Numerical Methods for Solving Ordinary Differential Equations in Two-Body Problem

Kamil Chaj

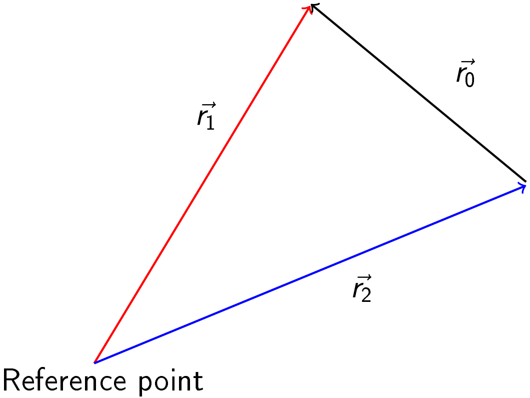
2023/2024

Two-body problem

- in system there exists only two bodies
- bodies have uniform mass distributions and are perfectly symmetrical

Two-body problem

$$\vec{r_0} = \vec{r_2} - \vec{r_1}$$



Derivation

$$\begin{cases} m_1 \frac{d^2 \vec{r_1}}{dt^2} = \vec{F} \\ m_2 \frac{d^2 \vec{r_2}}{dt^2} = -\vec{F} \end{cases} \tag{1}$$

$$\vec{F} = G\frac{m_1 m_2}{r_0^2}\hat{r_0}, \qquad \hat{r_0} = \frac{\vec{r_0}}{r_0} \tag{2}$$

$$\vec{r_0} = \vec{r_2} - \vec{r_1} \tag{3}$$

Derivation

$$\begin{cases} \dfrac{d^2\vec{r_1}}{dt^2} = \dfrac{Gm_2}{||\vec{r_2} - \vec{r_1}||^3}(\vec{r_2} - \vec{r_1}) \\ \dfrac{d^2\vec{r_2}}{dt^2} = \dfrac{Gm_1}{||\vec{r_2} - \vec{r_1}||^3}(\vec{r_1} - \vec{r_2}) \end{cases} \tag{4}$$

Initial conditions

$$\begin{bmatrix} \vec{r_1}(t_0) \\ \vec{r_2}(t_0) \\ \vec{v_1}(t_0) \\ \vec{v_2}(t_0) \end{bmatrix} = \begin{bmatrix} x_1(t_0) & y_1(t_0) & z_1(t_0) \\ x_2(t_0) & y_2(t_0) & z_2(t_0) \\ v_{x1}(t_0) & v_{y1}(t_0) & v_{z1}(t_0) \\ v_{x1}(t_0) & v_{y1}(t_0) & v_{z1}(t_0) \end{bmatrix}, \qquad t_0 = 0 \tag{5}$$

In MATLAB

```matlab
function dydt = base_ode(t, r, m_1, m_2, G)
    r_0 = r(10:12) - r(7:9);
    abs_r_0 = norm(r_0);
    vx1 = G .* m_2 .* r_0(1) ./ abs_r_0 .^3;
    vy1 = G .* m_2 .* r_0(2) ./ abs_r_0 .^3;
    vz1 = G .* m_2 .* r_0(3) ./ abs_r_0 .^3;
    vx2 = - G .* m_1 .* r_0(1) ./ abs_r_0 .^3;
    vy2 = - G .* m_1 .* r_0(2) ./ abs_r_0 .^3;
    vz2 = - G .* m_1 .* r_0(3) ./ abs_r_0 .^3;
    dydt(1:6) = [vx1, vy1, vz1, vx2, vy2, vz2];
    dydt(7:12) = r(1:6);
    dydt = dydt';
end
```

Idea behind numerical methods

$$\frac{d\vec{v}}{dt} = f(\vec{r}, t) \implies \frac{\Delta\vec{v}}{\Delta t} = f(\vec{r}, t_n) \tag{6}$$

$$\Delta\vec{v} = f(\vec{r}, t_n)\Delta t = f(\vec{r}, t_n)h \tag{7}$$

$$\Delta t = h$$

Euler method

$$
\begin{aligned}
\vec{r_{n+1}} &= \vec{r_n} + \Delta\vec{r} \\
&= \vec{r_n} + f(\vec{r_n}, t_{n+1})h
\end{aligned}
\tag{8}
$$

$$
\begin{bmatrix} \vec{r_{n+1}} \\ \vec{v_{n+1}} \end{bmatrix} = \begin{bmatrix} r_n + v_n h \\ v_n + f(\vec{r_n}, t_{n+1})h \end{bmatrix}
\tag{9}
$$

In MATLAB

```
function [t, r] = euler(func, tspan, h, initial_conditions, mass, G)
    k = 1; t(k) = 0;
    r = zeros(round(tspan(2)/h), 12);
    r(1, :) = initial_conditions(:);
    for k = 2:length(r)
        t(k) = t(k-1) + h;
        r(k, :) = r(k-1, :) + h .* func(0, r(k-1, :), mass(1), mass(2), G)';
    end
end
```

4th order Runge-Kutta method

$$\begin{aligned}
k_1 &= f(\vec{r_n}, t_0 + nh) \\
k_2 &= f\left(\vec{r_n} + k_1\frac{h}{2}, t_0 + h(n + \frac{1}{2})\right) \\
k_3 &= f\left(\vec{r_n} + k_2\frac{h}{2}, t_0 + h(n + \frac{1}{2})\right) \\
k_4 &= f\left(\vec{r_n} + k_3 h, t_0 + h(n + 1)\right)
\end{aligned} \tag{10}$$

$$\vec{v_{n+1}} = \vec{v_n} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{11}$$

$$\begin{bmatrix} \vec{r_{n+1}} \\ \vec{v_{n+1}} \end{bmatrix} = \begin{bmatrix} r_n + v_n h \\ v_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{bmatrix} \tag{12}$$

In MATLAB

```
function [t, r] = RK4(func, tspan, h, initial_conditions, mass, G)
    k = 1; t(k) = 0;
    r = zeros(round(tspan(2)/h), 12);
    r(1, :) = initial_conditions(:);
    for k = 2:length(r)
        t(k) = t(k-1) + h;
        k_1 = func(0, r(k-1, :), mass(1), mass(2), G)';
        k_2 = func(0, r(k-1, :) + h .* k_1 ./2 , mass(1), mass(2), G)';
        k_3 = func(0, r(k-1, :) + h .* k_2 ./2 , mass(1), mass(2), G)';
        k_4 = func(0, r(k-1, :) + h .* k_3, mass(1), mass(2), G)';
        r(k, :) = r(k-1, :) + h ./ 6 .* (k_1 + 2.*k_2 + 2.*k_3 + k_4);
    end
end
```

Convergence analysis

$$S_C(R_{n-1}, R_n) = \cos(\theta) = \frac{\langle \vec{R_{n-1}}, \vec{R_n} \rangle}{R_{n-1} R_n} \tag{13}$$

$$D_\theta(R_{n-1}, R_n) = \arccos(S_C(R_{n-1}, R_n)) = \theta \tag{14}$$

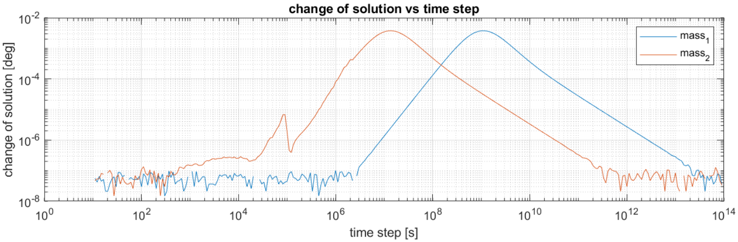


Figure: Euler method

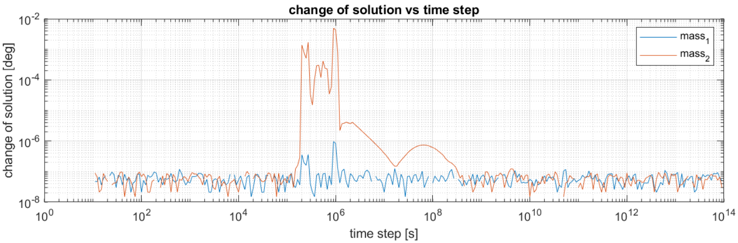


Figure: Runge-Kutta method

in MATLAB

```
function [test_value, dr1, dr2] = err_test(method, mass, initial_conditions, tspan)
    G = 6.67430e-11;
    n = 5e4;
    test_value = logspace(14, 3, 200);
    parfor k = 1:length(test_value)
        [temp1, temp2] = method(@base_ode, tspan, test_value(k), initial_conditions, mass, G);
        [tt, R(:, :, k)] = interpol_arr(temp2, temp1, n);
    end
    parfor k = 2:(length(test_value) - 1)
        norm1 = [norm_arr(R(:, 7:9, k)), norm_arr(R(:, 7:9, k - 1))];
        norm2 = [norm_arr(R(:, 10:12, k)), norm_arr(R(:, 10:12, k - 1))];
        dr1(k) = dot(norm1(:, 1), norm1(:, 2)) ./ (norm(norm1(:, 1)) .* norm(norm1(:, 2)));
        dr1(k) = abs(acos(dr1(k)));
        dr2(k) = dot(norm2(:, 1), norm2(:, 2)) ./ (norm(norm2(:, 1)) .* norm(norm2(:, 2)));
        dr2(k) = abs(acos(dr2(k)));
    end
end
```

Computation time

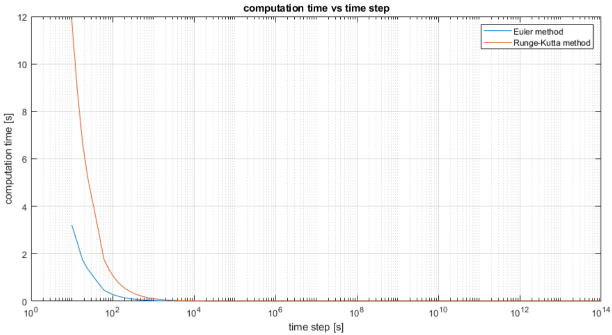


Figure: Comparison of computation time between methods

in MATLAB

```
function [test_value, T] = comp_time(method, mass, initial_conditions, tspan)
    G = 6.67430e-11;
    test_value = logspace(1, 14, 100);
    parfor k = 1:length(test_value)
        tic
        method(@base_ode, tspan, test_value(k), initial_conditions, mass, G);
        T(k) = toc;
    end
end
```

Demo