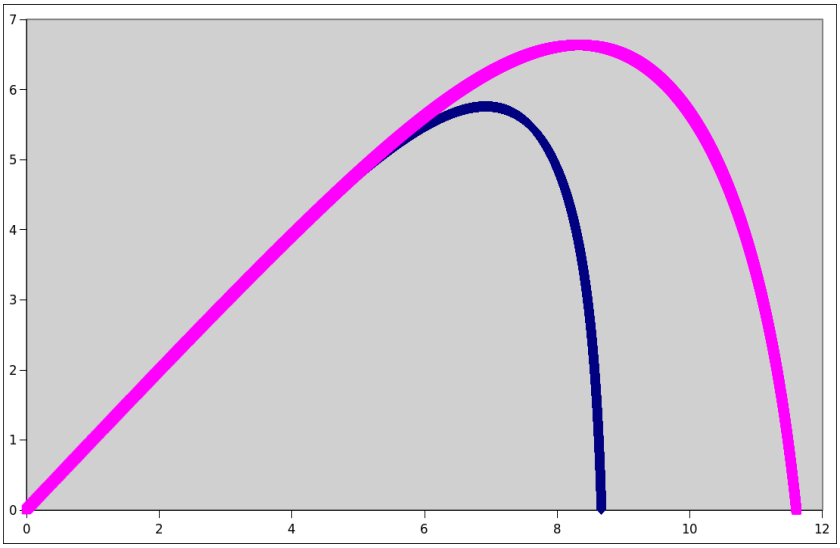


```
#include <iostream>
#include <cmath>
#include <vector>
#include <fstream>
#include <tuple>


int g = 9;
float rho = 1.225;
float C_d = 0.47;
float radius = 0.05;
float A = M_PI * radius * radius;
float m = 0.01;
float gamma_internal = 1.0;
float tau = 0.0001;
float velocityThreshold = 0.01;
float calculateDragForce(float velocity) {
    if (std::abs(velocity) < velocityThreshold) return 0;
    return 0.5 * C_d * rho * A * velocity * velocity;
}
float calculateInternalFriction(float velocity) {
    return gamma_internal * velocity;
}
void updatePositionAndVelocity(float &x, float &y, float &vx, float
&vy, float &t, float dt) {
    float velocity = std::sqrt(vx * vx + vy * vy);
    float dragForce = calculateDragForce(velocity);

    if (std::abs(velocity) < velocityThreshold) {
        dragForce = 0;
    }
```

```
    float ax = -dragForce * (vx / velocity) / m -
calculateInternalFriction(vx) / m - tau * vx;
    float ay = -g - (dragForce * (vy / velocity) / m) -
calculateInternalFriction(vy) / m - tau * vy;
    vx += ax * dt;
    vy += ay * dt;
    x += vx * dt;
    y += vy * dt;
    t += dt;
}
int main() {
    float x = 0.0;
    float y = 0.0;
    float vx = 50.0;
    float vy = 50.0;
    float t = 0.0;
    float dt = 0.00001;
    std::vector<std::tuple<float, float, float, float, float>>
trajectory;
    while (y >= 0 && t < 10.0) {
        trajectory.push_back(std::make_tuple(t, x, y, vx, vy));
        updatePositionAndVelocity(x, y, vx, vy, t, dt);
    }
    std::ofstream outFile("trajectory_corrected.gnumeric");
    if (!outFile) {
        std::cerr << "Error opening file!" << std::endl;
        return 1;
    }
    outFile << "Time (s)\tX Position (m)\tY Position (m)\tX Velocity
(m/s)\tY Velocity (m/s)\n";
    for (const auto& point : trajectory) {
        outFile << std::get<0>(point) << "\t" << std::get<1>(point) <<
"\t"
                << std::get<2>(point) << "\t" << std::get<3>(point) <<
"\t"
                << std::get<4>(point) << std::endl;
    }
    outFile.close();
    std::cout << "Final position: (" << x << ", " << y << ")\n";
    std::cout << "Final velocity: (" << vx << ", " << vy << ")\n";
    std::cout << "Total time: " << t << " seconds\n";

    return 0;
}
```