UNIVERSIDADE FEDERAL DE SÃO PAULO

INSTITUTO DE CIÊNCIA E TECNOLOGIA

LUCAS SAAVEDRA VAZ

Aplicação de Sistemas Computacionais na simulação Balística

São José dos Campos

2018

Introdução:

Neste trabalho foi realizado o projeto e execução de um sistema computacional de simulação balística, levando em conta diversas variáveis, dentre elas: colisões totalmente elásticas e inelásticas, atrito com o ar e massa do objeto. Também foram comparados os resultados entre os diferentes tipos de colisão.

Desenvolvimento:

Modelagem:

O primeiro desafio enfrentado foi encontrar o modelo matemático que rege o sistema balístico com arrasto. A partir da decomposição, em relação ao ângulo de lançamento, da velocidade inicial é possível encontrar as velocidades iniciais nos eixos X e Y do sistema. A partir disso é possível separar nos eixos X e Y as forças que agem no sistema:

$$F_x = F_{at}$$

$$F_y = F_p + F_{at}$$

A partir das definições da Segunda Lei de Newton e da Força de Atrito, podemos substituir nas equações das forças nos eixos:

$$F = ma$$

$$F_{at} = -kv$$

$$ma_x = -kv_x$$

$$ma_y = mg - kv_y$$

Portanto, organizando as equações e substituindo a aceleração por sua definição em Cálculo:

$$\frac{dv_x}{dt} = -k\frac{v_x}{m}$$

$$\frac{dv_y}{dt} = g - k\frac{v_y}{m}$$

Utilizando o Método de Euler e isolando a velocidade, podemos transformar a equação em algo computacionalmente possível:

$$\Delta v_x = (-k\frac{v_x}{m})\Delta t$$
$$\Delta v_y = (g - k\frac{v_y}{m})\Delta t$$

Como o passo Δt é definido pelo programador, precisamos apenas de Δv, que representa a variação de velocidade. Assim, precisamos somar a variação de velocidade com a velocidade atual para assim obter a velocidade resultante:

$$v_x(t + \Delta t) = v_x(t) \ + (-k\frac{v_x}{m})\Delta t$$
$$v_y(t + \Delta t) = v_y(t) \ + (g - k\frac{v_y}{m})\Delta t$$

Dessa maneira obtivemos funções iterativas, que, com o passo de Δt e com o valor $v(0)$ fornecidos pelo usuário, podemos calcular a velocidade em qualquer momento e consequentemente, se multiplicado pelo passo e com os valores $s_x(0)$ e $s_y(0)$, achar a equação iterativa do espaço:

$$s_x(t + \Delta t) = s_x(t) \ + v_x\Delta t$$
$$s_y(t + \Delta t) = s_y(t) \ + v_y\Delta t$$

Assim, com as equações de espaço e velocidade definidas, podemos achar a energia cinética em qualquer ponto:

$$E_c(t) = \frac{m}{2}\sqrt{v_x^2(t) + v_y^2(t)}$$

Implementação:

Com todas as equações necessárias definidas, podemos realizar a simulação física. Primeiramente é necessário obter todos os dados iniciais fornecidos pelo usuário e o tipo de colisão a ser realizada:

```
printf("Initial Velocity (M/S):\n");
scanf("%f",&V0);
printf("Gravity (Absolute value) (M/S^2):\n");
scanf("%f",&G);
printf("Initial Height (Meters):\n");
scanf("%f",&Sy);
printf("Initial Distance (Meters):\n");
scanf("%f",&Sx);
printf("Launch Angle (Degrees):\n");
scanf("%f",&Theta);
printf("Projectile Mass (Kg):\n");
scanf("%f",&M);
printf("Drag Coefficient:\n");
scanf("%f",&K);
printf("Enter \'1\' for elastic collision and \'2\' for inelastic collision:\n");
scanf("%d",&Collision);
```

Com os parâmetros iniciais definidos, podemos começar a calcular os valores de cada ponto no espaço, velocidades e energia, enquanto o objeto se mover ou o tempo de simulação for menor que 10s (sempre verificando com as condições de cada tipo de colisão:

```
while (((abs(Vy)>=0.001)||(Vx>=0.001))&&(Time<=10)){
    if (Sy<0){
      Sy=0;
      if (Collision == 1){  //elastic
     Vy=-Vy;
      } else{
     Vy=0;
     Vx=0;
     Ec=0;
      }
    }
```

```
        fprintf(file_ptr,"    %f    %f        %f    %f    %f
%f\n",Sx,Sy,Vx,Vy,Ec,Time);
        Vx=Vx+(-K/M*Vx)*DELTA_T;
        Vy=Vy+(-G-K/M*Vy)*DELTA_T;
        Time+=DELTA_T;
        Sx=Sx+Vx*DELTA_T;
        Sy=Sy+Vy*DELTA_T;
        Ec=M*sqrt(pow(Vx,2)+pow(Vy,2))*sqrt(pow(Vx,2)+pow(Vy,2))/2;
    }
```

