Nom étudiant: Samuel Ogulluk
Numéro étudiant: 21501479

LU3PY126 FOAD

TP 2 Résolution numérique d'équations différentielles ordinaires- 2ème partie : application au pendule chaotique

---

Table des matières

Introduction

Au cours de ce TP, nous nous intéresserons à la résolution numérique d'équations différentielles.
Ainsi, nous verrons notamment les point suivants :

- ➔ Etude d'un oscillateur harmonique amorti par la méthode RK4
- ➔ Etude d'un oscillateur harmonique amorti excité par la méthode RK4
- ➔ Etude de la période d'un pendule
- ➔ Etude d'un mouvement chaotique

Les codes présentés sont en C++ et Python pour la visualisation et sont joint à ce compte-rendu ainsi que disponibles dans le dépôt suivant : Dépôt Github

Question 13 Runge-Kutta d'ordre 4 sur le pendule amorti

On s'intéresse dans un premier temps à un pendule amorti sans excitation :

$$\ddot{\theta} + q\dot{\theta} + \Omega^2\theta = 0 \tag{1}$$

Afin de résoudre ce probleme, on vient utiliser la méthode de Runge-Kutta d'ordre 4.

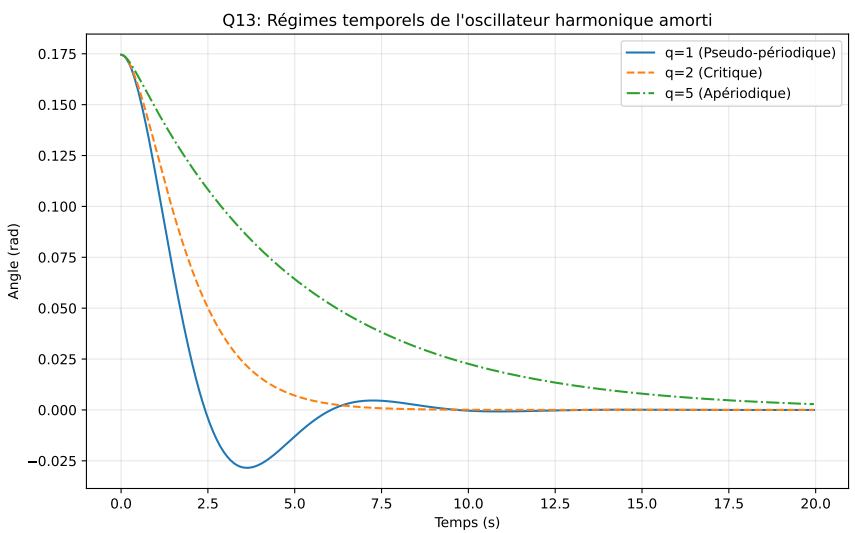

FIGURE 1 – Régimes de l'oscillateur harmonique amorti

On observe ainsi les 3 régimes possibles (critique, apériodique et pseudo-périodique) en fonction du coefficient d'amortissement.

TP2/q13.cpp

```cpp
void solve_q13() {
    g_non_lineaire = false;
    g_Fe = 0.0;
    double q_vals[] = {1.0, 2.0, 5.0};
    double t_max = 20.0;
    double dt = 0.05;

    std::vector<double> t_vec, th1, th2, th3;
    std::vector<double>* theta_vecs[] = {&th1, &th2,
     ↪  &th3};

    for(int i = 0; i < 3; ++i) { // on itère sur q
        g_q = q_vals[i];
        double y[2] = {10.0 * M_PI / 180.0, 0.0};

        for(double t = 0.0; t <= t_max; t += dt) { //
         ↪  on itèse sur le temps
            if(i == 0) t_vec.push_back(t);
            theta_vecs[i]->push_back(y[0]);
            rk4(2, t, y, dt, deriv);
        }
    }
```

Question 14 Ajout d'une excitation sinusoidale

On vient maintenant ajouter une excitation sinusoïdale et résoudre l'équation suivante :

$$\ddot{\theta} + q\dot{\theta} + \Omega^2\theta = F_e \sin(\Omega_e t) \tag{2}$$

L'évolution du système montre une transition d'une conservation d'énergie pure vers un régime forcé.

