PHYS580 Lab04 Report

Yicheng Feng
PUID: 0030193826

September 17, 2019

Workflow: I use the Linux system, and code C++ in terminals. For visualization, I use the C++ package – ROOT (made by CERN) to make plots. At the end, the reports are written in LaTeX.

The codes for this lab are written as the following files:

- `runge_kutta.h` and `runge_kutta.cxx` for the class `RungeKutta` to solve general ordinary differential equation sets.
- `pendulum.h` and `pendulum.cxx` for the class `Pendulum` to calculate ODE set of the pendulum.
- `lab4.cxx` for the main function to make plots.

To each problem of this lab report, I will attach the relevant parts of the code. If you want to check the validation of my code, you need to download the whole code from the link `https://github.com/YichengFeng/phys580/tree/master/lab4`.

(1) Modify the provided starter code (or write your own using any appropriate approximation) to study the Poincare section of the phase portrait and those of the time evolution of the dynamical variables θ, ω, and E (energy). Consider both θ-ω and ω-E pairings (2D). Produce the Poincaré sections for some model parameters in the normal periodic (with period $T = 2\pi/\Omega$, where Ω is the driving force angular frequency), chaotic, and period doubling (with period $2T$) regimes, choosing the sectioning and driving frequencies to be the same in all cases. Set the sectioning phase to 0 (i.e., do in phase sectioning). What are the characteristics of the different regimes reflected in these results? Explain how you can be sure that your computational parameters are sufficient to reach your conclusions..

Physics explanation:

We use the same parameters as those in the textbook: pendulum length $L = 9.8$ m, gravity acceleration $g = 9.8$ m/s^2, driving force magnitude $1.35 < f_d < 1.49$, driving force frequency $\Omega_d = 2/3$, section phase 0, time step $\Delta t = 0.04$ s. Initial conditions are $\theta(0) = 0.2$ rad, $\omega(0) = 0$.

To make the error small in the numerical calculation, we use the 4^{th}-order Runge-Kutta (RK4) approximation for this problem. In the Poincare section, we follow the textbook to remove the effect of the initial condition. First, we let the physical pendulum evolute 300 periods of the driving force. After that, the initial condition should decay enough to be neglected. Then, we record another 100 periods of the driving force for the poincare section. The results are shown in Fig. 1. We can see that, inside the range we covered, the **normal period regime** is $1.35 < f_d < 1.424$; the **period doubling regime** is $1.424 < f_d < 1.48$. We can clearly see the 2-period, 4-period regimes, but it is hard to follow the higher-period regimes. From the textbook, $f_d = 1.20$ is inside the **chaotic regime**. Therefore, we pick some specific values of f_d: 1.40, 1.44, 1.2.

The characteristics of the different regimes:

- **normal period regime** $f_d = 1.40$: The time evolution has obvious periodic pattern. The phase diagrams are closed when the initial condition decays away. The Poincare section will reach a single point when stable.

- **period doubling regime** $f_d = 1.44$, 2-period: The time evolution has obvious periodic pattern, but the period is now $2^n T_d$. For this particular case, the period is $2T_d$. The phase diagrams are closed when the initial condition decays away. The Poincare section will reach finite number of points (2^n) when stable, and 2 for this case.

- **chaotic regime** $f_d = 1.20$: The time evolution doesn't have periodic pattern. The effect of the initial condition will not decay away. The phase diagrams are open. The Poincare section will reach infinite number of points if the sampling time is infinitely long.

Explain how you can be sure that your computational parameters are sufficient to reach your conclusions.

The f_d scan of the Poincare section can show clearly the normal period regimes and the period doubling regimes. For the chaotic regimes, we can see Fig. 6 shows no regular pattern, and Fig. 7 is filled with color due to the openness of the phase diagram. Those are the characteristics of chao.

Plots:

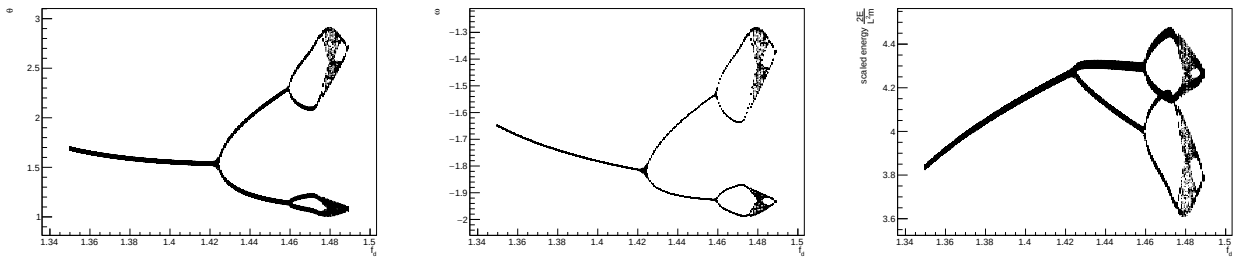

Figure 1: Poincare section of the physical pendulum with section period same as the period of the driving force. The left pad is θ depending on f_d, middle pad ω, and right pad scaled energy.

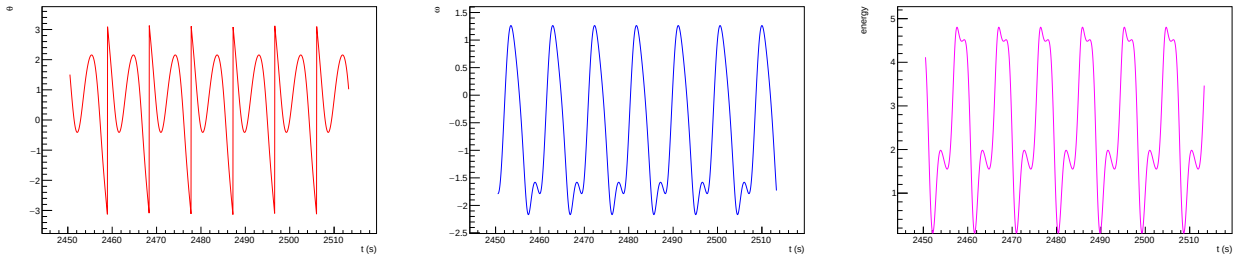

Figure 2: **Normal period regime**: $f_d = 1.40$. Time evolution of θ, ω and energy when stable.

