

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**

13 Runge-Kutta d'ordre 4 sur le pendule amorti	3	15 Résolution hors de l'approximation de McLaurin	6
14 Ajout d'une excitation sinusoidale	4	16 Etude des divergences du mouvement chaotique	8
		17 Diagramme de bifurcation	10

# Introduction

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

$$\ddot{\theta} + q\dot{\theta} + \Omega^2\theta = 0 \quad (1)$$

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

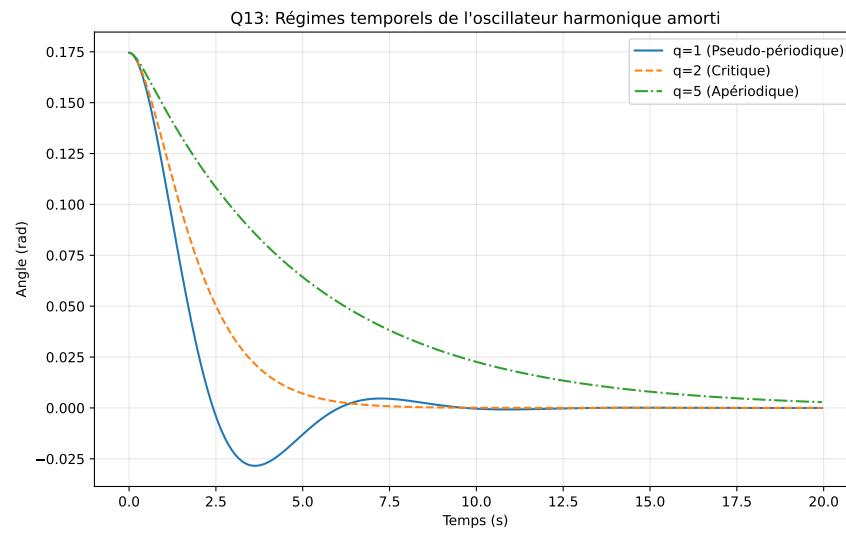


FIGURE 1 – Régimes de l'oscillateur harmonique amorti

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

**TP2/q13.cpp**

```
4 void solve_q13() {
5     g_non_Lineaire = false;
6     g_Fe = 0.0;
7     double q_vals[] = {1.0, 2.0, 5.0};
8     double t_max = 20.0;
9     double dt = 0.05;
10
11    std::vector<double> t_vec, th1, th2, th3;
12    std::vector<double*>* theta_vecs[] = {&th1, &th2,
13                                         &th3};
14
15    for(int i = 0; i < 3; ++i) { // on itère sur q
16        g_q = q_vals[i];
17        double y[2] = {10.0 * M_PI / 180.0, 0.0};
18
19        for(double t = 0.0; t <= t_max; t += dt) { // 
20            if(i == 0) t_vec.push_back(t);
21            theta_vecs[i]->push_back(y[0]);
22            rk4(2, t, y, dt, deriv);
23        }
24    }
25}
```

## Question 14 Ajout d'une excitation sinusoïdale

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

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

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  $q$ .