- Pendule libre ($q = 0, F_e = 0$) : La trajectoire est une ellipse fermée, traduisant la conservation de l'énergie mécanique totale du système.
- Pendule amorti ($q = 1, F_e = 0$) : Le système perd de l'énergie et la trajectoire spiralise vers l'origine $(0, 0)$, qui est un point d'équilibre stable (puits).
- Pendule avec excitation ($q = 1, F_e = 1$) : Après un régime transitoire, la trajectoire converge vers un cycle limite stable où l'énergie injectée par F_e équilibre exactement l'énergie dissipée par q.

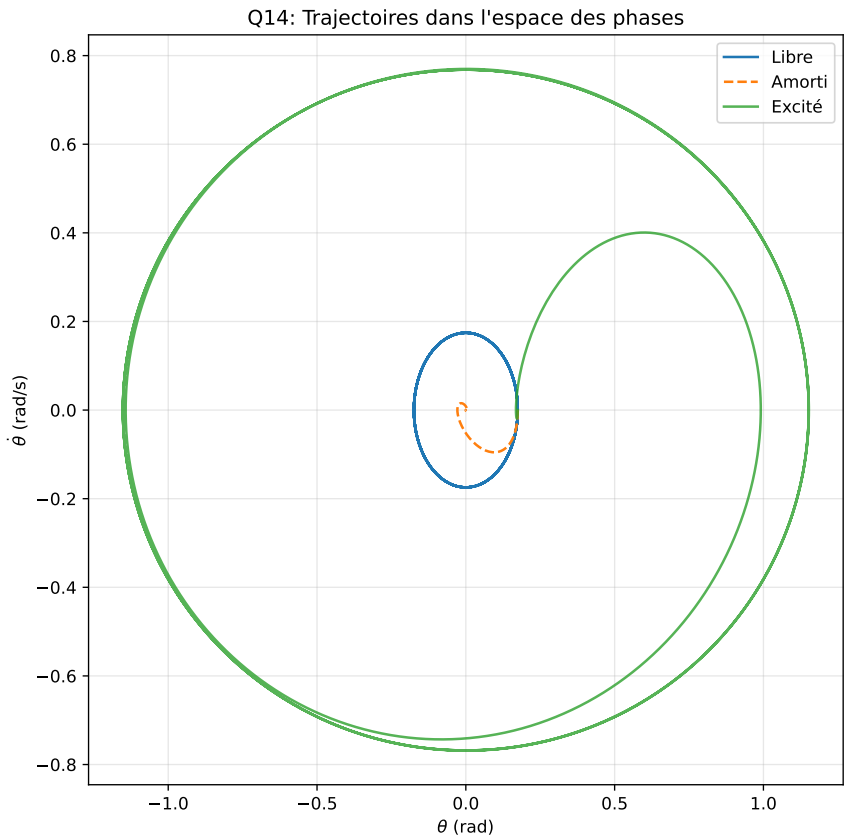

FIGURE 1 – Trajectoires dans l'espace des phases

TP2/q14.cpp

```cpp
void simulation_wrapper(double q, double Fe, std::vector<double>& th, std::vector<double>& dth) {
    g_q = q;
    g_Fe = Fe;
    double t = 0.0, dt = 0.04, t_max = 50.0;
    double y[2] = {10.0 * M_PI / 180.0, 0.0};

    while(t <= t_max) {
        th.push_back(y[0]);
        dth.push_back(y[1]);
        rk4(2, t, y, dt, deriv);
        t += dt;
    }
}

void solve_q14() {
    g_non_lineaire = false;

    std::vector<double> th_lib, dth_lib;
    std::vector<double> th_amort, dth_amort;
    std::vector<double> th_force, dth_force;

    simulation_wrapper(0.0, 0.0, th_lib, dth_lib);
    simulation_wrapper(1.0, 0.0, th_amort, dth_amort);
    simulation_wrapper(1.0, 1.0, th_force, dth_force);
```

Question 15 Résolution hors de l'approximation de McLaurin

A présent, on souhaite sortir de l'approximation des petits angles. Ainsi, l'équation 2 devient :

$$\ddot{\theta} + q\dot{\theta} + \Omega^2 \sin(\theta) = F_e \sin(\Omega_e t) \qquad (3)$$

L'augmentation de l'amplitude F_e brise successivement la symétrie temporelle du système, modifiant radicalement la périodicité de la réponse.