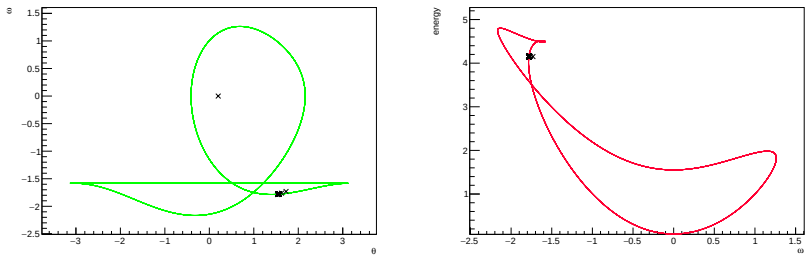

Figure 3: **Normal period regime**: $f_d = 1.40$. Poincare section (black cross markers) on the "**stable**" phase diagram: left pad θ-ω; right pad ω-energy. The single cross marker is the initial condition (or the decay of the initial condition).

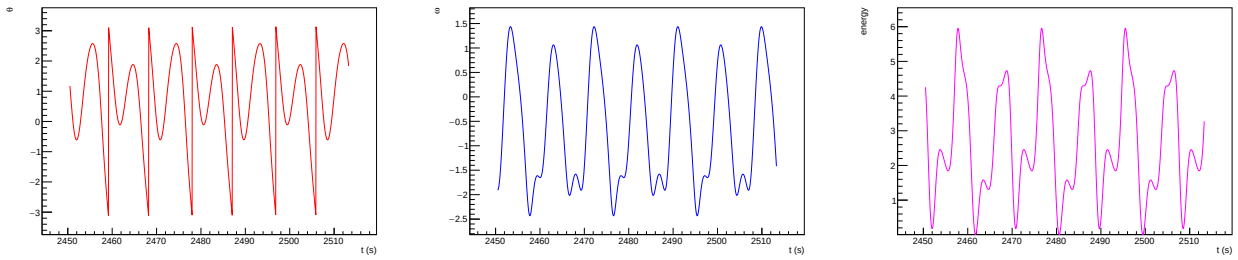

Figure 4: **Period doubling regime**: $f_d = 1.44$, 2-period. Time evolution of θ, ω and energy when stable.

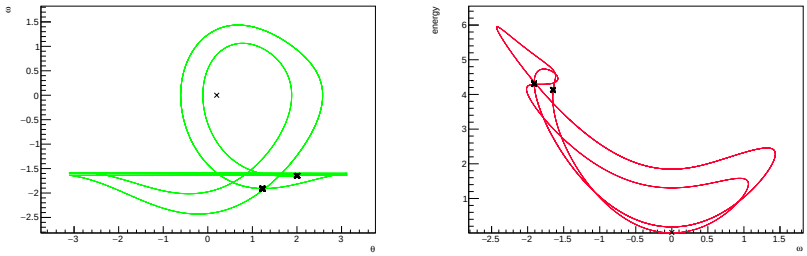

Figure 5: **Period doubling regime**: $f_d = 1.44$, 2-period. Poincare section (black cross markers) on the "**stable**" phase diagram: left pad θ-ω; right pad ω-energy. The single cross marker is the initial condition (or the decay of the initial condition).

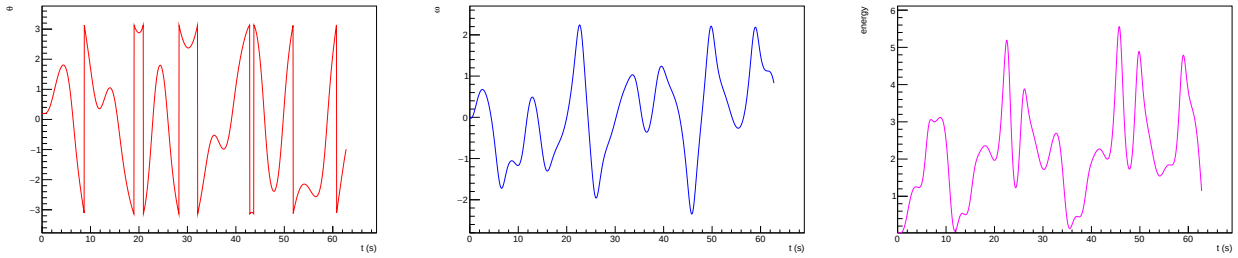

Figure 6: **Chaotic regime**: $f_d = 1.20$. Time evolution of θ, ω and energy.

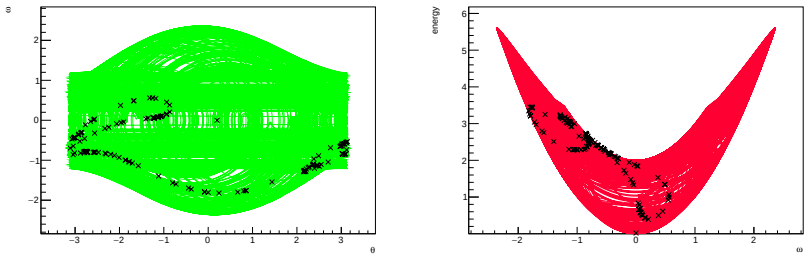

Figure 7: **Chaotic regime**: $f_d = 1.20$. Poincare section (black cross markers) on the phase diagram: left pad θ-ω; right pad ω-energy. The single cross marker is the initial condition (or the decay of the initial condition).

