$...$`, como pode ser visto em [26].

Deixemos um pequeno comentário em nosso código para nos lembrarmos disso. Temos

```
\[ \frac{v^2}{2} + \frac{p}{\rho} + g z = k \] %$$...$$ possui um
↪  resultado parecido com \[...\], mas seu uso não é recomendado
```

O código que vem após % é comentado e não tem qualquer impacto no PDF gerado. Note que isto vale apenas para o código que está na mesma linha que %: da linha seguinte em diante o código é compilado normalmente.

Em nossos exemplos, já visualizamos como escrever sobrescritos, como x^n — `\(x^n\)` — e subscritos, como x_i — `\(x_i\)`. Porém, perceba que `\(x^100\)` pode fornecer um resultado estranho: x^100. Os sobrescritos e subscritos se referem apenas ao caractere imediatamente após o símbolo de `^` ou `_`. Quando desejamos incluir mais de um caractere, os encapsulamos usando chaves. Para obter x^{100} escrevemos `\(x^{100}\)`.

Percebemos também uma pequena sugestão de como escrever letras gregas. Para obter ρ havíamos escrito* `\rho`. De modo análogo, qualquer letra grega pode ser obtida com base em seu nome `\alpha`, `\xi`, `\omega` fornecem α, ξ, ω e assim por diante. Letras maiúsculas são obtidas de modo análogo, mas iniciando o comando por uma letra maiúscula. `\Psi` fornece Ψ, `\Xi` fornece Ξ, etc. Notamos que letras gregas maiúsculas cujo desenho é idêntico a alguma letra latina não possuem comandos. `\Alpha` retornará um erro, por exemplo, visto que o alfa maiúsculo possui o mesmo desenho que o A maiúsculo do alfabeto latino.

Alguns sinais são inseridos de forma elementar: `a-b=c` retorna $a - b = c$, por exemplo. O sinal de multiplicação pode ser escrito como $\times$ (`\times`) ou $\cdot$ (`\cdot`). Frações podem ser escritas por meio do comando `\frac`, que recebe dois argumentos obrigatórios: um numerador e um denominador. Por exemplo, `\frac{a}{b}` retorna $\frac{a}{b}$.

Alguns outros símbolos comuns são a raiz n-ésima de um número a, que pode ser escrita como `\sqrt[n]{a}` ($\sqrt[n]{a}$), integrais, que podem ser escritas como `\int`, somatórios e produtórios, escritos como `\sum` e `\prod`, respectivamente. Pode-se adicionar limites a estes últimos operadores usando circunflexos e underslashes. Por exemplo,

*Vamos omitir os sinais de ambiente matemático quando eles forem evidentes pelo contexto.

```
\[ \sum_{n=1}^{\infty} \frac{(-1)^n}{n} = \log{2} \]
```

retornará

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{n} = \log 2,$$

vemos que $\log 2$ retorna o logaritmo apropriadamente formatado. Isto é diferente de $log2$, que obteríamos escrevendo `log{2}`, sem usar o comando `\log`. Enquanto o comando escreve log como texto, `log` formata a palavra como se fossem três variáveis sendo multiplicadas.

Finalmente, podemos obter equações numeradas com o uso do ambiente `equation`. A saber, o código

```
\begin{equation}
    F(s) = \frac{1}{2 \pi} \int_{-\infty}^{+\infty} f(t)e^{-i s t} dt.
    \label{eq: fourier}
\end{equation}
```

fornece a Eq. (1).

$$F(s) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(t)e^{-ist} dt. \tag{1}$$

Desta vez não precisamos adicionar uma legenda, pois a numeração do ambiente se encarrega disso. Podemos então fazer referência à equação normalmente, e, como esperaríamos, `Equação \ref{eq: fourier}` retornará Equação 1.

Notamos que o diferencial na Eq. (1) não está formatado de modo adequado: dt pode sugerir que d é uma variável, dado que está em itálico. O ideal seria termos $\mathrm{d}t$. Obteremos este resultado com o auxílio do pacote `physics`.

6.1 Lidando com Dúvidas e Buscando Comandos Desconhecidos

Há diversos outros comandos que podem ser utilizados em expressões matemáticas. Na verdade, são tantos que sequer vale a pena tentarmos listar neste documento. Por isso, sugerimos consultar [42] — um documento de mais de 300 páginas contendo mais de 14 000 símbolos que podem ser utilizados no LaTeX— ou [12] — um website que permite que o usuário desenhe o símbolo com o mouse e veja possíveis símbolos disponíveis no LaTeX que se assemelham ao desenhado. Em ambos os casos, os símbolos podem estar disponíveis no LaTeX "puro" ou por meio de pacotes.

É comum que diversas dúvidas acerca de LaTeX surjam ao longo do processo de escrita, e algumas fontes excelentes com tutoriais ou no estilo Q&A são [32, 48]. Aos dispostos a ler um pouco mais, [30] é uma referência excelente e completa.

7 Formatação de Texto

Façamos ainda alguns comentários sobre formatação de texto no LaTeX. Por exemplo, como podemos fazer listas, ou dar ênfase a alguma palavra em particular?

Listas podem ser obtidas usando os ambientes `enumerate`, `itemize` e `description`. `enumerate` se propõe a escrever listas ordenadas. Por exemplo,

```
\begin{enumerate}
    \item O Fantasma;
    \item Rei Cláudio;
    \item Rainha Gertrudes.
\end{enumerate}
```

fornece*

i. O Fantasma;

ii. Rei Cláudio;

iii. Rainha Gertrudes.

O ambiente `itemize` tem um funcionamento semelhante, mas não enumera as entradas.

```
\begin{itemize}
    \item O Fantasma;
    \item Christine Daaé;
    \item Visconde Raoul de Chagny.
\end{itemize}
```

fornece

- O Fantasma;
- Christine Daaé;
- Visconde Raoul de Chagny.

