22 Decaimento radioativo

Reações e desintegrações nucleares são importantes fontes de informações sobre a estrutura nuclear, pois são processos nos quais há um rearranjo da configuração dos nucleons no núcleo. Muitos desses processos ocorrem naturalmente e outros são produzidos artificialmente em laboratórios, aceleradores e reatores.

Alguns núcleos têm uma combinação de prótons e nêutrons que não leva a uma configuração estável. Núcleos instáveis tendem a se aproximar de uma configuração estável pela emissão de certas partículas alfa (α) e beta (β). Os núcleos que resultam desses decaimentos podem encontrar-se em um estado excitado e decaem para o seu estado fundamental emitindo radiação gama (γ).

A *lei do decaimento radioativo* é uma função que descreve quantos núcleos radioativos existem em uma amostra a partir do conhecimento do número inicial de núcleos radioativos e da taxa de decaimento. É obtida a partir da hipótese de que o número dN de núcleos que decaem em um intervalo de tempo dt é proporcional ao número de núcleos radioativos existentes e ao próprio intervalo de tempo dt:

$$dN = -\lambda N dt$$

onde λ é a constante de decaimento do material. O sinal negativo indica que há uma diminuição no número de nucleons. A integração da equação leva à lei do decaimento radioativo:

$$N(t) = N_0 e^{-\lambda t}$$

onde $N(t)$ é o número de átomos radioativos no instante t e N_0 é o número de átomos radioativos no instante $t = 0$.

A constante de decaimento λ está relacionada à meia-vida $T_{1/2}$, definida como o tempo necessário para que metade dos átomos instáveis de uma amostra decaiam, pela relação:

$$\lambda = \ln 2 / T_{1/2}$$

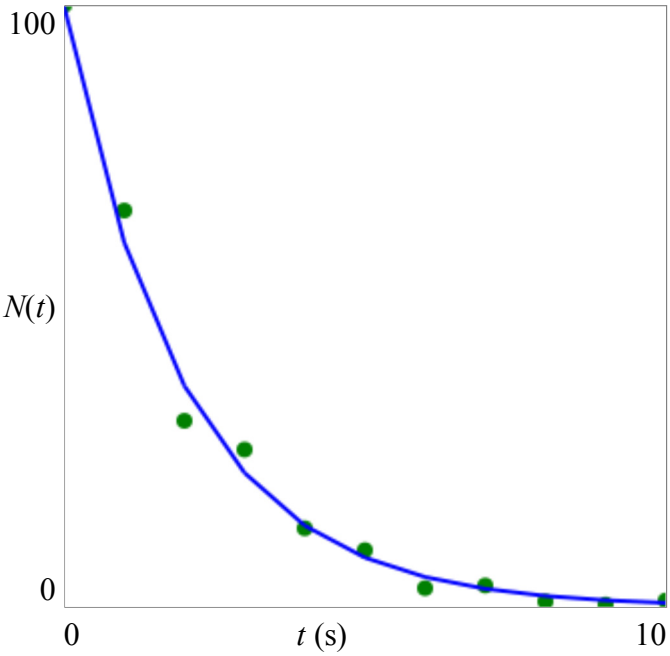
O script a seguir simula a população de nuclídeos radioativos de uma certa amostra em função do tempo.