Relevant code:

For the RK4:

```
//-------------------------------------------------------------------------//

void RungeKutta::cal_rk4() {
        if(!check() || !clear()) {
                cout << "WARNING: check() not passed, no calculation!" << endl;
                return;
        }
        //cout << "check() passed" << endl;

        _n_stps = 0;

        double t = _t_start;
        vector<double> x = _x_start;

        vector<double> F1(_n_eqns);
        vector<double> F2(_n_eqns);
        vector<double> F3(_n_eqns);
        vector<double> F4(_n_eqns);

        vector<double> x1(_n_eqns);
        vector<double> x2(_n_eqns);
        vector<double> x3(_n_eqns);
        vector<double> x4(_n_eqns);

        double t1, t2, t3, t4;

        int nfv = _fv.size(); // nfv = _n_eqns
        while(!_stop(t, x)) {
                // Fill into output
                _t.push_back(t);
                for(int ifv=0; ifv<nfv; ifv++) {
                        _x[ifv].push_back(x[ifv]);
                }
                _n_stps ++;

                // F1
                t1 = t;
                for(int ifv=0; ifv<nfv; ifv++) {
                        x1[ifv] = x[ifv];
                }
                for(int ifv=0; ifv<nfv; ifv++) {
                        F1[ifv] = _fv[ifv](t1, x1);
                }
                // F2
                t2 = t + 0.5*_dt;
                for(int ifv=0; ifv<nfv; ifv++) {
                        x2[ifv] = x[ifv] + F1[ifv]*0.5*_dt;
                }
                for(int ifv=0; ifv<nfv; ifv++) {
                        F2[ifv] = _fv[ifv](t2, x2);
                }
                // F3
                t3 = t + 0.5*_dt;
                for(int ifv=0; ifv<nfv; ifv++) {
                        x3[ifv] = x[ifv] + F2[ifv]*0.5*_dt;
                }
                for(int ifv=0; ifv<nfv; ifv++) {
                        F3[ifv] = _fv[ifv](t3, x3);
                }
                // F4
```

```
                t4 = t + _dt;
                for(int ifv=0; ifv<nfv; ifv++) {
                        x4[ifv] = x[ifv] + F3[ifv]*_dt;
                }
                for(int ifv=0; ifv<nfv; ifv++) {
                        F4[ifv] = _fv[ifv](t4, x4);
                }

                t += _dt;
                for(int ifv=0; ifv<nfv; ifv++) {
                        x[ifv] += (F1[ifv]/6 + F2[ifv]/3 + F3[ifv]/3 + F4[ifv]/6)*_dt;
                }
        }

        _t.push_back(t);
        for(int ifv=0; ifv<nfv; ifv++) {
                _x[ifv].push_back(x[ifv]);
        }
        _n_stps ++;
}
```

For the physical pendulum with driving force and dissipation:

```
//------------------------------------------------------------------------//

// x[0]: theta; x[1]: omega
double Pendulum::f_theta(double t, const vector<double> &x) {
        // dtheta/dt = omega
        return x[1];
}

//------------------------------------------------------------------------//

double Pendulum::f_omega_physics(double t, const vector<double> &x) {
        return -_g/_l*sin(x[0]) - _q*x[1] + _F*sin(_O*t+_P);
}

//------------------------------------------------------------------------//

double Pendulum::f_energy_physics(double t, const vector<double> &x) {
        // 2E/m/l^2
        return _g/_l*(1-cos(x[0]))+x[1]*x[1];
}

//------------------------------------------------------------------------//
```

For the solution of the ODE set and Poincare section:

```
//------------------------------------------------------------------------//

void chao_plot(double f_d, double theta_start, double omega_start, double psphi, double psomega) {

        Pendulum pd;
        pd.set_mode(1); // physics pendulum
        pd.set_alg(4);
        //pd.set_theta_start(0.2);
        //pd.set_omega_start(0);
        pd.set_theta_start(theta_start);
        pd.set_omega_start(omega_start);
        pd.set_q(0.5);
        pd.set_F(f_d);
```

```
        pd.set_O(2./3);
        pd.set_dt(0.04);
        pd.set_n_periods(400);

        pd.cal();

        // solution of the ODE set
        vector<double> t = pd.get_t();
        vector< vector<double> > x = pd.get_x();
        vector<double> energy = pd.get_energy();
        int n_stps = pd.get_n_stps();

        theta_to_twopi(x[0]);

        // fixed poincare section
        vector<double> ps_f_d;
        vector<double> ps_theta;
        vector<double> ps_omega;
        vector<double> ps_energy;
        int ps_n_pts = 0;

        //double psphi = 0;
        //double psomega = 2./3;
        double dt = pd.get_dt();

        for(int i_stps=0; i_stps<n_stps; i_stps++) {
                if(fabs(psomega*t[i_stps]-psphi-int((psomega*t[i_stps]-psphi)/2./M_PI)*2*M_PI)
                > 0.5*psomega*dt*0.999999) {
                        continue;
                }
                ps_f_d.push_back(f_d);
                ps_theta.push_back(x[0][i_stps]);
                ps_omega.push_back(x[1][i_stps]);
                ps_energy.push_back(energy[i_stps]);
                ps_n_pts ++;
        }

        // ... plotting code ...
}

//------------------------------------------------------------------------------//
```

(2) Second, investigate the effects, if any, of using different initial conditions $\theta(0)$ and $\omega(0)$, while keeping all other parameters the same as in (1). Explicitly compare results with those of part (1).

Physics explanation:

To investigate the effect of the initial conditions, I set three sets of $\theta(0)$-$\omega(0)$:

(a) $\theta(0) = 0.2$ rad, $\omega(0) = 0$;

(b) $\theta(0) = 1.2$ rad, $\omega(0) = 0$;

(c) $\theta(0) = 0$, $\omega(0) = 1.2$ rad/s.

In this problem, the plots will be drawn in the sequence: (a) left pad, (b) middle pad, and (c) right pad. The initial condition (a) is the same as in (1).

In the **normal period regime** ($f_d = 1.40$) and the **period doubling regime** ($f_d = 1.44$), the left and middle pad in Fig. 8 and Fig. 9 show the same Poincare section results when stable. However, the right pad shows a different Poincare section result. Therefore, the initial condition**can** change the Poincare section in those two regimes. This could be understood in the following way. The pendulum ODE set are symmetrical about $\theta = 0$ and $\omega = 0$, whereas the phase diagrams are not, so there could be several possible phase diagrams, among which one can become another by mirror transformation about $\theta = 0$ or $\omega = 0$ (or combined, e.g. Fig. 8 from (b) to (c)). This difference can effectively contribute to the section like a phase.

In the **chaotic regime** ($f_d = 1.20$), the Poincare section is **independent** from the initial condition. All the three pads in Fig. 10 show the same Poincare section results when stable.