`description`, por fim, se presta a fornecer descrições dos itens listados. Como exemplo,

```
\begin{description}
    \item[O Fantasma] espírito de Jacob Marley, sócio de Scrooge que
    ↪  morreu há sete anos;
    \item[Ebenezer Scrooge] sovina sem coração que odeia o Natal;
    \item[Bob Cratchit] empregado de Scrooge.
\end{description}
```

*Os contadores podem ser personalizados. Neste documento, utilizamos um estilo de contagem usando numerais romanos minúsculos, mas o padrão do LaTeX utiliza algarismos arábicos.

fornece

O Fantasma espírito de Jacob Marley, sócio de Scrooge que morreu há sete anos;

Ebenezer Scrooge sovina sem coração que odeia o Natal;

Bob Cratchit empregado de Scrooge.

Em textos digitados com máquinas de escrever, é comum dar ênfase a certas palavras as sublinhando. Isso pode ser obtido no LaTeX passando ao comando `\underline` o texto que desejamos sublinhar como argumento obrigatório. `\underline{sublinhado}` fornece sublinhado, por exemplo. Textos modernos frequentemente preferem *itálico*, que é obtido com o comando `\textit`. `\textit{itálico}` retorna *itálico*. Negrito é obtido de modo semelhante, com `\textbf{negrito}` retornando **negrito**.

Porém, na posição de autor, nenhum desses comandos deveria ser relevante*. O objetivo do autor com o uso de, por exemplo, negrito, não é que a palavra fique em negrito, mas sim que ela possua um destaque distinto do restante do texto. O objetivo é dar ênfase. Escrever `\textbf{negrito}` quebra com a filosofia *WYSIWYM* que nos trouxe ao LaTeX em primeiro lugar, pois estamos escrevendo o que queremos ver, não o que queremos dizer. Por isso, damos preferência a escrever `\emph{negrito}`, que retorna *negrito*.

A princípio, o comando `\emph` pode aparecer apenas uma forma distinta de escrever em itálico. Contudo, há duas diferenças

i. `\emph` tem o significado de dar ênfase, enquanto `\textit` tem o significado de formatar em itálico;

ii. os comandos de fato de comportam de modos diferentes.

Tomemos o exemplo

```
\textit{Escrevemos algum texto. Em particular, escrevemos uma palavra em
↪  \textit{itálico} e mais uma com \emph{ênfase}, para efeitos de
↪  comparação.}
```

O resultado é "*Escrevemos algum texto. Em particular, escrevemos uma palavra em itálico e mais uma com* ênfase*, para efeitos de comparação.*" Enquanto `\textit` não tem efeito em palavras que já estão em itálico, `\emph` remove o itálico, de modo que a ênfase é dada de toda forma.

Isto não significa que comandos como `\textbf` ou `\textit` são inúteis. A capa deste documento usa negrito, por exemplo. O que estamos colocando em jogo é como estes comandos são usados. Na posição de autor, o que é relevante é dar ênfase a esta ou àquela palavra. Porém, ao criar um modelo de documento em LaTeX do zero, é conveniente poder usar negrito, itálico e outras possibilidades de formatação para atribuir uma determinada aparência a cada tipo de objeto.

*Roubo as palavras de [30].

Fazemos ainda alguns comentários finais sobre boas práticas ao digitar texto em LaTeX. Aspas não devem ser incluídas utilizando `"..."`. Isto resulta em "...", que é mal–formatado. `''...''` (duas aspas simples seguidas ao invés de uma única aspa dupla) fornece ”...”, que também é mal–formatado, dado que os dois pares de aspas estão direcionados para o mesmo lado. O modo correto de utilizar aspas é escrevendo ``` ``...'' ``` (com dois acentos graves no começo) para obter “...”.

Escrever ... (três pontos consecutivos) também é uma má prática, pois espaça o sinal gráfico ...erroneamente. O modo correto é `\ldots`, que espaça as reticências de forma apropriada.

Por fim — não por falta de assuntos sobre os quais podemos ser pedantes, mas sim porque precisamos parar em algum momento — o LaTeX possui quatro tipos de *dashes*. Um deles é o menos numérico: escrever `\(-1\)` retorna -1. No modo de texto, temos outros três: o hífen (-), a meia-risca (–) e o travessão (—). Podemos digitar o hífen com `-`, a meia-risca com `--` e o travessão com `---`. Os detalhes de quando utilizar cada um deles depende não da tipografia, mas da gramática da língua sendo utilizada. O uso correto desses sinais gráficos pode fornecer uma compreensão mais clara ao texto: se mencionamos o livro de Choquet-Bruhat–DeWitt-Morette–Dillard-Bleick, podemos concluir que o livro possui três autoras. Enquanto o hífen ocorre em nomes compostos — como Yvonne Choquet-Bruhat, Cécile DeWitt-Morette e Margaret Dillard-Bleick, que são as autoras de [9] — a meia-risca ocorre ao ligarmos os nomes de duas pessoas distintas. Assim, sabemos que as equações de Navier–Stokes são nomeadas em homenagem a duas pessoas, enquanto o símbolo de Levi-Civita homenageia apenas uma.

8 Administrando Bibliografias

Um dos grandes poderes do LaTeX é sua facilidade para lidar com bibliografias. Nesta seção, apresentaremos como escrever bibliografias com o pacote `biblatex`[24], seguindo a estrutura geral de [35]. Outras opções seriam utilizar o pacote `natbib`[11] ou a estrutura nativa do LaTeX, o `bibtex`. Estes são apresentados em [36, 37], mas note que o próprio Overleaf recomenda o uso do `biblatex`, visto que ele “provê localização em diversas línguas, ser é ativamente desenvolvido e torna a administração de bibliografia mais fácil e flexível”[36]. A depender do contexto, pode haver desvantagens no uso do `biblatex` — como por exemplo alguma revista não admitir o uso de citações em `biblatex`. Uma explicação sobre as vantagens e defeitos de diferentes modos de se escrever bibliografias em LaTeX está disponível em [14].