- Pour $F_e = 1.44$, le pendule ne revient à son état initial qu'après deux cycles de la force d'excitation $(2T)$.
- Ce processus se répète $(4T, 8T)$ pour des variations de plus en plus petites de F_e jusqu'à atteindre un régime apériodique.
- À $F_e = 1.5$, le mouvement devient imprévisible et sensible aux conditions initiales, perdant toute structure périodique simple.

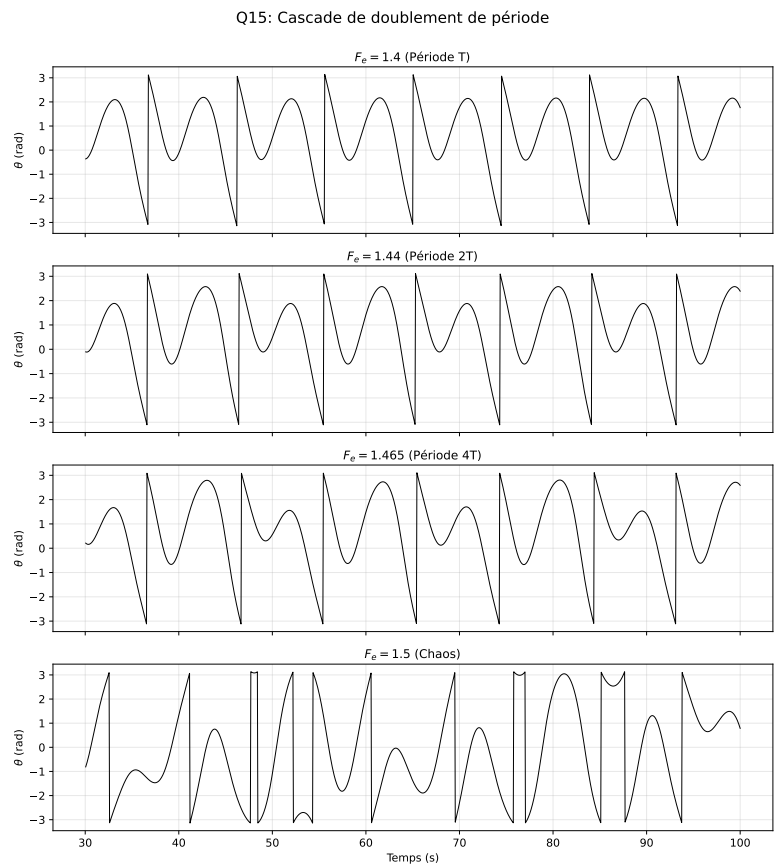

FIGURE 2 – θ en fonction du temps pour différentes excitations F_e

TP2/q15.cpp

```cpp
void solve_q15() {
    g_non_lineaire = true;
    g_q = 0.5;

    std::vector<double> Fe_vals = {1.4, 1.44, 1.465, 1.5};
    double t_max = 100.0;
    double dt = 0.05;

    std::vector<double> t_vec;
    std::vector<std::vector<double>> resultats; // Stocke les theta pour chaque Fe

    run_simulations(Fe_vals, t_max, dt, t_vec, resultats);

    std::vector<std::vector<double>> data_to_save;
    data_to_save.push_back(t_vec);
    for(auto& v : resultats) data_to_save.push_back(v);

    save_csv("resultats/q15_data.csv",
             {"t", "Fe_1.4", "Fe_1.44", "Fe_1.465", "Fe_1.5"},
             data_to_save);
}
```

TP2/q15.cpp

```cpp
void run_simulations(const std::vector<double>& Fe_vals, double t_max, double dt,
                     std::vector<double>& t_vec,
                     std::vector<std::vector<double>>& resultats) {
    for (double Fe : Fe_vals) {
        g_Fe = Fe;
        double t = 0.0;
        double y[2] = {10.0 * M_PI / 180.0, 0.0};
        std::vector<double> current_th;

        while (t <= t_max) {
            if (resultats.empty()) t_vec.push_back(t); // on remplis t une seule fois

            current_th.push_back(y[0]);

            rk4(2, t, y, dt, deriv);

            // on se ramène dans [-pi, pi]
            if (y[0] > M_PI) y[0] -= 2.0 * M_PI;
            if (y[0] < -M_PI) y[0] += 2.0 * M_PI;

            t += dt;
        }
        resultats.push_back(current_th);
    }
}
```

Question 16 Etude des divergences du mouvement chaotique

— **Sensibilité initiale** : Un écart de 10^{-3} degrés conduit à une divergence visible dès $t \approx 85$ s.