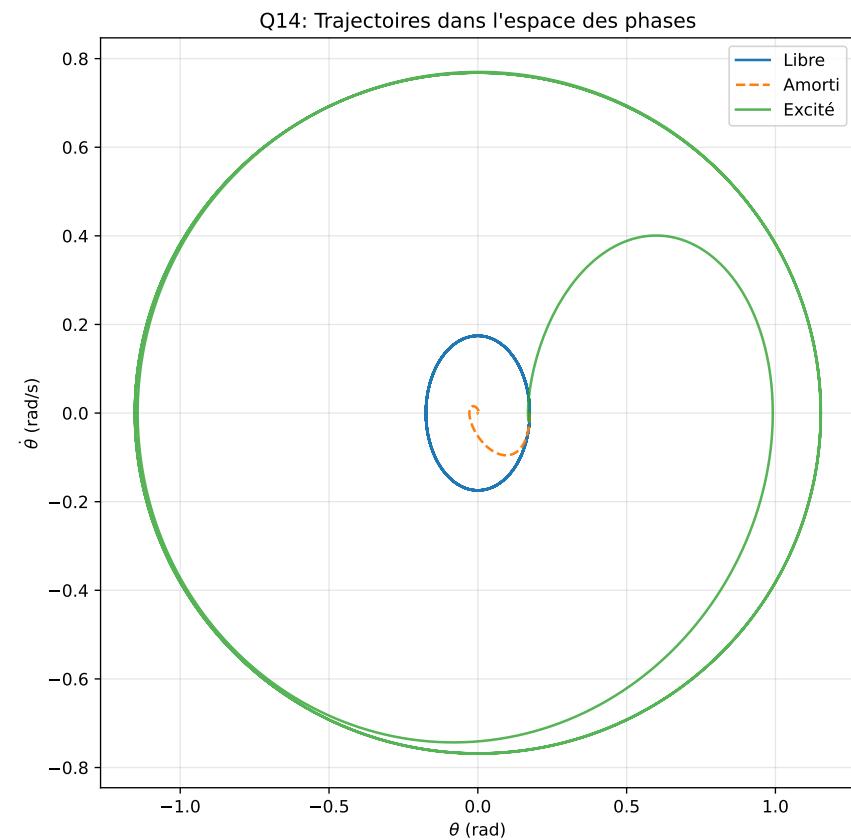


FIGURE 1 – Trajectoires dans l'espace des phases

## TP2/q14.cpp

```
4 void simulation_wrapper(double q, double Fe, std::vector<double>& th, std::vector<double>& dth) {
5     g_q = q;
6     g_Fe = Fe;
7     double t = 0.0, dt = 0.04, t_max = 50.0;
8     double y[2] = {10.0 * M_PI / 180.0, 0.0};
9
10    while(t <= t_max) {
11        th.push_back(y[0]);
12        dth.push_back(y[1]);
13        rk4(2, t, y, dt, deriv);
14        t += dt;
15    }
16}
17
18 void solve_q14() {
19     g_non_ligneaire = false;
20
21     std::vector<double> th_lib, dth_lib;
22     std::vector<double> th_amort, dth_amort;
23     std::vector<double> th_force, dth_force;
24
25     simulation_wrapper(0.0, 0.0, th_lib, dth_lib);
26     simulation_wrapper(1.0, 0.0, th_amort, dth_amort);
27     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 :

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

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.

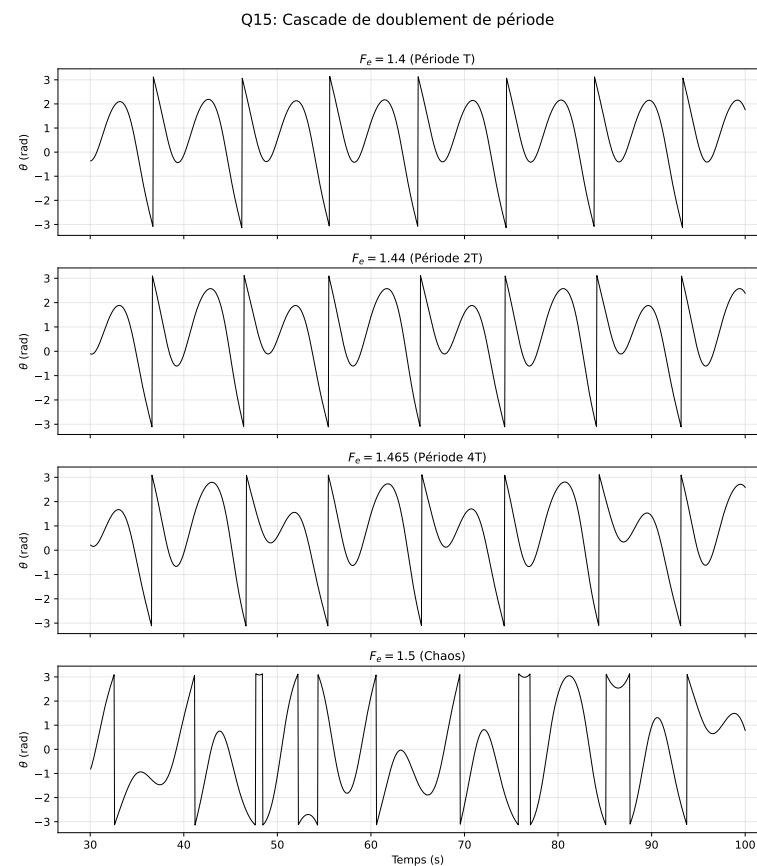


FIGURE 2 –  $\theta$  en fonction du temps pour différentes excitations  $F_e$

## TP2/q15.cpp

