```cpp
#include <iostream>
#include "MonteCarlo.hpp"
#include <cmath>

MonteCarloResult run_one_trial(
    std::mt19937 &rng,
    const RocketState &x0_nom,
    double launch_angle_nom_deg,
    double dt,
    int    numSteps,
    double t_burn,
    double mass_flow_rate,
    double Cd,
    double A_ref,
    double mass_empty,
    const Vector3 &F_B_nom,
    const Vector3 &M_B_nom,
    const Vector3 &g_I
)
{
    // Random distributions
    std::normal_distribution<double> angle_noise(0.0, 0.5);  // deg
    std::normal_distribution<double> mass_noise(1.0, 0.01);  // scale

    // random angle and mass scale
    double angle_deg  = launch_angle_nom_deg + angle_noise(rng);
    double angle_rad  = angle_deg * M_PI / 180.0;
    double mass_scale = mass_noise(rng);

    RocketState x0 = x0_nom;

    // Randomized pitch
    x0.euler.y = angle_rad;

    // Randomized mass
    x0.mass = x0_nom.mass * mass_scale;

    // Time
    std::vector<RocketState> states;
    states.reserve(static_cast<size_t>(numSteps) + 1);

    simulateEuler(
        x0,
        dt,
        numSteps,
        F_B_nom,
        M_B_nom,
        g_I,
        t_burn,
        mass_flow_rate,
        Cd,
        A_ref,
        mass_empty,
        states
    );

    double max_alt      = 0.0;
    double impact_time  = dt * numSteps;
    double ground_range = 0.0;

    for (int k = 0; k < states.size(); ++k)
    {
        const RocketState &s = states[k];
        double t = k * dt;

        if (s.r_I.z > max_alt)
        {
            max_alt = s.r_I.z;
        }

        if (s.r_I.z <= 0.0 && k > 0 && s.v_I.z < 0.0)
        {
            impact_time = t;
            break;
        }
    }

    const RocketState &xf = states.back();
    ground_range = std::sqrt(
        xf.r_I.x * xf.r_I.x +
        xf.r_I.y * xf.r_I.y
    );


    MonteCarloResult res;
    res.launch_angle_deg = angle_deg;
    res.mass_scale       = mass_scale;
    res.impact_time      = impact_time;
    res.ground_range     = ground_range;
    res.max_altitude     = max_alt;
    return res;
}
```