

Chaos in the Three Body Problem

Restricted Planar Circular Three-Body Problem

The gravitational three-body problem is not integrable. Trajectories may be chaotic, depending on the masses of the three bodies and the initial conditions. To simplify the problem as much as possible, assume that one of the bodies has negligible mass (restricted), that the trajectories are confined to a plane in 3-dimensional space (planar), and that the two massive bodies are in a Kepler orbit with zero eccentricity (circular).

This is a reasonable approximation if the massive bodies are taken to be the Sun and Jupiter, and the massless body an asteroid, or even a lighter planet like Earth. This system is straightforward to simulate and study. “Jupiter! The Three Body Problem”[1] is a good web reference with demonstration programs.

Co-rotating Frame of Reference

Because the two massive bodies move in perfect circular orbits about their center of mass, it is convenient to choose a frame of reference in which they are stationary. In this co-rotating or synodic reference frame we only need to consider the motion of the third massless body.

Choose units such that

$$G_N = 1, \quad m_1 + m_2 + m_3 = m_1 + m_2 = 1, \quad |\mathbf{r}_1 - \mathbf{r}_2| = 1. \quad (1)$$

Choose the center of mass of the two massive bodies to be the origin of coordinates. Choose x and y axes so that

$$\mathbf{r}_1 = (-\mu, 0), \quad \mathbf{r}_2 = (1 - \mu, 0), \quad \mu = \frac{m_2}{m_1 + m_2}. \quad (2)$$

From Kepler’s third law, the period and angular speed of the circular orbits are given by

$$T = 2\pi, \quad \Omega = \frac{2\pi}{T} = 1. \quad (3)$$

In the co-rotating reference frame, the third mass has coordinates (x, y) and generalized momenta

$$p_x = \dot{x} - y, \quad p_y = \dot{y} + x. \quad (4)$$

Its

The coordinates of m_3 in the inertial center of mass frame can be computed from the coordinates in the co-rotating frame:

$$(x \cos(t) - y \sin(t), x \sin(t) + y \cos(t)). \quad (5)$$

The Hamiltonian function in the co-rotating frame is

$$H = \frac{1}{2}(p_x^2 + p_y^2) + p_x y - p_y x - \frac{1 - \mu}{r_1} - \frac{\mu}{r_2}, \quad (6)$$

where $r_1 = \sqrt{(x + \mu)^2 + y^2}$ and $r_2 = \sqrt{(x - 1 + \mu)^2 + y^2}$.

The Jacobi integral

$$C \equiv -2H = x^2 + y^2 + \frac{2(1 - \mu)}{r_1} + \frac{2\mu}{r_2} - \dot{x}^2 - \dot{y}^2 \quad (7)$$

is conserved. This is the only first integral of motion in the 4-d phase space x, y, p_x, p_y . Since the number of first integrals is less than $4/2 = 2$, it is possible and even likely that some trajectories will be chaotic, which happens for example when $C = 4$.

Hamilton's equations are

$$\frac{d}{dt} \begin{pmatrix} x \\ y \\ p_x \\ p_y \end{pmatrix} = \begin{pmatrix} p_x + y \\ p_y - x \\ p_y - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3} \\ -p_x - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} \end{pmatrix}. \quad (8)$$

C++ Program for Trajectories and Poincaré Section

The following program uses adaptive fourth-order Runge-Kutta to integrate Hamilton's equations in the co-rotating reference frame.

The trajectories in the center-of-mass frame can easily be obtained by rotating all coordinates by angle $2\pi t/T = t$.

Because the Jacobi integral is conserved, the trajectories lie on a 3-dimensional hypertorus in the 4-dimensional phase space.

To visualize the trajectories using a Poincaré section, a 2-dimensional slice of the phase space can be chosen, and the intersection points of the trajectory with this plane recorded.

One possibility is to choose the $x - p_x$ plane located at $y = 0$. To further simplify the section plot, points can be recorded only when $p_y > 0$.

_____ Program 1: <http://www.physics.buffalo.edu/phy410-505/topic4/jupiter.cpp> _____

```

/*
  The Restricted Planar Circular Massless Three-Body Problem
  Choose units: G = 1, M_Sun + M_Jupiter = 1, |r_Jupiter - r_Sun| = 1
  In the co-rotating frame r_Sun = (-mu, 0), r_Jupiter = (1 - mu, 0)
  In these units mu = 0.00095 and
  the physical initial conditions for Earth are
  x = 1/5.203 = 0.192, v_y = 1/sqrt(x) = 2.28
*/

#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;

#include "linalg.hpp" // linear algebra for vector<double>
#include "odeint.hpp" // to use adaptive Runge-Kutta
using namespace cpl;

double mu; // mass of Jupiter

double Jacobi(