— **Divergence exponentielle** :
L'évolution de l'écart suit la loi :

$$|\delta\theta(t)| \propto e^{\lambda t}$$

— **Exposant de Lyapunov** :
La croissance linéaire de $\ln|\theta_1 - \theta_2|$ permet d'estimer $\lambda \approx 0.15\ \text{s}^{-1}$.

— **Caractère chaotique** :
Puisque $\lambda > 0$, l'incertitude est amplifiée exponentiellement, rendant toute prédiction impossible à long terme.

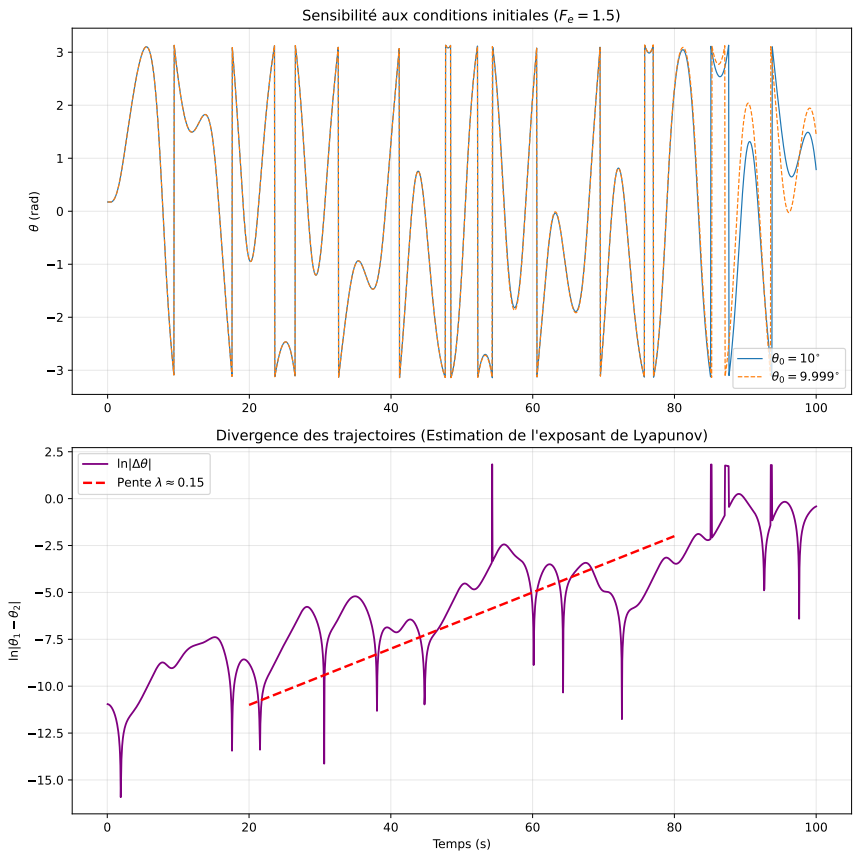

FIGURE 3 – Estimation de l'exposant de Lyapunov

TP2/q16.cpp

```cpp
void run_divergence_simulation(double t_max, double dt, double yA[2], double yB[2],
    std::vector<double>& t_vec, std::vector<double>& th_A, std::vector<double>& th_B,
    std::vector<double>& diff) {
    double t = 0.0;
    auto wrap = [](double& val) { // on ramene dans [-pi, pi]
        if (val > M_PI) val -= 2.0 * M_PI;
        if (val < -M_PI) val += 2.0 * M_PI;
    };

    while (t <= t_max) {
        t_vec.push_back(t);
        th_A.push_back(yA[0]);
        th_B.push_back(yB[0]);
        diff.push_back(std::abs(yA[0] - yB[0]));

        rk4(2, t, yA, dt, deriv);
        rk4(2, t, yB, dt, deriv);

        wrap(yA[0]);
        wrap(yB[0]);

        t += dt;
    }
}
```

