DOI: http://dx.doi.org/10.5007/2175-7941.2016v33n2p667

Física Moderna e técnicas computacionais: como "ver" o átomo de Hidrogênio[+*]

Nelson Canzian da Silva[1]
Departamento de Física – Universidade Federal de Santa Catarina
Florianópolis – SC

Resumo

Este artigo busca construir uma ligação entre dois temas que têm recebido cada vez mais atenção nos anos finais da educação básica e nos anos iniciais do ensino superior nas áreas de ciência e tecnologia: a introdução à física moderna e a introdução a técnicas básicas de programação. Como pretexto, o artigo descreve como gerar figuras que representam, através de um pontilhado ou uma escala de cores, as densidades de probabilidade para o elétron em um átomo de hidrogênio. Isso é feito apresentando brevemente a física do problema de base e comentando algoritmos que (a) implementam as equações do problema, (b) permitem gerar valores aleatórios a partir de distribuições de probabilidades arbitrárias e (c) efetivamente produzem as figuras. O objetivo fundamental é mostrar ao leitor como técnicas básicas de modelagem e visualização de dados são efetivamente implementadas utilizando um tema que em princípio é de seu interesse. A implementação é feita em HTML/CSS/JavaScript, uma tecnologia extremamente versátil e acessível que constitui a base de todo o conteúdo na internet hoje, incluindo muitos ambientes educacionais virtuais.

Palavras-chave: *Física moderna; Átomo de hidrogênio; Modelagem computacional; Tratamento e visualização de informação.*

[+] Modern physics and computational techniques: how to "see" the hydrogen atom

[*] *Recebido: julho de 2015.*
Aceito: março de 2016.

[1] E-mail: nelson.canzian@ufsc.br

Abstract

This paper aims to bridge two subjects that have received increased attention in the final years of the basic education and the first years of undergraduate education in science and technology areas: a bridge between an introduction to modern physics and an introduction to computer programming. As an example, the paper describes how to produce figures that represent, as a point density diagram or a color-coded diagram, the probability density for an electron in the hydrogen atom. This is accomplished through a short introduction to the basic physics of the problem and through detailed comments on algorithms that (a) implement the equations of the problem, (b) allow to generate random values from an arbitrary probability distribution and (c) effectively produce the figures. The main goal is to show the reader how modeling and visualization techniques are implemented using a subject that, in principle, is of her/his interest. The implementation uses HTML/CSS/JavaScript, an extremely versatile and accessible technology that underlays almost all internet content nowadays, including many education platforms.

Keywords: *Modern Physics; Hydrogen atom; Computer models; Data treatment and visualization*

I. Introdução

Possivelmente as imagens mais emblemáticas associadas à Física Moderna e Contemporânea vistas por alunos de Ensino Médio e Ensino Superior são as que tentam representar as possíveis configurações eletrônicas de um elétron em um átomo de hidrogênio. Para além da apreciação de sua beleza estética, a sua real compreensão requer um elevado grau de abstração a respeito das relações entre as grandezas cinemáticas do elétron (posição, velocidade) e distribuições de probabilidades. A fim de proporcionar imediata clareza sobre o objeto deste trabalho, a Fig. 1 mostra duas representações dessas distribuições, produzidas da maneira que descreveremos a seguir.

Após décadas de preocupações quanto à inserção de conteúdos de física moderna no Ensino Médio (OSTERMANN e MOREIRA, 2000), a introdução desse tema e, portanto, a apresentação e discussão de imagens e conceitos como os discutidos aqui, ainda hoje é precária e fragmentada (DOMINGUINI, 2012). Imagens mais complexas são frequentemente encontradas em livros de química quando abordam o tema dos orbitais. Via de regra os textos optam por figuras pontilhadas que remetem, ainda que imprecisamente, à distribuição eletrônica. ROZENTALSKI (2013), em sua dissertação de mestrado em ensino de ciências pela

USP, dedica um capítulo inteiro a uma análise da representação icônica dos orbitais feita em diversos livros didáticos. Em um livro do pioneiro Linus Pauling (PAULING, 1966), por exemplo, as figuras são artisticamente tão bem elaboradas que o ilustrador (Roger Hayward) é identificado com um destaque não mais encontrado nos livros de hoje.

No Ensino Superior, essas imagens são encontrados nos livros didáticos (EISBERG; RESNICK, 1986; BLATT, 1992; TIPLER; LLEVELLYN, 2001) de disciplinas tipicamente identificados como "Física Moderna" ou "Estrutura da Matéria", que os alunos dos cursos de física e algumas engenharias cursam após o típico ciclo básico de quatro semestres. São imagens também abundantemente encontradas na internet (NAVE, 2012; WIKIMEDIA, 2015).

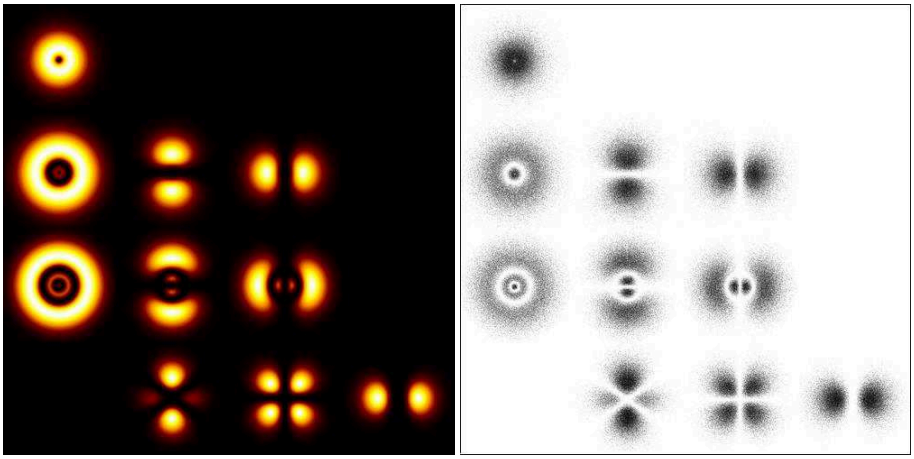

Fig. 1 – Distribuições de probabilidades de se encontrar um elétron em torno do núcleo de um átomo de hidrogênio. À esquerda, um mapa de cores (conhecido como "dark body gradiente") está associado à distribuição de probabilidade. À direita, representação que utiliza uma densidade de pontos proporcional à densidade de probabilidade. Na primeira linha, o estado fundamental (n = 1), camada K). Na segunda linha, os estados degenerados para o primeiro estado excitado (n = 2, camada L); na terceira e quarta linhas, os estados degenerados do segundo estado excitado (n - 3, camada K).

II. Fundamentação técnica

Soluções da equação de Schroedinger

As ilustrações da Fig. 1 são tentativas de representar as funções (ou variações delas) resultantes da solução da equação de Schroedinger para um elétron sujeito ao potencial coulombiano do núcleo. Detalhes da solução e das equações podem ser encontrados em todos os livros didáticos para o Ensino Superior citados anteriormente. Para a discussão que segue, entretanto, é importante recuperar alguns aspectos:

1. No tratamento desse problema são utilizadas coordenadas esféricas, em que um ponto no espaço, em vez de ser representado pelas usuais coordenadas cartesianas (*x*,*y*,*z*), é representado por outra trinca de números (r, θ, ϕ), onde *r* é a distância à origem, θ o ângulo com relação ao eixo vertical e ϕ um ângulo com relação ao plano *xz*.

2. Nesse sistema, a solução da equação de Schroedinger, $\Psi(r, \theta, \phi)$, pode ser representada pelo produto de três funções, que dependem exclusivamente de apenas uma das coordenadas. Isto é, $\Psi(r, \theta, \phi) = R(r)\Theta(\theta)\Phi(\phi)$.

3. A probabilidade de se encontrar o elétron em algum ponto entre (r, θ, ϕ) e ($r + dr, \theta + d\theta, \phi + d\phi$) é proporcional ao módulo quadrado da função de onda ($\Psi^*\Psi$) multiplicado pelo elemento de volume dV em coordenadas esféricas ($dV = r^2 \sin\theta dr d\theta d\phi$), de modo que $P(r, \theta, \phi)dV = \Psi^*\Psi r^2 \sin\theta dr d\theta d\phi$ (veja referências citadas anteriormente sobre a definição do elemento de volume).

No que segue serão utilizadas algumas técnicas matemáticas e de programação de computadores, além de algumas opções arbitrárias, para produzir diferentes visualizações de uma dessas soluções em particular.

Para concretizar minimamente o problema, será tratada extensivamente a solução com número quântico principal $n = 3$ (segundo estado excitado, camada *M*), número quântico orbital (associado ao momento angular) ℓ (orbital *p*), número quântico magnético (projeção do momento angular) $m_\ell = 0$. Para este estado, as soluções da equação de Schroedinger para as três variáveis são dadas por:

$$
\begin{aligned}
R_{31}(r) &= \frac{4\sqrt{2}}{3}\left(\frac{Z}{3a_0}\right)^{3/2}\left(\frac{Zr}{a_0}\right)\left(1 - \frac{Zr}{6a_0}\right)e^{-Zr/3a_0} \\
\Theta_{10}(\theta) &= \sqrt{\frac{3}{4\pi}}\cos\theta \\
\Phi_0(\phi) &= 1
\end{aligned}
$$

sendo *z* o número atômico do núcleo e a_0 o raio de Bohr.

Ferramentas computacionais

Neste trabalho são empregadas as principais tecnologias utilizadas na produção de páginas para a internet: HTML (*Hypertext Markup Language*) (W3C, 2014), CSS (*Cascade Style Sheets*) (W3C, 2011) e JavaScript (ECMA, 2014).

Um dos principais motivos para a escolha dessas tecnologias deve-se ao fato de estarem disponíveis e operacionais em qualquer computador que tenha um navegador instalado. Para utilizá-las não é preciso baixar, comprar, instalar e compilar nada. Basta abrir um editor de textos qualquer, digitar seu texto de maneira apropriada, salvar como texto sem formatação e com a extensão ".html", e carregar no navegador como uma página qualquer. O que for produzido dessa maneira vai refazer os cálculos, reproduzir os gráficos e gerar as visualizações em qualquer computador ou dispositivo móvel do planeta, não importando em que sistema

operacional esteja sendo executado (Windows, MacOS, Linux, Android etc), desde que tenha instalado um navegador razoavelmente padronizado (Firefox, Chrome, Opera etc).

Outro bom motivo é o fato dessas tecnologias funcionarem de maneira totalmente integrada dentro de um único documento HTML. Com isso, é possível atribuir ao documento características dinâmicas tanto quando é carregado pelo navegador quanto posteriormente, através de ações do usuário (clicar botões, deslizar controles) ou de temporizadores programáveis. Para completar, essas tecnologias podem ser utilizadas para produzir desenhos e gráficos (estáticos, dinâmicos e interativos) de altíssima qualidade.

Provavelmente a parte mais difícil está relacionada à compreensão dos comandos, das regras de sintaxe e da lógica de programação necessária para utilizar o JavaScript. Àqueles interessados em desenvolver essas habilidades, recomendamos a utilização de plataformas de autoinstrução, como por exemplo a Codecademy (SIMA e BUBINSKI, 2015), a consulta constante a materiais de referência e repositórios de exemplos (REFSNES, 2015) e a textos que sistematizam sua aplicação no contexto do ensino de física (SILVA, 2016).

III. Cálculo e visualização de funções de uma variável

O primeiro passo para a produção de tabelas, gráficos ou outra forma de visualização de informações quantitativas é a geração dos números associados aos valores possíveis das variáveis da função e da função propriamente dita. O pequeno trecho de código a seguir mostra como fazer isso para a função radial $R_{31}(r)$ apresentada na introdução. Para visualizar os resultados, é preciso copiá-lo em um editor de textos, salvá-lo e visualizá-lo no navegador. Em um editor de textos básico, como o "Bloco de Notas" ou o "Gedit", o documento deve ser simplesmente salvo como ".html"; em um processador de textos como o Word ou o LibreOffice, é preciso certificar-se que o texto seja salvo como "texto sem formatação" e com a extensão ".html".

Listagem 1 - Script para gerar valores de r e $R_{31}(r)$.

```
<script>
var Z = 1;      // carga do núcleo
var a0 = 1;     // raio de Bohr
function R31(r) {
    //calcula a função de onda radial para n=3 e l= 1
    return 4*Math.sqrt(2)/3*Math.pow((Z/3/a0),(3/2))*
           (Z*r/a0)*(1-Z*r/6/a0)*Math.exp(-Z*r/3/a0);
}
// imprime resultados com duas casas decimais
for (var r=0;r<=20;r=r+0.5) {
   document.write(r.toFixed(2) + "," + R31(r).toFixed(2) + "<br>");
}
</script>
```

Ao ser salvo, o texto acima constitui um documento que contém exclusivamente instruções em JavaScript (delimitadas pelos rótulos `<script>...</script>`). Mais

informações sobre como definir variáveis (`var...`), funções (`function...`) e seus valores de retorno (`return...`), bem como sobre as funções matemáticas `Math.sqrt(x)` (raiz quadrada de x), `Math.pow(x,a)` (x elevado a a) e `Math.exp(x)` (e elevado a x) e muitas outras podem ser obtidas em sites de referência sobre JavaScript (REFSNES, 2015). Em resumo, para obter os valores foi necessário (a) definir variáveis; (b) definir a função e (c) utilizar uma estrutura de repetição para calcular e imprimir o raio e o valor da função com uma formatação específica.

O resultado da execução do script é uma lista com quarenta pares de valores r, $R_{31}(r)$ = { (0.00,0.00), (0.50,0.14), ..., (19.50,-0.02), (20.00,-0.02) }, impressos com duas casas decimais (`.toFixed(2)`), separados por uma quebra de linha (`<br>`).

Gráficos cartesianos

O próximo passo óbvio é fazer um gráfico dessa informação, isto é, de R(r) × r. Para tanto, é preciso avançar um pouco mais na integração entre HTML, CSS e JavaScript e entender como incluir e utilizar bibliotecas de terceiros no documento, como ilustrado a seguir.

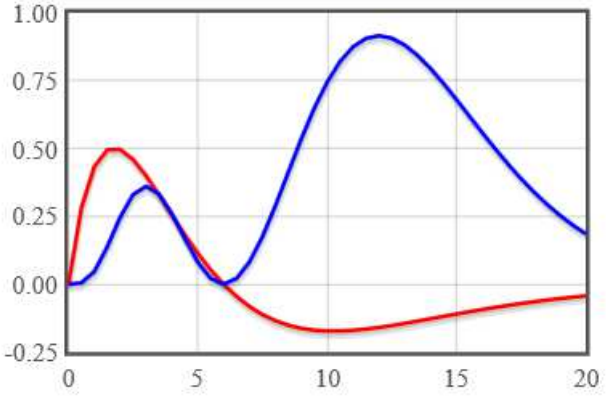

Fig. 2 – Gráfico da parte radial função de onda $R_{31}(r)$ (em vermelho) e da densidade de probabilidade radial $r^2|R_{31}(r)|^2$ (em azul), em função do raio. Ao gráfico da função de onda foi aplicado um fator 2 para melhorar a visualização.

Listagem 2 - Documento utilizado para gerar o gráfico da Fig. 2.

```
<html>

<script src="file://C:/flot/jquery.js"></script>
<script src="file://C:/flot/jquery.flot.js"></script>

<body>

<div id='grR31' style='width:300;height:200;margin:auto'></div>

<script>
var Z = 1;
var a0 = 1;
function R31(r) {
    return 4*Math.sqrt(2)/3*Math.pow((Z/3/a0),(3/2))*
           (Z*r/a0)*(1-Z*r/6/a0)*Math.exp(-Z*r/3/a0);
```

```
}
var ptsR31 = [];
var ptsR31sqr = [];
for (var r=0;r<=20;r=r+0.5) {
   ptsR31.push([r,2*R31(r)]);
   ptsR31sqr.push([r,r*r*R31(r)*R31(r)]);
}
$.plot("#grR31",[ptsR31,ptsR31sqr],{ colors:["red","blue"] });
</script>

</body>
</html>
```

O bloco acima começa e termina (`<html>...</html>`) com a especificação do trecho que deve ser tratado como um documento HTML (existem outros tipos de documento lidos pelos navegadores, como XML, PHP etc., que não abordaremos neste artigo).

A seguir, o documento contém declarações (`<script src=...>...</script>`) que incluem bibliotecas no código. No caso, incluímos dois arquivos de uma biblioteca gráfica relativamente simples (LAURSEN e SCHNUR, 2015) previamente instalada em uma pasta na raiz do sistema de arquivos de um computador rodando Windows. Essa especificidade pode ser contornada utilizando-se um link direto para o site do projeto na internet ou adaptada para outros sistemas operacionais. Existem muitas outras bibliotecas disponíveis (BOSTOK, 2015; HIGHSOFT, 2015), com foco nas mais diversas finalidades (engenharias, dados demográficos, mercado financeiro etc.). Devido à elevada padronização das estruturas de dados utilizadas no JavaScript (ECMA, 2013) é relativamente fácil migrar de uma para outra conforme a necessidade.

Em seguida há um bloco que especifica o início e o fim do corpo do documento (`<body>...</body>`), que é onde as informações serão efetivamente apresentadas. O corpo do documento tem apenas dois itens: (a) uma *divisão* (`<div>...</div>`) vazia que funciona como um *container* que receberá o gráfico da função, com dois parâmetros que definem sua largura (`width`) e sua altura (`height`), em pixels; e (b) um *script* que, além de incluir a declaração de variáveis e da função feitas anteriormente, declara a variável que armazenará os pontos do gráfico (`var ptsR31 = []`), a estrutura de repetição que preenche essa variável com os pontos (`for ...`) e o comando que evoca a função da biblioteca gráfica que efetivamente faz o gráfico (`$.plot(...)`). No exemplo, essa função recebe dois parâmetros, a identidade da divisão que vai receber o gráfico (`graficoR31`) e a matriz com os pontos a serem visualizados (`[ptsR31]`). A função pode receber parâmetros adicionais, informando opções de desenho (tipo de gráfico, limites das escalas, cores, fundo, legenda etc.) mais informações e exemplos podem ser encontrados na documentação sobre a biblioteca (LAURSEN; SCHNUR, 2015).