Dado que utilizaremos um pacote, devemos começar o carregando no preâmbulo. Escrevemos

```
\usepackage[backend=biber]{biblatex}
```

O argumento opcional `backend=biber` diz ao LaTeX com qual programa desejamos interpretar os arquivos de bibliografia que utilizaremos. As opções são `biber` — que é mais versátil e moderno e provê mais funcionalidades ao `biblatex` — e `bibtex` — mais

estável, amplamente utilizado, mas limitado. Uma discussão mais aprofundada sobre as diferenças entre os dois pode ser encontrada em [14].

Há três comandos principais com os quais devemos nos preocupar neste trecho: `\addbibresource`, `\cite` e `\printbibliography`.

`\addbibresource` nos permite importar um arquivo de bibliografia `.bib`, os quais discutiremos em breve. `\addbibresource` recebe como argumento obrigatório o caminho para o arquivo `.bib` que desejamos importar. Por exemplo, se temos um arquivo chamado `bib.bib` na mesma pasta que o arquivo `.tex` que está sendo compilado, escrevemos `\addbibresource{bib.bib}` no código.

O comando `\cite` funciona de modo semelhante ao `\ref`, mas utiliza os rótulos e informações definidos no arquivo `.bib`.

Por fim, o comando `\printbibliography` imprime a bibliografia com a formatação desejada, exibindo apenas as referências que foram citadas no texto. Caso desejemos adicionar referências à bibliografia sem as mencionar no texto, podemos usar o comando `\nocite`, que funciona de modo análogo a `\cite`, mas não imprime no PDF a referência à obra sendo mencionada. Podemos ainda usar `\nocite{*}` para referenciar todas as obras nos arquivos `.bib` que importamos sem que as referências sejam escritas no PDF — a menos da bibliografia, claro.

Façamos um MWE* Temos

```
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[brazilian]{babel}
\usepackage[T1]{fontenc}

\usepackage[backend=biber]{biblatex}
    \addbibresource{bib.bib}

\begin{document}
    Uma boa introdução aos métodos da análise dimensional é
    ↪ \cite{paper:trancanelli}. \cite{book:paterson1983}, por outro
    ↪ lado, é uma boa referência para o estudo de Mecânica dos Fluidos
    ↪ e \cite{phd:mauro2003} apresenta um ponto de vista moderno sobre
    ↪ a Mecânica de Koopman-- von Neumann. Em suma, recomendamos a
    ↪ leitura de
    ↪ \cite{paper:trancanelli,phd:mauro2003,book:paterson1983}. A
    ↪ leitura de \cite{book:burden2011} também é recomendada.

    \printbibliography
\end{document}
```

*Minimal Working Example, ou exemplo funcional mínimo, em português. É um documento em LaTeX tão curto quanto possível, mas capaz de reproduzir as características ou problemas que se quer exemplificar. São vastamente recomendados para perguntas almejando solução de problemas no TeXStack Exchange[48].

Para este documento, utilizamos o arquivo `bib.bib`

```
@book{book:paterson1983,
    author = {Paterson, A. R.},
    title = {A First Course in Fluid Dynamics},
    year = {1983},
    address = {Cambridge},
    publisher = {Cambridge University Press},
    keywords = {fluids},
}

@book{book:burden2011,
    author = {Burden, Richard L. and Faires, J. Douglas},
    title = {Numerical Analysis},
    publisher = {Brooks/Cole},
    year = {2011},
    address = {Boston},
    keywords = {numerical},
}

@phdthesis{phd:mauro2003,
    author = {{Mauro}, D.},
    title = {Topics in Koopman--von Neumann Theory},
    year = {2002},
    institution = {Università degli Studi di Trieste},
    archivePrefix = {arXiv},
    eprint = {quant-ph/0301172},
    keywords = {mechanics},
}

@article{paper:trancanelli,
    author = {Diego Trancanelli},
    title = {Grandezas Físicas e Análise Dimensional: da Mecânica à
    ↪  Gravidade Quântica},
    doi = {https://doi.org/10.1590/1806-9126-RBEF-2015-0003},
    journal = {Revista Brasileira do Ensino de Física},
    volume = {38},
    year = {2016},
    ISSN = {1806-1117},
    publisher = {Scielo},
    keywords = {dimensional},
}
```

8.1 Estilos de Bibliografia e Citação

Note que o último comando `\cite` mencionou os rótulos das três obras separados por vírgulas, mas sem espaços. Isto permite que as citações apareçam em conjunto, como [27, 44, 51], ao invés de soltas como [9][6].

Podemos personalizar os estilos de citação e bibliografia ao fornecer argumentos opcionais ao pacote `biblatex`, bem como a forma de ordenamento da bibliografia. Para que o sistema de referências seja formatado tal qual o destas notas, usamos

```
\usepackage[backend = biber, bibstyle = nature, sorting = nty,
↪  citestyle = numeric-comp]{biblatex}
```

`bibstyle = nature` faz com que a bibliografia seja formatada no mesmo estilo da revista *Nature*. `sorting = nty` define o ordenamento das entradas na bibliografia segundo nome, título e ano (ou, em inglês, *year*). `citestyle = numeric-comp` assegura que as citações ao longo do texto apareçam na forma "[973]", com o cuidado de que citações consecutivas aparecem como "[973-975]" ao invés de "[973, 974, 975]". Há diversos outros estilos possíveis, e mais informações e exemplos podem ser encontrados em [24, 33, 34].

8.2 Arquivo `.bib`

Vamos analisar agora o arquivo `bib.bib` que está armazenando as informações bibliográficas. Tomemos por exemplo o trecho que usamos para citar [44], visto que sua simplicidade nos permite analisar o funcionamento geral do código. Temos

```
@book{book:burden2011,
    author = {Burden, Richard L. and Faires, J. Douglas},
    title = {Numerical Analysis},
    publisher = {Brooks/Cole},
    year = {2011},
    address = {Boston},
    keywords = {numerical},
}
```

Em primeiro lugar, nós declaramos o tipo da entrada. A saber, `book`, de modo que o LaTeX sabe que deverá formatar a entrada na bibliografia como sendo um livro. Há diversos tipos de entradas, e pode-se encontrar alguns exemplos em [24, 35]. Perceba que o nosso exemplo utilizou, além da entrada `book`, as entradas `article` (para artigos) e `phdthesis` (para teses de doutorado).