Plots:

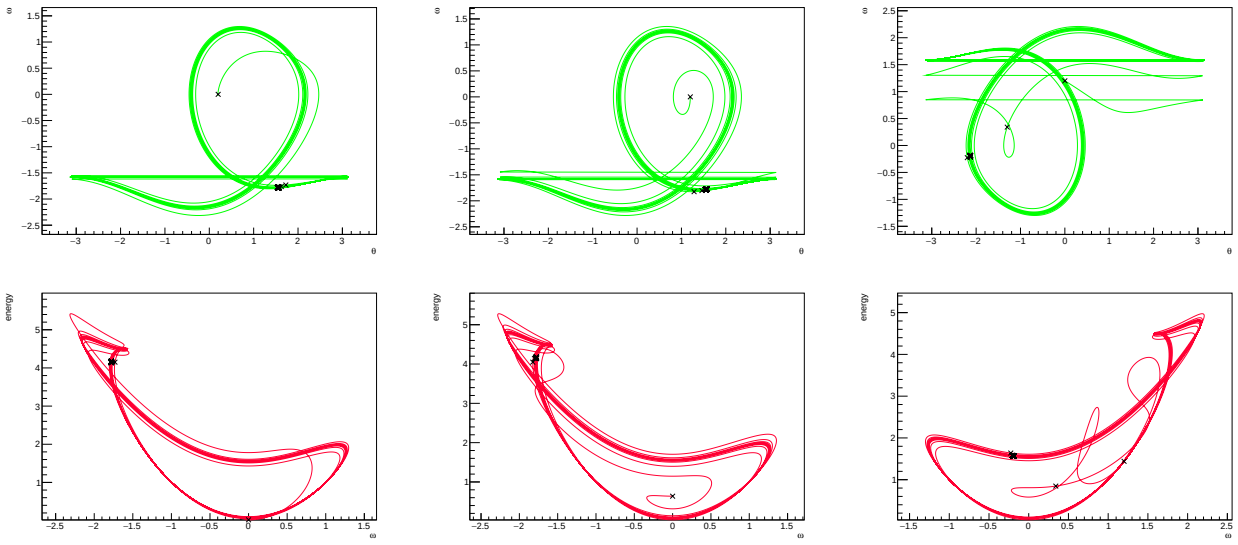

Figure 8: **Normal period regime**: $f_d = 1.40$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the initial condition: $\theta(0) = 0.2$ rad and $\omega(0) = 0$; the middle column has the initial condition: $\theta(0) = 1.2$ rad and $\omega(0) = 0$; the right column has the initial condition: $\theta(0) = 0$ and $\omega(0) = 1.2$ rad/s.

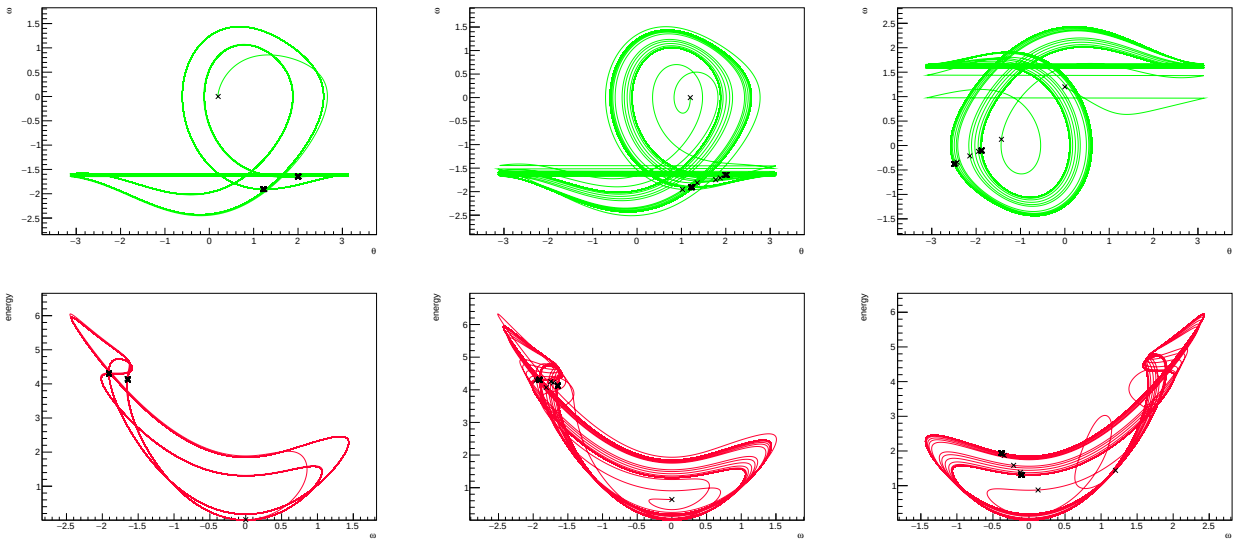

Figure 9: **Period doubling regime**: $f_d = 1.44$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the initial condition: $\theta(0) = 0.2$ rad and $\omega(0) = 0$; the middle column has the initial condition: $\theta(0) = 1.2$ rad and $\omega(0) = 0$; the right column has the initial condition: $\theta(0) = 0$ and $\omega(0) = 1.2$ rad/s.