De maneira totalmente análoga, podemos construir o gráfico da função angular $\Theta(\theta)$, apenas substituindo `R31` por `T01` em todos os pontos do documento e trocando o valor de retorno da função por `return Math.sqrt(3/4/Math.PI) * Math.cos(x).`

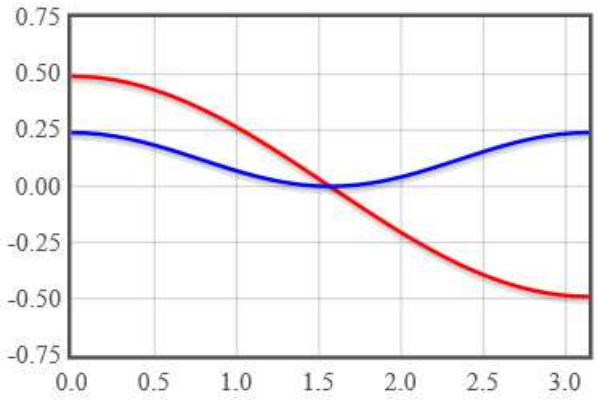

Fig. 3 – Gráfico da parte angular (polar) da função de onda $\Theta_{10}(\theta)$ (em vermelho) e da densidade de probabilidade angular $|\Theta_{10}(\theta)|^2$ (em azul) em função do ângulo em radianos.

Listagem 3 - Documento utilizado para gerar o gráfico da Fig. 3.

```
<html>

<script src="file://C:/flot/jquery.js"></script>
<script src="file://C:/flot/jquery.flot.js"></script>

<body>

<div id='grT10' style='width:300;height:200;margin:auto'></div>

<script>
function T10(x) {
    return Math.sqrt(3/4/Math.PI)*Math.cos(x);
}
var ptsT10 = [];
var ptsT10sqr = [];
for (var theta=0;theta<=Math.PI;theta=theta+Math.PI/36) {
   ptsT10.push([theta,T10(theta)]);
   ptsT10sqr.push([theta,T10(theta)*T10(theta)]);
}
$.plot("#grT10",[ptsT10,ptsT10sqr],{ colors:["red","blue"] });
</script>

</body>
</html>
```

Gráficos polares

Uma forma eventualmente mais adequada de visualizar essa informação é através de um gráfico polar. Para ilustrar como utilizar as capacidades de desenho do JavaScript, o exemplo a seguir mostra como fazê-lo sem utilizar bibliotecas gráficas, que em geral dispõem de métodos que facilitam a produção de figuras desse tipo.

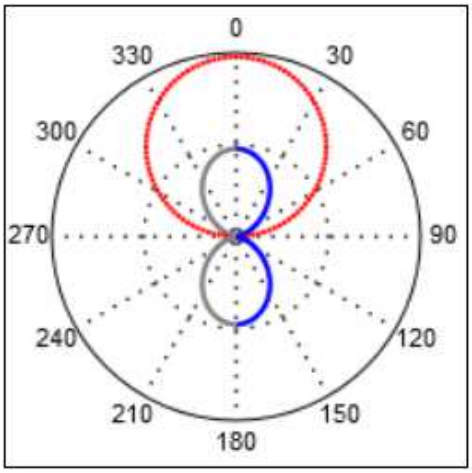

Fig. 4 – Gráfico polar da função de onda $\Theta_{10}(\theta)$ (em vermelho) e da densidade de probabilidade angular $|\Theta_{10}(\theta)|^2$ (em azul). Para maior clareza sobre a consequência dessa distribuição no espaço, a densidade de probabilidade angular foi completada para θ de π a 2π (em cinza).

Listagem 4 - Documento utilizado para gerar o gráfico da Fig. 4.

```
<html>
<body>

<canvas id="cnvPolar" width="200" height="200"
    style="border:1px solid black"></canvas>

<script>

function T10(x) {
    return Math.sqrt(3/4/Math.PI)*Math.cos(x);
}

// captura do contexto gráfico e transformação de coordenadas
var cnv = document.getElementById("cnvPolar");
var ctx = cnv.getContext("2d");
var cnvW = cnv.width;
var cnvH = cnv.height;
var grfW = 1.25;
var grfH = 1.25;
ctx.transform(cnvW/grfW,0,0,cnvH/grfH,cnvW/2,cnvH/2);
ctx.lineWidth = grfW/cnvW;

// desenho do círculo externo
ctx.beginPath();
ctx.strokeStyle = "black";
ctx.arc(0,0,0.5,0,Math.PI*2,0);
ctx.stroke();

// desenho das linhas pontilhadas radiais
for (var ang=0;ang<2*Math.PI;ang+=Math.PI/6) {
    for (var i=1;i<25;i+=2) {
        var r = 0.5*i/25;
        var x = r*Math.cos(ang);
```

```
        var y = r*Math.sin(ang);
        ctx.beginPath();
        ctx.fillStyle = "black";
        ctx.arc(x,y,0.005,2*Math.PI,0);
        ctx.fill();
    }
}

// desenho da linha pontilhada circular
var da = 2*Math.PI/36;
var r = 0.5/2;
for (var j=0;j<36;j+=1) {
    var ang = j*da;
    var x = r*Math.cos(ang);
    var y = r*Math.sin(ang);
    ctx.beginPath();
    ctx.strokeStyle = "black";
    ctx.arc(x,y,0.005,0,2*Math.PI,0);
    ctx.fill();
}

// texto com os angulos
ctx.save();
ctx.scale(grfW/cnvW,grfH/cnvH);
ctx.textAlign = "center";
ctx.textBaseline = "middle";
ctx.font = "11px sans-serif";
ctx.fillStyle = "black";
for (var t=-Math.PI/2;t<(2*Math.PI-Math.PI/2)*0.9;t+=Math.PI/6) {
    deg = (t*180/Math.PI+90).toFixed(0);
    var x = 200/2*0.9*Math.cos(t);
    var y = 200/2*0.9*Math.sin(t);
    ctx.beginPath();
    ctx.fillText(deg,x,y);
}
ctx.restore();

// desenho das funções proprimente ditas
ctx.rotate(-Math.PI/2);
var N = 100;
var da = Math.PI/N;
for (var i=0;i<N;i++) {
    var a = i*da;
    // função
    var f = T10(a);
    var x = f*Math.cos(a);
    var y = f*Math.sin(a);
    ctx.beginPath();
    ctx.fillStyle = "red";
    ctx.arc(x,y,0.0075,0,2*Math.PI,0);
    ctx.fill();
    // quadrado da função
    var f = T10(a)*T10(a);
    var x = f*Math.cos(a);
    var y = f*Math.sin(a);
    ctx.beginPath();
    ctx.fillStyle = "blue";
    ctx.arc(x,y,0.0075,0,2*Math.PI,0);
    ctx.fill();
```

```
    // complemento do quadrado da função
    ctx.beginPath();
    ctx.fillStyle = "gray";
    ctx.arc(x,-y,0.0075,0,2*Math.PI,0);
    ctx.fill();
}
</script>

</body>
</html>
```

O documento descrito na listagem 4 começa com a declaração de que trata-se de um documento HTML, seguida pela declaração do corpo do documento. O corpo do documento tem apenas dois elementos: um canvas e um script.

O canvas proporciona o *contexto* gráfico onde o JavaScript pode, através de instruções específicas, fazer desenhos. Nesse exemplo, o canvas tem quatro parâmetros: `id`, ao qual é atribuída a sua identidade única, `width`, que especifica sua largura em pixels, `height`, que especifica sua altura em pixels e `style`, que lhe atribui uma borda contínua com 1 pixel de espessura (ao parâmetro `style` podem ser atribuídas várias declarações de estilo definidas no CSS, separadas por ponto-e-vírgula).

O script contém seis trechos distintos, cada um com uma finalidade, identificadas com comentários (as linhas iniciadas com `//`), descritas em detalhes a seguir.

Captura do contexto gráfico

O JavaScript precisa saber onde deve desenhar, isto é, precisa do "endereço", ou, como é conhecido no jargão computacional, da *referência* ao elemento onde deve atuar. Esse conhecimento é obtido primeiro atribuindo à variável `cnv` uma referência ao objeto que tem o `id="cnvPolar"`. Isso é feito com o método `getElementById()` (um método extremamente útil para a manipulação dos dados de todo o documento). A partir da referência a esse objeto, é possível acessar seus métodos (funções) e obter as suas propriedades, tais como o contexto gráfico (`cnv.getContext("2d")`), a sua largura (`cnv.width`) e a sua altura (`cnv.height`).

Essas informações serão utilizadas para adequar o sistema de coordenadas do desenho ao sistema de coordenadas do canvas. Nesse problema em particular, o canvas tem 200 × 200 pixels e a origem (0,0) está, por padrão, no canto superior esquerdo. Por conveniência, é mais interessante que o espaço do desenho tenha dimensões de 1,25 × 1,25 unidades arbitrárias e que a origem esteja no centro. Essa transformação de coordenadas é feita utilizando-se o método `transform(...)`, que por sua vez utiliza algumas variáveis auxiliares (`grfW` e `grfH`).

Transformações de coordenadas

Uma transformação de coordenadas T leva um ponto $X = (x, y)$ no sistema de coordenadas específico do problema a ser tratado a um ponto $X' = (x', y')$ no sistema de

coordenadas intrínseco do canvas. O método `transform(m11, m12, m21, m22, dx, dy)` recebe seis parâmetros, interpretados como os elementos de uma matriz:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & dx \\ m_{21} & m_{22} & dy \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

ou, equivalentemente, como coeficientes de duas equações lineares:

$$\begin{aligned} x' &= m_{11}x + m_{12}y + dx \\ y' &= m_{21}x + m_{21}y + dy \end{aligned}$$

Por exemplo, uma translação do sistema de coordenadas, de modo a colocar a origem no centro do canvas, seria dada por:

```
ctx.transform(1,0,0,1,cnvW/2,cnvH/2)
```

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \text{cnvW}/2 \\ 0 & 1 & \text{cnvH}/2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{aligned} x' &= 1x + 0y + dx &&= x + \text{cnvW}/2 \\ y' &= 0x + 1y + dy &&= y + \text{cnvH}/2 \end{aligned}$$

Uma transformação de escala do sistema de coordenadas, de modo que todas as coordenadas expressas no sistema do problema sejam multiplicadas por um fator S_x na direção *x* e um fator S_y na direção *y* é dada por:

```
ctx.transform(Sx,0,0,Sy,0,0)
```

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{aligned} x' &= S_x x + 0 \cdot y + 0 \cdot 1 &&= S_x x \\ y' &= 0 \cdot x + S_y y + 0 \cdot 1 &&= S_y y \end{aligned}$$

Uma rotação do sistema de coordenadas, de modo que todas as coordenadas expressas no sistema do problema sejam apresentadas rodadas de um ângulo θ no sistema do canvas é dada por:

```
ctx.transform(Math.cos(t),Math.sin(t),-Math.sin(t),Math.cos(t),0,0)
```

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta + 0 \cdot 1 &&= x\cos\theta + y\sin\theta \\ y' &= -x\sin\theta + y\cos\theta + 0 \cdot 1 &&= -x\sin\theta + y\cos\theta \end{aligned}$$

No caso do gráfico polar da listagem 4, é feita uma transformação de escala e uma translação simultaneamente. Por fim, a espessura das linhas que serão desenhadas no canvas são adaptadas à nova escala (`ctx.lineWidth`).

Desenho do círculo externo

O segundo bloco do script desenha o círculo mais externo do gráfico. A primeira instrução, `ctx.beginPath()`, informa o JavaScript que um novo bloco de instruções de desenho (um *path*, ou caminho ou trajetória da "caneta") deve ser iniciado. A instrução seguinte atribui a cor preta (*black*) à caneta que vai desenhar o contorno (`stroke`) do objeto (no caso, um círculo). A instrução `arc(x,y,r,ti,tf,s)` desenha um arco de circunferência no canvas, centrado em `x`,`y`, com raio `r` e ângulos inicial e final `ti` e `tf`. Para círculos completos, `ti = 0` e `tf = 2*Math.PI`. O último parâmetro, `s`, especifica se o arco deve ser desenhado no sentido horário (`1`) ou anti-horário (`0`), o que é irrelevante quando se desenha um círculo completo. A última instrução do bloco, `ctx.stroke()`, faz o JavaScript efetivamente desenhar no canvas o contorno do arco, que até então só estava construído na memória.

Desenho das linhas pontilhadas radiais

As linhas pontilhadas radiais são desenhadas utilizando-se dois laços `for (...)` aninhados. O primeiro, mais externo, controla o ângulo em que serão desenhados, que vai de 0 a 2π em passos de $\pi/6$. O segundo controla onde, ao longo do raio, serão desenhados os 12 pontos que representam o raio (note que a variável *i* inicia em 1, vai efetivamente até 24 e é incrementada de 2 em 2 unidades). Dentro dos laços, é calculada a distância *r* do ponto ao centro do círculo e as coordenadas polares *r* e θ são convertidas para as coordenadas cartesianas necessárias para desenhar no canvas.

Nesse bloco é utilizada a instrução `ctx.fillStyle` que atribui a cor preta à área delimitada pelo arco a ser desenhado (diferentemente da instrução `strokeStyle` no bloco anterior, que atribuía a cor ao contorno da figura). Novamente, a instrução `arc(...)` é utilizada para especificar que círculos devem ser desenhados nas coordenadas calculadas. Finalmente, é utilizada a instrução `ctx.fill()` para efetivamente fazer o preenchimento dos círculos diminutos (diferentemente de `ctx.stroke()`, que desenharia apenas o contorno).

Desenho da linha pontilhada circular

A linha pontilhada circular a meio caminho entre o centro e a borda da circunferência externa é desenhada dentro de um laço `for (...)` que calcula ângulos entre 0 e 2π em intervalos de 10 graus e as coordenadas *x,y* onde devem ser colocados os pontos a partir de um raio *r* fixo. No mais, os pontos são desenhados como no bloco anterior.

Texto com ângulos

A colocação de textos no canvas requer cuidados com relação a escalas, rotações, alinhamentos, altura de linha e espaçamento de caracteres. Uma referência bastante completa sobre como lidar com textos no canvas pode ser encontrada em MDN (2015).

Neste exemplo, é utilizada uma estratégia que introduz algumas novas instruções convenientes quando se lida com o canvas. A primeira instrução do bloco, `ctx.save()`, salva o *estado* (sistema de coordenadas, cores de contorno e preenchimento, espessura de linha etc.) do canvas naquele ponto. A instrução `ctx.restore()` no final do bloco restaura o estado salvo. Isso significa que tudo o que for feito entre esses dois pontos (transformações de coordenadas, alterações de cores e espessuras de linhas etc.) somente terá efeito entre essas duas instruções.

A segunda instrução do bloco (`ctx.scale(sx,sy)`) é uma versão simplificada de `ctx.transform(...)` que apenas aplica um fator de escala ao sistema de coordenadas (existem outras, como `ctx.rotate(ang)` e `ctx.translate(dx,dy)`, com resultados óbvios). Nesse caso, foi restaurada, temporariamente, a escala original do canvas.

As instruções `ctx.textAlign="center"` e `ctx.textBaseline="middle"` especificam que o texto deve ser desenhado centralizado horizontalmente e verticalmente, respectivamente, em torno do ponto de referência (a ser especificado posteriormente). Um laço `for (...)` define os ângulos onde os textos devem ser desenhados. Como no canvas os ângulos são especificados no sentido horário a partir do eixo horizontal, os ângulos onde os textos devem ser desenhados vão de $-\pi/2$ a $3\pi/2$ em intervalos de $\pi/6$ radianos. Dentro do laço o valor do ângulo é convertido para graus e formatado para 0 casas decimais (`.toFixed(0)`). O texto, armazenado na variável `deg`, é efetivamente desenhado no canvas pela instrução `ctx.fillText(deg,x,y)` tendo como referência o ponto *x,y*.

Desenho das funções propriamente ditas

O último bloco do script praticamente não apresenta novidades em relação aos blocos anteriores. Talvez digno de nota sejam: (a) o uso da instrução `ctx.rotate(-Math.PI/2)`, que reorienta o sistema de coordenadas de modo que a origem dos ângulos esteja no topo do canvas, e (b) o desenho do complemento do quadrado da função de onda angular, que foi adicionado para auxiliar na compreensão da distribuição espacial real das probabilidades.

IV. Cálculo e visualização de funções de duas variáveis

As figuras mostradas na introdução são visualizações do produto $|R(r)|^2 r^2 |\Theta(\theta)|^2$, isto é, do produto do gráfico da Fig. 1 com o gráfico da Fig. 3, para todos os valores de r e θ. São utilizados dois esquemas bastante distintos para representar o valor da função em cada ponto, descritos a seguir.

Gradientes de cores

Na Fig. 1, à esquerda, os valores da função são mapeados a um gradiente de cores. Gradientes de cores são tipicamente implementados através da atribuição de valores para as componentes *R* (*red*, vermelho), *G* (*green*, verde) e *B* (*blue*, azul) que definem uma cor. $R = G = B = 0$ produz preto, $R = G = B = 1$ produz branco, quaisquer outros valores de $R = G = B$ produzem tons de cinza indo do preto ao branco. Em geral os sistemas de vídeo e impressoras dividem o intervalo entre 0 e 1 em 256 níveis (para cada cor separadamente), totalizando cerca de 16 milhões de cores possíveis.

Um gradiente linear de cores pode ser especificado informando-se os seus pontos de parada (*stops*) e os valores que cada cor componente deve ter nesses pontos. Por exemplo, uma escala de tons de cinza, indo do preto ao branco linearmente pode ser especificada como:

$$\begin{aligned} S &= [0{,}00, 1{,}00] \\ R &= [0{,}00, 1{,}00] \\ G &= [0{,}00, 1{,}00] \\ B &= [0{,}00, 1{,}00] \end{aligned}$$

S contém os pontos de parada, que neste caso são os extremos dos valores possíveis da função — 0 corresponde ao valor mínimo assumindo pela função (z_{min}) e 1 ao valor máximo (z_{max}). *r*, *G* e *B* contém as intensidades que cada componente deve ter em cada ponto de parada. Nesse caso, as três componentes vão variar de 0,00 a 1,00 igualmente, produzindo tons de cinza indo do preto ($R = G = B = 0{,}00$) ao branco ($R = G = B = 1{,}00$).

O exemplo a seguir mostra como especificar um gradiente linear que vai do vermelho para $z = z_{min}$ ao azul para $z = z_{max}$, passando pelo verde no centro do intervalo. A primeira metade do intervalo contém cores resultantes da mistura do vermelho com o verde, produzidos à medida que a componente vermelha diminui de 1,00 a 0,00 e a verde aumenta de 0,00 a 1,00. A segunda metade do intervalo contém cores resultantes da mistura do verde com o azul, produzidos à medida que a componente verde diminui de 1,00 a 0,00 e a azul aumenta de 0,00 a 1,00.

$$\begin{aligned} S &= [0{,}00, 0{,}50, 1{,}00] \\ R &= [1{,}00, 0{,}00, 0{,}00] \\ G &= [0{,}00, 1{,}00, 0{,}00] \\ B &= [0{,}00, 0{,}00, 1{,}00] \end{aligned}$$

O gradiente conhecido como *dark body*, implementado na biblioteca de análise e visualização de dados do CERN (CERN, 2015) contém tons que vão do preto ao branco, passando por tons de vermelho, laranja e amarelo, e é especificado como:

$$\begin{aligned} S &= [0{,}00, 0{,}25, 0{,}50, 0{,}75, 1{,}00] \\ R &= [0{,}00, 0{,}50, 1{,}00, 1{,}00, 1{,}00] \\ G &= [0{,}00, 0{,}00, 0{,}55, 1{,}00, 1{,}00] \\ B &= [0{,}00, 0{,}00, 0{,}00, 0{,}00, 1{,}00] \end{aligned}$$

Este gradiente divide o intervalo total em 4 partes iguais. No primeiro (0,00 a 0,25), o vermelho aumenta de 0,00 a 0, 50, e não há contribuições nem do verde nem do azul. No

segundo intervalo (0,25 a 0,50), o vermelho aumenta de 0,50 a 1,00, o verde aumenta de 0,00 a 0,55, e o azul permanece em 0,00. No terceiro intervalo o vermelho permanece em 1,00, o verde aumenta de 0,55 a 1,00 e o azul permanece em 0,00. No quarto intervalo (0,75 a 1, 00), o vermelho e o verde permanecem em 1,00 e o azul aumenta de 0, 00 a 1, 00.

A Fig. 5 é uma tentativa de representar a contribuição de cada componente de cor dos três gradientes discutidos acima, bem como o espectro de cores gerado por cada um dos esquemas. Frequentemente a escolha dos gradientes de cores utilizados para a visualização de dados é vista, indevidamente, mais como uma questão de opção estética (uma questão de "gosto") do que como uma opção técnica. Àqueles que desejam aprofundar seus conhecimentos a respeito das consequências da escolha de diferentes esquemas de cores, recomendamos a leitura de Rogowitz e Couet (2015).

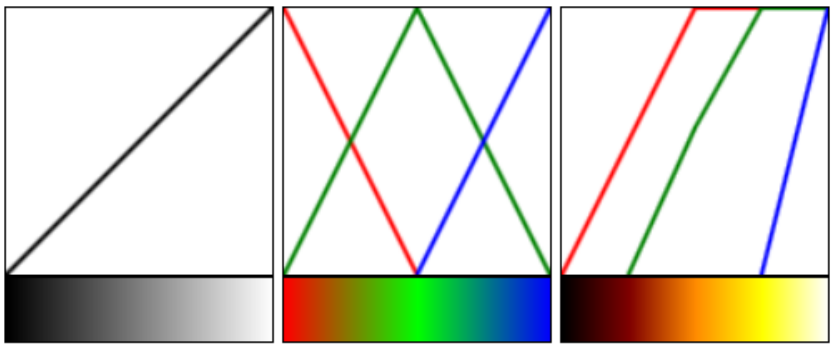

Fig. 5 – Gráficos das contribuições das componentes R (vermelha), G (verde) e B (azul) na formação de alguns gradientes de cores. À esquerda, tons de cinza, em que as contribuições das componentes são iguais. Ao centro, um gradiente do vermelho ao azul, passando pelo verde e tons intermediários. À direita, o gradiente dark body. Em cima da figura, a contribuição relativa de cada componente de cor em cada intervalo; embaixo, o espectro de cores gerado.

A biblioteca de rotinas de análise e visualização de dados do CERN, citada anteriormente, apresenta uma implementação muito eficiente para gerar e utilizar gradientes lineares de cores. A listagem 5 mostra como construir um documento que gera a Fig. 6 utilizando uma versão não tão eficiente (mas igualmente genérica e eventualmente mais didática) que implementa uma função que retorna uma cor dentro do esquema *dark body* para valores entre 0 e 1 da função a ser visualizada.

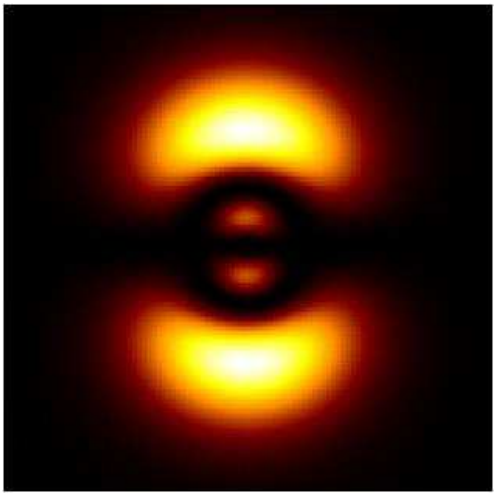

Fig. 6 – Densidade de probabilidade $r^2|R_{31}(r)|^2|\Theta_{10}(\theta)|^2$ de se encontrar o elétron do átomo de hidrogênio calculada para um plano que passa pelo centro do átomo. A extensão total da figura, tanto na vertical quanto na horizontal, corresponde a 50 raios de Bohr.

Listagem 5 - Documento utilizado para gerar a imagem da Fig. 6.

```
<html>
<body>

<canvas id="cnvDarkBody" width="250" height="250"
    style="border:1px solid black"></canvas>

<script>

function R31(r) {
    return 4*Math.sqrt(2)/3*Math.pow((Z/3/a0),(3/2))*
           (Z*r/a0)*(1-Z*r/6/a0)*Math.exp(-Z*r/3/a0);
}

function T10(x) {
    return Math.sqrt(3/4/Math.PI)*Math.cos(x);
}

// definição da função que define a cor do ponto
function darkBody(z) {

    var S = [0.00, 0.25, 0.50, 0.75, 1.00];
    var R = [0.00, 0.50, 1.00, 1.00, 1.00];
    var G = [0.00, 0.00, 0.55, 1.00, 1.00];
    var B = [0.00, 0.00, 0.00, 0.00, 1.00];

    if (z<0) r = g = b = 0;
    else if (z>1) r = g = b = 255;
    else {
        // descobre entre que "stops" está z
        var s;
        for (s=0; s<S.length-1; s++)
            if ((z>=S[s])&&(z<=S[s+1])) break;

        // interpola as componentes das cores
```

```
        r = parseInt(Math.floor(255*(R[s] +
                (R[s+1]-R[s])/(S[s+1]-S[s])*(z-S[s]))));
        g = parseInt(Math.floor(255*(G[s] +
                (G[s+1]-G[s])/(S[s+1]-S[s])*(z-S[s]))));
        b = parseInt(Math.floor(255*(B[s] +
                (B[s+1]-B[s])/(S[s+1]-S[s])*(z-S[s]))));
    }
    return "rgb("+r+","+g+","+b+")";
}

// captura do contexto gráfico e transformação de coordenadas
var cnv = document.getElementById("cnvDarkBody");
var ctx = cnv.getContext("2d");
var cnvW = cnv.width;
var cnvH = cnv.height;
var wrdW = 50;
var wrdH = 50;
ctx.transform(cnvW/wrdW,0,0,-cnvH/wrdH,cnvW/2,cnvH/2)
ctx.rotate(Math.PI/2)
ctx.lineWidth = wrdW/cnvW;

// cálculo e armazenamento dos valores da função
var nX = nY = 100;
var dx = wrdW/nX;
var dy = wrdH/nY;
var zmax = -Infinity;
var mapa = [];
for (var i=0;i<nX;i++) {
    var x = -wrdW/2 + i*dx;
    for (var j=0;j<nY;j++) {
        var y = -wrdH/2 + j*dy;
        var r = Math.sqrt(x*x+y*y)
        var theta = Math.atan2(y,x);
        var z = Math.pow( (r * R31(r) * T10(theta)),2);
        if (z>zmax) zmax = z;
        mapa.push([x,y,z]);
    }
}

// construção da imagem propriamente dita
for (var i=0; i<mapa.length; i++) {
    ctx.beginPath();
    ctx.fillStyle = darkBody(mapa[i][2]/zmax);
    ctx.strokeStyle = ctx.fillStyle;
    ctx.rect(mapa[i][0],mapa[i][1],dx,dy);
    ctx.stroke();
    ctx.fill();
}

</script>

</body>
</html>
```

Como no exemplo anterior (listagem 4), o documento contém apenas dois elementos: um canvas e um script. Quanto ao canvas, a única diferença é o seu tamanho, um pouco maior. Quanto ao script, contém quatro trechos distintos, novamente identificados com linhas de comentários e descritos em detalhes a seguir.

