Lançamento de projetil usando Javascript

Jorge Luís Gurgel Fernandes

Universidade Federal do Rio Grande do Norte

jorgelgf@ufrn.edu.br

1. Introdução

O trabalho consiste em analisar o comportamento de uma partícula em um lançamento de projetil com o auxílio de um programa desenvolvido na linguagem Javascript.

O tema foi escolhido devido a junção dos conhecimentos que adquiri no decorrer do curso de Ciências e Tecnologia (C&T). Achei bastante interessante o desafio de criar uma lógica para resolver, computacionalmente e através dos conceitos físicos, o comportamento de uma partícula em um arremesso no plano xy.

O ensaio consiste em:

- Definição do problema a ser estudado;
- Mostrar o algoritmo feito e a lógica utilizada;
- Uso do programa para analisar o comportamento de uma partícula em seu lançamento de acordo com a mudança do ângulo aplicado com a horizontal;
- Coletar os dados informados pelo programa;
- Apontar, de acordo com os dados obtidos, o ângulo que foi capaz de atingir uma maior distância e explicar o porquê da eficiência;
- Conclusão;
- Bibliografia;

2. Definição do problema

Um projétil é qualquer sólido lançado “com uma velocidade inicial e que segue uma trajetória determinada exclusivamente pela aceleração da gravidade e pela resistência do ar” (YOUNG & FREDMAN, 2008, p.77).

Para analisar esse tipo de movimento, o programa implementado despreza a resistência do ar e os efeitos da curvatura e rotação da Terra. Vale ressaltar que o algoritmo construído considerada a aceleração da gravidade, em que ela é constante em módulo, direção e sentido.

De início, será atribuído uma velocidade fixa, em metros por segundo (m/s), e o ângulo (dado em graus) em relação a horizontal. Serão aplicados ângulos diferentes em cada teste, de modo que a velocidade inicial será a mesma para cada evento. Por fim, teremos cinco amostras diferentes, no qual será verificado qual dos casos foi obtido uma maior distância.

3. Algoritmo feito e lógica utilizada

O algoritmo foi desenvolvido em Javascript, que é uma linguagem de programação que trabalha dentro de um arquivo em HTML.

A escolha dessa linguagem foi devido aos conhecimentos obtidos no semestre passado, 2017.1, na disciplina de Lógica de Programação ministrada pelo professor Orivaldo Vieira. Disciplina em que pude absorver muitos conceitos práticos no desenvolvimento de pequenos programas de computadores.

Para implementação do código foi gerado quatro arquivos, ditos como principais, que são:

- index2.html → Página inicial a ser aberta;
- fisica.js→ Arquivo contendo as principais instruções em Javascript;
- index.html → Arquivo para chamar uma nova aba no navegador;
- Chart.min.js → Arquivo em javascript contendo instruções para construção de um gráfico;

O programa também contém algumas bibliotecas em Javascript dentro do seu próprio diretório (pastas *libraries*).

Ilustração 1: Diretório principal do programa

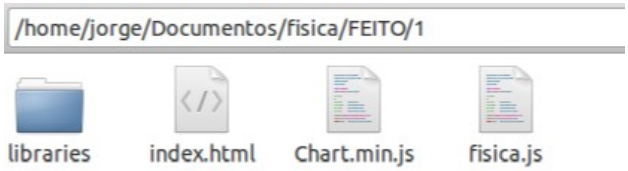

Ilustração 2: Diretório referente aos arquivos geradores do gráfico

Código contido no arquivo index2.html:

```
1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2. <html><head>
3. <meta charset="UTF-8">
4. <script language="javascript" type="text/javascript" src="libraries/p5.js"></script>
5. <script language="javascript" type="text/javascript" src="fisica.js"></script>
6. <form><INPUT TYPE="button" VALUE="Atualizar" onClick='parent.location="javascript:location.reload()"'></form>
7. <a href = "1/index.html" target="_blank">Clique aqui para gerar um gráfico</a>
```

```
8. </body>
9. </html>
```

Tabela 1: Código no index2.html

O código acima tem as devidas chamadas das bibliotecas utilizadas e a chamada do arquivo fisica.js com os códigos das instruções principais em javascript.