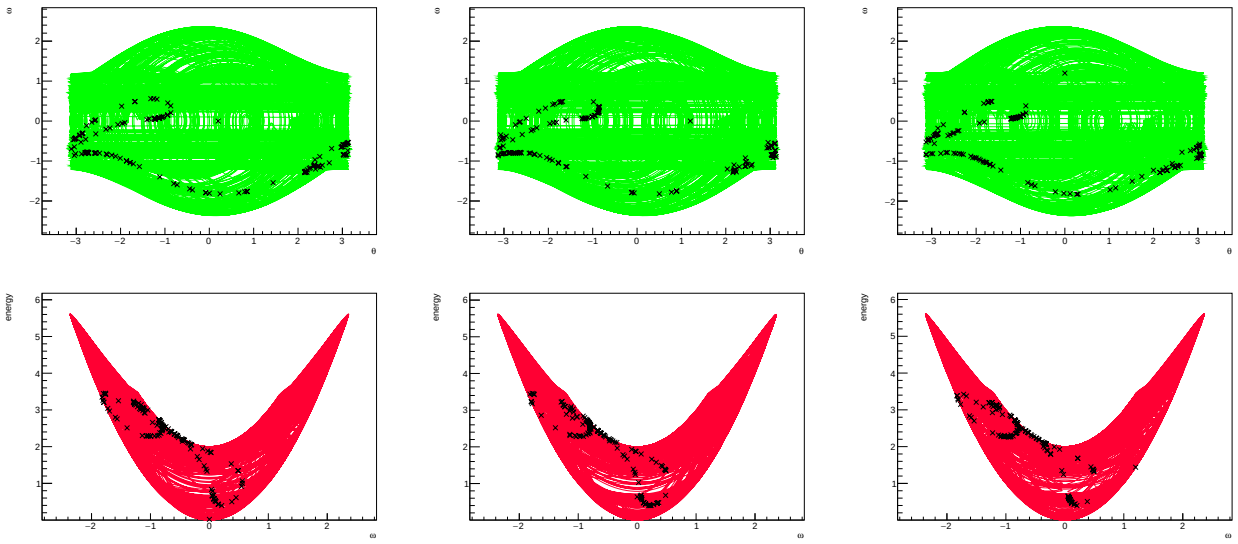

Figure 10: **Chaotic regime**: $f_d = 1.20$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the initial condition: $\theta(0) = 0.2$ rad and $\omega(0) = 0$; the middle column has the initial condition: $\theta(0) = 1.2$ rad and $\omega(0) = 0$; the right column has the initial condition: $\theta(0) = 0$ and $\omega(0) = 1.2$ rad/s.

Relevant code: The relevant code is the same as listed in the Problem (1)

```
//-----------------------------------------------------------------------//

void chao_plot(double f_d, double theta_start, double omega_start, double psphi, double psomega) {

        Pendulum pd;
        pd.set_mode(1); // physics pendulum
        pd.set_alg(4);
        //pd.set_theta_start(0.2);
        //pd.set_omega_start(0);
        pd.set_theta_start(theta_start);
        pd.set_omega_start(omega_start);
        pd.set_q(0.5);
        pd.set_F(f_d);
        pd.set_O(2./3);
        pd.set_dt(0.04);
        pd.set_n_periods(400);

        pd.cal();

        // solution of the ODE set
        vector<double> t = pd.get_t();
        vector< vector<double> > x = pd.get_x();
        vector<double> energy = pd.get_energy();
        int n_stps = pd.get_n_stps();

        theta_to_twopi(x[0]);

        // fixed poincare section
        vector<double> ps_f_d;
        vector<double> ps_theta;
        vector<double> ps_omega;
        vector<double> ps_energy;
        int ps_n_pts = 0;

        //double psphi = 0;
        //double psomega = 2./3;
        double dt = pd.get_dt();

        for(int i_stps=0; i_stps<n_stps; i_stps++) {
                if(fabs(psomega*t[i_stps]-psphi-int((psomega*t[i_stps]-psphi)/2./M_PI)*2*M_PI)
                > 0.5*psomega*dt*0.999999) {
                        continue;
                }
                ps_f_d.push_back(f_d);
                ps_theta.push_back(x[0][i_stps]);
                ps_omega.push_back(x[1][i_stps]);
                ps_energy.push_back(energy[i_stps]);
                ps_n_pts ++;
        }

        // ... plotting code ...
}

//-----------------------------------------------------------------------//
```

(3) Third, do the same as in (1) but with changing the phase cut to $\pi/2$, π, and $37°$. Compare the results with those of parts (1) and (2).

Physics explanation:

In the plots, the left pad has the section phase π; the middle pad has the section phase $\pi/2$; and the right pad has the section phase $37°$.

In the **normal period regime** ($f_d = 1.40$) and the **period doubling regime** ($f_d = 1.44$), different section phases ($\pi/2$, π, $37°$ in (3), and 0 in (1)) make the Poincare section results different. (Fig. 11 and Fig. 12) However, all dots got from Poincare section are on the phase diagram. As the section phase increases, the section dots are moving towards one direction of the phase diagram.

In the **chaotic regime** ($f_d = 1.20$), different section phases make the Poincare section results different. The upper middle pad (θ-ω, section phase π) in Fig. 13 is similar to the Fig. 7 bottom left pad in (1). If we do the mirror transformation about $\theta = 0$ and then the mirror transformation about $\omega = 0$, one can become the other. The lower middle pad (ω-E, section phase π) in Fig. 13 is similar to the Fig. 7 bottom right pad in (1). If we do the mirror transformation about $\omega = 0$, one can become the other.

This could be understood. The physical pendulum ODE has mirror symmetry about $\theta = 0$ and $\omega = 0$. The section phase π (half of the phase period 2π) can therefore effectively make the mirror transformation.

Plots:

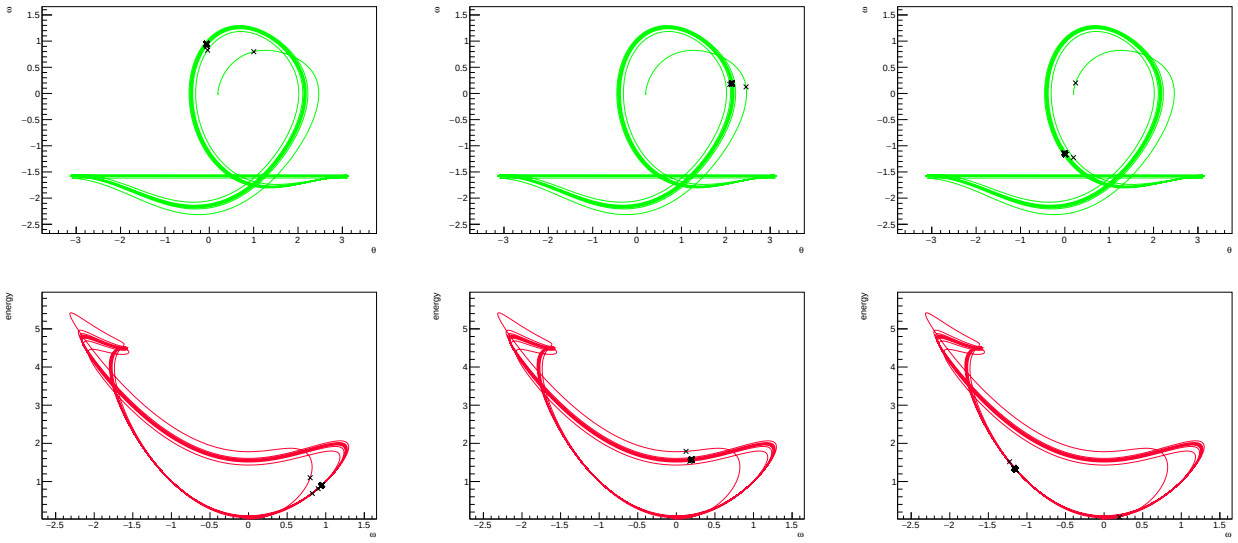

Figure 11: **Normal period regime**: $f_d = 1.40$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the section phase $\pi/2$; the middle column has the section phase π; the right column has the section phase $37°$.