Logo após a entrada, digita-se o rótulo da citação. No caso, `book:burden2011`. Isto é apenas o rótulo que usaremos para referência cruzada. É uma boa prática iniciar os rótulos com o tipo de entrada, o que pode facilitar o trabalho do autor em identificar o rótulo que procura. Um método comum para rotular as entradas é utilizando o padrão `autorANOtítulo`, onde `ANO` é o ano de publicação. Naturalmente, a escolha é do autor, e estas são apenas algumas sugestões de práticas.

A seguir temos os campos em que colocamos as informações sobre os documentos. Eles são razoavelmente autoexplicativos. `author` recebe o autor, `title` o título, `year` o ano e assim por diante. [24, 35] também possuem listas com diversos tipos de campos que podem ser utilizados. Porém, notamos que nem todos os campos preenchidos serão utilizados pelo LaTeX: apenas aqueles que constam no estilo de bibliografia escolhido aparecerão no documento final. Perceba ainda que ao fim de cada campo há uma vírgula, para que o LaTeX saiba que terminamos de inserir a informação relacionada àquele campo. A vírgula após o último campo é opcional.

Uma menção especial deve ser feita ao campo `author`: frequentemente desejamos inserir obras com mais de um autor. Para separar os autores, usamos a palavra `and`, como em `author = {Burden, Richard L. and Faires, J. Douglas},`. Caso desejássemos incluir mais autores, bastaria seguir os separando por `and`. Por exemplo, as informações bibliográficas de [9] são inseridas neste documento como

```
@book{choquet-bruhat1982,
    author = {Choquet-Bruhat, Yvonne and DeWitt-Morette, Cécile and
    ↪  Dillard-Bleick, Margaret},
    title = {Analysis, Manifolds, and Physics},
    publisher = {North-Holland},
    year = {1982},
    address = {Amsterdam},
}
```

8.3 Personalizando a Bibliografia

A bibliografia permite mais customização por meio do fornecimento de argumentos opcionais ao comando `\printbibliography`. Para adicionar um título específico à bibliografia, podemos fornecer o argumento opcional `title={TÍTULO DESEJADO}`. Por exemplo,

```
\printbibliography[title={Lista de Referências}]
```

fornecerá uma bibliografia com o título "Lista de Referências".

Podemos ainda fazer bibliografias parciais. Para uma bibliografia que liste apenas as entradas do tipo `article`, podemos utilizar o argumento `type = article`. De modo semelhante, o argumento `keyword = {fluids}` restringe a bibliografia às entradas que possuem `fluids` em sua lista de palavras-chave — a qual especificamos por meio do campo `keyword` no arquivo `.bib`.

No caso comum em que desejamos que a bibliografia seja adicionada ao sumário, utilizamos o argumento opcional `heading=bibintoc`. `heading=subbibintoc` se comporta de modo análogo, mas a bibliografia é tratada como uma seção ao invés de um capítulo (se a classe utilizada permitir capítulos) ou como uma subseção ao invés de uma seção (se a classe utilizada não permitir capítulos). Isto permite construções como

```
\printbibliography[heading=bibintoc, title={Bibliografia Completa}]

\printbibliography[heading=subbibintoc,keyword={fluids},title={Mecânica
↪  dos Fluidos}]
```

Por fim, [31] pode ser uma ferramenta valiosa para escrever o arquivo `.bib` a ser utilizado.

9 Pacotes para Matemática

9.1 `amsmath`

O pacote primordial para a escrita de qualquer documento em LaTeXque envolve expressões matemáticas é o pacote `amsmath`[23], que fortalece diversas das capacidades matemáticas do LaTeX. Este é provavelmente o mais interessante, mas certamente não o único, pacote da American Mathematical Society. Na Seção 9.2 na página 24 discutiremos brevemente ainda o `amsfonts` e o `amssymb`, que estendem os símbolos que podemos acessar no modo matemático.

Algumas das funcionalidades implementadas pelo `amsmath` são o maior controle de ambientes matemáticos para sequências de equações (como o ambiente `align`), escrita de texto em modo matemático por meio do comando `\text`, criação de novos operadores matemáticos com `\DeclareMathOperator`, e muito mais.

É possível configurar o pacote, por meio de argumentos opcionais, para que a numeração das equações fique centralizada, deslocada para baixo ou deslocada para cima em equações que possuem divisões horizontais (como expressões envolvendo frações ou coeficientes binomiais). Além disso, é possível configurar a disposição dos limites de "operadores grandes" para que se dê à frente ou acima e abaixo dos símbolos. Também é possível configurar se a disposição dos números das equações se dará alinhada à esquerda, alinhada à direita ou a uma certa distância específica. Detalhes sobre essas configurações podem ser encontrados em [23].

O pacote introduz diversos novos ambientes matemáticos para a escrita de sequências de equações, sejam elas alinhadas ou não. O ambiente `align` permite a escrita de equações alinhadas, o ambiente `gather` a escrita de equações sem se preocupar com alinhamento, `split` permite a escrita de trechos alinhados dentro de outros ambientes matemáticos, e assim por diante. A menos dos ambientes que tem o propósito de ser utilizados dentro de outros ambientes matemáticos, todos contam com versões estreladas — como `align*` — que omitem a numeração das equações. Em particular, é introduzido o ambiente `equation*`, cujo funcionamento é idêntico aos comandos `\[...\]` — de fato, o `amsmath` redefine `\[...\]` para que seja exatamente o comando `equation*`, o que permite mais colaboração de `\[...\]` com as funcionalidades do pacote*.