TP2/q16.cpp

```cpp
void solve_q16() {
    g_non_lineaire = true;
    g_q = 0.5;
    g_Fe = 1.5;

    double t_max = 100.0, dt = 0.05;

    // CI proches
    double yA[2] = {10.0 * M_PI / 180.0, 0.0};
    double yB[2] = {9.999 * M_PI / 180.0, 0.0};

    std::vector<double> t_vec, th_A, th_B, diff;

    run_divergence_simulation(t_max, dt, yA, yB, t_vec, th_A, th_B, diff);

    save_csv("resultats/q16_data.csv",
             {"t", "th_A", "th_B", "delta_abs"},
             {t_vec, th_A, th_B, diff});
}
```

Question 17 Diagramme de bifurcation

Enfin, afin de mettre en évidence la transition du système vers le régime chaotique, on vient tracer un diagramme de bifurcation :

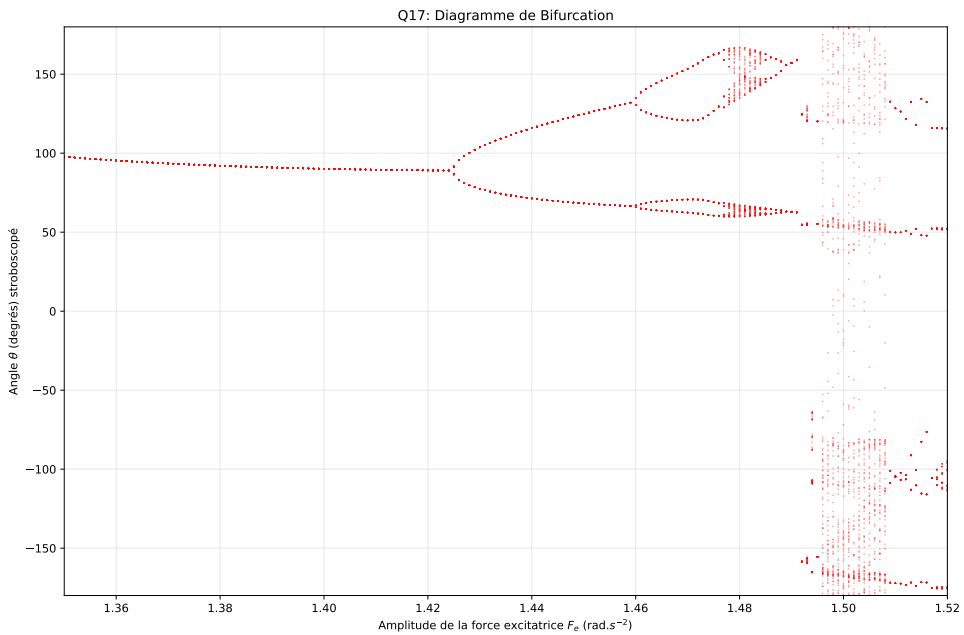

FIGURE 2 – Diagramme de bifurcation d'un pendule excité

On constate ainsi qu'on obtient un diagramme de bifurcation similaire à celui de l'exemple :

- Doublement de période en cascade
- Régime chaotique au delà de $F_e = 1.49$
- Bandes blanches au sein de la zone chaotique $\rightarrow$ comportement régulier

TP2/q17.cpp

```cpp
void run_transient(int p_transitoire, int steps,
    double dt, double& t, double y[2]) {
    for(int p = 0; p < p_transitoire; ++p) {
        for(int s = 0; s < steps; ++s) {
            rk4(2, t, y, dt, deriv);
            if (y[0] > M_PI) y[0] -= 2.0 * M_PI;
            if (y[0] < -M_PI) y[0] += 2.0 * M_PI;
            t += dt;
        }
    }
}

void run_capture(int p_capture, int steps, double dt,
    double& t, double y[2],
                 double Fe, std::ofstream& f) {
    for(int p = 0; p < p_capture; ++p) {
        for(int s = 0; s < steps; ++s) {
            rk4(2, t, y, dt, deriv);
            if (y[0] > M_PI) y[0] -= 2.0 * M_PI;
            if (y[0] < -M_PI) y[0] += 2.0 * M_PI;
            t += dt;
        }
        f << Fe << "," << (y[0] * 180.0 / M_PI) <<
            std::endl;
    }
}
```