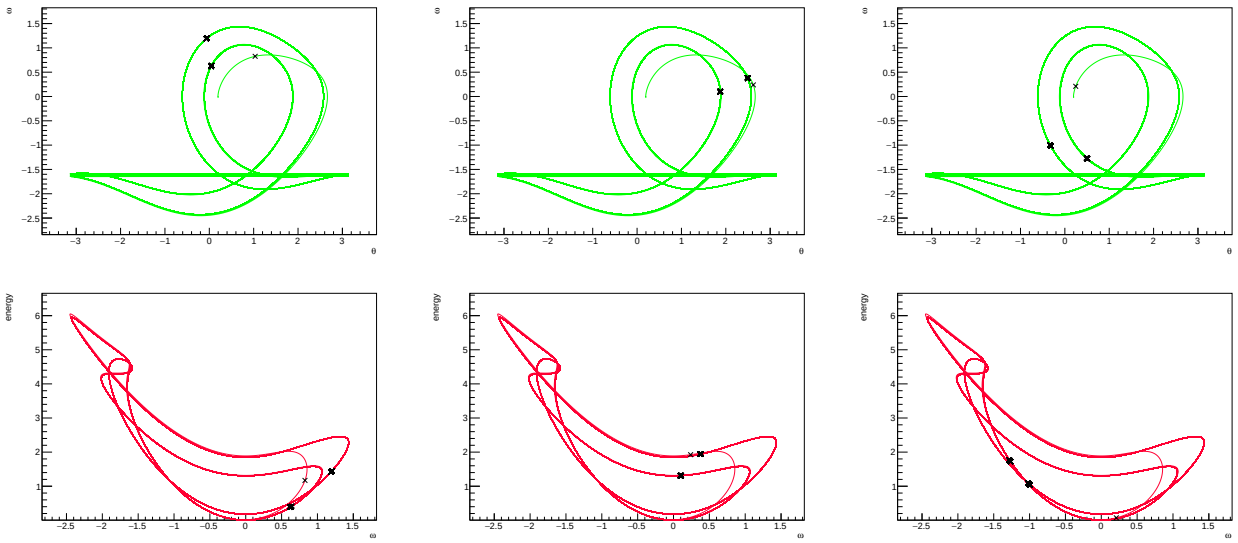

Figure 12: **Period doubling regime**: $f_d = 1.44$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the section phase $\pi/2$; the middle column has the section phase π; the right column has the section phase 37°.

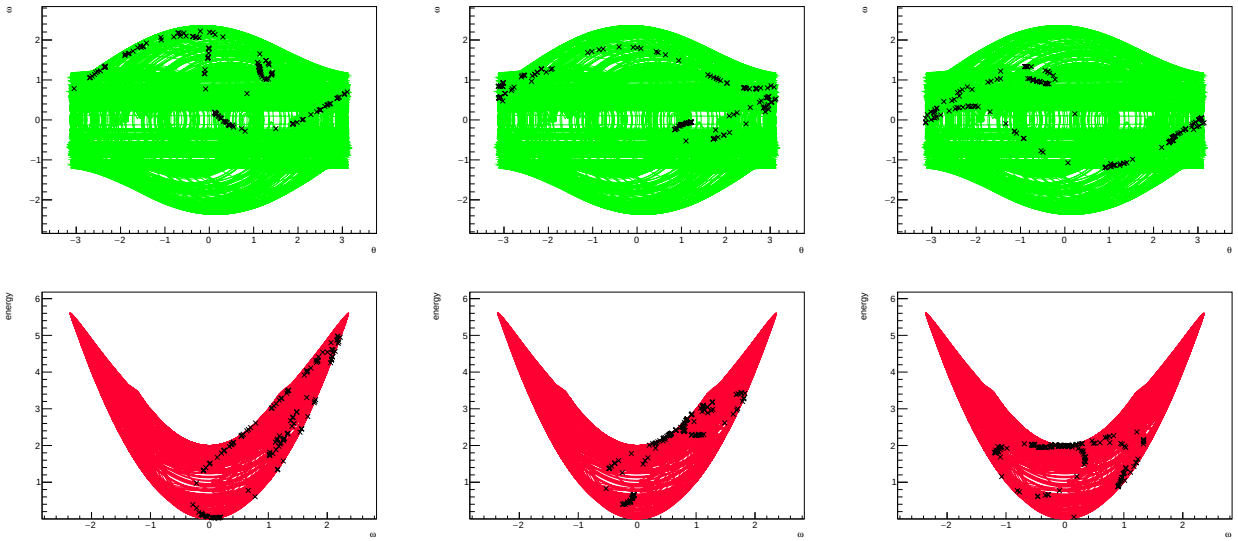

Figure 13: **Chaotic regime**: $f_d = 1.20$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the section phase $\pi/2$; the middle column has the section phase π; the right column has the section phase 37°.