São introduzidos também diversos comandos menores para lidar com a formatação das equações. A numeração das equações pode ser substituída com o comando `\tag`. Este também pode ser estrelado (`\tag*`), o que omitirá os parênteses que cercam a marcação da

*Esta é uma das razões para se dar preferência a `\[...\]` ao invés de `$$...$$`, conforme [47].

equação. Anotações à margem de cada expressão podem ser feitas utilizando o comando `\text`, que em geral permite a escrita de texto em meio ao ambiente matemático.

Tomemos um exemplo. Escrevemos

```
\begin{align}
    a &= b, \\
    ab &= b^2, \\
    ab - a^2 &= b^2 - a^2, \\
    a(b-a) &= (a+b)(b-a), \tag*{Verdadeiro} \\
    a &= a + b, \tag{Falso} \\
    a &= 2a, \\
    1 &= 2, \text{ que é absurdo.}
\end{align}
```

e obtemos

$$\begin{aligned}
a &= b, && (2)\\
ab &= b^2, && (3)\\
ab - a^2 &= b^2 - a^2, && (4)\\
a(b-a) &= (a+b)(b-a), && \text{Verdadeiro}\\
a &= a + b, && (\text{Falso})\\
a &= 2a, && (5)\\
1 &= 2, \text{ que é absurdo.} && (6)
\end{aligned}$$

O ambiente `gather` funciona de modo análogo, mas sem o alinhamento de equações — e portando não utiliza os símbolos de `&`. Como ambos os ambientes ocupam todo o espaço de uma linha de texto, pode-se querer as versões que não o fazem. Elas são dadas pelos ambientes `gathered` e `aligned`. Outros ambientes e mais detalhes sobre estes podem ser encontrados em [23].

Matrizes podem ser digitadas com facilidade usando os ambientes `matrix`, `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` e `Vmatrix`. Cada um deles utiliza delimitadores diferentes para as "paredes" da matriz (nenhum, parênteses, colchetes, chaves, barra vertical simples e barra vertical dupla, respectivamente). A sintaxe das matrizes é semelhante à das tabelas, com a exceção de que não é preciso se preocupar com o número de colunas. Também é possível inserir matrizes dentro de matrizes para escrever matrizes por blocos. Como exemplo, o código

```
\[\Lambda = \begin{pmatrix}
    \begin{matrix} \gamma & -\beta\gamma \\ -\beta\gamma & \gamma
    ↪  \end{matrix} & 0 \\
        0 & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}
\end{pmatrix}.\]
```

retorna

$$\Lambda = \begin{pmatrix} \gamma & -\beta\gamma & & 0 \\ -\beta\gamma & \gamma & & \\ & & 1 & 0 \\ 0 & & 0 & 1 \end{pmatrix}.$$

Matrizes, sendo entidades matemáticas, devem estar em ambientes matemáticos. Há também um ambiente `smallmatrix` que pode ser usado para digitar matrizes em modo matemático em linha. `\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}` retorna $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$.

Estes são apenas alguns exemplos das funcionalidades do pacote, e muitas outras funcionalidades podem ser encontradas em sua documentação, disponível em [23], ou em [15]. Sugerimos o uso do pacote `mathtools`[21] em conjunto com o `amsmath`, dado que aquele corrige alguns problemas deste, além de adicionar algumas funcionalidades novas.

9.2 `amssymb`

O pacote `amsfonts`[2] incrementa as fontes matemáticas disponíveis ao usuário, adicionando símbolos *blackboard bold* ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, ...), *fraktur* ($\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$) e outros.

O pacote `amssymb` expande ainda mais esta biblioteca de símbolos, utilizando as fontes carregadas pelo `amsfonts`. Assim, carregar o `amssymb` automaticamente carrega o `amsfonts`.

Detalhes mais específicos sobre o pacote `amsfonts`, incluindo a história das fontes que ele introduz, podem ser encontrados em [2]. Mais detalhes sobre símbolos definidos por diversos pacotes do LaTeX estão disponíveis em [42].

9.3 `physics`

O pacote `physics`[13] possui ferramentas extremamente poderosas para a escrita de equações recorrentes em Física, além de sua documentação ser clara e concisa. Com ele, tem-se facilitado o trabalho de inserção de derivadas em notação de Leibniz (ordinárias e parciais), operadores comuns em cálculo vetorial, notações com vetores, matrizes, e muito mais. Em geral, os operadores definidos ou redefinidos pelo pacote possuem adaptação automática do tamanho dos delimitadores. Ou seja, não é preciso se preocupar com o tamanho de parênteses, colchetes e etc.

Estão inclusos comandos para escrever valores absolutos (`\abs{a}` fornece $|a|$), normas (`\norm{a}` fornece $\|a\|$), avaliar variáveis (`\eval{x}_0^1` e $x\big|_0^1$), comutadores de matrizes (`\comm{A}{B}` $[A, B]$), colchetes de Poisson (`\pb{p}{q}` $\{p, q\}$), e muitos outros.

Em particular, o pacote lida com uma ampla gama de notações vetoriais. O operador ∇ ganha diversas formas para abarcar cada uma de suas tradicionais variações. Além disso, há notações de rápido acesso para vetores. Estas estão listadas na Tabela 2

As funções trigonométricas, bem como outras tradicionais, são redefinidas pelo `physics` de forma a ganharem novas funcionalidades. Por exemplo, tem-se agora que `\sin[2](x)` fornece $\sin^2(x)$, sendo que parênteses se adaptam à altura do argumento.

Tabela 2: *Notações para vetores definidas pelo pacote* `physics`.

Entrada	Saída	Entrada	Saída
`\grad{f}`	$\boldsymbol{\nabla} f$	`\vb{x}`	$\mathbf{x}$
`\div{\vb{F}}`	$\boldsymbol{\nabla} \cdot \mathbf{F}$	`\va{x}`	$\vec{\mathbf{x}}$
`\curl{\vb{F}}`	$\boldsymbol{\nabla} \times \mathbf{F}$	`\vu{x}`	$\hat{\mathbf{x}}$
`\laplacian{f}`	$\nabla^2 f$	`\vb{x} \cp \vb{y}`	$\mathbf{x} \times \mathbf{y}$
`\vb{x} \vdot \vb{y}`	$\mathbf{x} \cdot \mathbf{y}$		

Há diversas notações usadas para derivadas e diferenciais, que podem ser vistas na Tabela 3.