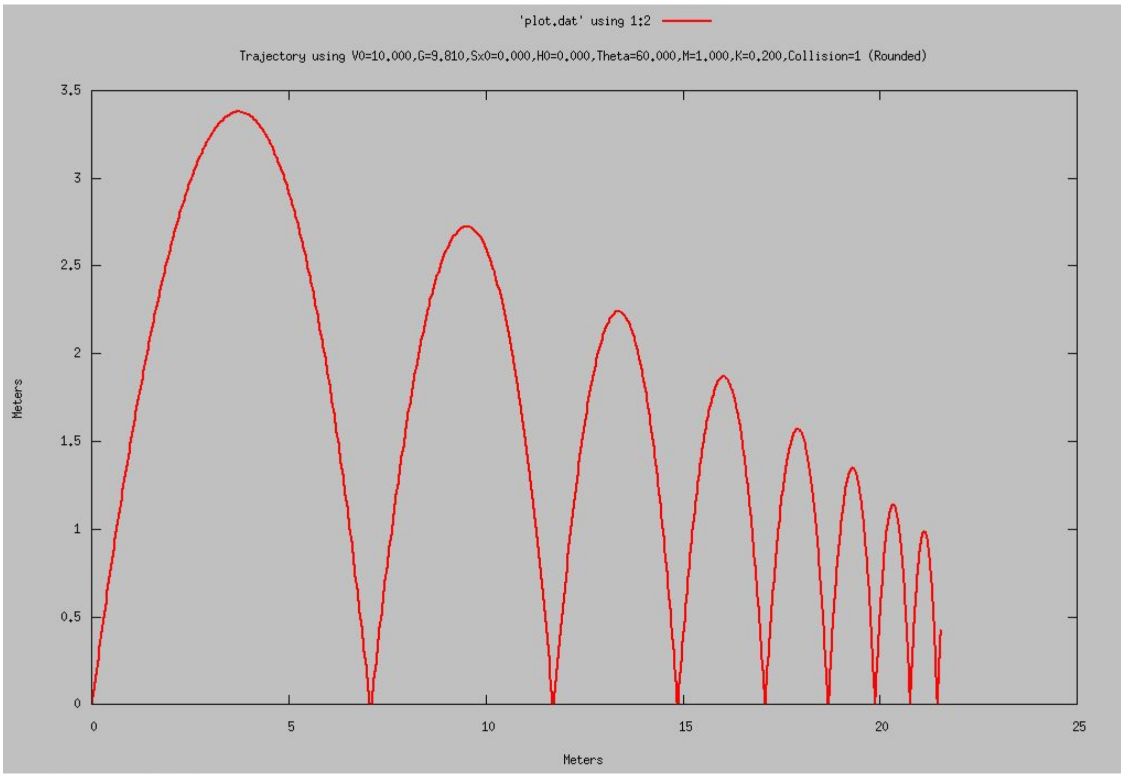
A cada ponto calculado, os valores são escritos em um arquivo organizado para funcionar com o *Gnuplot* e assim plotar a trajetória. Em seguida os parâmetros e chamada do *Gnuplot* são passados para o sistema:

```
snprintf(CallPlot,sizeof(CallPlot),"gnuplot -p -e \"set terminal x11 enhanced background rgb \'grey\';set key outside; set key center top;set title \'Trajectory using V0=%.3f,G=%.3f,Sx0=%.3f,H0=%.3f,Theta=%.3f,M=%.3f,K=%.3f,Collision=%.0d (Rounded)\';set xlabel \'Meters\';set ylabel \'Meters\';plot \'plot.dat\' using 1:2 with line lt -1 lw 2 lc rgb \'red\'\"",V0,G,Sx,Sy,Theta,M,K,Collision);
system(CallPlot);
```