Relevant code: The relevant code is the same as listed in the Problem (1)

```
//-------------------------------------------------------------------------//

void chao_plot(double f_d, double theta_start, double omega_start, double psphi, double psomega) {

        Pendulum pd;
        pd.set_mode(1); // physics pendulum
        pd.set_alg(4);
        //pd.set_theta_start(0.2);
        //pd.set_omega_start(0);
        pd.set_theta_start(theta_start);
        pd.set_omega_start(omega_start);
        pd.set_q(0.5);
        pd.set_F(f_d);
        pd.set_O(2./3);
        pd.set_dt(0.04);
        pd.set_n_periods(400);

        pd.cal();

        // solution of the ODE set
        vector<double> t = pd.get_t();
        vector< vector<double> > x = pd.get_x();
        vector<double> energy = pd.get_energy();
        int n_stps = pd.get_n_stps();

        theta_to_twopi(x[0]);

        // fixed poincare section
        vector<double> ps_f_d;
        vector<double> ps_theta;
        vector<double> ps_omega;
        vector<double> ps_energy;
        int ps_n_pts = 0;

        //double psphi = 0;
        //double psomega = 2./3;
        double dt = pd.get_dt();

        for(int i_stps=0; i_stps<n_stps; i_stps++) {
                if(fabs(psomega*t[i_stps]-psphi-int((psomega*t[i_stps]-psphi)/2./M_PI)*2*M_PI)
                > 0.5*psomega*dt*0.999999) {
                        continue;
                }
                ps_f_d.push_back(f_d);
                ps_theta.push_back(x[0][i_stps]);
                ps_omega.push_back(x[1][i_stps]);
                ps_energy.push_back(energy[i_stps]);
                ps_n_pts ++;
        }

        // ... plotting code ...
}

//-------------------------------------------------------------------------//
```

(4) Lastly, keeping all the other parameters the same as in (1), use an arbitrary sectioning frequency that differs from the driving frequency Ω and is irrationally related to or incommensurate with it. As an example, you can try $\omega_0 = (g/l)^{1/2}$ (the natural angular frequency of the linearized oscillator) or $\omega_1 = (g/l)^{1/2}\pi/[2K(\sin(\theta_m/2))]$ (that of the nonlinear oscillator without dissipation or driving force where θ_m is the amplitude). Discuss the results.

Physics explanation:

From my observation, the Poincare section only depends on the driving force. The nature angular frequency of linearized pendulum is 1. If we choose this to be the section frequency, both the 1-period regime (Fig. 14 middle) and the 2-period regime (Fig. 15 middle) have three dots on the phase diagrams. This is because the section period 2π and the driving force period 3π have common multiple, like 6π. If we wait 6π seconds, we will sample three different dots in Poincare section, and then go back to next common multiple period.

If we choose the natural angular frequency of the physical pendulum 0.9975, the common multiple period would be extremely large, so we can get lots of dots. (Fig. 14 right, Fig. 15 right) Theoratically, the frequency is actually an irrational number, so the common multiple should be infinity, and the dot number should also be infinity in this Poincare sampling.

In the chaotic regime, the dots show a finite pattern when the section frequency is 2/3 or 1, although the dots of the latter cover more area. However, if we choose the section frequency 0.9975, the dots seem everywhere on the phase diagram.

Plots:

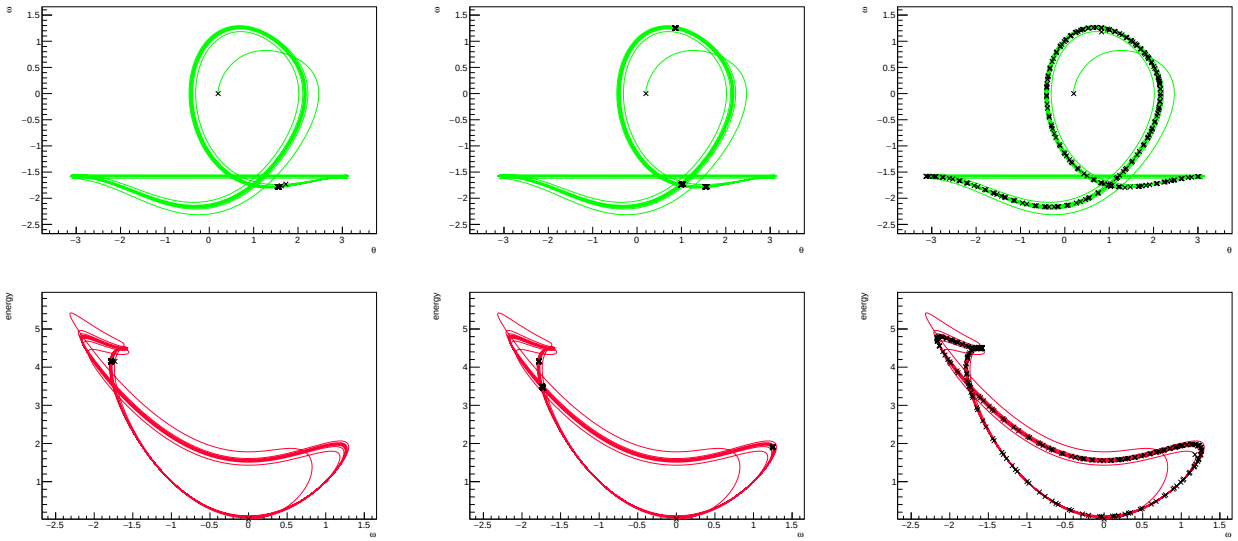

Figure 14: **Normal period regime**: $f_d = 1.40$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the section frequency 2/3 equal to the driving force frequency Ω; the middle column has the section frequency 1; the right column has the section frequency 0.9975.