Tabela 3: *Notações para derivadas definidas pelo pacote* `physics`.

Entrada	Saída	Entrada	Saída
`\dd{x}`	$\mathrm{d}x$	`\dd[n]{x}`	$\mathrm{d}^n x$
`\dd(\cos\theta)`	$\mathrm{d}(\cos\theta)$	`\dv{x}`	$\dfrac{\mathrm{d}}{\mathrm{d}x}$
`\dv{y}{x}`	$\dfrac{\mathrm{d}y}{\mathrm{d}x}$	`\dv[n]{y}{x}`	$\dfrac{\mathrm{d}^n y}{\mathrm{d}x^n}$
`\pdv{x}`	$\dfrac{\partial}{\partial x}$	`\pdv{u}{x}`	$\dfrac{\partial u}{\partial x}$
`\pdv{u}{x}{y}`	$\dfrac{\partial^2 u}{\partial x \partial y}$	`\pdv[n]{u}{x}`	$\dfrac{\partial^n u}{\partial x^n}$
`\var{F[g(x)]}`	$\delta F[g(x)]$	`\var(E-TS)`	$\delta(E - TS)$
`\fdv{g}`	$\dfrac{\delta}{\delta g}$	`\fdv{F}{g}`	$\dfrac{\delta F}{\delta g}$
`\fdv{V}(E-TS)`	$\dfrac{\delta}{\delta V}(E - TS)$		

Em particular, as funcionalidades do pacote nos permitem escrever expressões envolvendo Cálculo com uma tipografia muito mais limpa. Por exemplo, o comando `\dd` nos permite escrever a Eq. (1) na página 14 como

$$F(s) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(t) e^{-ist} \, \mathrm{d}t \,, \tag{7}$$

onde agora o diferencial $\mathrm{d}t$ é claramente distinto de um produto de variáveis dt.

De modo semelhante, o poder do pacote para a escrita sem dificuldades de expressões envolvendo vetores e cálculo vetorial nos permite escrever

```
\[\rho\qty(\pdv{\vb{v}}{t} + \qty(\vb{v} \vdot \grad) \vb{v}) + \grad p -
↪  \eta \laplacian \vb{v} - \qty(\zeta + \frac{1}{3}
↪  \eta)\grad(\div{\vb{v}}) = \vb{f},\]
```

para obter

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\boldsymbol{\nabla})\mathbf{v}\right) + \boldsymbol{\nabla} p - \eta\nabla^2\mathbf{v} - \left(\zeta + \frac{1}{3}\eta\right)\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\mathbf{v}) = \mathbf{f},$$

que faz uso de diversas funcionalidades do pacote para obter uma expressão fácil de digitar, mas com excelente tipografia.

Em particular, utilizamos o comando `\qty`, que permite obter delimitares de tamanho automático. Ao escrever `\qty(algo)`, os parênteses se ajustaram automaticamente ao tamanho deste `algo`, o que não acontece usualmente no LaTeX. O comando também funciona com colchetes e chaves.

O pacote disponibiliza outras funcionalidades interessantes, como alguns upgrades na escrita de matrizes. Tem-se comandos simples para escrever matrizes identidade e nula de ordem n, matrizes de Pauli, matrizes diagonais, entre outros. Para mais informações sobre isso, e sobre o pacote `physics` no geral, consulte [13].

10 Criação de Comandos

Por fim, vamos discutir como criar novos comandos no LaTeX. Isto é particularmente prático para nos apegarmos mais à filosofia *WYSIWYM* e mesmo adicionar praticidade à digitação.

O LaTeX fornece o comando `\newcommand`, que recebe dois argumentos obrigatórios (nome e efeito do comando) e dois opcionais (número de argumentos e um valor padrão, que será usado caso se queira colocar um argumento opcional no comando*). Comandos criados devem ser inseridos no preâmbulo do documento.

Para exemplificar o processo, criaremos um comando que insira no PDF uma linha para assinatura, com o nome do signatário abaixo, alinhada à direita da página.

O nome do signatário é um argumento que deve ser recebido pelo comando, e portanto é um argumento. No caso, o único argumento. Deste modo, a estrutura geral que buscamos é

```
\newcommand{\signhere}[1]{INSTRUÇÕES}
```

Queremos que a linha de assinatura esteja alinhada à direita. Assim, utiizaremos o ambiente `flushright`, que alinha o seu conteúdo à direita. Temos

```
\newcommand{\signhere}[1]{%
    \begin{flushright}%

    \end{flushright}%
}
```

*Não é possível, sem o uso de pacotes, definir comandos que recebam mais de um argumento opcional

Note que expandimos o último argumento do comando `\newcommand` em várias linhas. Não há problema, dado que as chaves delimita o início e fim do argumento. Desde modo, temos mais conveniência ao escrever.

Perceba ainda o uso de comentários (%) ao fim de cada linha de código. Eles se prestam a evitar espaços indesejados na definição do comando.

A seguir, gostaríamos que o nome do signatário esteja centralizado sob a linha de assinatura. Para obter isso, utilizaremos o ambiente `minipage`, que se comporta como uma mini-página do LaTeX. Dentro deste ambiente, usamos o comando `\centering` para que a linha de assinatura e o nome do signatário estejam centralizado em relação ao `minipage`, que por sua vez está alinhado à direita. Temos então

```
\newcommand{\signhere}[1]{%
    \begin{flushright}%
        \begin{minipage}{0.5\textwidth}%
            \centering%

        \end{minipage}%
    \end{flushright}%
}
```

O argumento obrigatório que fornecemos ao `minipage`, `0.5\textwidth`, informa o tamanho do `minipage` sendo criado. No caso, estamos criando um `minipage` com a metade da largura do bloco de texto.