exemplo-22-1.html

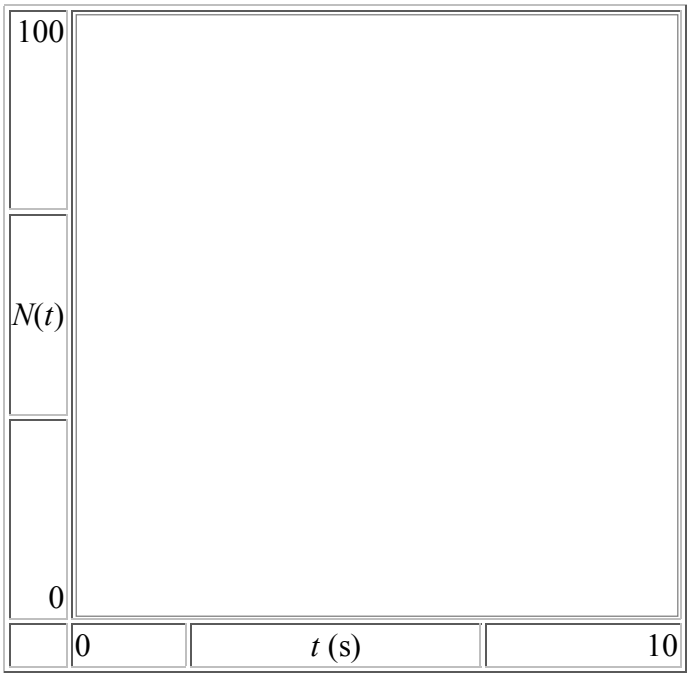
```
<table align='center' border='0'>
<tr>
   <td valign='top' align='right'>
      100
   </td>
   <td rowspan='3' colspan='3'>
      <canvas style='border: 1px solid gray;'
            id="cnv" width='300' height='300'>
      </canvas>
   </td>
</tr>
<tr>
   <td valign='center' align='right'><i>N</i>(<i>t</i>)</td>
</tr>
<tr>
   <td valign='bottom' align='right'>0</td>
</tr>
<tr>
   <td> </td>
   <td align='left'>0</td>
   <td align='center'><i>t</i> (s)</td>
   <td align='right'>10</td>
</tr>
</table>

<script>
function TransCoord(cLrg,cAlt,mxi,myi,mxf,myf) {
   this.Bx = cLrg/(mxf-mxi);
   this.Ax = - this.Bx * mxi;
   this.By = cAlt/(myi-myf);
   this.Ay = - this.By * myf;
}
TransCoord.prototype.cx = function(mx) {
   return this.Ax + this.Bx * mx;
}
TransCoord.prototype.cy = function(my) {
   return this.Ay + this.By * my;
}

var tc = new TransCoord(300,300,0,0,10,100);
var ctx = document.getElementById('cnv').getContext('2d');

var npts = 11;
var t = new Array(npts);
var Ne = new Array(npts);
var Nc = new Array(npts);
var No = 100;
var lambda = 0.5;

for (var i=0;i<npts;i++) {
   t[i]  = i;
   Nc[i] = No * Math.exp(-lambda*t[i]);
   Ne[i] = Nc[i] + (Math.random()*2 - 1) * Math.sqrt(Nc[i]);
}

for (i=0;i<t.length;i++) {
   ctx.beginPath();
   ctx.arc(tc.cx(t[i]),tc.cy(Ne[i]),4,0,2*Math.PI,2);
   ctx.fillStyle = "green";
   ctx.fill();
}
```

```
for (i=0;i<t.length-1;i++) {
   ctx.beginPath();
   ctx.strokeStyle = "blue";
   ctx.lineWidth = 2;
   ctx.moveTo(tc.cx(t[i]),tc.cy(Nc[i]));
   ctx.lineTo(tc.cx(t[i+1]),tc.cy(Nc[i+1]));
   ctx.stroke();
}
</script>
```

Resultado:



O corpo do documento HTML contém uma tabela sem bordas centralizada na página. Essa tabela está estruturada para conter o gráfico e os extremos e nomes das escalas. Se você alterar o valor do atributo `border` para `1` verá o seguinte resultado:



A tabela deve ser pensada como tendo 4 linhas e 4 colunas. O número de linhas é definido pelo número de pares `<tr>...</tr>` que contém. O número de colunas é definido pela linha com o maior número de pares `<td>...</td>`, no caso a 4a. linha. Algumas células foram aglutinadas para formar o grande bloco que contém o gráfico. Isto é feito utilizando os atributos `rowspan` (extensão de linhas) e `colspan` (extensão de colunas).

A primeira linha da tabela contém duas células. A primeira célula contém o máximo da escala vertical (100) e o alinhamento do texto é no topo (`valign='top'`) e à direita (`align='right'`). A segunda célula estende-se por três linhas (`rowspan='3'` e três colunas (`colspan='3'`) e contém o `canvas`, que é a "tela" onde são desenhados os elementos gráficos.

O canvas é formatado utilizando-se o atributo `style`, que lhe dá uma borda sólida de 1 pixel de espessura e cor cinza (`gray`). Os atributos `width` e `height` o definem como uma área quadrada de 300 × 300 pixels e o atributo `id` lhe dá uma identificação única `cnv` que será utilizada pelo script que faz os desenhos para "encontrá-lo" na página.