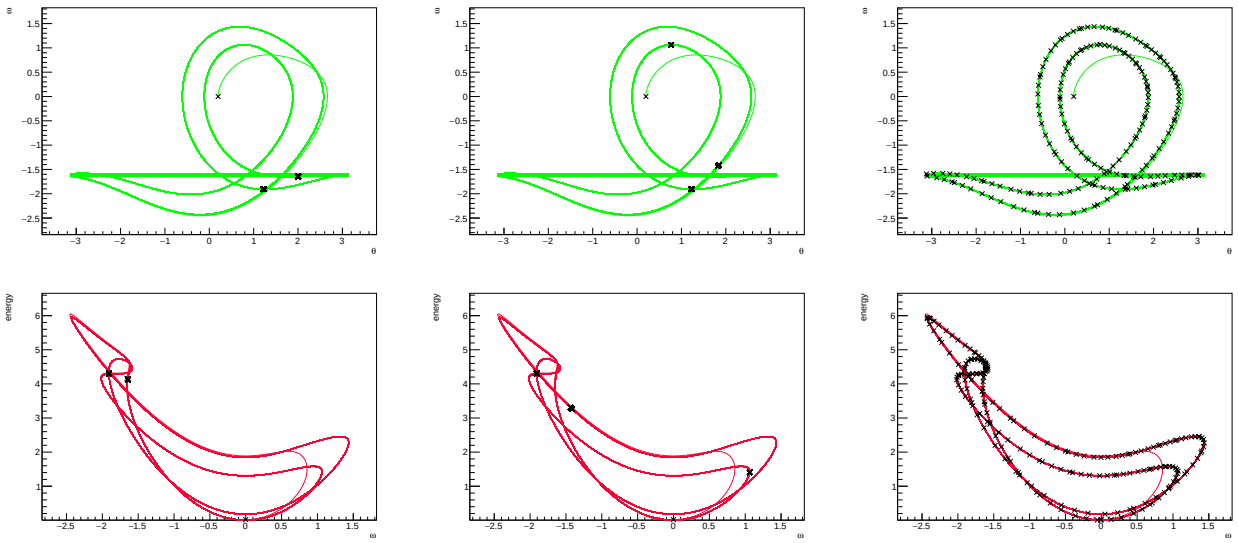

Figure 15: **Period doubling regime**: $f_d = 1.44$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the section frequency 2/3 equal to the driving force frequency Ω; the middle column has the section frequency 1; the right column has the section frequency 0.9975.

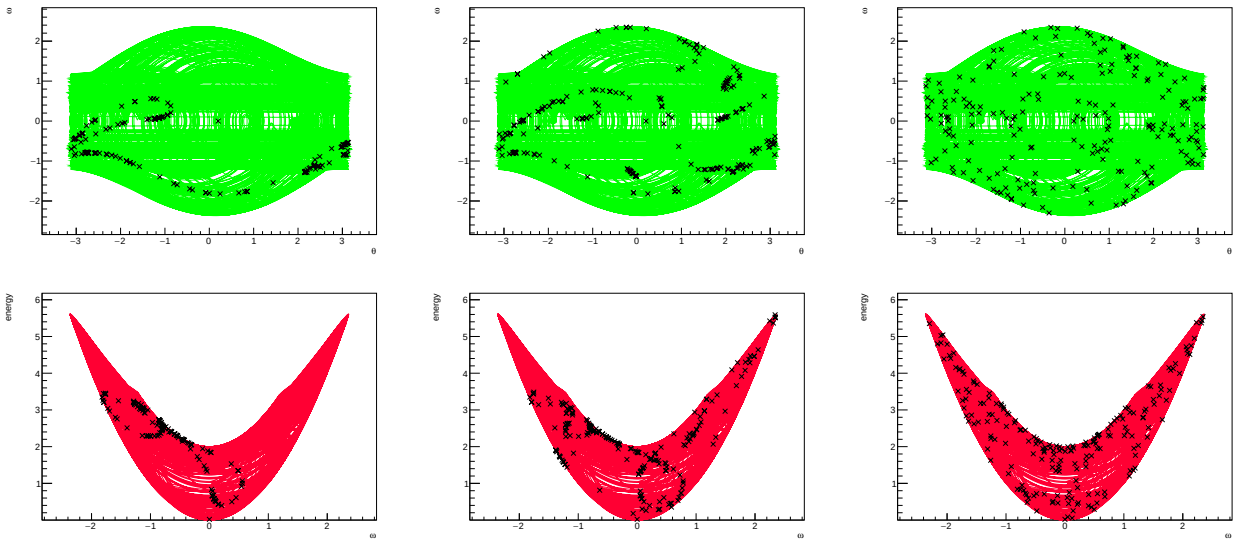

Figure 16: **Chaotic regime**: $f_d = 1.20$. The upper three plots are the θ-ω diagram, and the lower three plots are the ω-energy diagrams. The left column has the section frequency 2/3 equal to the driving force frequency Ω; the middle column has the section frequency 1; the right column has the section frequency 0.9975.

Relevant code: The natural angular frequency of the physical pendulum:

```
//------------------------------------------------------------------------//

double ellint_sin_half_theta(double theta_max) {

        double period;
        period = M_PI/2;
        //period *= 1 + pow(theta_max,2)/16 + pow(theta_max,4)*11/3072;
        period *= 1 + pow(theta_max,2)/16 + pow(theta_max,4)*11/3072 + pow(theta_max,6)*173/737280;
}

//------------------------------------------------------------------------//

// ... other code ...
double omega_1 = M_PI/2./ellint_sin_half_theta(0.2);
```

The other relevant code is the same as listed in the Problem (1)

```
//------------------------------------------------------------------------//

void chao_plot(double f_d, double theta_start, double omega_start, double psphi, double psomega) {

        Pendulum pd;
        pd.set_mode(1); // physics pendulum
        pd.set_alg(4);
        //pd.set_theta_start(0.2);
        //pd.set_omega_start(0);
        pd.set_theta_start(theta_start);
        pd.set_omega_start(omega_start);
        pd.set_q(0.5);
        pd.set_F(f_d);
        pd.set_O(2./3);
        pd.set_dt(0.04);
        pd.set_n_periods(400);

        pd.cal();

        // solution of the ODE set
        vector<double> t = pd.get_t();
        vector< vector<double> > x = pd.get_x();
        vector<double> energy = pd.get_energy();
        int n_stps = pd.get_n_stps();

        theta_to_twopi(x[0]);

        // fixed poincare section
        vector<double> ps_f_d;
        vector<double> ps_theta;
        vector<double> ps_omega;
        vector<double> ps_energy;
        int ps_n_pts = 0;

        //double psphi = 0;
        //double psomega = 2./3;
        double dt = pd.get_dt();

        for(int i_stps=0; i_stps<n_stps; i_stps++) {
                if(fabs(psomega*t[i_stps]-psphi-int((psomega*t[i_stps]-psphi)/2./M_PI)*2*M_PI)
                > 0.5*psomega*dt*0.999999) {
                        continue;
```

```
                }
                ps_f_d.push_back(f_d);
                ps_theta.push_back(x[0][i_stps]);
                ps_omega.push_back(x[1][i_stps]);
                ps_energy.push_back(energy[i_stps]);
                ps_n_pts ++;
        }

        // ... plotting code ...
}

//--------------------------------------------------------------------------//
```