Por fim, devemos adicionar a linha de assinatura e o nome do signatário. Para a linha de assinatura, utilizaremos o comando `\rule`. No caso que nos interessa, ele receberá dois argumentos obrigatórios: o comprimento da reta que desenhará e sua espessura. Como comprimento, usaremos `\textwidth`. A princípio, isto pode soar estranho, dado que o `minipage` tem `0.5\textwidth` de largura, mas como o argumento `\rule` se encontrará dentro do `minipage`, ele usará o valor de `\textwidth` do interior do `minipage`. Como espessura da reta desenhada, usaremos `0.4pt`.

Queremos evitar que o LaTeX indente a linha com a reta, e portando garantimos a ausência de indentação usando o comando `\noindent`. Para assegurar que o parágrafo da reta seja encerrado e um novo se inicie onde é escrito o nome do signatário, usamos o comando `\par`. Ele é semelhante a deixar uma linha em branco no código, mas mais conveniente de se utilizar ao definir comandos.

Por fim, adicionamos o nome do signatário, que está armazenado no argumento `#1` do comando. Adicionamos ainda um `\noindent` para assegurar que a linha não seja indentada.

O produto final é

```
\newcommand{\signhere}[1]{%
    \begin{flushright}%
        \begin{minipage}{0.5\textwidth}%
            \centering%
```

```
            \noindent\rule{\textwidth}{0.4pt}\par%
            \noindent#1%
        \end{minipage}%
    \end{flushright}%
}
```

Adicionando este trecho de código ao preâmbulo do documento, podemos utilizar o comando `\signhere{John Doe}` para obter

John Doe

Naturalmente, esta não é a única solução para obter uma linha de assinatura em LaTeX. [19, 43] possuem abordagens completamente diferentes para resolver o mesmo problema. Criação de comandos é, frequentemente, um exercício de criatividade.

11 Aprofundamentos

Deste modo concluímos esta exposição inicial ao LaTeX. Há certamente diversas outras ferramentas que podem ser utilizadas ao longo da escrita de documentos, e mencionamos aqui algumas fontes que podem ser estudadas para aprofundamento.

[18] é um pequeno "guia de sobrevivência" para a escrita de documentos longos. Em particular, ele menciona os pacotes `varioref`[28], `hyperref`[45] e `cleveref`[10], que formam uma tríade fortíssima para uma melhor administração das referências cruzadas. [8] também é uma referência valiosa e, entre outras coisas, ensina o uso do pacote `geometry`[52], que permite configurar as margens das páginas.

Sendo este um tutorial em português brasileiro, não podemos deixar de mencionar a existência da suíte `abntex2`[1], que permite a escrita de documentos em LaTeX formatados segundo os padrões da Associação Brasileira de Normas Técnicas.

Para encontrar informações sobre outros pacotes, pode-se sempre consultar [50] — a principal referência usada para pacotes ao longo deste documento.

Por fim, recomenda-se a visita aos tutoriais[40] e modelos[41] do Overleaf. Estes materiais são úteis tanto para sanar dúvidas e relembrar conceitos básicos como para aprender a extrair o máximo possível da incrível ferramenta que é o LaTeX.

Referências

1. Abntex2 team. *abntex2 — Typeset technical and scientific Brazilian documents based on ABNT rules* https://ctan.org/pkg/abntex2?lang=en.
2. American Mathematical Society. *amsfonts — TeX fonts from the American Mathematical Society* https://ctan.org/pkg/amsfonts?lang=en.
3. Bersch, C. *pst-optexp — Drawing optical experimental setups* https://ctan.org/pkg/pst-optexp?lang=en.

4. BWW (https://tex.stackexchange.com/users/511/bww). *Which graphics formats can be included in documents processed by latex or pdflatex?* https://tex.stackexchange.com/questions/1072.
5. Carlisle, D., LaTeX3 Project & Rahtz, S. *graphicx — Enhanced support for graphics* https://ctan.org/pkg/graphicx?lang=en.
6. Carroll, L. *Alice's Adventures in Wonderland & Through the Looking-Glass* (Bantam, New York, 1981).
7. Cassidy, J. *Beamer Presentations: A Tutorial for Beginners* https://www.overleaf.com/learn/latex/Beamer_Presentations:_A_Tutorial_for_Beginners_(Part_1)%E2%80%94Getting_Started.
8. Cassidy, J. *How to Write a Thesis in LaTeX* https://www.overleaf.com/learn/latex/How_to_Write_a_Thesis_in_LaTeX_(Part_1):_Basic_Structure.
9. Choquet-Bruhat, Y., DeWitt-Morette, C. & Dillard-Bleick, M. *Analysis, Manifolds, and Physics* (North-Holland, Amsterdam, 1982).
10. Cubitt, T. *cleveref — Intelligent cross-referencing* https://ctan.org/pkg/cleveref?lang=en.
11. Daly, P. W. & Ogawa, A. *natbib — Flexible bibliography support* https://ctan.org/pkg/natbib?lang=en.
12. *DeTeXify* [Ferramenta para obtenção do código de símbolos arbitrários]. http://detexify.kirelabs.org/classify.html.
13. De la Barrera, S. C. *physics — Macros supporting the Mathematics of Physics* https://ctan.org/pkg/physics?lang=en.
14. doncherry (https://tex.stackexchange.com/users/4012/doncherry). *bibtex vs. biber and biblatex vs. natbib* https://tex.stackexchange.com/questions/25701.
15. Downes, M. & Beeton, B. *Short Math Guide for LaTeX* http://tug.ctan.org/info/short-math-guide/short-math-guide.pdf.
16. Evan Aad (https://tex.stackexchange.com/users/21685/evan-aad). *Is there any reason to use inputenc?* https://tex.stackexchange.com/questions/370278.
17. Happel, P. *lipsum Easy access to the Lorem Ipsum dummy text* https://ctan.org/pkg/lipsum?lang=en.
18. Helsø, M. *The TeXpert's Guide to Survival* https://www.mn.uio.no/math/studier/masterstudier/om-masteroppgaven/Maler/survivalguide.pdf.
19. ismail (https://tex.stackexchange.com/users/132970/ismail). *Creating a Signature Line* https://tex.stackexchange.com/questions/368227.
20. The LaTeX Team. *fontenc — Standard package for selecting font encodings* https://ctan.org/pkg/fontenc?lang=en.
21. The LaTeXTeam, Madsen, L., Høgholm, M., Robertson, W. & Wright, J. *mathtools — Mathematical tools to use with amsmath* https://ctan.org/pkg/mathtools?lang=en.