TP2/q17.cpp

```cpp
void run_bifurcation_scan(double Fe_start, double
    Fe_end, double dFe,
                          int p_transitoire, int
                              p_capture, int steps,
                              double dt,
                          std::ofstream& f) {
    for (double Fe = Fe_start; Fe <= Fe_end; Fe +=
        dFe) {
        g_Fe = Fe;
        double t = 0.0;
        double y[2] = {10.0 * M_PI / 180.0, 0.0};

        run_transient(p_transitoire, steps, dt, t, y);
        run_capture(p_capture, steps, dt, t, y, Fe,
            f);
    }
}
```

TP2/q17.cpp

```cpp
void solve_q17() {
    g_non_lineaire = true;
    g_q = 0.5;

    // Paramètres de scan
    double Fe_start = 1.35;
    double Fe_end = 1.52;
    double dFe = 0.001;

    // Paramètres stroboscopiques
    double Te = 2.0 * M_PI / OMEGA_E;
    int steps = 200;
    double dt = Te / (double)steps;

    int p_transitoire = 300; // Périodes ignorées
    int p_capture = 100;     // Périodes gardées

    std::ofstream f("resultats/q17_bifurcation.csv");
    f << "Fe,theta_deg" << std::endl;

    run_bifurcation_scan(Fe_start, Fe_end, dFe, p_transitoire, p_capture, steps, dt, f);
}
```

Annexe

TP2/tp2.cpp

```cpp
#include <iostream>
#include <fstream>
#include "tp2.hpp"

double g_q = 0.0;
double g_Fe = 0.0;
bool g_non_lineaire = false;

void deriv(int n, double t, double y[], double dy[]) {
    dy[0] = y[1];
    double rappel = g_non_lineaire ? std::sin(y[0]) : y[0];
    dy[1] = -g_q * y[1] - OMEGA * OMEGA * rappel + g_Fe * std::sin(OMEGA_E * t);
}

void save_csv(const std::string& filename, const std::vector<std::string>& headers,
              const std::vector<std::vector<double>>& data) {
    std::ofstream f(filename);
    for(size_t i=0; i<headers.size(); ++i) {
        f << headers[i] << (i < headers.size()-1 ? "," : "");
    }
```

TP2/tp2.cpp

```cpp
    f << std::endl;
    size_t rows = data[0].size();
    size_t cols = data.size();
    for(size_t i=0; i<rows; ++i) {
        for(size_t j=0; j<cols; ++j) {
            f << data[j][i] << (j < cols-1 ? "," : "");
        }
        f << std::endl;
    }
}

int main() {
    std::cout << "Lancement des simulations" << std::endl;
    solve_q13();
    solve_q14();
    solve_q15();
    solve_q16();
    solve_q17();

    std::cout << "Terminé" << std::endl;
    return 0;
}
```

TP2/visualisation.py

```
def plot_q13():
    try:
        path = os.path.join(RES_DIR, 'q13_data.csv')
        df = pd.read_csv(path)
        plt.figure()
        plt.plot(df['t'], df['th_pseudo'], label='q=1 (Pseudo-périodique)')
        plt.plot(df['t'], df['th_crit'], label='q=2 (Critique)', ls='--')
        plt.plot(df['t'], df['th_aper'], label='q=5 (Apériodique)', ls='-.')
        plt.title("Q13: Régimes temporels de l'oscillateur harmonique amorti")
        plt.xlabel("Temps (s)")
        plt.ylabel("Angle (rad)")
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.savefig(os.path.join(FIG_DIR, 'q13_regimes.pdf'), dpi=150, bbox_inches='tight')
        plt.close()
    except Exception as e:
        print(f"Erreur Q13: {e}")
```

TP2/visualisation.py

```
def plot_q14():
    try:
        path = os.path.join(RES_DIR, 'q14_data.csv')
        df = pd.read_csv(path)
        plt.figure(figsize=(8, 8))
        plt.plot(df['th_lib'], df['dth_lib'], label='Libre')
        plt.plot(df['th_amort'], df['dth_amort'], label='Amorti', ls='--')
        plt.plot(df['th_force'], df['dth_force'], label='Excité', alpha=0.8)
        plt.title("Q14: Trajectoires dans l'espace des phases")
        plt.xlabel(r"$\theta$ (rad)")
        plt.ylabel(r"$\dot{\theta}$ (rad/s)")
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.savefig(os.path.join(FIG_DIR, 'q14_phases.pdf'), dpi=150, bbox_inches='tight')
        plt.close()
    except Exception as e:
        print(f"Erreur Q14: {e}")
```