```
29 void solve_q15() {
30     g_non_lineaire = true;
31     g_q = 0.5;
32
33     std::vector<double> Fe_vals = {1.4, 1.44, 1.465,
34                                     → 1.5};
35     double t_max = 100.0;
36     double dt = 0.05;
37
38     std::vector<double> t_vec;
39     std::vector<std::vector<double>> resultats; // Stocke les theta pour chaque Fe
40
41     run_simulations(Fe_vals, t_max, dt, t_vec,
42                      → resultats);
43
44     std::vector<std::vector<double>> data_to_save;
45     data_to_save.push_back(t_vec);
46     for(auto& v : resultats)
47         → data_to_save.push_back(v);
48
49     save_csv("resultats/q15_data.csv",
50             {"t", "Fe_1.4", "Fe_1.44", "Fe_1.465",
51              → "Fe_1.5"},
52             data_to_save);
53 }
```

## TP2/q15.cpp

```
4 void run_simulations(const std::vector<double>&
5     → Fe_vals, double t_max, double dt,
6     std::vector<double>& t_vec,
7     → std::vector<std::vector<double>>&
8     → resultats) {
9
10    for (double Fe : Fe_vals) {
11        g_Fe = Fe;
12        double t = 0.0;
13        double y[2] = {10.0 * M_PI / 180.0, 0.0};
14        std::vector<double> current_th;
15
16        while (t <= t_max) {
17            if (resultats.empty()) t_vec.push_back(t);
18            → // on remplit t une seule fois
19            current_th.push_back(y[0]);
20            rk4(2, t, y, dt, deriv);
21
22            // on se ramène dans [-pi, pi]
23            if (y[0] > M_PI) y[0] -= 2.0 * M_PI;
24            if (y[0] < -M_PI) y[0] += 2.0 * M_PI;
25
26            t += dt;
27        }
28        resultats.push_back(current_th);
29    }
30 }
```

# 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.

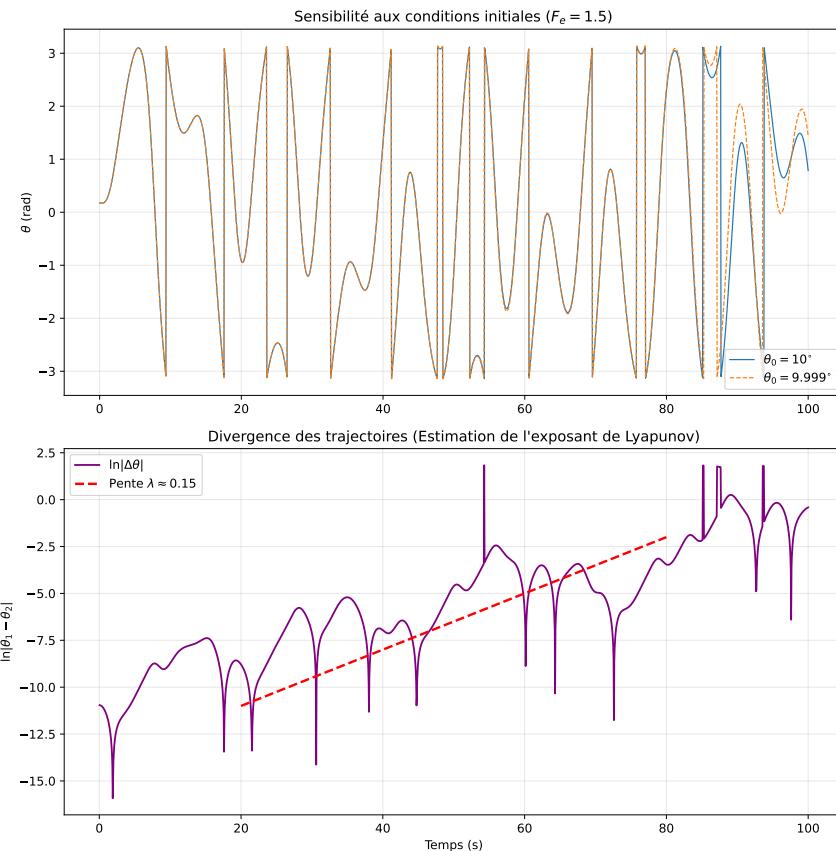


FIGURE 3 – Estimation de l'exposant de Lyapunov

## TP2/q16.cpp

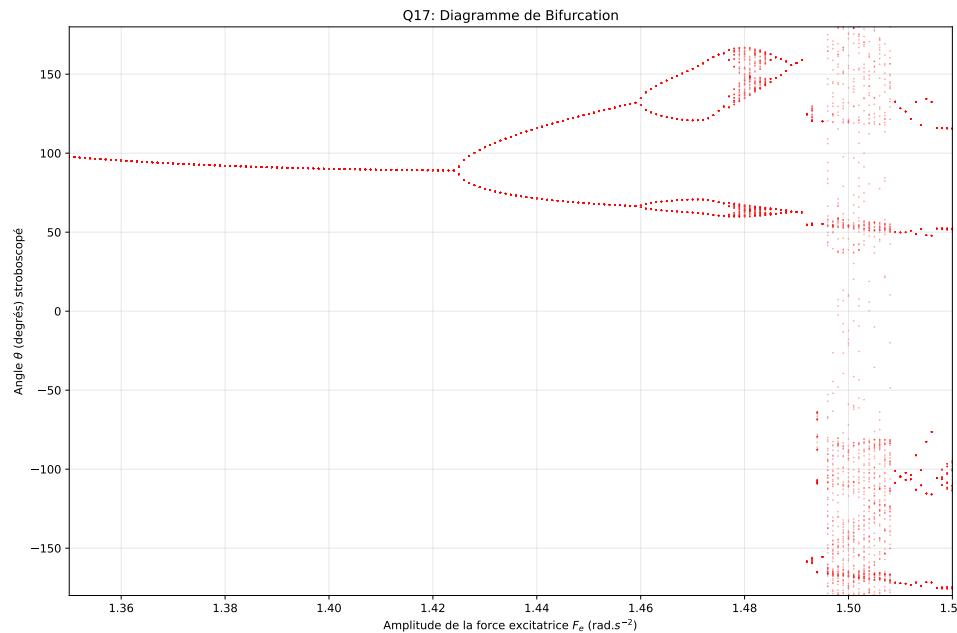
```
4 void run_divergence_simulation(double t_max, double
  ↵ dt, double yA[2], double yB[2],
  ↵ std::vector<double>& t_vec, std::vector<double>&
  ↵ th_A, std::vector<double>& th_B,
  ↵ std::vector<double>& diff) {
5   double t = 0.0;
6   auto wrap = [] (double& val) { // on ramene dans
  ↵ [-pi, pi]
  ↵   if (val > M_PI) val -= 2.0 * M_PI;
  ↵   if (val < -M_PI) val += 2.0 * M_PI;
9   };
10
11  while (t <= t_max) {
12    t_vec.push_back(t);
13    th_A.push_back(yA[0]);
14    th_B.push_back(yB[0]);
15    diff.push_back(std::abs(yA[0] - yB[0]));
16
17    rk4(2, t, yA, dt, deriv);
18    rk4(2, t, yB, dt, deriv);
19
20    wrap(yA[0]);
21    wrap(yB[0]);
22
23    t += dt;
24  }
25 }
```

## TP2/q16.cpp