22. The LaTeX Team, Mittelbach, F. & Jeffrey, A. *inputenc — Accept different input encodings* https://ctan.org/pkg/inputenc?lang=en.

23. LaTeX3 Project & American Mathematical Society. *amsmath — AMS mathematical facilities for LaTeX* https://ctan.org/pkg/amsmath?lang=en.

24. Lehman, P., Wright, J., Boruvka, A. & Kime, P. *BibLaTeX Sophisticated Bibliographies in LaTeX* https://ctan.org/pkg/biblatex?lang=en.

25. Marco Daniel (https://tex.stackexchange.com/users/5239/marco-daniel). *How to influence the position of float environments like figure and table in LaTeX?* https://tex.stackexchange.com/questions/39017.

26. Mark Meckes (https://tex.stackexchange.com/users/206/mark-meckes). *Are \(and \) preferable to dollar signs for math mode?* https://tex.stackexchange.com/questions/510.

27. Mauro, D. *Topics in Koopman–von Neumann Theory* tese de dout. (Università degli Studi di Trieste, 2002). arXiv: quant-ph/0301172.

28. Mittelbach, F., The LaTeX3 Project *et al. varioref — Intelligent page references* https://ctan.org/pkg/varioref?lang=en.

29. Murrish, R. *LaTeX is More Powerful than you Think - Computing the Fibonacci Numbers and Turing Completeness* https://www.overleaf.com/learn/latex/Articles/LaTeX_is_More_Powerful_than_you_Think_-_Computing_the_Fibonacci_Numbers_and_Turing_Completeness.

30. Oetiker, T., Partl, H., Hyna, I. & Schlegl, E. *The Not So Short Introduction to LaTeX 2ε* https://tobi.oetiker.ch/lshort/lshort.pdf.

31. Otto, J. *et al. OttoBib* https://www.ottobib.com/.

32. *Overleaf* [Editor de LaTeX online]. https://www.overleaf.com/.

33. Overleaf Team. *Biblatex bibliography styles* https://pt.overleaf.com/learn/latex/Biblatex_bibliography_styles.

34. Overleaf Team. *Biblatex citation styles* https://pt.overleaf.com/learn/latex/Biblatex_citation_styles.

35. Overleaf Team. *Bibliography management in LaTeX* https://pt.overleaf.com/learn/latex/Bibliography_management_in_LaTeX.

36. Overleaf Team. *Bibliography management with bibtex* https://pt.overleaf.com/learn/latex/Bibliography_management_with_bibtex.

37. Overleaf Team. *Bibliography management with natbib* https://pt.overleaf.com/learn/latex/Bibliography_management_with_natbib.

38. Overleaf Team. *Chess notation* https://www.overleaf.com/learn/latex/Chess_notation.

39. Overleaf Team. *CircuiTikZ package* https://www.overleaf.com/learn/latex/CircuiTikz_package.

40. Overleaf Team. *Documentation* https://www.overleaf.com/learn.

41. Overleaf Team. *Templates* https://www.overleaf.com/latex/templates.

42. Pakin, S. *The Comprehensive LaTeXSymbol List Symbols accessible from LaTeX* https://ctan.org/pkg/comprehensive/?lang=en.

43. Pat McDaniel (https://tex.stackexchange.com/users/12074/pat-mcdaniel). *Signature/date line with fixed width* https://tex.stackexchange.com/questions/48152.

44. Paterson, A. R. *A First Course in Fluid Dynamics* (Cambridge University Press, Cambridge, 1983).

45. Rahtz, S., Oberdiek, H., Oberdiek Package Support Group, The LaTeX3 Project *et al. hyperref — Extensive support for hypertext in LaTeX* https://ctan.org/pkg/hyperref?lang=en.

46. Rein, H., Sultanik, E., Randez, L. & Robson, A. *LaTeXCoffee Stains* https://www.overleaf.com/latex/examples/latex-coffee-stains/qsjjwwsrmwnc.

47. Sophie Alpert (https://tex.stackexchange.com/users/9/sophie-alpert). *Why is \[... \] preferable to $$ ... $$?* https://tex.stackexchange.com/questions/503.

48. *TeXStackExchange* [Centro de perguntas e respostas acerca de TeX e LaTeX]. https://tex.stackexchange.com/.

49. Tantau, T., Mileti, V., Wright, J. & Stuart, L. *beamer — A LaTeXclass for producing presentations and slides* https://ctan.org/pkg/beamer?lang=en.

50. *The Comprehensive TeX Archive Network* https://ctan.org/.

51. Trancanelli, D. Grandezas Físicas e Análise Dimensional: da Mecânica à Gravidade Quântica. *Revista Brasileira do Ensino de Física* **38.** ISSN: 1806-1117 (2016).

52. Umeki, H. & Carlisle, D. *geometry — Flexible and complete interface to document dimensions* https://ctan.org/pkg/geometry?lang=en.

53. Wilson, P. R. *sudokubundle — A set of sudoku-related packages* https://ctan.org/pkg/sudokubundle.

54. Wilson, P. R. & Madsen, L. *memoir — Typeset fiction, non-fiction and mathematical books* https://ctan.org/pkg/memoir?lang=en.

55. xport (https://tex.stackexchange.com/users/2099/xport). *What is the difference between font encoding and input encoding?* https://tex.stackexchange.com/questions/6448.