TP2/visualisation.py

```
def plot_q15():
    try:
        path = os.path.join(RES_DIR, 'q15_data.csv')
        df = pd.read_csv(path)
        fig, axes = plt.subplots(4, 1, sharex=True, figsize=(10, 12))
        cols = ['Fe_1.4', 'Fe_1.44', 'Fe_1.465', 'Fe_1.5']
        titles = [r'$F_e=1.4$ (Période T)',
                  r'$F_e=1.44$ (Période 2T)',
                  r'$F_e=1.465$ (Période 4T)',
                  r'$F_e=1.5$ (Chaos)']

        for i, col in enumerate(cols):
            mask = df['t'] > 30
            axes[i].plot(df.loc[mask, 't'], df.loc[mask, col], color='black', lw=0.8)
            axes[i].set_title(titles[i], fontsize=11, pad=5)
            axes[i].set_ylabel(r"$\theta$ (rad)")
            axes[i].grid(True, alpha=0.3)

        axes[-1].set_xlabel("Temps (s)")
        plt.suptitle("Q15: Cascade de doublement de période", y=0.995, fontsize=14)
        plt.tight_layout(rect=[0, 0.03, 1, 0.99])
        plt.savefig(os.path.join(FIG_DIR, 'q15_chaos.pdf'), dpi=150, bbox_inches='tight')
        plt.close()
    except Exception as e:
        print(f"Erreur Q15: {e}")

```

TP2/visualisation.py

```python
def plot_q16():
    try:
        path = os.path.join(RES_DIR, 'q16_data.csv')
        df = pd.read_csv(path)

        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10))
        ax1.plot(df['t'], df['th_A'], label=r'$\theta_0 = 10^{\circ}$', lw=1)
        ax1.plot(df['t'], df['th_B'], label=r'$\theta_0 = 9.999^{\circ}$', ls='--', lw=1)
        ax1.set_title("Sensibilité aux conditions initiales ($F_e=1.5$)")
        ax1.set_ylabel(r"$\theta$ (rad)")
        ax1.legend()
        ax1.grid(True, alpha=0.3)

        mask = (df['t'] > 0) & (df['delta_abs'] > 1e-12)
        t_log = df.loc[mask, 't']
        log_delta = np.log(df.loc[mask, 'delta_abs'])

        ax2.plot(t_log, log_delta, label=r'$\ln|\Delta \theta|$', color='purple')
        t_slope = np.linspace(20, 80, 100)
        lambda_est = 0.15
        intercept = -14
        ax2.plot(t_slope, lambda_est * t_slope + intercept, 'r--', lw=2, label=r'Pente $\lambda \approx 0.15$')
        ax2.set_title("Divergence des trajectoires (Estimation de l'exposant de Lyapunov)")
        ax2.set_xlabel("Temps (s)")
        ax2.set_ylabel(r"$\ln |\theta_1 - \theta_2|$")
        ax2.legend()
        ax2.grid(True, alpha=0.3)
        plt.tight_layout()
        plt.savefig(os.path.join(FIG_DIR, 'q16_lyapunov.pdf'), dpi=150, bbox_inches='tight')
        plt.close()
```

TP2/visualisation.py

```
117 def plot_q17():
118     try:
119         path = os.path.join(RES_DIR, 'q17_bifurcation.csv')
120         df = pd.read_csv(path)
121         plt.figure(figsize=(12, 8))
122         plt.scatter(df['Fe'], df['theta_deg'], s=0.1, c='red', alpha=0.6)
123         plt.title("Q17: Diagramme de Bifurcation")
124         plt.xlabel(r"Amplitude de la force excitatrice $F_e$ (rad.$s^{-2}$)")
125         plt.ylabel(r"Angle $\theta$ (degrés) stroboscopé")
126         plt.xlim(df['Fe'].min(), df['Fe'].max())
127         plt.ylim(df['theta_deg'].min(), df['theta_deg'].max())
128         plt.grid(True, alpha=0.3)
129         plt.tight_layout()
130         plt.savefig(os.path.join(FIG_DIR, 'q17_bifurcation.pdf'), dpi=150, bbox_inches='tight')
131         plt.close()
132     except Exception as e:
133         print(f"Erreur Q17: {e}")
```