```
27 void solve_q16() {
28   g_non_lineaire = true;
29   g_q = 0.5;
30   g_Fe = 1.5;
31
32   double t_max = 100.0, dt = 0.05;
33
34   // CI proches
35   double yA[2] = {10.0 * M_PI / 180.0, 0.0};
36   double yB[2] = {9.999 * M_PI / 180.0, 0.0};
37
38   std::vector<double> t_vec, th_A, th_B, diff;
39
40   run_divergence_simulation(t_max, dt, yA, yB,
  ↵ t_vec, th_A, th_B, diff);
41
42   save_csv("resultats/q16_data.csv",
43     {"t", "th_A", "th_B", "delta_abs"},
44     {t_vec, th_A, th_B, diff});
45 }
```

## 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 :



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 → comportement régulier

FIGURE 2 – Diagramme de bifurcation d'un pendule excité

## TP2/q17.cpp

```
4 void run_transient(int p_transitoire, int steps,
→ double dt, double& t, double y[2]) {
5   for(int p = 0; p < p_transitoire; ++p) {
6     for(int s = 0; s < steps; ++s) {
7       rk4(2, t, y, dt, deriv);
8       if (y[0] > M_PI) y[0] -= 2.0 * M_PI;
9       if (y[0] < -M_PI) y[0] += 2.0 * M_PI;
10      t += dt;
11    }
12  }
13 }
14
15 void run_capture(int p_capture, int steps, double dt,
→ double& t, double y[2],
→ double Fe, std::ofstream& f) {
16   for(int p = 0; p < p_capture; ++p) {
17     for(int s = 0; s < steps; ++s) {
18       rk4(2, t, y, dt, deriv);
19       if (y[0] > M_PI) y[0] -= 2.0 * M_PI;
20       if (y[0] < -M_PI) y[0] += 2.0 * M_PI;
21       t += dt;
22     }
23     f << Fe << "," << (y[0] * 180.0 / M_PI) <<
→ std::endl;
24   }
25 }
26 }
```

## TP2/q17.cpp

```
28 void run_bifurcation_scan(double Fe_start, double
→ Fe_end, double dFe,
29                           int p_transitoire, int
→ p_capture, int steps,
→ double dt,
→ std::ofstream& f) {
30   for (double Fe = Fe_start; Fe <= Fe_end; Fe +=
→ dFe) {
31     g_Fe = Fe;
32     double t = 0.0;
33     double y[2] = {10.0 * M_PI / 180.0, 0.0};
34
35     run_transient(p_transitoire, steps, dt, t, y);
36     run_capture(p_capture, steps, dt, t, y, Fe,
→ f);
37   }
38 }
39 }
```

## TP2/q17.cpp

```
42 void solve_q17() {
43     g_non_lineaire = true;
44     g_q = 0.5;
45
46     // Paramètres de scan
47     double Fe_start = 1.35;
48     double Fe_end = 1.52;
49     double dFe = 0.001;
50
51     // Paramètres stroboscopiques
52     double Te = 2.0 * M_PI / OMEGA_E;
53     int steps = 200;
54     double dt = Te / (double)steps;
55
56     int p_transitoire = 300; // Périodes ignorées
57     int p_capture = 100;    // Périodes gardées
58
59     std::ofstream f("resultats/q17_bifurcation.csv");
60     f << "Fe,theta_deg" << std::endl;
61
62     run_bifurcation_scan(Fe_start, Fe_end, dFe, p_transitoire, p_capture, steps, dt, f);
63 }
```

# Annexe

## TP2/tp2.cpp

```
1 #include <iostream>
2 #include <fstream>
3 #include "tp2.hpp"
4
5 double g_q = 0.0;
6 double g_Fe = 0.0;
7 bool g_non_lineaire = false;
8
9 void deriv(int n, double t, double y[], double dy[])
10 { dy[0] = y[1];
11   double rappel = g_non_lineaire ? std::sin(y[0]) :
12     y[0];
13   dy[1] = -g_q * y[1] - OMEGA * OMEGA * rappel +
14     g_Fe * std::sin(OMEGA_E * t);
15 }
16 void save_csv(const std::string& filename, const
17   std::vector<std::string>& headers,
18   const std::vector<std::vector<double>>&
19   data) {
20   std::ofstream f(filename);
21   for(size_t i=0; i<headers.size(); ++i) {
22     f << headers[i] << (i < headers.size()-1 ? "," :
23       : "\n");
24   }
25 }
```

## TP2/tp2.cpp

```
21   f << std::endl;
22   size_t rows = data[0].size();
23   size_t cols = data.size();
24   for(size_t i=0; i<rows; ++i) {
25     for(size_t j=0; j<cols; ++j) {
26       f << data[j][i] << (j < cols-1 ? "," :
27         "\n");
28     }
29   }
30 }
31
32 int main() {
33   std::cout << "Lancement des simulations" <<
34   std::endl;
35   solve_q13();
36   solve_q14();
37   solve_q15();
38   solve_q16();
39   solve_q17();
40
41   std::cout << "Terminé" << std::endl;
42 }
```

## TP2/visualisation.py

```
21 def plot_q13():
22     try:
23         path = os.path.join(RES_DIR, 'q13_data.csv')
24         df = pd.read_csv(path)
25         plt.figure()
26         plt.plot(df['t'], df['th_pseudo'], label='q=1
27             ↪ (Pseudo-périodique)')
28         plt.plot(df['t'], df['th_crit'], label='q=2
29             ↪ (Critique)', ls='--')
30         plt.plot(df['t'], df['th_aper'], label='q=5
31             ↪ (Apériodique)', ls='-.')
32         plt.title("Q13: Régimes temporels de
33             ↪ l'oscillateur harmonique amorti")
34         plt.xlabel("Temps (s)")
35         plt.ylabel("Angle (rad)")
36         plt.legend()
37         plt.grid(True, alpha=0.3)
38         plt.savefig(os.path.join(FIG_DIR,
39             ↪ 'q13_regimes.pdf'), dpi=150,
40             ↪ bbox_inches='tight')
41         plt.close()
42     except Exception as e:
43         print(f"Erreur Q13: {e}")
```

## TP2/visualisation.py

```
40 def plot_q14():
41     try:
42         path = os.path.join(RES_DIR, 'q14_data.csv')
43         df = pd.read_csv(path)
44         plt.figure(figsize=(8, 8))
45         plt.plot(df['th_lib'], df['dth_lib'],
46             ↪ label='Libre')
47         plt.plot(df['th_amort'], df['dth_amort'],
48             ↪ label='Amorti', ls='--')
49         plt.plot(df['th_force'], df['dth_force'],
50             ↪ label='Excité', alpha=0.8)
51         plt.title("Q14: Trajectoires dans l'espace des
52             ↪ phases")
53         plt.xlabel(r"$\theta$ (rad)")
54         plt.ylabel(r"$\dot{\theta}$ (rad/s)")
55         plt.legend()
56         plt.grid(True, alpha=0.3)
57         plt.savefig(os.path.join(FIG_DIR,
58             ↪ 'q14_phases.pdf'), dpi=150,
59             ↪ bbox_inches='tight')
60         plt.close()
61     except Exception as e:
62         print(f"Erreur Q14: {e}")
```

## TP2/visualisation.py

```
58 def plot_q15():
59     try:
60         path = os.path.join(RES_DIR, 'q15_data.csv')
61         df = pd.read_csv(path)
62         fig, axes = plt.subplots(4, 1, sharex=True, figsize=(10, 12))
63         cols = ['Fe_1.4', 'Fe_1.44', 'Fe_1.465', 'Fe_1.5']
64         titles = [r'$F_e=1.4$ (Période T)', 
65                   r'$F_e=1.44$ (Période 2T)', 
66                   r'$F_e=1.465$ (Période 4T)', 
67                   r'$F_e=1.5$ (Chaos)']
68
69         for i, col in enumerate(cols):
70             mask = df['t'] > 30
71             axes[i].plot(df.loc[mask, 't'], df.loc[mask, col], color='black', lw=0.8)
72             axes[i].set_title(titles[i], fontsize=11, pad=5)
73             axes[i].set_ylabel(r"$\theta$ (rad)")
74             axes[i].grid(True, alpha=0.3)
75
76             axes[-1].set_xlabel("Temps (s)")
77             plt.suptitle("Q15: Cascade de doublement de période", y=0.995, fontsize=14)
78             plt.tight_layout(rect=[0, 0.03, 1, 0.99])
79             plt.savefig(os.path.join(FIG_DIR, 'q15_chaos.pdf'), dpi=150, bbox_inches='tight')
80             plt.close()
81     except Exception as e:
82         print(f"Erreur Q15: {e}")
83
```

## TP2/visualisation.py

```
84 def plot_q16():
85     try:
86         path = os.path.join(RES_DIR, 'q16_data.csv')
87         df = pd.read_csv(path)
88
89         fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10))
90         ax1.plot(df['t'], df['th_A'], label=r'$\theta_0 = 10^{\circ}$', lw=1)
91         ax1.plot(df['t'], df['th_B'], label=r'$\theta_0 = 9.999^{\circ}$', ls='--', lw=1)
92         ax1.set_title("Sensibilité aux conditions initiales ($F_e=1.5$)")
93         ax1.set_ylabel(r"$\theta$ (rad)")
94         ax1.legend()
95         ax1.grid(True, alpha=0.3)
96
97         mask = (df['t'] > 0) & (df['delta_abs'] > 1e-12)
98         t_log = df.loc[mask, 't']
99         log_delta = np.log(df.loc[mask, 'delta_abs'])
100
101        ax2.plot(t_log, log_delta, label=r'$\ln|\Delta \theta|$', color='purple')
102        t_slope = np.linspace(20, 80, 100)
103        lambda_est = 0.15
104        intercept = -14
105        ax2.plot(t_slope, lambda_est * t_slope + intercept, 'r--', lw=2, label=r'Pente $\lambda \approx 0.15$')
106        ax2.set_title("Divergence des trajectoires (Estimation de l'exposant de Lyapunov)")
107        ax2.set_xlabel("Temps (s)")
108        ax2.set_ylabel(r"$\ln |\theta_1 - \theta_2|$")
109        ax2.legend()
110        ax2.grid(True, alpha=0.3)
111        plt.tight_layout()
112        plt.savefig(os.path.join(FIG_DIR, 'q16_lyapunov.pdf'), dpi=150, bbox_inches='tight')
113        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}")
```