A linha seguinte, a segunda da tabela, define apenas uma célula que contém o nome do eixo vertical, $N(t)$. As outras três células da linha estão "tomadas" pelo canvas (lembre-se do `rowspan='3'`.

A terceira linha da tabela também define apenas uma célula que contém o mínimo da escala vertical (0). Agora o alinhamento vertical é no fundo (`bottom`) e o horizontal à direita.

A quarta linha da tabela volta a definir quatro células pois a abrangência do `rowspan` na célula que contém o canvas já terminou. A primeira célula contém apenas um espaço em branco (` `) e as outras três o mínimo, o nome e o máximo da escala horizontal. O mínimo está centralizado à esquerda, o máximo à direita e o nome no centro.

Vamos agora ao script. Ele está organizado em dois blocos: a definição do objeto `TransCoord` e o código executado quando o documento é carregado. O objeto `TransCoord` possui um *construtor* que recebe seis parâmetros: a largura (`cLrg`) e a altura (`cAlt`) do canvas e os

limites do "mundo" que estamos modelando, na direção *x* (`mxi` e `mxf`) e na direção *y* (`mxi` e `mxf`). Quando uma instância do objeto é criada esses valores são passados para o construtor e utilizados para os cálculos dos coeficientes lineares e angulares para as transformações de coordenadas na direção *x* e na direção *y*.

As expressões utilizadas no construtor vêm da solução dos sistemas lineares:

Direção *x*		**Direção *y***	
Modelo	Canvas	Modelo	Canvas
`mxi` `mxf`	`0` `cLrg`	`myi` `myf`	`cAlt` `0`
`0 = Ax + Bx * mxi` `cLrg = Ax + Bx * mxf`		`cAlt = Ay + By * myi` `0 = Ay + By * myf`	

O construtor define quatro atribudos do objeto. Estes atributos são declarados utilizando-se o prefixo `this` para indicar que se referem a *este* objeto. A seguir, são declarados dois *métodos* do objeto. A declaração do método é feita especificando o nome do objeto a que pertence, seguido da palavra reservada `prototype` e então o nome da função. As duas funções recebem um único parâmetro (`mx` no caso de `cx()` e `my` no caso de `cy()`).

O trecho do script executado quando o documento é carregado faz essencialmente 6 ações: (a) cria uma instância do objeto `TransCoord`; (b) obtém a referência para o objeto `canvas` definido no documento HTML; (c) define uma série de variáveis utilizadas para modelar o sistema físico em questão; (d) desenha os pontos simulados; e (e) desenha a curva teórica.

Após instanciar uma transformação de coordenadas, a função `init()` busca no documento HTML o elemento `cnv`, que foi o `id` dado ao canvas, e usa o método `getContext()` para obter a referência ao contexto gráfico.

São definidas três variáveis do tipo `Array` que armazenarão os `npts` valores de tempo, de dados "experimentais" e de valores da função que descreve o decaimento radioativo para os instantes de tempo especificados. O número de nuclídeos iniciais foi definido como 100 e a constante de decaimento $\lambda = 0.5\ s^{-1}$, o que corresponde a uma meia-vida de 1.8 s.

O laço `for` que se segue define os valores do tempo (`t[i]`) do número de nuclídeos calculados pelo modelo teórico (`Nc[i]`) e o número de nuclídeos "experimentais" (`Ne[i]`), que é calculado introduzindo uma flutuação estatística no número calculado com o modelo teórico.

A seguir, são desenhados os pontos "experimentais" e a linha teórica. Os métodos disponíveis para desenhar no canvas são muitos e permitem operações bastante sofisticadas. Para operações simples como desenhar um gráfico, a ordem é relativamente simples: inicia-se um "caminho" (*path*) com o método `beginPath()`, escolhe-se a cor (`strokeStyle`) e a espessura (`lineWidth` da linha, além do preenchimento de figuras geométricas (`fillStyle`), constrói-se o elemento (`arc`, `moveTo`, `lineTo`), encerra-se o caminho (`closePath()`) e finalmente desenha-se o conjunto (`stroke`).

O método `arc` recebe seis parâmetros, na ordem: `arc(`*x*`,`*y*`,`*r*`,`θ_i`,`θ_f`,`*sent*`)`, onde *x* e *y* são as coordenadas do centro do arco e *r* o seu raio, em unidades do canvas, θ_i e θ_f os ângulos inicial e final, em radianos. O último parâmetro pode ser `true` ou `false`, e diz se o arco deve ser desenhado no sentido anti-horário ou não (o que é indiferente no caso de arcos

completos). Note o uso da transformação de coordenadas `tc` para levar os valores de `t[i]` e `Ne[i]` para o sistema de coordenadas da tela.

Para desenhar as linhas unindo os pontos calculados, optamos por primeiro mover (`moveTo(xi,yi)`) a "caneta" para o ponto inicial e desenhar a linha até o ponto final (`lineTo(xf,yf)`) para cada segmento da curva. Note que o laço `for` é realizado de `t.length-1` vezes, e não `t.length`, como no caso dos pontos, pois o índice do final da linha é `i+1`.

Exercícios

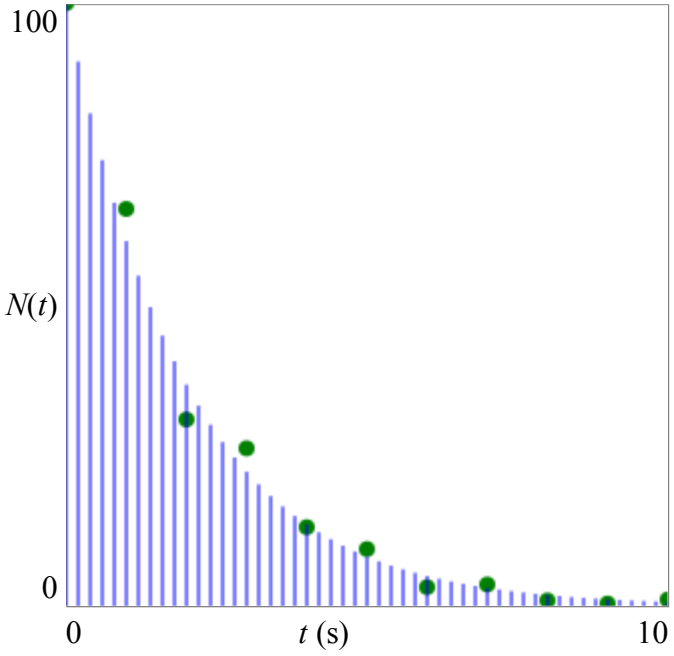
1. O script acima utiliza uma constante de decaimento de 0.5 s^{-1}. Modifique seu valor para 0.05 s^{-1} e depois para 5.0 s^{-1} e faça as demais alterações no programa para que ele mostre um gráfico que "encaixe" no canvas de maneira semelhante ao deste exemplo.

2. O script acima utiliza 100 como o número inicial de nuclídeos instáveis. Modifique seu valor para 10 e depois para 1000 e faça as demais alterações no programa para que ele mostre um gráfico que "encaixe" no canvas de maneira semelhante ao deste exemplo. Observe particularmente como ficam as flutuações estatísticas nos dados "experimentais".

3. Modifique o script de modo a fazer com que o canvas tenha (a) o dobro da largura e (b) o dobro da altura, mas de modo que os gráficos de dados e da função tenham as mesmas características (isto é, se você imprimir o canvas inicial e o final e colocar um sobre o outro, os pontos e a curva fiquem superpostos.

4. O script acima primeiro calcula os pontos teóricos e depois introduz a flutuação para simular os pontos experimentais. Isto significa que os pontos experimentais são tantos quanto os teóricos. Modifique o script de modo a fazer com que simule um ponto experimental apenas a cada 10 pontos teóricos (isso fará a curva teórica muito mais suave).

5. Modifique o script de modo a fazer com que, ao invés de uma curva ligando os pontos, ele desenhe barras verticais saindo do eixo horizontal e chegando aos pontos da curva (um gráfico de barras). Neste caso, é interessante fazer os cálculos para um grande número de pontos teóricos.



6. Modifique a transformação de coordenadas de modo a deixar a escala vertical logaritmica. Se funcionar, o gráfico se tranformará em uma reta descendente (lembre-se que se $y = A\, e^{ax}$, $\ln(y) = \ln(A) + ax$).