Abaixo, o código contido no arquivo fisica.js:

```
1. //Chamada da funcao inicial
2. inicio();
3. //variaveis trabalhadas
4. var v; var xzero=30;  var yzero =475;  var deg; var t=1; var g=9.80665;
5. var vx, vy;
6. function inicio(){
7.  xzero=30;
8.  yzero =485;
9.  t=1;
10. g=9.80665;
11. vx, vy;
12. v = parseInt(prompt("Digite a velocidade inicial (m/s): "));
13. grau  = parseInt(prompt("Digite o angulo aplicado (em graus): "));
14.                 //chamada fufuncao coseno
15.                 vx = (getcosDeg(grau));
16.                 //chamad funcao seno
17.                 vy = (getsinDeg(grau));
18. }                           //funcao pra gerar coseno
19.                             function getcosDeg(grau) {
20.                             var rad = grau * Math.PI/180;
21.                             if(grau==90|| grau%450==0){
22.                                     return 0;
23.                             }else{
24.                             return (v*(Math.cos(rad)));
25.                         }}
26.                                             //funcao pra gerar seno
27.                                             function getsinDeg(grau) {
28.                                             var radi = grau*Math.PI/180;
29.                                             return (v*(Math.sin(radi)));
```

```
30.                                                 }
31. //criando cenário no canvas atraves da funcao setup
32. function setup() {
33. //eixo X,Y
34.   createCanvas(1080,500);
35.   frameRate(30);
36. }//funcao da animacao no canvas
37. function draw() {
38.                     background(200); //cor amarelo do fundo da tela
39.                     textSize(20); //tamanho do texto inserido na pagina
40.  if (yzero<490){
41. //partindo da esquerda pra direita, posicao de x da elipse
42.           xzero = (xzero + vx*t);
43. //partindo de cima para baixo, posicao de y da elipse
44.           yzero = (yzero -vy*t +(g*t*t)/2);
45.           t = t+(0.1); //foi escolhido um intervalo de tempo pequeno para poder
    observar melhor o movimento da bola
46.           }else if (yzero>490){
47.           //funcao em looping para sempre retornar a pergunta da velocidade e
    angulo
48.           //inicio();
49.           }
50. //calculo real da posicao de x
51. //encontrando o tempo
52. var tr;
53. tr =(2*vy)/g;
54. //velocidade final em Y, y'(t) = Vo +gt
55. var vfy = (vy - g*(tr));
56. // altura maxima, onde V(t) = 0
57. var tmp = vy/g;
58. var alt = +vy*tmp -(g*tmp*tmp)/2;
59. if (yzero>480){
60. var x  = parseFloat(vx*tr);
61. text("Posicao Xfinal: "+x+ " m", 700, 230);//posicao final de x
62. text("Altura Max: "+alt+ " m", 700, 260);//altura max y
63. text("Vx: "+vx+ " m/s", 700, 290);//velocidade de x
64. text("Vy: "+vy+ " m/s", 700, 320);//velocidade de y
65. text("Tempo de queda "+tr+ " s", 700, 350);//tempo total de queda
66. }
```

```
67.                 ellipse(xzero, yzero, 20, 20); //esse e o elipse que se movienta no cenario
68. }
```

Tabela 2: Código do arquivo fisica.js

- As anotações do código estão na cor vermelha

Abaixo, o código do arquivo index.html:

```
1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2. <html><head>
3. Azul Y(t)<br> Azul escuro X(t)<br>Cinza Vy(t)
4.  <meta charset="UTF-8">
5.                 <script language="javascript" type="text/javascript" src="libraries/p5.js"></script>
6.  <script language="javascript" type="text/javascript" src="fisica.js"></script>
7.  <meta content="width=device-width, initial-scale=1" name="viewport">
8.    <script src="Chart.min.js"></script>
9.    <style type="text/css">
10.   *{
11.      font-family: calibri;
12.   }
13.   .box {
14.      margin: 0px auto;
15.      width: 70%;
16.   }
17.   .box-chart {
18.      width: 100%;
19.      margin: 0 auto;
20.      padding: 10px;
21.   }
22.   </style>
23.   <script type="text/javascript">
24.      var randomnb = function(){ return Math.round(Math.random()*100)};
25.   </script>  </head><body>
26.   <div class="box">
27.      <div class="box-chart">
28.         <canvas id="GraficoLine" style="width:100%;"></canvas>
29.         <script type="text/javascript">
30. //variaveis trabalhadas
```

```
var g =9.80665;
var y = vy; //velocidade inicial
var t =(2*y)/g; //tempo velocidade final
var Vy = y -g*t; //velocidade no instante final
var tmmp = y/g; //tempo onde a velocidade eh zero (altura maxima)
var Vy2 = y -g*tmmp; //velocidade no instalnte de tempo zero
var alt2 = +vy*tmmp -(g*tmmp*tmmp)/2; //altura maxima
var xt; //posicao de X(t)
var x = vx;
xt = x*t;// equacao pra encontrar posicao final de X
var xtmeio;
xtmeio = x*tmmp; //posicao onde se encontra x na altura maxima;
            var options = {
                responsive:true
            };
            var data = {
                labels: [0,tmmp,t,"t(s)"],
                datasets: [
                    {
                        label: "Velocidade Y",
                        fillColor: "rgba(220,220,220,0.2)",
                        strokeColor: "rgba(220,220,220,1)",
                        pointColor: "rgba(220,220,220,1)",
                        pointStrokeColor: "#fff",
                        pointHighlightFill: "#fff",
                        pointHighlightStroke: "rgba(220,220,220,1)",
                        data: [y,Vy2 ,Vy]    },
                    {
                        label: "Posicao Y",
                        fillColor: "rgba(151,187,205,0.2)",
                        strokeColor: "rgba(151,187,205,1)",
                        pointColor: "rgba(151,187,205,1)",
                        pointStrokeColor: "#fff",
                        pointHighlightFill: "#fff",
                        pointHighlightStroke: "rgba(151,187,205,1)",
                        data: [0,alt2,0]},
                    {
                        label: "Posicao X",
```

```
69.                     fillColor: "rgba(130,130,130,0.2)",
70.                     strokeColor: "rgba(130,130,130,1)",
71.                     pointColor: "rgba(130,130,130,1)",
72.                     pointStrokeColor: "#fff",
73.                     pointHighlightFill: "#fff",
74.                     pointHighlightStroke: "rgba(130,130,130,1)",
75.                     data: [0,xtmeio,xt]
76.                 }
77.             ]
78.         };
79.         window.onload = function(){
80.  var ctx = document.getElementById("GraficoLine").getContext("2d");
81.             var LineChart = new Chart(ctx).Line(data, options);
82.         }
83. </script>  </div>    </div></body></html>
```

Tabela 3: Código do arquivo index.html

Para o arquivo Chart.min.js, devido à complexidade de elaborar um gráfico do zero, decidi pegar um modelo pronto em javascript, no qual foi utilizado o recurso da biblioteca Chart.JS. Com isso, pude adaptá-lo para as demandas do programa que foi desenvolvido.

Arquivo Chart.min.js:

Download arquivo

Tabela 4: Link para download referente ao modelo de gráfico utilizado

Segue o link para download o programa:

Download aqui

Tabela 5: Link para download do programa

3.1 Definição da lógica aplicada

Para elaboração da animação do programa, onde aparece uma elipse sendo lançada em um plano, foi utilizada o recurso do Canvas. Canvas é um elemento em HTML muito utilizado para o desenvolvimento de gráficos, desenhos e animações, tudo isso usando a linguagem Javascript.

O eixo de coordenadas do Canvas começa na esquina superior a esquerda da tela.

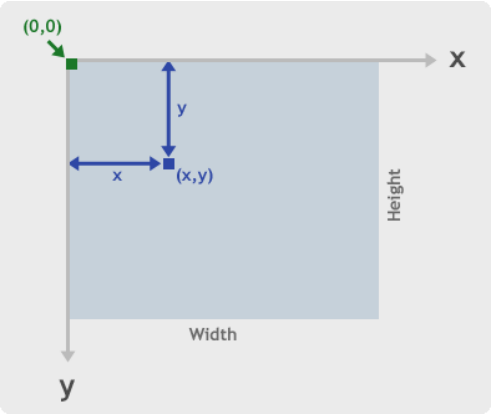

Ilustração 3: Coordenadas do Canvas

As variáveis manipuladas na animação do projétil (uma elipse) que aparece dentro do navegador, no cenário do Canvas, diferem das variáveis contidas nos resultados imprimidos na tela. Pois a animação segue o eixo **Y** de sentido de cima para baixo. Devido a isso, tive que manipular outras variáveis com o eixo de coordenadas sendo o convencional para poder facilitar a impressão correta dos dados mostrados no programa.

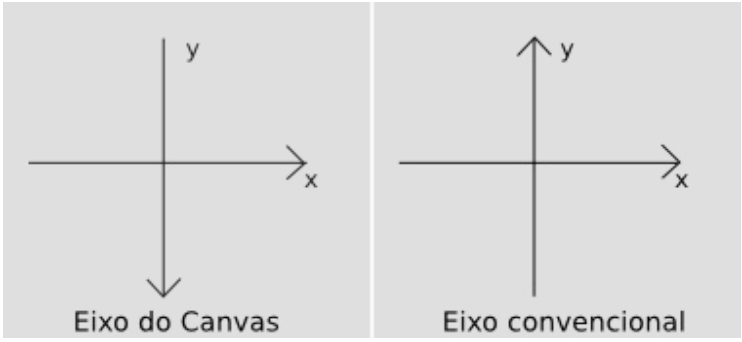

Ilustração 4: Coordenadas

Para manipulação de ambas as situações (comportamento da elipse e dos dados demonstrados no programa), tive que adotar os conceitos de lançamento oblíquo.

Sabendo-se que a velocidade é uma grandeza vetorial, tive que decompor em relação aos eixos **X** e **Y**. Ficando assim:

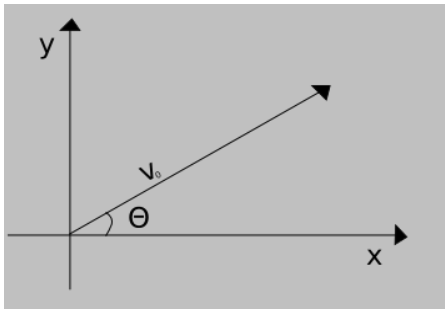

Ilustração 5: Demonstração da velocidade aplicada

$V_{0x} = V_0 . \cos(\theta)$

$V_0y = V_0 . \sin(\theta)$

A partícula, em seu ponto de início, se encontra na origem do seu eixo de referência, isso faz com que as posições iniciais $\mathbf{X_0}$ e $\mathbf{Y_0}$ sejam iguais a zero. Com isso, temos que:

$$X = X_0 + V_{0x} \cdot \cos(\theta) \cdot t \quad \Rightarrow \quad X = V_{0x} \cdot \cos(\theta) \cdot t$$

Tabela 6: Equação posição X(t), MRU

Como o movimento em relação a direção no eixo **X** não sofre influências em relação a gravidade, é adotado que o movimento é retilíneo uniforme (MRU).

Para o eixo **Y**, temos que:

$$Y = Y_0 + V_{0y} \cdot \sin(\theta) \cdot t + (½) \cdot g \cdot t^2 \quad \Rightarrow \quad Y = V_{0y} \cdot \sin(\theta) \cdot t + (½) \cdot g \cdot t^2$$

Tabela 7: Equação posição de Y(t), MUV

O movimento em relação a direção do eixo **Y** sofre influências em relação a gravidade. Para isso, é adotado o movimento uniformemente variado (MUV).

Para encontrar a distância máxima percorrida tive que isolar o tempo em **Y(t)**. A partícula saí da origem até voltar pra origem (em **Y**), ficando assim:

$$Y = Y_0 = 0$$

$$0 = V_{0y} \cdot \sin(\theta) \cdot t + (½) \cdot g \cdot t^2$$

$$0 = t \cdot (V_{0y} \cdot \sin(\theta)) + (½) \cdot g \cdot t) \quad \ldots \quad - V_{0y} \cdot \sin(\theta) = (½) \cdot g \cdot t$$

$$t = - 2 \cdot (V_{0y} \cdot \sin(\theta)) \cdot (g)^{-1}$$

Tabela 8: Encontrando tempo de queda da partícula

Assumindo que a gravidade tem sentido contrário ao eixo **Y**, temos que:

$$t = 2(V_{0y} \cdot \sin(\theta) \cdot (g)^{-1}$$

Tabela 9: Tempo de queda da partícula

Com o tempo definido, bastou substituir na equação **X(t)** e encontrar a distância máxima percorrida.

Para informar a altura máxima, tive que usar o seguinte raciocínio:

$$V_y(t) = Y'(t) \quad \Rightarrow \quad V_y = V_{0y} \cdot \sin(\theta) + g.t$$

Tabela 10: Definindo velocidade no eixo Y

Sabendo que $\mathbf{V_y} = 0$, isso será o tempo em que a partícula não terá velocidade no eixo **Y** e que atingirá sua altura máxima.

$$V_{0y} . \sin(\theta) + g.t = 0 \quad => \quad t = - (V_{0y} . \sin(\theta)) . g^{-1}$$

Com g<0, então:

$$t = (V_{0y} . \sin(\theta)) . g^{-1}$$

Tabela 11: Tempo em que a velocidade em Y será zero

Com o tempo definido, bastou apenas substituir o valor de **t** para a equação da posição de **Y(t).**

4. Uso do programa

Ao abrir o diretório do programa, foi executado o arquivo inicial, o arquivo index2.html. Ao abrir o programa pede, como dados de entrada, a V_0 (1) e o ângulo aplicado com a horizontal (2).

Ilustração 6: Diretório com o arquivo index2.html

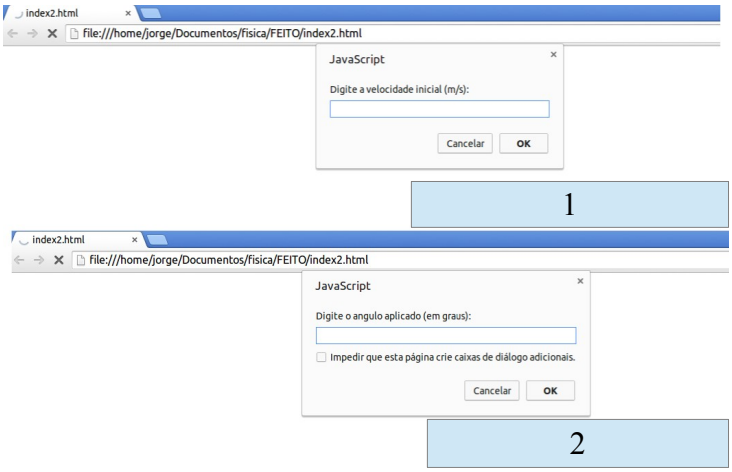

Ilustração 7: Tela inicial do programa

Com isso, para análisar a influência do ângulo dada uma velocidade inicial, foi informado os seguintes valores no programa:

Velocidade em m/s (v_0)	Ângulo em graus θ
19	28º
	35º
	45º
	53º
	60º

Tabela 12: Dados utilizados no programa

Para todos os casos, em relação aos gráficos:

Preto Y(t) ; **Cinza** X(t)

O primeiro caso:

V_0 = 19 m/s ; Ângulo = 28º

Velocidade $_y$: 8,91 m/s

Velocidade $_x$: 16,77 m/s

Altura Máxima: 4,05 m

Tempo de queda: 1,81 s

Distância em X: 30,51 m

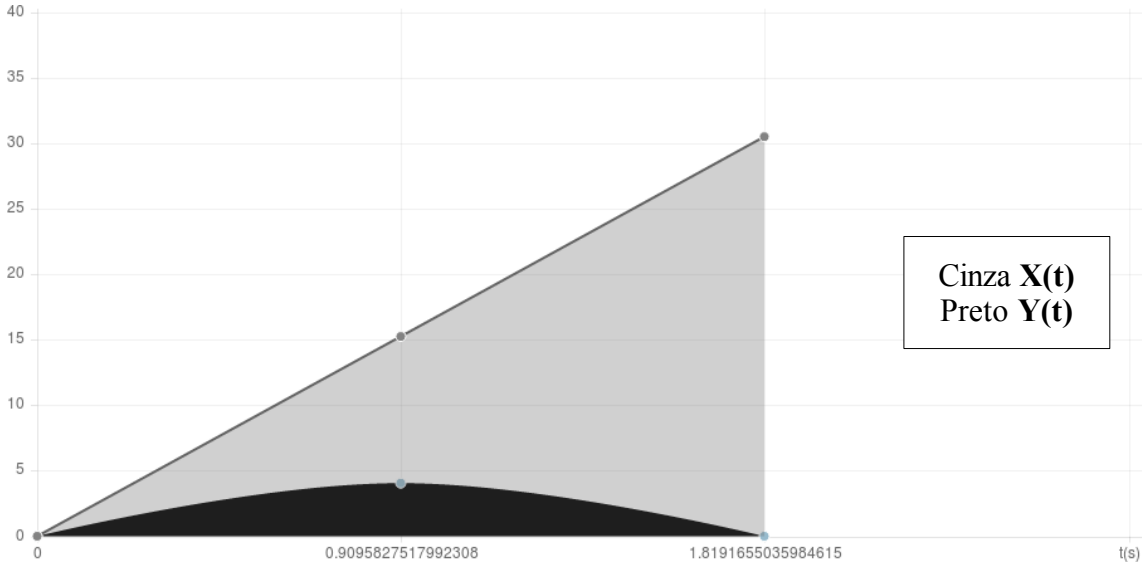

Ilustração 8: Caso 1

O segundo caso:

V_0 = 19 m/s ; Ângulo = 35º

Velocidade $_y$: 10,89 m/s

Velocidade $_x$: 15,56 m/s

Altura Máxima: 6,05 m

Tempo de queda: 2,22 s

Distância em X: 34,59 m

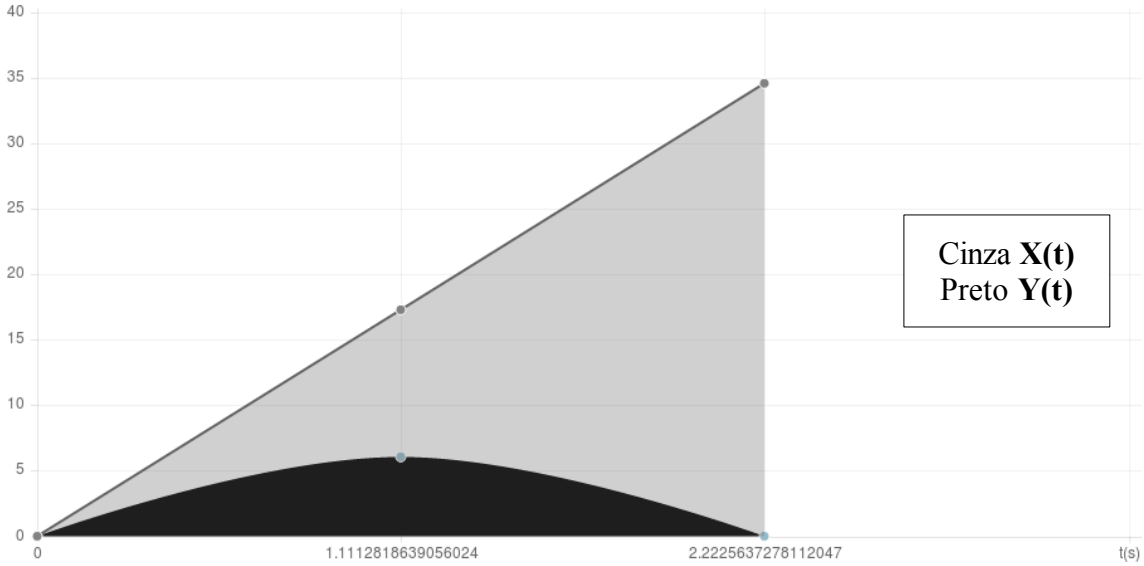

Ilustração 9: Caso 2

O terceiro caso:

V_0 = 19 m/s ; Ângulo = 45º

Velocidade $_y$: 13,43 m/s

Velocidade $_x$: 13,43 m/s

Altura Máxima: 9,20 m

Tempo de queda: 2,73 s

Distância em X: 36 m

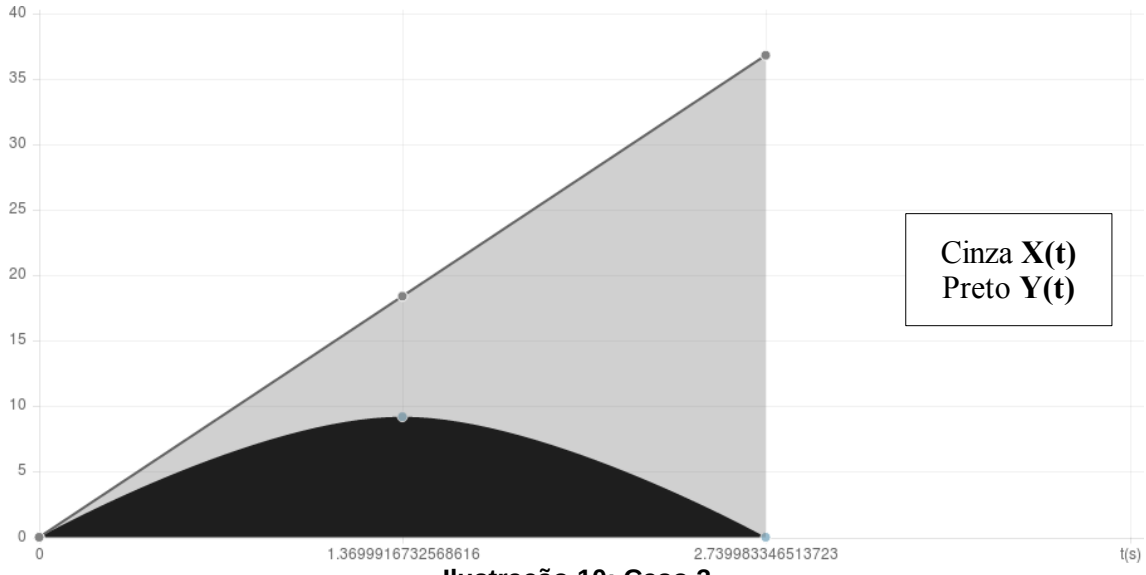

Ilustração 10: Caso 3

O quarto caso:

V_0 = 19 m/s ; Ângulo = 53º

Velocidade $_y$: 15,17 m/s

Velocidade $_x$: 11,43 m/s

Altura Máxima: 11,73 m

Tempo de queda: 3,09 s

Distância em X: 35,38 m

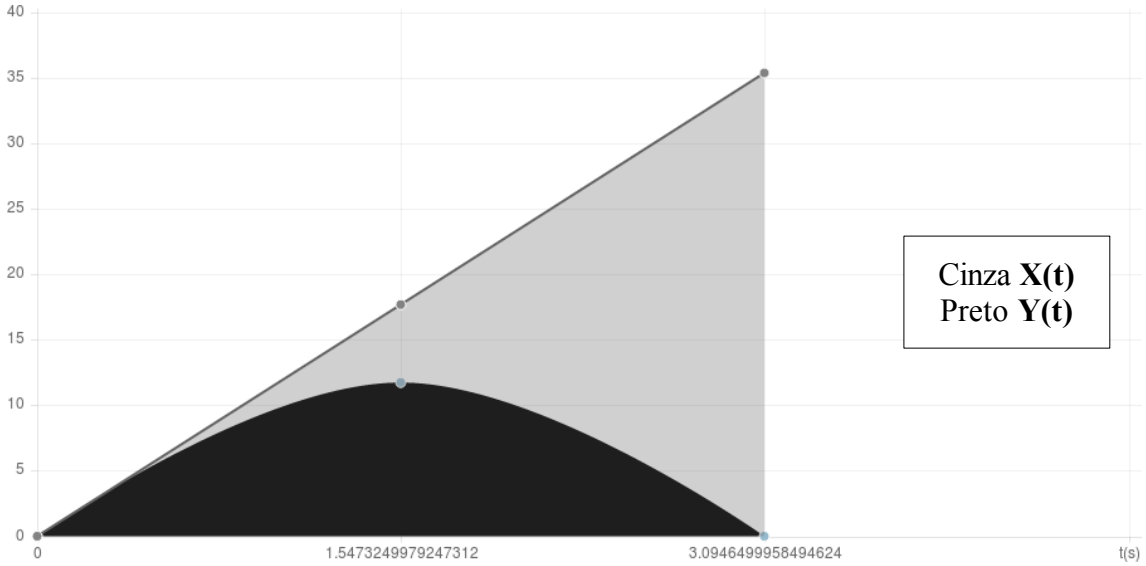

Ilustração 11: Caso 4

O quinto caso:

V_0 = 19 m/s ; Ângulo = 60º

Velocidade $_y$: 16,45 m/s

Velocidade $_x$: 9,50m/s

Altura Máxima: 13,80m

Tempo de queda: 3,35 s

Distância em X: 31,87 m

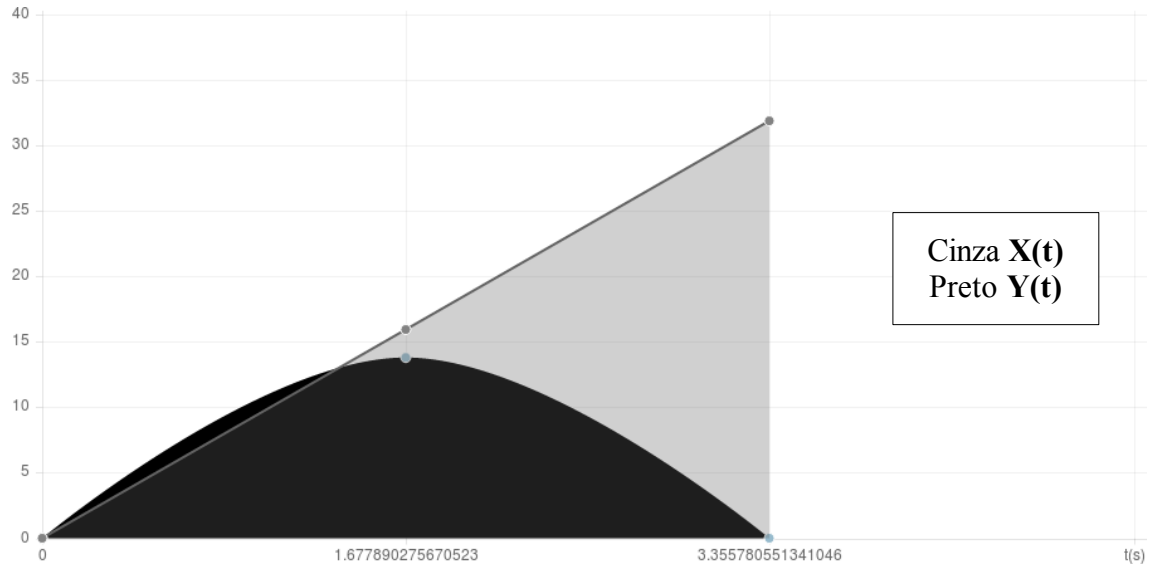

Ilustração 12: Caso 5

5. Coleta dos dados

Casos	Velocidade em metros por segundo	Ângulo em graus	Distância X em metros
1	19 m/s	28	30,51 m
2		35	34,59 m
3		45	36,81 m
4		53	35,38 m
5		60	31,87 m

Tabela 13: Dados coletados no programa

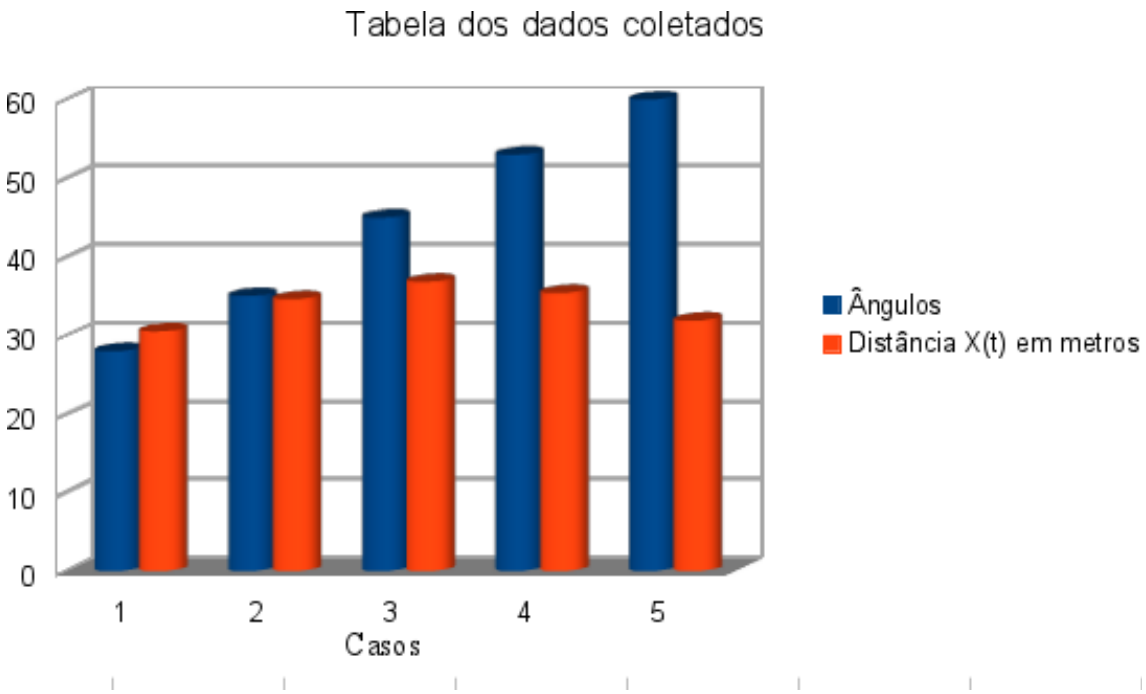

Ilustração 13: Tabela dos resultados obtidos

6. Resultados e observações

De acordo com os dados os obtidos, o ângulo em que foi alcançado a maior distância, dada uma velocidade inicial de 19 m/s, foi o ângulo de 45º.

Esse é o ângulo mais eficiente para se obter o maior alcance no eixo **X,** devido ao fator demonstrado abaixo:

Em todo o lançamento de partícula temos duas componentes

$V_x = V_0 . \cos(\theta)$
$V_y = V_0 . \sin(\theta)$

Na posição horizontal temos apenas movimento retilíneo uniforme (MRU)

Com $X_0 = 0$
$X(t) = V_{0x} . t$

Tabela 14: Equação 1

A velocidade em **X** é constante

$X(t) = V_{0x} . t$
$X'(t) = V_x \quad ... \; Vx = V_0 . \cos(\theta)$

Tabela 15: Velocidade no eixo X

O alcance está ligado a componente horizontal e o quanto de tempo ela permanece em movimento, se tratando de uma partícula.

Na posição vertical temos movimento uniformemente variado (MUV), pois há influência da aceleração da gravidade em seu percurso.

Equação da velocidade em **Y**

$V_y = V_{0y} - g . t$

Tabela 16: Velocidade no eixo Y

Quando a velocidade em $\mathbf{V_{0y}}$ for igual a zero, encontraremos a altura máxima do projétil. Nesse caso, se tratando de uma partícula em que seu ponto de partida $\mathbf{(Y_0)}$ e destino $\mathbf{(Y_f)}$ se encontram na origem do seu eixo de coordenadas, a altura máxima atingida será a metade da trajetória total alcançada em **X**.

$t = (V_y /g)$
Como **t** de subida é o mesmo **t** de descida, logo:
$2t$ = tempo total $\quad ... \; 2t = 2 . (V_y /g)$

Tabela 17: Tempo total

Pegando esse tempo substituindo na equação 1

$X = V_{0x} . t \quad ... \quad X = V_{0x} . (2 . (V_y /g)) \quad ... \quad X = (2/g) . (V_y . V_{0x})$
$X = (2/g). (v_0 . \sin(\theta) \; . v_0 \cos(\theta)) \; ... \quad X = (2v^2_0/g) . (\sin(\theta) . \cos(\theta))$

Tabela 18: Aplicando tempo em X(t)

$V_x = V_{0x}$, devido à velocidade em X não ter aceleração.

A única variável que vamos manipular é o ângulo

$f(\theta) = (2v^2_0/g) . (\sin(\theta) . \cos(\theta))$
$df/d\theta = (2v^2_0/g). [\cos^2(\theta) + (- \sin^2(\theta))]$

Tabela 19: Derivando a função em relação ao ângulo

Encontrando ponto máximo, em que **$df/d\theta = 0$**, em que **$2v^20/g$** é uma constante

$0 = (2v^2 /g). [\cos^2(\theta) + (- \sin^2(\theta))] \quad . . . \quad 0 = [\cos^2(\theta) + (- \sin^2(\theta))]$
$\cos^2(\theta) = \sin^2(\theta) \quad => \quad \cos(\theta) = \sin(\theta)$

Tabela 20: Encontrando o ponto máximo da função

Com isso, entre 0 a 90 graus, o ângulo em que respeita a igualdade

$\cos(\theta) = \sin(\theta)$
$\cos(45) = (2)^{1/2} . (½)$
$\sin(45) = (2)^{1/2} . (½)$

Tabela 21: Ângulos com mesmo valor

7. Conclusão

O trabalho foi bastante interessante, revi muitos assuntos aprendidos no semestre passado (conceitos sobre a linguagem javascript) e pude adaptá-los para os casos que envolvem lançamento oblíquo (aprendidos nas aulas de física clássica).

Tive que me atentar com algumas limitações da própria linguagem JavaScript. Quando fui usar a função *Math.cos* (função *coseno* que se encontra na linha 24 do arquivo **fisica.js**), em ângulos que são múltiplos de 90 (com exceção ao ângulo de 360 graus), o seu resultado dava um número próximo a zero. Isso ocorria em alguns navegadores. Para corrigir esse problema, tive que impor uma condição em que o resultado desse zero (caso o ângulo inserido seja 90 graus o resultado será zero). Essa correção se encontra na linha 21 do arquivo **física.js**.

Outra dificuldade na construção do código, foi referente ao eixo de coordenadas do Canvas (recurso utilizado para animação do lançamento do projétil) e seus dados obtidos. Tive que criar outras variáveis para poder facilitar os resultados imprimidos na tela.

Um fato curioso, para mim, foi a descoberta do por que o ângulo de 45 graus é o mais eficiente para alcançar maiores distâncias. Antes de fazer o trabalho eu já tinha ciência que esse ângulo é o mais competente, porém… desconhecia a forma que foi provada através das derivadas das funções obtidas (explicação no tópico **Resultado** p. 15).

Dessa forma, o trabalho foi concluído, demandou bastante tempo mas foi bastante prazeroso. Pude ampliar mais meus conhecimentos envolvendo casos físicos e aplicações computacionais, duas áreas em que admiro bastante.

8. Referências

- Young & Freedman, Física I, Edição 12
- http://prolina.df.ibilce.unesp.br/walter/fis_geral/proj/projetil2.html
- https://aleteia.wordpress.com/2009/04/06/lancamentos-obliquos-alcance-maximo/
- https://webdesign.tutsplus.com/pt/tutorials/build-a-dynamic-dashboard-with-chartjs--webdesign-14363
- https://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/Canvas_tutorial
- http://br.ccm.net/faq/2680-javascript-introducao-a-linguagem-javascript
- https://www.youtube.com/watch?v=l0K95at8BVg