Definição da função que atribui a cor ao ponto

O primeiro bloco define a função que retorna uma cor no esquema *dark body* para cada valor de *z* fornecido. Nos exemplos anteriores, as funções realizam um simples cálculo e retornam um valor numérico. Neste caso, o corpo da função constitui um programa à parte, contendo variáveis locais (`var ...`), estruturas de controle e repetição (`if ...`, `for (...)`) e tudo o mais que pode ser utilizado em um programa comum. O valor retornado nesse caso não é um valor numérico, mas uma expressão literal (*string*) que especifica uma cor.

O corpo da função inicia definindo os parâmetros do gradiente de cor utilizado. A seguir, testa se o valor do parâmetro fornecido é menor que 0. Se for, a função fixa as três componentes de cor em 0 (isto é, prepara a cor preta como retorno). Da mesma maneira, a função testa se o parâmetro fornecido é maior que 1. Se for, a função fixa as três componentes de cor em 255 (isto é, prepara a cor branca como retorno). Para valores intermediários entre 0 e 1 a função calcula as contribuições de cada cor segundo o esquema explicado anteriormente. Primeiro, utiliza um laço `for (...)` para descobrir em que trecho do gradiente o parâmetro fornecido se encontra. A seguir utiliza essa informação para obter o valor de cada componente de cor segundo a especificação do intervalo através de uma interpolação linear. Por fim, o valor retornado é uma expressão literal que especifica uma cor (por exemplo, `rgb(130,214,35)`).

Captura do contexto gráfico e transformação de coordenadas

Esse bloco é em tudo semelhante ao apresentado na análise da listagem 4. Dignos de nota talvez sejam os parâmetros da instrução `ctx.transform(...)`, em particular o quarto parâmetro, com sinal negativo, que inverte a direção do eixo y (no sistema do canvas, o eixo y aponta para baixo; no sistema do problema, o eixo y aponte para cima).

Cálculo e armazenamento dos valores da função

Nesse bloco o plano com 50 × 50 raios de Bohr é dividido em 100 × 100 regiões. A instrução `mapa = []` informa ao JavaScript que a variável deve ser tratada como um vetor (ou matriz, no jargão computacional), isto é, uma variável que irá armazenar uma lista de itens do mesmo tipo. Dentro dos laços que fazem a varredura são calculadas as coordenadas cartesianas `x` e `y` de cada ponto, que serão usadas para informar ao canvas o início de cada pequena região a ser colorida. As coordenadas retangulares são convertidas para coordenadas polares `r` e `theta` e o valor de `z` é calculado passando estas coordenadas como parâmetros para as funções `R31(r)` e `T10(theta)`. A última instrução do laço mais interno adiciona ao vetor `mapa` um vetor com os três números associados a cada região do gráfico a ser colorida (as coordenadas *x* e *y* e o valor *z* da função).

Ao final do processo, a variável `mapa` conterá 10 mil vetores de três elementos cada, constituindo um vetor bidimensional 10000 x 3. Simultaneamente, é realizado o processo de

busca do máximo da função, que atualiza o valor de `zmax` cada vez que a instrução `if (z>zmax)` for verdadeira. Antes do laço ser iniciado, o valor de `zmax` deve ser inicializado com o menor valor possível. (`-Infinity`).

Construção da imagem propriamente dita

O terceiro e último bloco contém um laço que percorre todos os 10000 elementos da variável `mapa`. Para cada um deles utiliza o terceiro "sub-elemento" normalizado pelo máximo da função (`mapa[i][2]/zmax`) como parâmetro para a função que calcula a cor. A cor obtida é atribuída para as propriedades cor de preenchimento (`fillStyle`) e cor de contorno do canvas (`strokeStyle`). A instrução `ctx.rect(x,y,dx,dy)` desenha e preenche um retângulo com início em `x,y` e dimensões `dx,dy`.

Densidade de pontos

Outra maneira frequentemente utilizada para mostrar este tipo de informação consiste em construir uma figura na qual a densidade de pontos em uma determinada região do plano (ou do espaço, no caso 3D) é proporcional à grandeza que se quer visualizar. No caso das funções de onda do átomo de hidrogênio, esta representação é particularmente interessante por dois motivos. Por um lado, apresenta os dados como se fossem a superposição de uma série de medidas da mesma coisa, cada uma com um resultado diferente, seguindo uma distribuição de probabilidades. Por outro, a própria imagem é construída probabilisticamente (sorteando números) e não deterministicamente como no caso anterior. Apesar das duas maneiras levarem ao mesmo resultado, este procedimento incorpora claramente à imagem resultante a natureza probabilística do fenômeno.

Concretamente, o problema consiste em distribuir pontos aleatoriamente sobre uma superfície. Essa aleatoriedade, entretanto, não é uma aleatoriedade qualquer, mas deve seguir uma distribuição de probabilidades específica, dada pelas soluções da equação de Schroedinger para as componentes radial e angular da função de onda.

Nesse problema, devem ser sorteados um valor para o raio e um valor para o ângulo em que se encontra o elétron no momento em que é feita a "fotografia". A cada nova fotografia, um novo raio e um novo ângulo devem são sorteados, sempre seguindo a distribuição de probabilidades dada pela solução da equação de Schroedinger.

Computacionalmente, esses sorteios são feitos utilizando-se geradores de números aleatórios. Entretanto, é inviável incorporar nas linguagens de programação geradores de números aleatórios para a infinidade de distribuições de probabilidades associadas a fenômenos reais da física, da biologia, da economia etc. Em geral as linguagens de programação dispõem de apenas um gerador de números aleatórios, que fornece números que seguem a distribuição uniforme. A partir deles técnicas matemáticas são utilizadas para obter números que obedecem distribuições de probabilidades arbitrárias.

Distribuições de probabilidades

Um gerador de números aleatórios que obedece a uma distribuição uniforme produz números entre dois extremos (entre 0 e 1, por exemplo) com igual probabilidade. Ou seja, é aquela em que a probabilidade $p(x)$ de se gerar um número entre x e $x + dx$ é dada por:

$$p(x)dx = dx$$

para $0 < x < 1$ e 0 para qualquer outro intervalo. A distribuição de probabilidades é normalizada (a probabilidade de gerar qualquer número é 1):

$$\int_{-\infty}^{+\infty} p(x)dx = 1$$

A Fig. 7 apresenta o gráfico de frequências obtido para 100 mil números aleatórios sorteados entre 0 e 1 a partir da distribuição uniforme, distribuídos em 100 canais ao longo do eixo x.

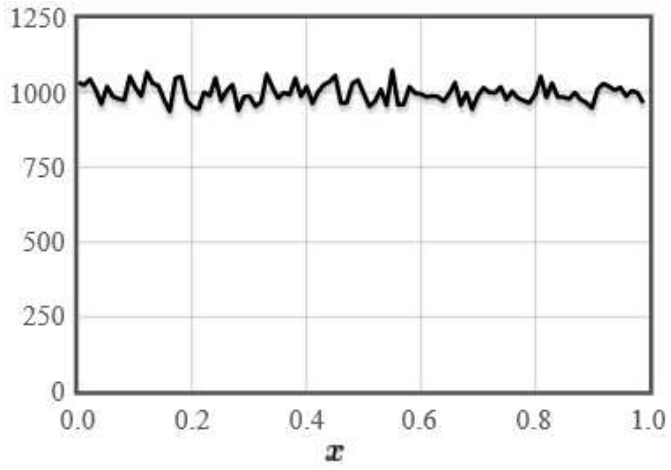

Fig. 7 – Gráfico da frequência de 100 mil números aleatórios sorteados utilizando a distribuição uniforme $\mathrm{p}(x)dx$ *e distribuídos em 100 canais ao longo do eixo x. O gráfico deixa evidente que, apesar da pequena flutuação estatística, todos os canais contém aproximadamente o mesmo valor (~1000).*

Lei da transformação das probabilidades

A lei fundamental da transformação das probabilidades (ANGUS, 1994; PRESS et al., 2007) permite relacionar números que obedecem uma distribuição de probabilidade $p(y)$ a números que obedecem outra distribuição uniforme $p(x)$

$$|p(y)dy| = |p(x)dx|$$

de modo que:

$$p(y) = p(x)|\frac{dx}{dy}|$$

Fazendo $p(x) = 1$ (distribuição uniforme) e integrando a equação obtém-se:

$$x = \int_0^y p(y')dy'$$

Invertendo essa equação, obtém-se a função $y(x)$ que, dado um número x sorteado de uma distribuição uniforme $p(x)$, retorna um número y que obedece a distribuição $p(y)$.

Um exemplo concreto pode deixar o procedimento mais claro. A distribuição de probabilidades:

$$p(y)dy = 2ydy, \text{ para} 0 < y < 1$$

fornece números aleatórios que obedecem uma "rampa" linear entre 0 e 1 (o fator dois é a constante de normalização, de modo que a integral da função no intervalo considerado seja 1). Utilizando a lei de transformação de probabilidades, fazendo $p(x) = 1$, integrando a equação e invertendo o resultado obtém-se:

$$p(y) = p(x)|\frac{dx}{dy}| = 2y$$

$$x = y^2$$

$$y = \sqrt{x}$$

Em palavras, isso significa que a raiz quadrada de números sorteados de uma distribuição uniforme obedecem à rampa linear. A Fig. 8 mostra o gráfico da frequência de 100 mil números aleatórios sorteados utilizando a distribuição $p(y)dy = 2ydy$.

Densidade de pontos

No caso das funções de onda do átomo de hidrogênio, é possível calcular analiticamente as integrais das distribuições de probabilidade $r^2|R_{31}(r)|^2$ e $|\Theta_{10}(\theta)|^2$. Entretanto, não é possível invertê-las para obter expressões analíticas para $r(x)$ e $\theta(x)$ (como acontece, aliás, na vasta maioria dos problemas práticos, onde frequentemente sequer se conhece exatamente a distribuição de probabilidade). Nesses casos, em geral a integral é calculada numericamente, armazenada em uma tabela (com a precisão desejada), e o valor de y é obtido a partir de uma interpolação entre dois valores previamente calculados. Por tratar-se de um procedimento relativamente simples e aplicável a todo tipo de distribuição (teórica ou experimental, integrável ou não, inversível ou não), é aplicado na solução de todo tipo de problema.

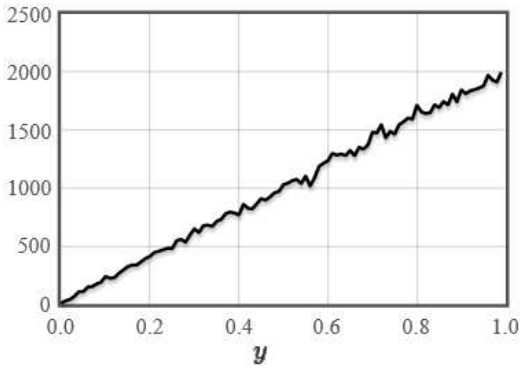

Fig. 8 – Gráfico da frequência de 100 mil números aleatórios sorteados utilizando a distribuição $p(y)dy = 2ydye$ distribuídos em 100 canais ao longo do eixo y. Apesar da flutuação estatística, o gráfico deixa evidente que os números são sorteados com uma probabilidade que cresce linearmente entre y = 0 e y = 1.

A Fig. 9 mostra os gráficos das integrais das distribuições de probabilidade $r^2|R_{31}(r)|^2$ e $|\Theta_{10}(\theta)|^2$ calculadas e apresentadas no documento descrito na listagem 6. Também na listagem 6 está detalhado o processo de obtenção da Fig. 10, que mostra densidade de probabilidade de se encontrar um elétron em um plano que passa pelo núcleo do átomo de hidrogênio, obtida através do sorteio de 50000 pontos (*r*,*θ*) que obedecem a essas distribuições de probabilidades.

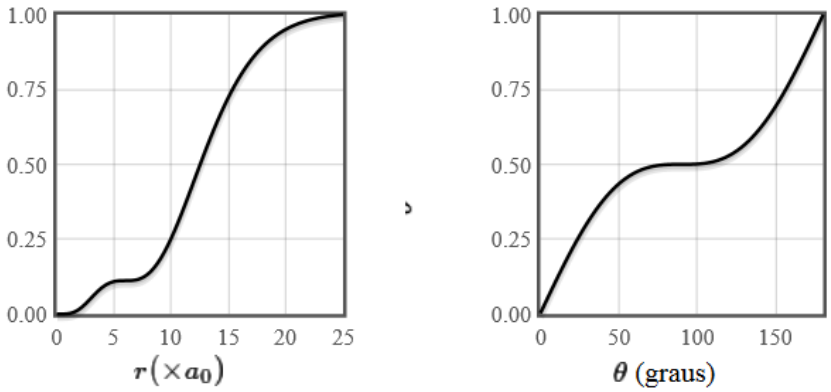

Fig. 9 – Gráficos das integrais das densidades de probabilidade $r^2|R_{31}(r)|^2$ (esquerda) e $|\Theta_{10}(\theta)|^2$ (direita) obtidas numericamente. Em ambos os casos foram geradas tabelas que calculam os valores das integrais com uma precisão da ordem de 0,1%.

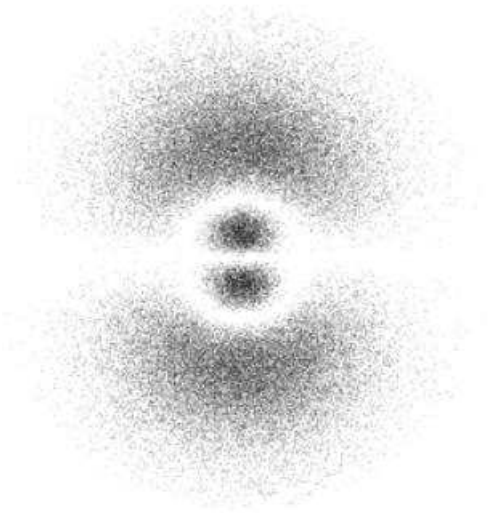

Fig. 10 – Representação da densidade de probabilidade de se encontrar um elétron em um plano que passa pelo núcleo do átomo de hidrogênio, obtida através do sorteio de 50000 pontos (r,θ) que seguem as distribuições de probabilidade e $r^2|R_{31}(r)|^2$ *e* $|\Theta_{10}(\theta)|^2$.

Listagem 6 - Documento utilizado para gerar as Fig. 9 e 10.

```
<html>

<script src="file://C:/flot/jquery.js"></script>
<script src="file://C:/flot/jquery.flot.js"></script>

<body>

<div id="grIntR31" style="width:200;height:200"></div>
<div id="grIntT10" style="width:200;height:200"></div>
<canvas id="densPts" width="250" height="250"></canvas>

<script>

var Z = 1;
var a0 = 1;

function R31(r) {
    return 4*Math.sqrt(2)/3*Math.pow((Z/3/a0),(3/2))*
           (Z*r/a0)*(1-Z*r/6/a0)*Math.exp(-Z*r/3/a0);
}

function T10(x) {
    return Math.sqrt(3/4/Math.PI)*Math.cos(x);
}

// preenche o vetor intR31 com a integral de R31
var intR31 = [];
var a0 = 1;
var nRadius = 1000;
var dRadius = 25*a0/nRadius;
var radius = [];
for (var i=0; i<=nRadius; i++) radius.push(i*dRadius);
```

```
var int = 0;
for (var i=0; i<=nRadius; i++) {
    int += R31(i*dRadius)*R31(i*dRadius)*
           (i*dRadius)*(i*dRadius)*dRadius;
    intR31.push(int);
}
// garante que a integral seja normalizada
for (var i=0;i<intR31.length;i++) intR31[i] /= int;
    console.log(int);

// preenche o vetor intT10 com a integral de T10
var intT10 = [];
var nTheta = 720;
var dTheta = Math.PI/nTheta;
var theta = [];
for (var i=0; i<=nTheta; i++) theta.push(i*dTheta);
var int = 0;
for (var i=0; i<=nTheta; i++) {
    int += T10(i*dTheta)*T10(i*dTheta) * dTheta;
    intT10.push(int);
}
// garante que a integral seja normalizada
for (var i=0;i<intT10.length;i++) intT10[i] /= int;
    console.log(int);

// definição da função que obtem raio aleatório
function randomR31() {
    var x = Math.random();
    for (var i=1;i<intR31.length;i++)
             if ((x>=intR31[i-1])&&(x<intR31[i]))
                return (i-1)*dRadius + (x-intR31[i-1]) *
                       dRadius/(intR31[i]-intR31[i-1]);
}

// definição da função que obtem o ângulo aleatório
function randomT10() {
    // theta foi definido de 0 a pi. Para preencher todo o espaco
    // é sorteado outro número que define o sinal do ângulo
    var x = Math.random();
    if (Math.random()>0.5) sgn = 1; else sgn = -1;
    for (var i=1;i<intT10.length;i++)
             if ((x>intT10[i-1])&&(x<=intT10[i]))
                return sgn * ( (i-1)*dTheta + (x-intT10[i-1]) *
                       dTheta/(intT10[i]-intT10[i-1]) );
}

// constroi o vetor com os pontos [r,f]
//para o gráfico da integral em r
var grIntR31 = [];
for (var i=0;i<intR31.length;i++) {
    var r = radius[i];
    var f = intR31[i];
    grIntR31.push( [r,f] );
}
$.plot( "#grIntR31", [grIntR31],
        { yaxis:{ max: 1 }, colors: ["black"] } );

// constroi o vetor com os pontos [r,t]
// para o gráfico da integral em theta
```

```
var grIntT10 = [];
for (var i=0;i<intT10.length;i++) {
    var t = theta[i]*180/Math.PI;
    var f = intT10[i]
    grIntT10.push( [t,f] );
}
$.plot("#grIntT10", [grIntT10],
       { yaxis: { max: 1}, colors: ["black"] } );

// captura o canvas e transforma as coordenadas
var cnv = document.getElementById("densPts");
var ctx = cnv.getContext("2d");
var cnvW = cnv.width;
var cnvH = cnv.height;
var wrdW = 50;
var wrdH = 50;
ctx.transform(cnvW/wrdW,0,0,-cnvH/wrdH,cnvW/2,cnvH/2)
ctx.rotate(Math.PI/2)
ctx.lineWidth = wrdW/cnvW;

// sorteia r e t para desenhar o ponto
for (var i=0; i<50000; i++) {
    var r = randomR31();
    var t = randomT10();
    var x = r * Math.cos(t);
    var y = r * Math.sin(t);
    ctx.beginPath();
    ctx.arc(x,y,wrdW/1000,0,2*Math.PI,0);
    ctx.fill();
}

</script>

</body>
</html>
```

A listagem 6 tem comentários que a esta altura devem deixá-la quase autoexplicativa. Começa incluindo os pacotes necessários para desenhar os gráficos com o FLOT e declarando as divisões e o canvas que vão receber os gráficos e a figura com a densidade de pontos, respectivamente. No início do script estão as definições do número atômico e do raio de Bohr (em unidades convenientes) e das funções que calculam $R_{31}(r)$ e $\Theta_{10}(\theta)$.

O bloco seguinte define as estruturas de dados que receberão os valores possíveis de *r* (`radius = []`) e da integral de $r^2|R_{31}(r)|^2$ (`intR31 = []`) e usa dois laços `for (...)` distintos, um para apensar valores sucessivos do raio ao vetor `radius` e outro para apensar os valores da integral da função ao vetor `intR31`, em ambos os casos utilizando o método `push()`. Ao final do bloco, todos os elementos do vetor `intR31` são divididos pelo valor da integral total da função, para garantir que a integral da distribuição de probabilidade seja 1.

Procedimento semelhante é utilizado para definir e preencher as estruturas de dados que receberão os valores possíveis de θ e da integral de $|\Theta_{10}(\theta)|^2$.

Em seguida, é definida a função `randomR31()`, que retorna valores de *r* que seguem sua respectiva distribuição de probabilidades. Dentro da função, é sorteado um número aleatório

que obedece à distribuição uniforme (`x = Math.random()`). Um laço `for (...)` varre a tabela com a integral verificando entre que elementos encontra-se o valor sorteado (`if ((x>=intR31[i-1])&&(x<intR31[i]))`) e retorna um valor interpolado linearmente entre eles.

A função `randomT01()` é definida essencialmente da mesma maneira, exceto pelo uso de um número aleatório adicional que define um sinal para o ângulo, de modo que a figura apareça completa, e não apenas em um hemisfério (pois θ é definido somente entre 0 e π).

Os dois blocos seguintes constroem os gráficos das integrais utilizando o FLOT. Nestes exemplos, a função `$.plot` recebe três parâmetros: a referência à `div` que vai receber o gráfico, a matriz com os pontos a serem visualizados e uma lista de opções que, no caso, ajustam o máximo do eixo *y* para 1 e a cor preta para o gráfico (são dezenas as opções que podem ser especificadas nesta estrutura (LAURSEN; SCHNUR, 2015)), que é estruturada empregando um formato amplamente utilizado na internet (ECMA, 2013).

O bloco final captura o canvas e realiza as transformações de coordenadas adequadas antes sortear os 50 mil pontos que comporão a figura. Tanto o número de pontos quanto o tamanho deles (definido no terceiro parâmetro da chamada a `ctx.arc(...)`) influenciam na qualidade da imagem e vários valores devem ser testados até obter-se o efeito desejado.

V. Conclusão

Em uma sociedade que gera e disponibiliza dados em uma escala cada vez maior, o domínio de ferramentas que auxiliem a sua visualização e análise é uma habilidade de extremo valor pessoal e profissional (particularmente no caso de profissionais da área de ciência e tecnologia, em qualquer nível). Dada a quantidade e complexidade desses dados, é inevitável a utilização de ferramentas computacionais para a consolidação da informação e apresentação em formatos que possibilitem sua interpretação (ou seja, a transformação de dados em informação).

Nesse trabalho foram apresentadas metodologias que utilizam as tecnologias de especificação, formatação e manipulação de informações atualmente mais disseminadas no planeta (HTML, CSS e JavaScript) para auxiliar na visualização de conceitos associadas a um dos mais importantes tópicos da física moderna: as distribuições de probabilidades associadas às funções de onda do elétron no átomo de hidrogênio.

Foram apresentados exemplos concretos de como incorporar em um único documento metodologias de cálculo de valores e geração de gráficos e imagens que utilizam modelos matemáticos relativamente simples (transformações de coordenadas, gradientes de cores e distribuições de probabilidades) que facilitam a interpretação das informações.

A expectativa é de que, independentemente de seu conteúdo específico, este artigo tenha ajudado a perceber que alguma familiaridade com técnicas de modelagem e visualização de informações pode vir a ser relevante nos mais diversos contextos pessoais e profissionais.

Agradecimentos

Agradeço aos alunos das disciplinas "Estrutura da Matéria I" e "Física Computacional" do Curso de Graduação em Física da UFSC pela receptividade e atenção dispensadas às discussões em sala de aula, que muito estimularam a organização desse conteúdo na forma de um artigo. Agradeço também aos pareceristas que avaliaram o artigo pela leitura minuciosa do texto, apontando alguns erros e oferecendo sugestões valiosas.

Referências bibliográficas

ALONSO, M.; FINN, E. **University Physics, Vol. III: Quantum and Statistical Physics.** Reading: Addison-Wesley, 1968.

ANGUS, J. E. The probability integral transform and related results. **SIAM Review**, v. 36, n. 4, p. 652-654, Dec. 1994.

BLATT, F. J. **Modern Physics.** New York: McGraw-Hill, 1992.

BOSTOK, M. **D3 data manipulation library.** Disponível em: <https://d3js.org/>. Acesso em: 06 out. 2015.

CERN **ROOT Data Analysis Framework: TColor class.** Disponível em: <http://root.cern.ch/root/html/src/TColor.cxx.html#Autqp>. Acesso em: 29 jun. 2015.

DOMINGUINI, L. Física moderna no Ensino Médio: com a palavra os autores dos livros didáticos do PNLEM. **Revista Brasileira de Ensino de Física**, v. 34, n. 2, p. 2502, 2012.

ECMA International. **ECMA-262 ECMAScript Language Specification**, 6. ed. June 2015. Geneve: ECMA International, 2015. Acesso em: 06 out. 2015.

ECMA International. **The JSON Data Interchange Format, 1st Edition, October 2013.** Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em: 22 jun. 2015.

EISBERG, R.; RESNICK, R. **Física Quântica.** Rio de Janeiro: Campus, 1986.

HIGHSOFT AS. **Highcharts charting library.** Disponível em: < http://www.highcharts.com/products/highcharts>. Acesso em: 29 abr. 2015.

LAURSEN, O.; SCHNUR, D. **FLOT: an attractive JavaScript plotting for jQuery**, v. 0.8.3 Disponível em: <http://www.flotcharts.org/>. Acesso em: 29 abr. 2015.

MDN (Mozilla Developer Network). **Drawing text.** Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_text>. Acesso em: 29 abr. 2015.

NAVE, C. R. **HyperPhysics, Department of Physics and Astronomy, Georgia State University, 2012.** Disponível em: <http://hyperphysics.phy-astr.gsu.edu/hbase/hframe.html>. Acesso em: 28 jun. 2015.

OSTERMANN, F.; MOREIRA, M. A. Uma revisão bibliográfica sobre a área de pesquisa "Física Moderna e Contemporânea no Ensino Médio". **Investigações em Ensino de Ciências**, v. 5, n. 1, p. 23-48, 2000.

PAULING, L. **Química Geral.** Rio de Janeiro: Ao Livro Técnico S. A., 1966.

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P. **Numerical Recipes** Third Edition. Cambridge University Press, 2007, p. 907-910. Disponível em: <http://apps.nrbook.com/empanel/index.html>. Acesso em: 29 abr. 2015.

REFSNES Data **w3schools Tutorials.** Disponível em: <http://www.w3schools.com>. Acesso em: 22 jun. 2015.

ROGOWITZ, B.; COUET, O. **The Rainbow Color Map.** ROOT Data Analysis Framework. Disponível em: <https://root.cern.ch/drupal/content/rainbow-color-map>. Acesso em: 29 jun. 2015.

ROZENTALSKI, E. F. **O estatuto ontológico e epistemológico do conceito de orbital em livros didáticos de Química Geral no século XX:** uma análise de seus fundamentos, suas representações e implicações para a aprendizagem. 2013. Dissertação (Mestrado em Ensino de Ciências) - Universidade de São Paulo. Disponível em: <http://www.teses.usp.br/index.php?option=com_jumi&fileid=11&Itemid=76&lang=en&filtro=rozentalski>. Acesso em: 18 jul. 2015.

SILVA, N. C. da. **Física com JavaScript.** 2. ed. Florianópolis: Universidade Federal de Santa Catarina, 2016. 192 p. Disponível em: <http://canzian.fsc.ufsc.br/ead/fiscomp/fisica-com-javascript-2016.pdf>. Acesso em: 02 mar. 2016.

SIMA, Z.; BUBINSKI, R. **Codecademy.** Disponível em: <http://www.codecademy.com>. Acesso em: 22 jun. 2015.

TIPLER, P. A.; LLWELLYN, R. A. **Física Moderna**. 3. ed. Rio de Janeiro: LTC Editora, 2001.

W3C (World Wide Web Consortium). **Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, W3C Recommendation 07 June 2011.** Disponível em: <http://www.w3.org/TR/CSS21/>. Acesso em: 29 abr. 2015.

W3C (World Wide Web Consortium). **HTML5: A vocabulary and associated APIs for HTML and XHTML. W3C Proposed Recommendation 16 September 2014.** Disponível em: <http://www.w3.org/TR/html5/>. Acesso em: 29 abr. 2015.

WIKIMEDIA Foundation, Inc. **Wikipedia.** Disponível em:
<https://en.wikipedia.org/wiki/Quantum_mechanics>. Acesso em: 28 jun. 2015.