```

```

    const vector<double>& trv
) {
    double t = trv[0], x = trv[1], y = trv[2], vx = trv[3], vy = trv[4],
        r1 = sqrt((x + mu) * (x + mu) + y * y),
        r2 = sqrt((x - 1 + mu) * (x - 1 + mu) + y * y);
    return x * x + y * y + 2 * (1 - mu) / r1 + 2 * mu / r2 - vx * vx - vy * vy;
}

bool switch_t_and_y = false;          // to zero in on section point

// derivative vector in co-rotating frame
vector<double> f(
    const vector<double>& trv
) {
    double t = trv[0], x = trv[1], y = trv[2], vx = trv[3], vy = trv[4],
        r1 = sqrt((x + mu) * (x + mu) + y * y),
        r2 = sqrt((x - 1 + mu) * (x - 1 + mu) + y * y);
    vector<double> f(5);
    f[0] = 1;
    f[1] = vx;
    f[2] = vy;
    f[3] = 2 * vy + x - (1 - mu) * (x + mu) / (r1 * r1 * r1)
        - mu * (x - 1 + mu) / (r2 * r2 * r2);
    f[4] = - 2 * vx + y - (1 - mu) * y / (r1 * r1 * r1)
        - mu * y / (r2 * r2 * r2);

    if (switch_t_and_y)
        f /= vy;

    return f;
}

// compute v_y given mu, C, x, and assuming y = 0 and vx = 0
double v_y(
    const double C,          // Jacobi integral
    const double x          // x-coordinate in co-rotating frame
) {
    double y = 0, vx = 0,
        r1 = sqrt((x + mu) * (x + mu) + y * y),
        r2 = sqrt((x - 1 + mu) * (x - 1 + mu) + y * y);

    double vy_sqd =
        - C + x * x + y * y + 2 * (1 - mu) / r1 + 2 * mu / r2 - vx * vx;

    if (vy_sqd < 0) {
        cerr << " Sorry, C = " << C << " too large, cannot solve for v_y"
            << endl;
        exit(EXIT_FAILURE);
    }
}

```

```

    return sqrt(vy_sqd);
}

int main() {
    cout << " Jupiter! The Three-Body Problem\n"
         << " -----\n"
         << " Enter mu in range [0,1]: ";
    cin >> mu;
    double t = 0, C, x, y = 0, vx = 0, vy;
    cout << " Enter x: ";
    cin >> x;
    cout << " Do you wish to specify C? Enter 1 for yes, 0 for no: ";
    bool yes;
    cin >> yes;
    if (yes) {
        cout << " Enter Jacobi Integral C: ";
        cin >> C;
        vy = v_y(C, x);
    } else {
        cout << " Enter v_y: ";
        cin >> vy;
        vy -= x;          // transform to co-rotating frame
    }

    vector<double> trv(5); // (t, x, y, vx, vy) in co-rotating frame
    trv[0] = t;
    trv[1] = x;
    trv[2] = y;
    trv[3] = vx;
    trv[4] = vy;
    cout << " Jacobi Integral C = " << Jacobi(trv) << '\n'
         << " v_y in co-rotating frame = " << vy << endl;

    cout << " Enter number of section points: ";
    int section_points;
    cin >> section_points;
    ofstream trajectory_file("trajectory.data");
    ofstream section_file("section.data");

    RK4 rk4;
    double dt = 0.001;
    double accuracy = 1e-6;
    rk4.set_step_size(dt);
    rk4.set_accuracy(accuracy);

    int crossing = 0;
    while (true) {

        t = trv[0];
        x = trv[1] * cos(t) - trv[2] * sin(t);
        y = trv[1] * sin(t) + trv[2] * cos(t);

```

```

trajectory_file << t << '\t' << x << '\t' << y << '\n';
double y_old = trv[2];

rk4.adaptive_step(f, trv);

if (y_old < 0.0 && trv[2] >= 0.0 && trv[4] >= 0.0) {

    ++crossing;
    switch_t_and_y = true;
    double dt_save = rk4.get_step_size();

    vector<double> trv_map = trv;
    double dy = -trv_map[2];
    rk4.set_step_size(dy);
    rk4.step(f, trv_map);

    switch_t_and_y = false;
    rk4.set_step_size(dt_save);
    t = trv_map[0]; x = trv_map[1]; y = trv_map[2];
    cout << "Crossing " << crossing << " x = " << trv_map[1]
         << " v_x = " << trv_map[3] << " Jacobi = "
         << Jacobi(trv_map) << endl;
    section_file << trv_map[1] << '\t' << trv_map[3] << '\n';
}

if (crossing >= section_points)
    break;

double r = sqrt(trv[1]*trv[1] + trv[2]*trv[2]);
if (r > 100) {
    cout << "Earth at " << r << " escaping Solar System ..." << endl;
    break;
}
}

trajectory_file.close();
section_file.close();
}

```

Homework Problem

Explore the trajectories of the restricted planar circular three-body problem for various values of μ and C . See the references in the notes for ideas on what to look for. Dr. Sethna's "Jupiter" website has some interesting orbit pictures and software you can download and play with. Choose three different trajectories that you find most interesting and describe what you learned by generating and examining them.

References

- [1] J. Sethna, “Jupiter! The Three Body Problem”,
<http://pages.physics.cornell.edu/sethna/teaching/sss/jupiter/jupiter.htm>.
- [2] M. Gidea and J.J. Masdemont, “Geometry of homoclinic connections in a planar circular restricted three-body problem”, http://www.ma.utexas.edu/mp_arc-bin/mpa?yn=06-22.