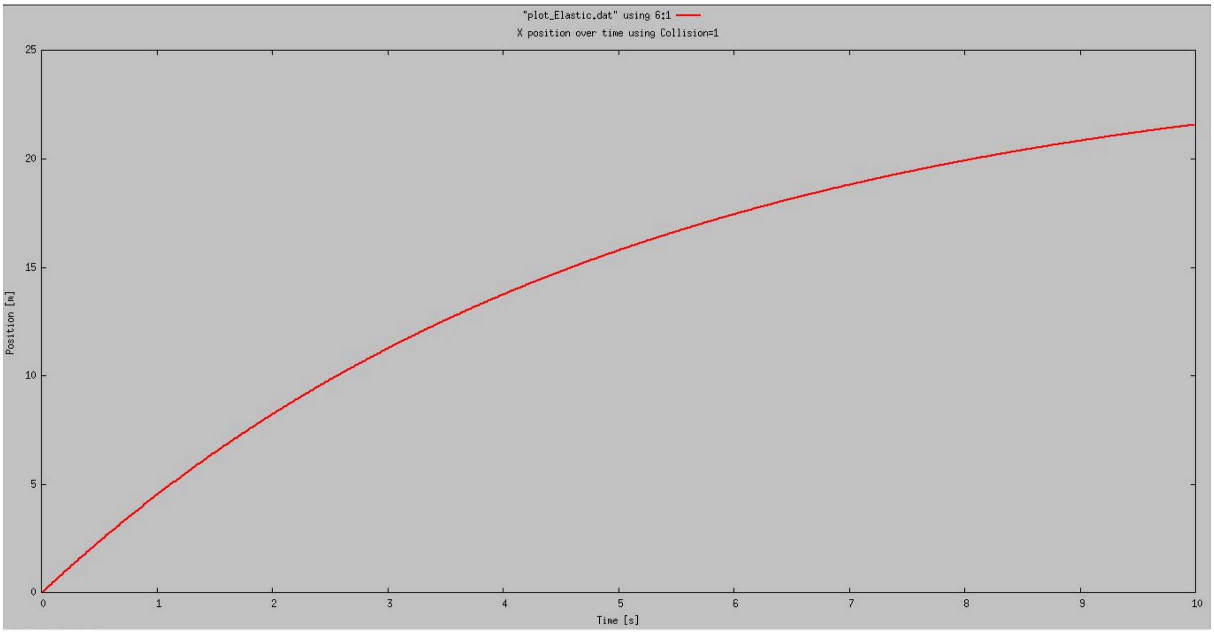
Resultados:

Os gráficos a seguir foram gerados para uma colisão totalmente elástica utilizando V0 = 10 m/s, G=9.81 m/s², Sx0 = 0 m, Sy0 = 0 m, Angulo Theta = 60º, Massa = 1 kg e Coef. de Atrito = 0.2:

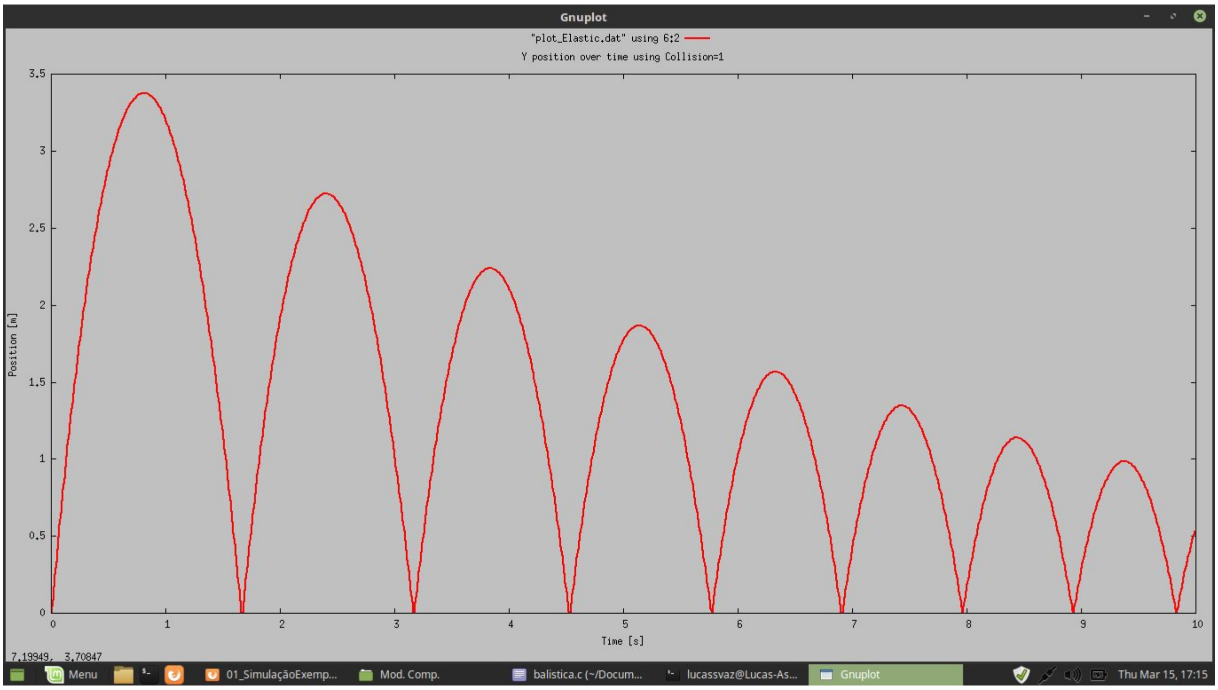
Trajetória:



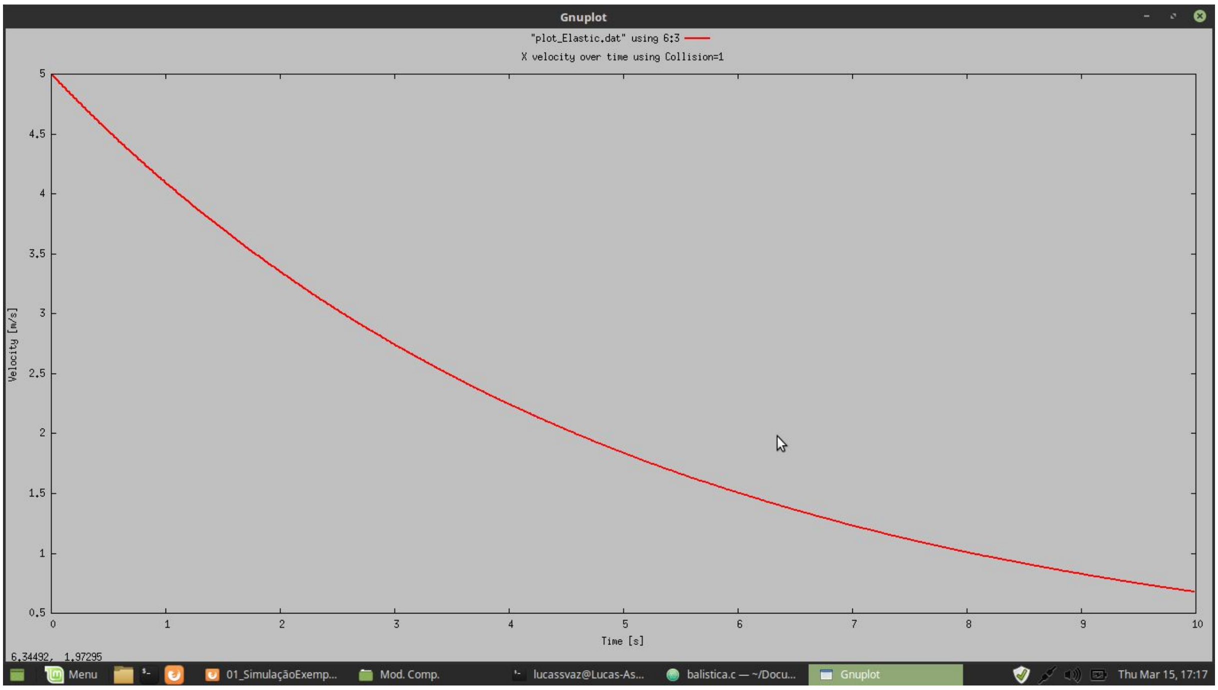
Posição em X:



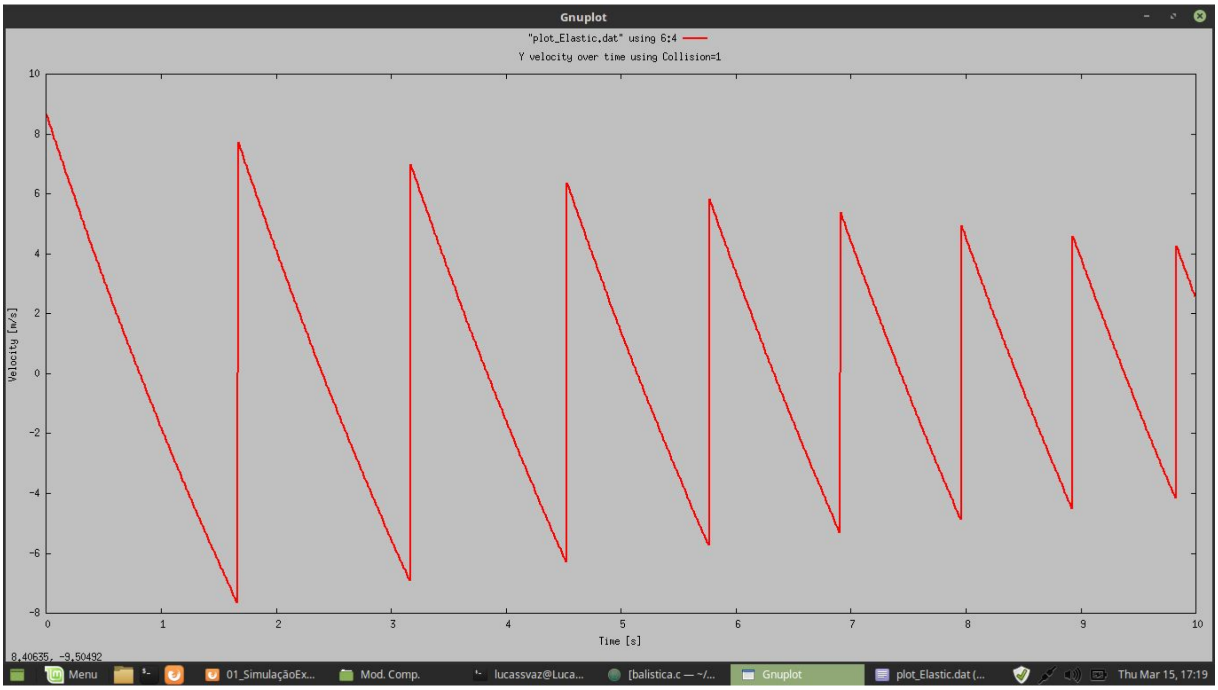
Posição em Y:



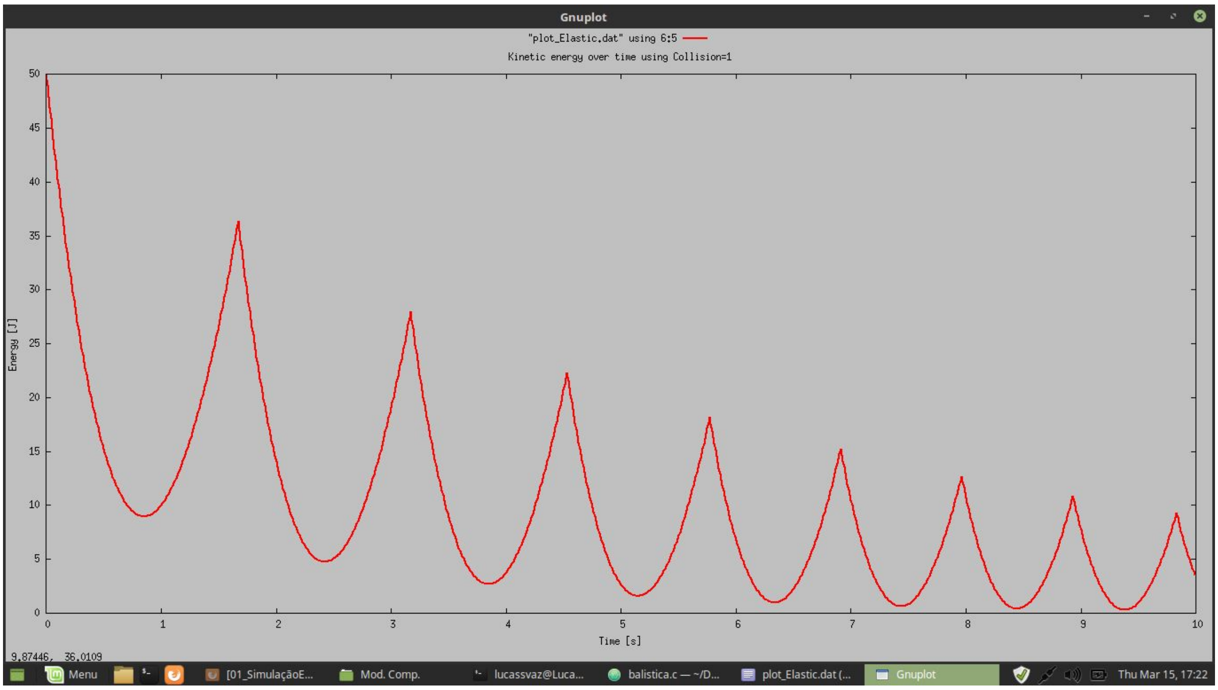
Velocidade em X:



Velocidade em Y:

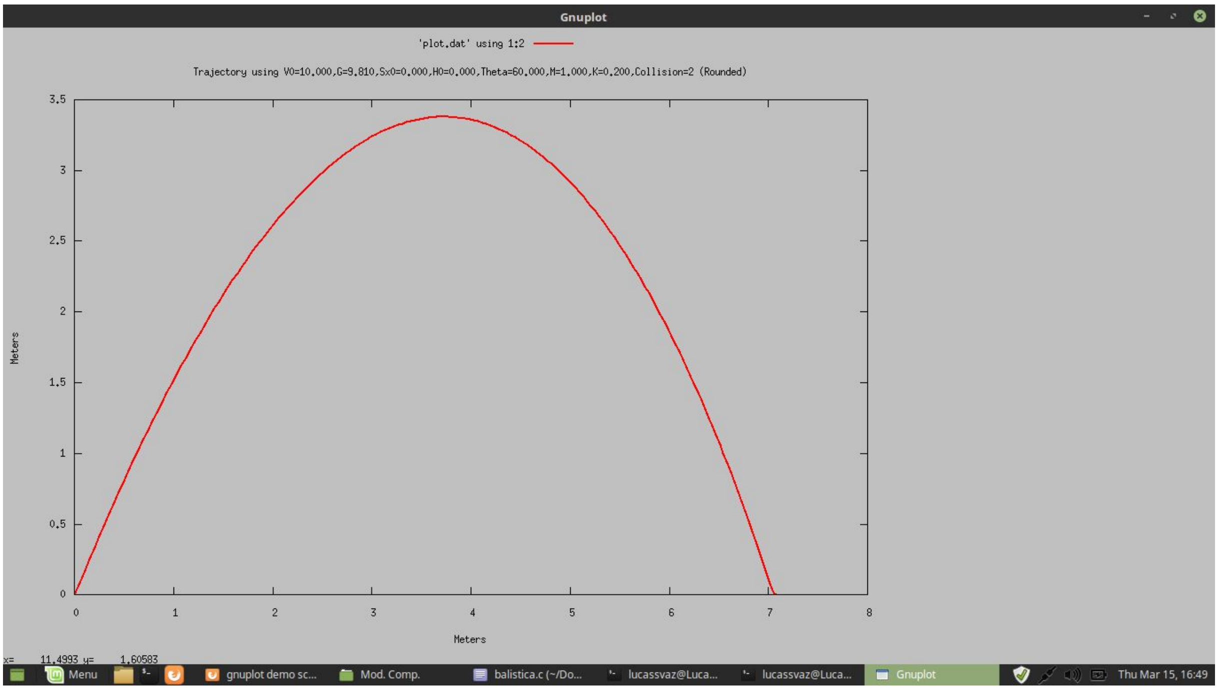


Energia Cinética:

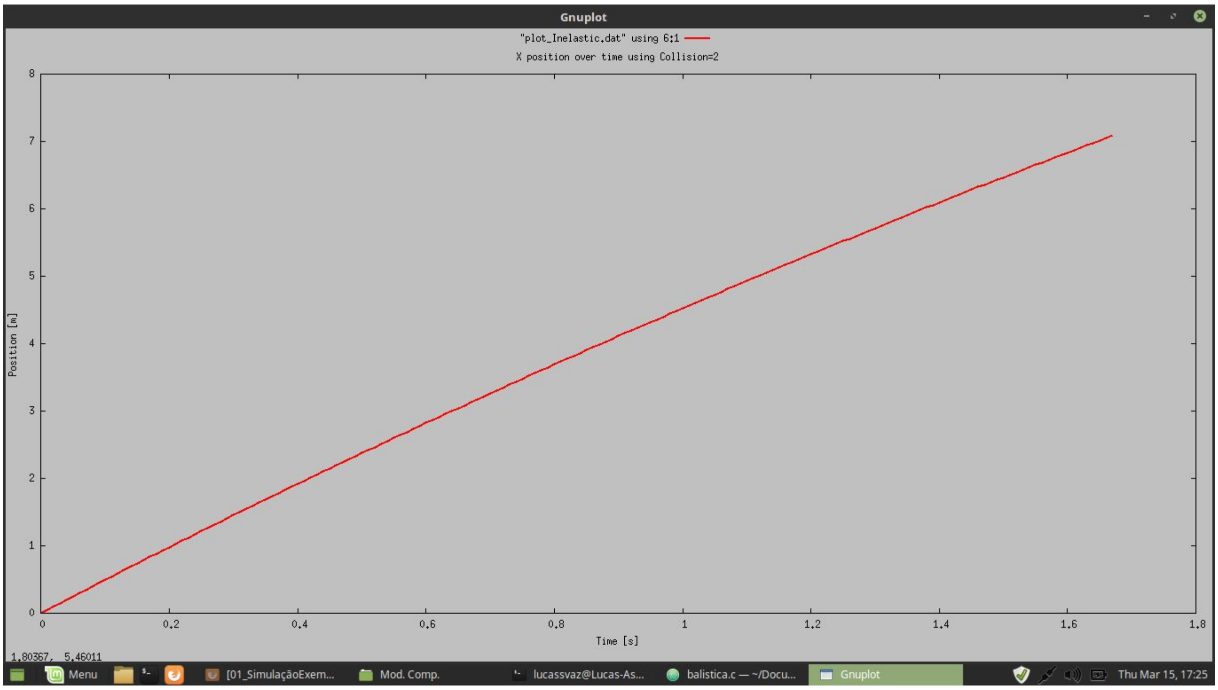


Os próximos gráficos são para uma colisão totalmente inelástica (onde toda energia é transferida na colisão):

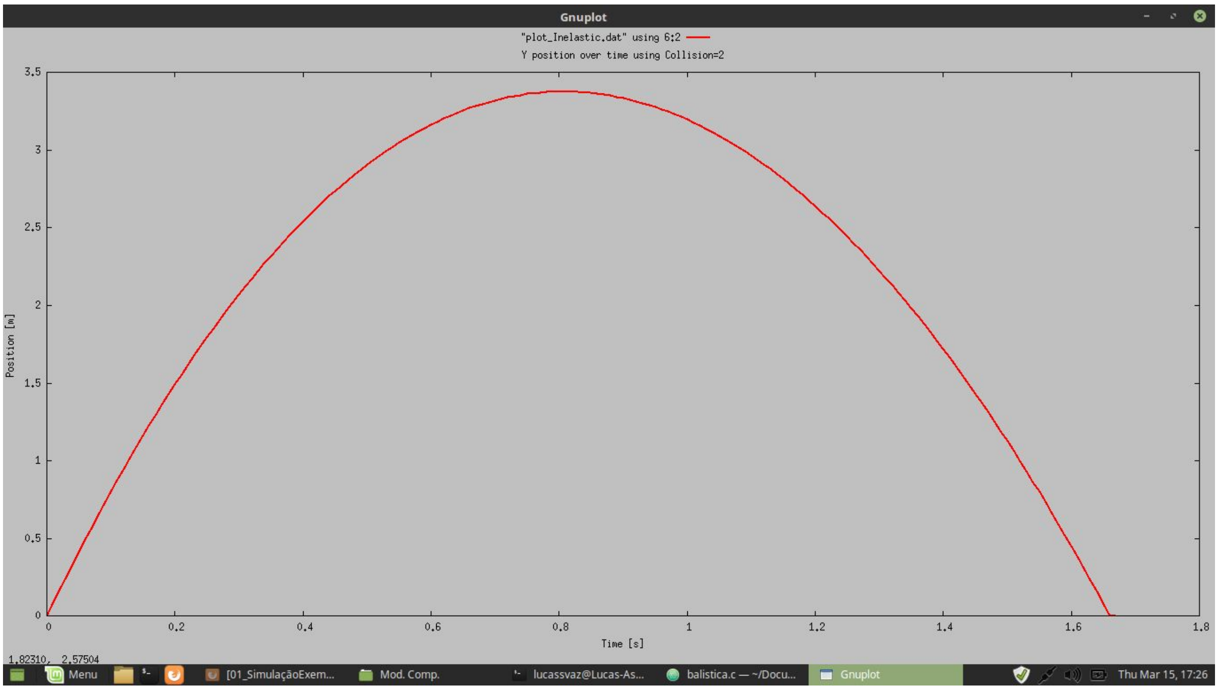
Trajetória:



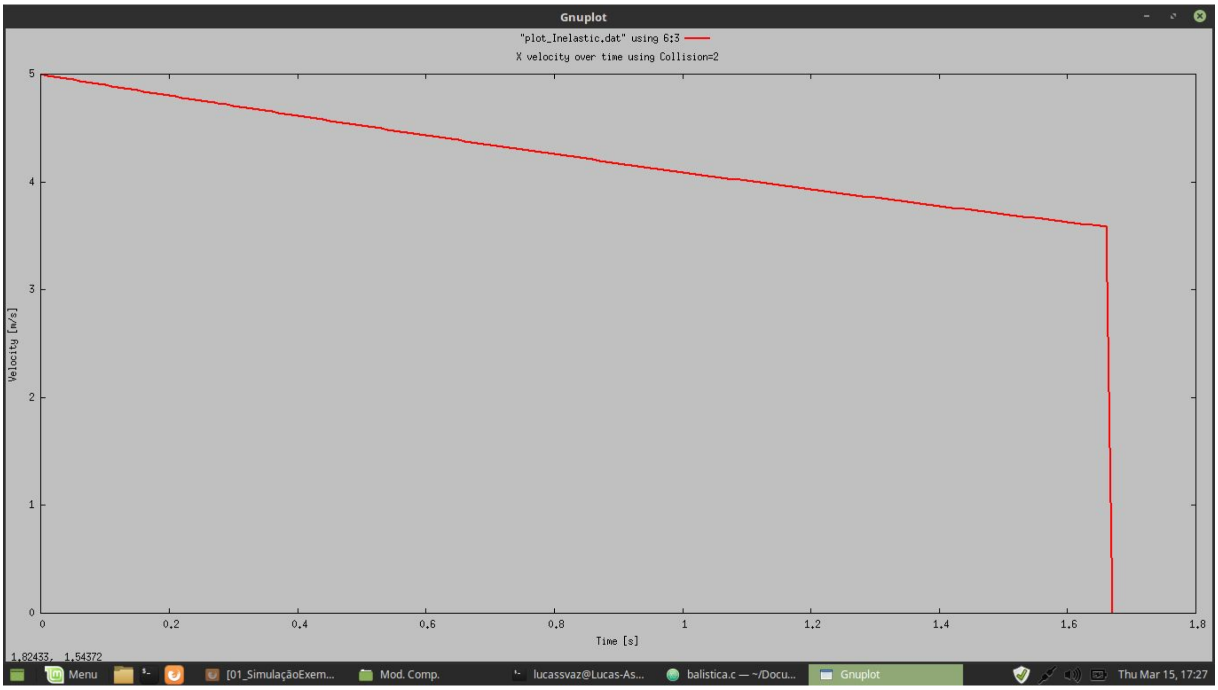
Posição em X:



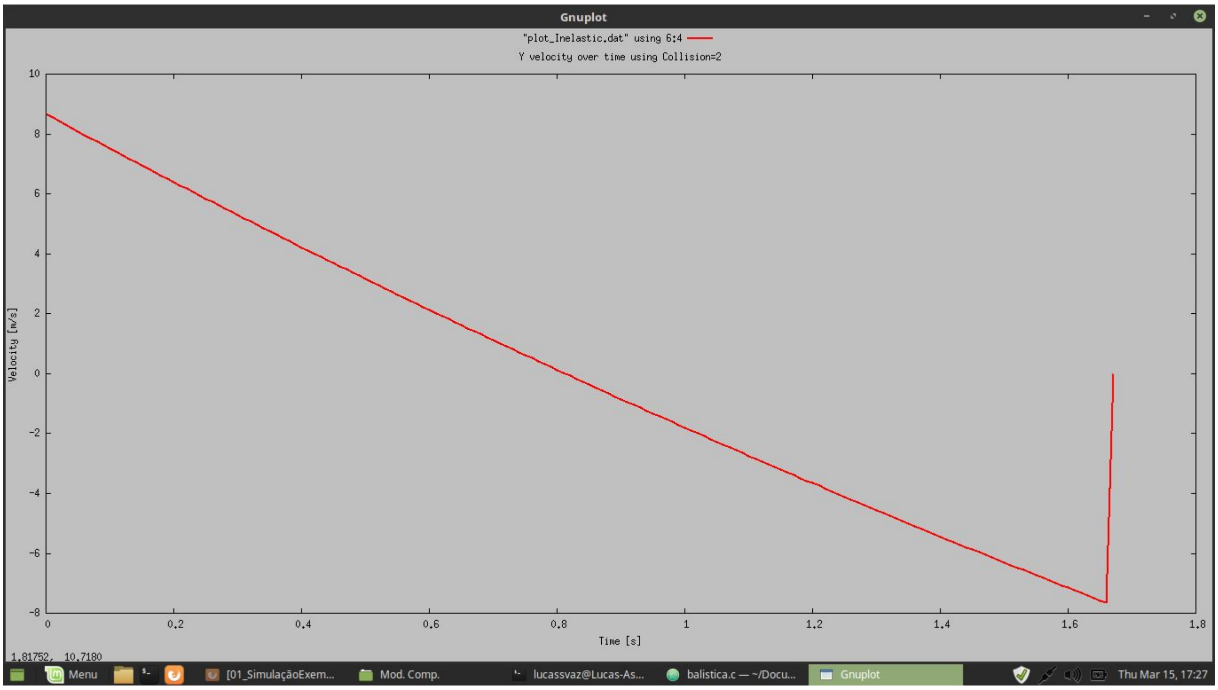
Posição em Y:



Velocidade em X:



Velocidade em Y:



Energia Cinética: