Modélisation des battements du cœur

- Plan:

I) Modèle de Van der Pol

II) Modèle de Zeeman

I) Modèle de Van der Pol

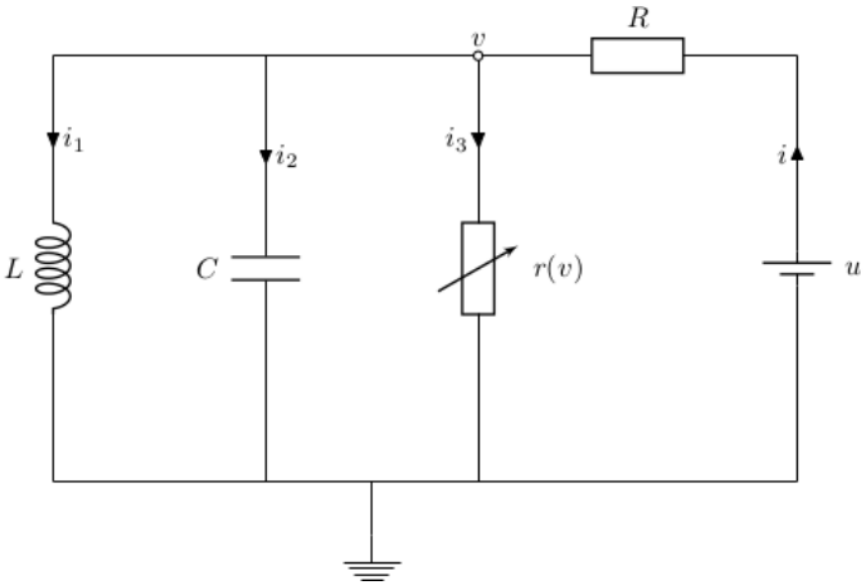
Contextualisation

- Équation générale: $$\ddot{x} - \varepsilon(1 - x^2)\dot{x} + \omega_0^2 x = 0$$

Où:

- ϵ = frottement;
- ω_0 = pulsation du système

Analogie électrocinétique du cœur



Crédit: CNRS

$$\frac{\mathrm{d}^2 v}{\mathrm{d}t^2} - \frac{1}{RC}\frac{\mathrm{d}v}{\mathrm{d}t} + \frac{1}{C}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{v}{r(v)}\right) + \frac{1}{LC}v = 0$$

$$r(v) = \frac{RV_0^2}{v^2} \Rightarrow \frac{v}{r(v)} = \frac{v^3}{RV_0^2}$$

$$\Rightarrow \frac{d(\frac{v}{r(v)})}{dt} = \frac{3v^2}{RV_0^2}\frac{dv}{dt}$$

$$Posons \;\; x = \frac{\sqrt{3}v}{V_0}$$

$$\boxed{\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \frac{1}{RC}(1-x^2)\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{1}{LC}x = 0}$$

Résolution numérique: méthode d'Euler

$$\ddot{x} - \varepsilon(1 - x^2)\dot{x} + \omega_0^2 x = 0 \Leftrightarrow \begin{cases} y = \dot{x} \\ \dot{y} = \varepsilon(1 - x^2)\dot{x} - \omega_0^2 x = 0 \end{cases}$$

$$En\ posant\ X = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}$$

$$Notons\ h\ = \frac{t_{max}}{N-1}$$

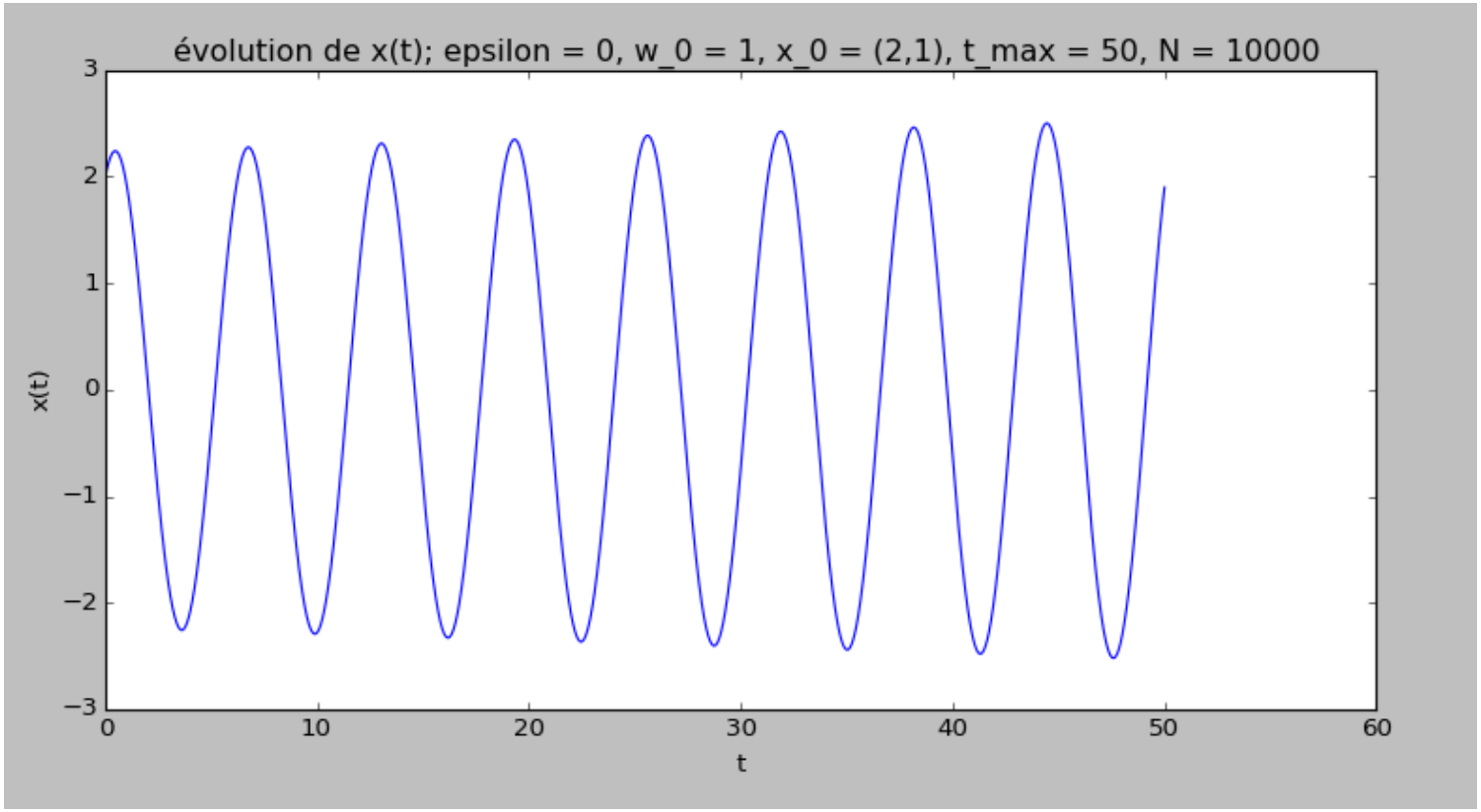
Où:

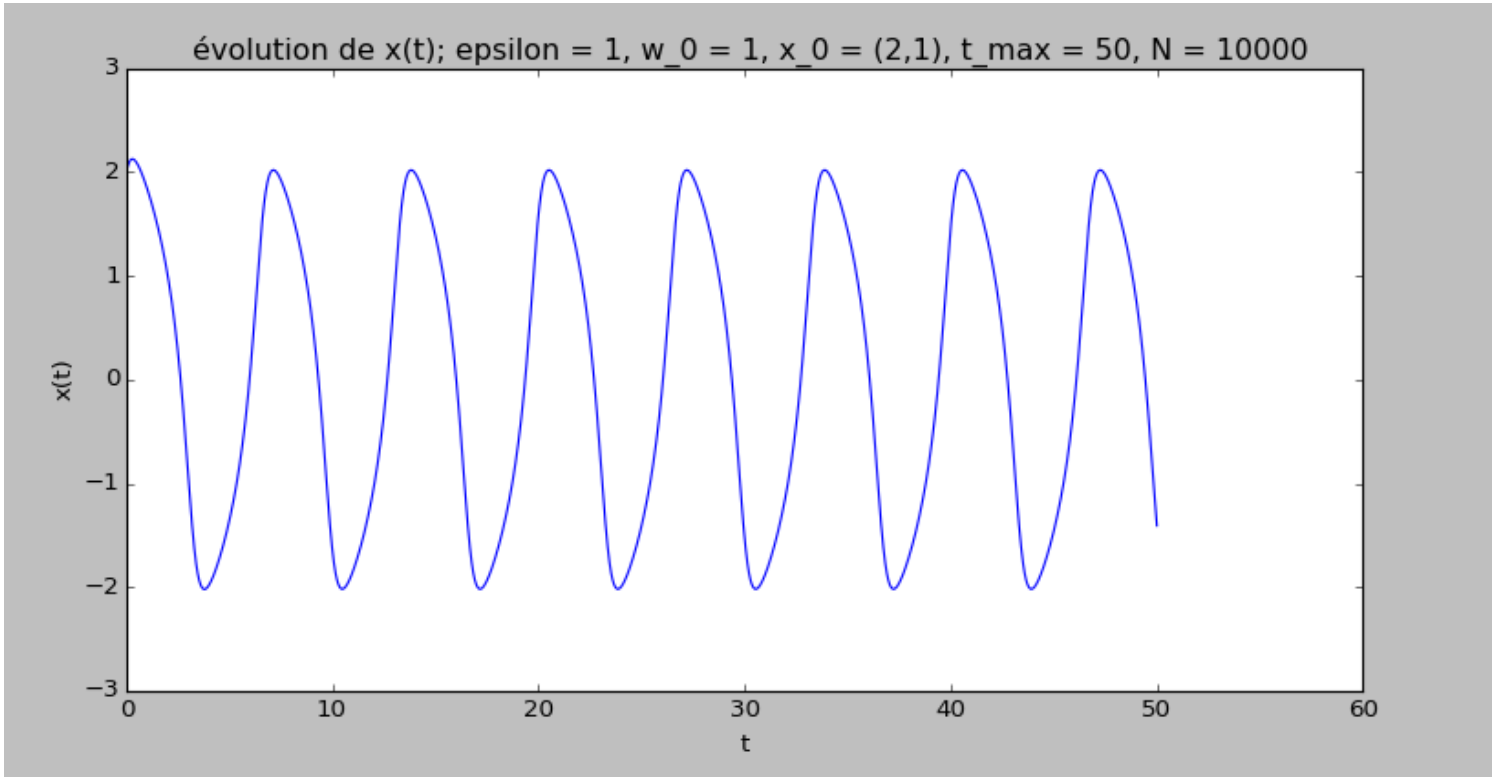
- t_{max}= temps d'intégration;
- N = nombre de points

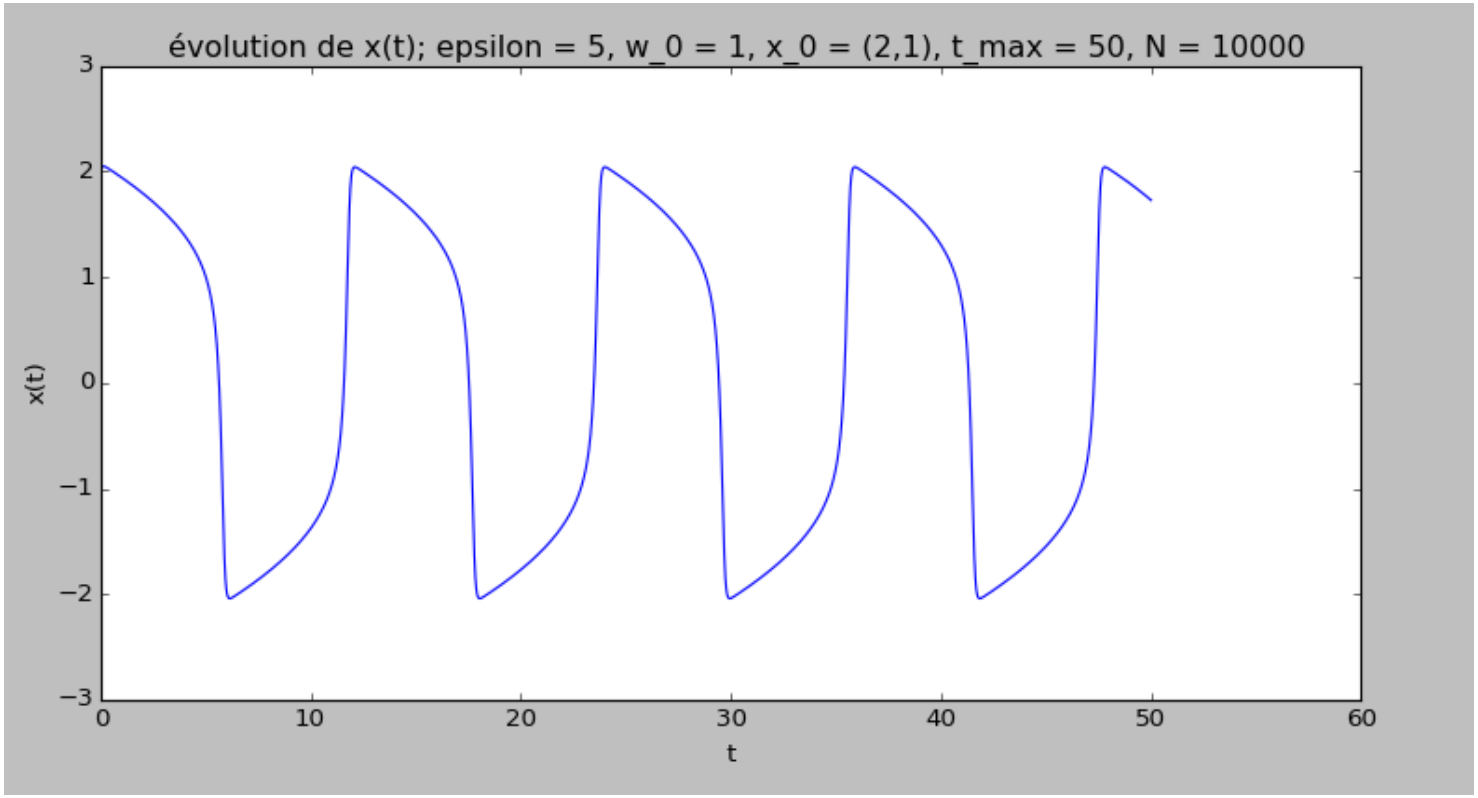
A la i^{ème} itération:

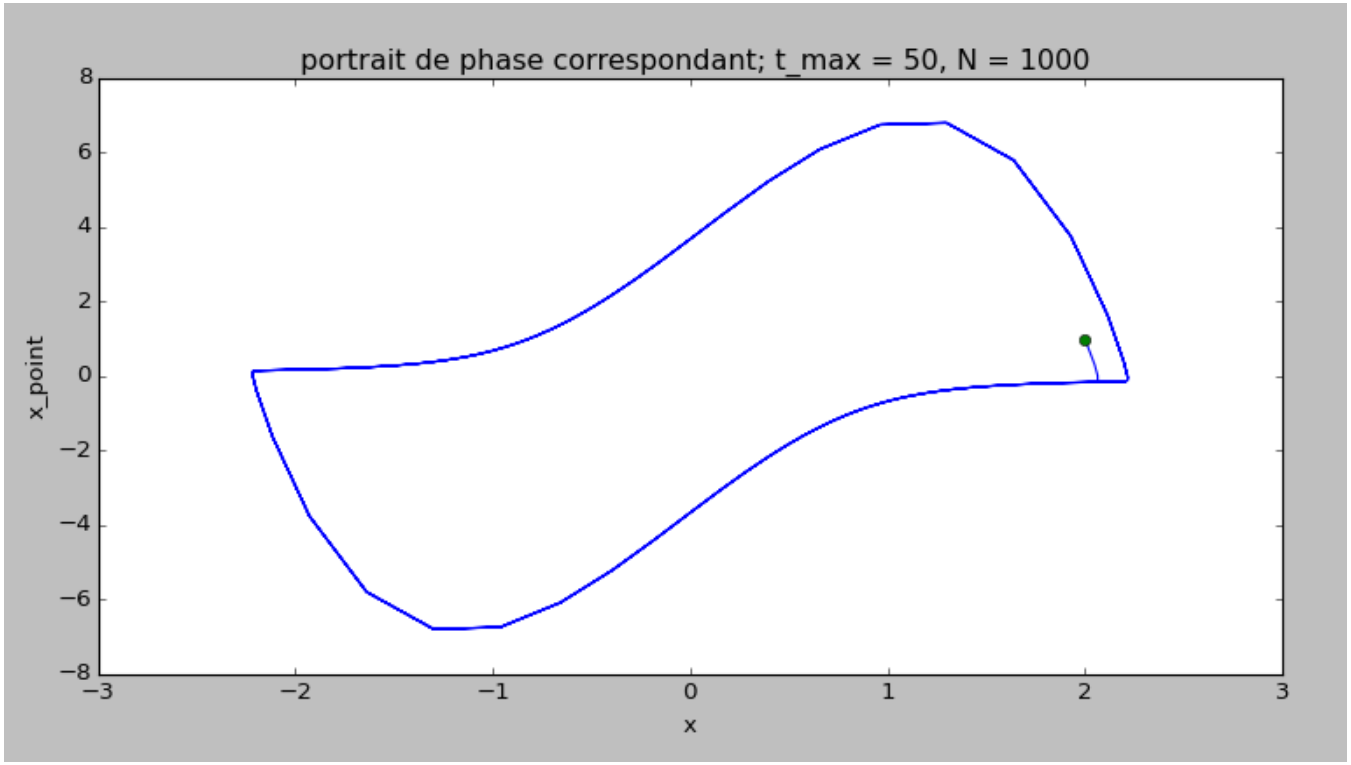
$$\begin{cases} t_i = t_0 + i \times h \\ f(X(t_i), \varepsilon) = \frac{X(t_{i+1}) - X(t_i)}{h} \end{cases}$$

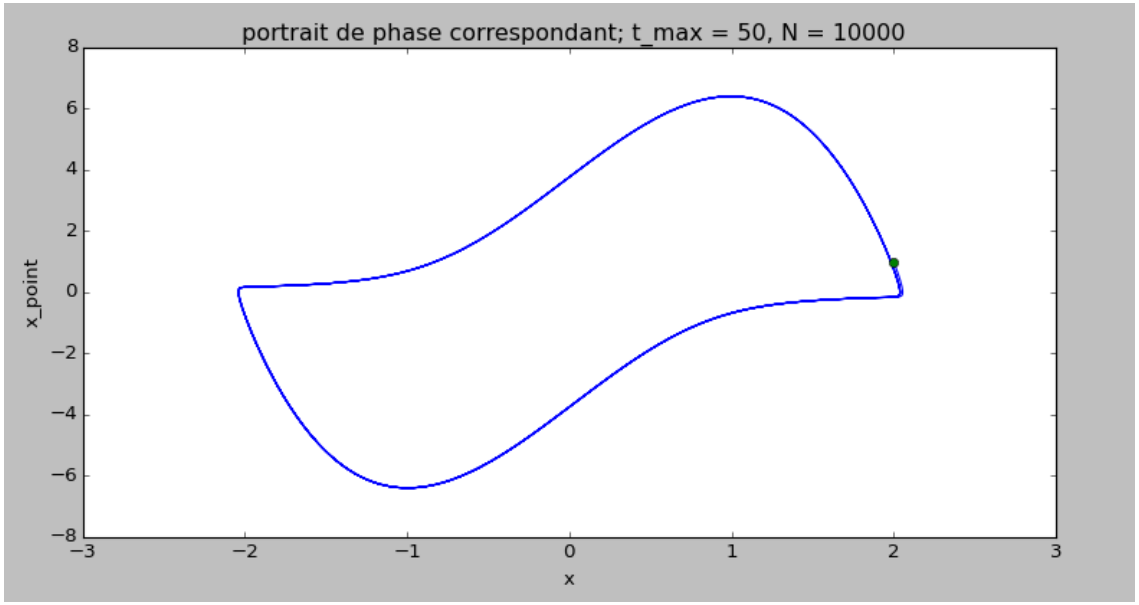
$$\Rightarrow X(t_{i+1}) = X(t_i) + f(X(t_i), \varepsilon)$$

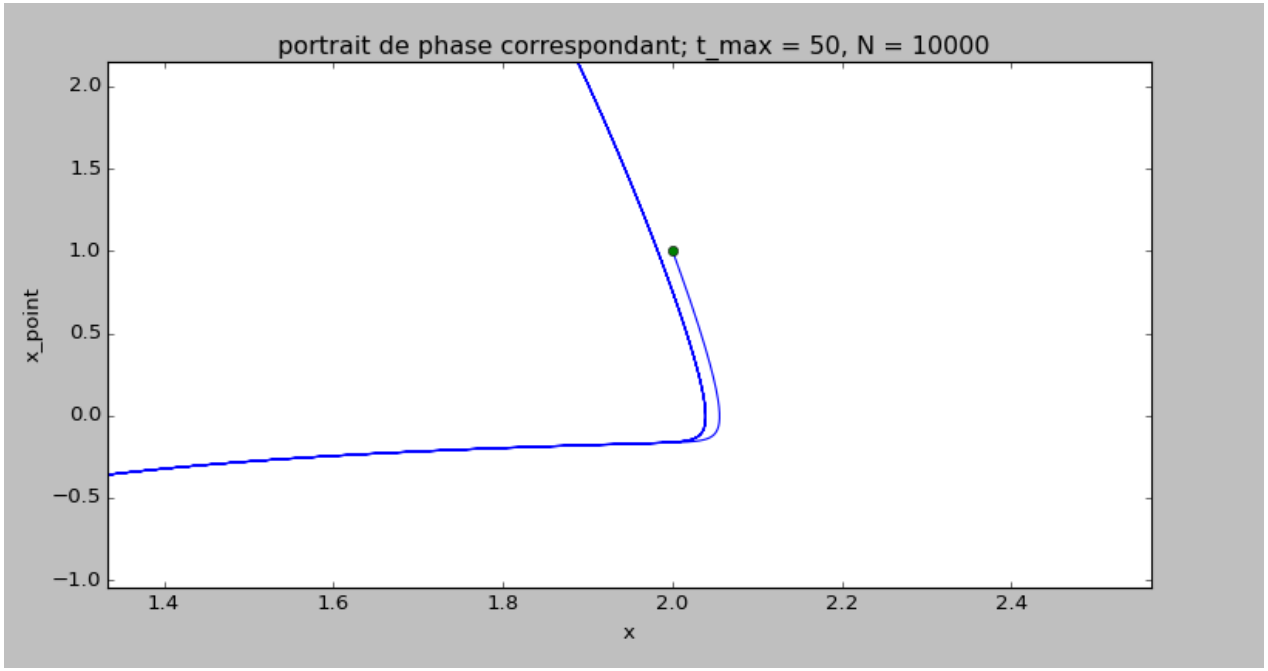

évolution de x(t); epsilon = 0, w_0 = 1, x_0 = (2,1), t_max = 50, N = 10000
x(t)
t
3
2
1
0
−1
−2
−3
0
10
20
30
40
50
60

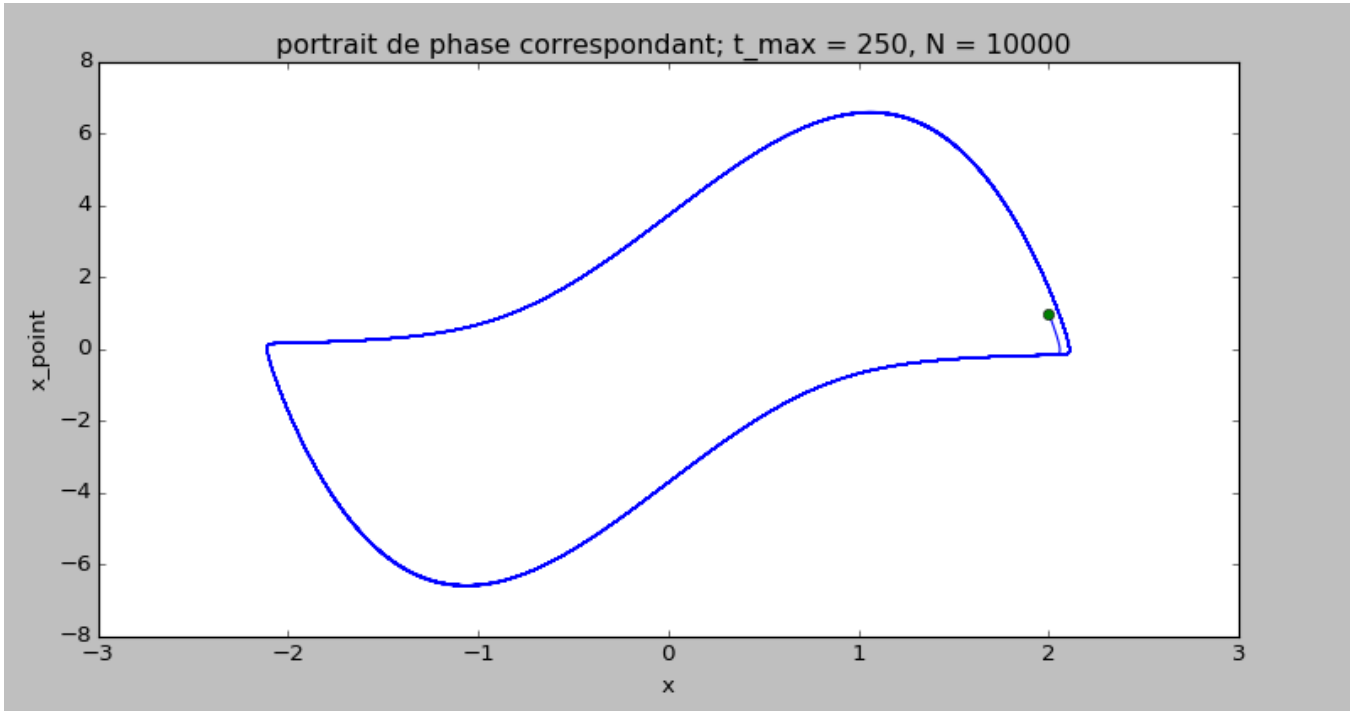

évolution de x(t); epsilon = 1, w_0 = 1, x_0 = (2,1), t_max = 50, N = 10000
x(t)
t
3
2
1
0
−1
−2
−3
0
10
20
30
40
50
60


évolution de x(t); epsilon = 5, w_0 = 1, x_0 = (2,1), t_max = 50, N = 10000
x(t)
t
3
2
1
0
−1
−2
−3
0
10
20
30
40
50
60


portrait de phase correspondant; t_max = 50, N = 1000
x_point
x
8
6
4
2
0
−2
−4
−6
−8
−3
−2
−1
0
1
2
3


portrait de phase correspondant; t_max = 50, N = 10000
x_point
x
8
6
4
2
0
−2
−4
−6
−8
−3
−2
−1
0
1
2
3


portrait de phase correspondant; t_max = 50, N = 10000
x_point
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
1.4
1.6
1.8
2.0
2.2
2.4


portrait de phase correspondant; t_max = 250, N = 10000
x_point
x
8
6
4
2
0
−2
−4
−6
−8
−3
−2
−1
0
1
2
3

Erreur en O(h)

Étude des bifurcations

$$\ddot{x} - \varepsilon(1 - x^2)\dot{x} + \omega_0^2 x = 0 \Leftrightarrow \begin{cases} y = \dot{x} \\ \dot{y} = \varepsilon(1 - x^2)\dot{x} - \omega_0^2 x = 0 \end{cases}$$

$$f \quad : \quad (x, y) \quad \mapsto (\ f_1(x, y), f_2(x, y)\ )$$

$$\text{où} \begin{cases} f_1(x, y) = y \\ f_2(x, y) = \epsilon(1 - x^2)y - \omega_0^2 x \end{cases}$$

$$f(x,y) \simeq df(0,0)(x,y) = (y, -\omega_0^2 x + \varepsilon y)$$

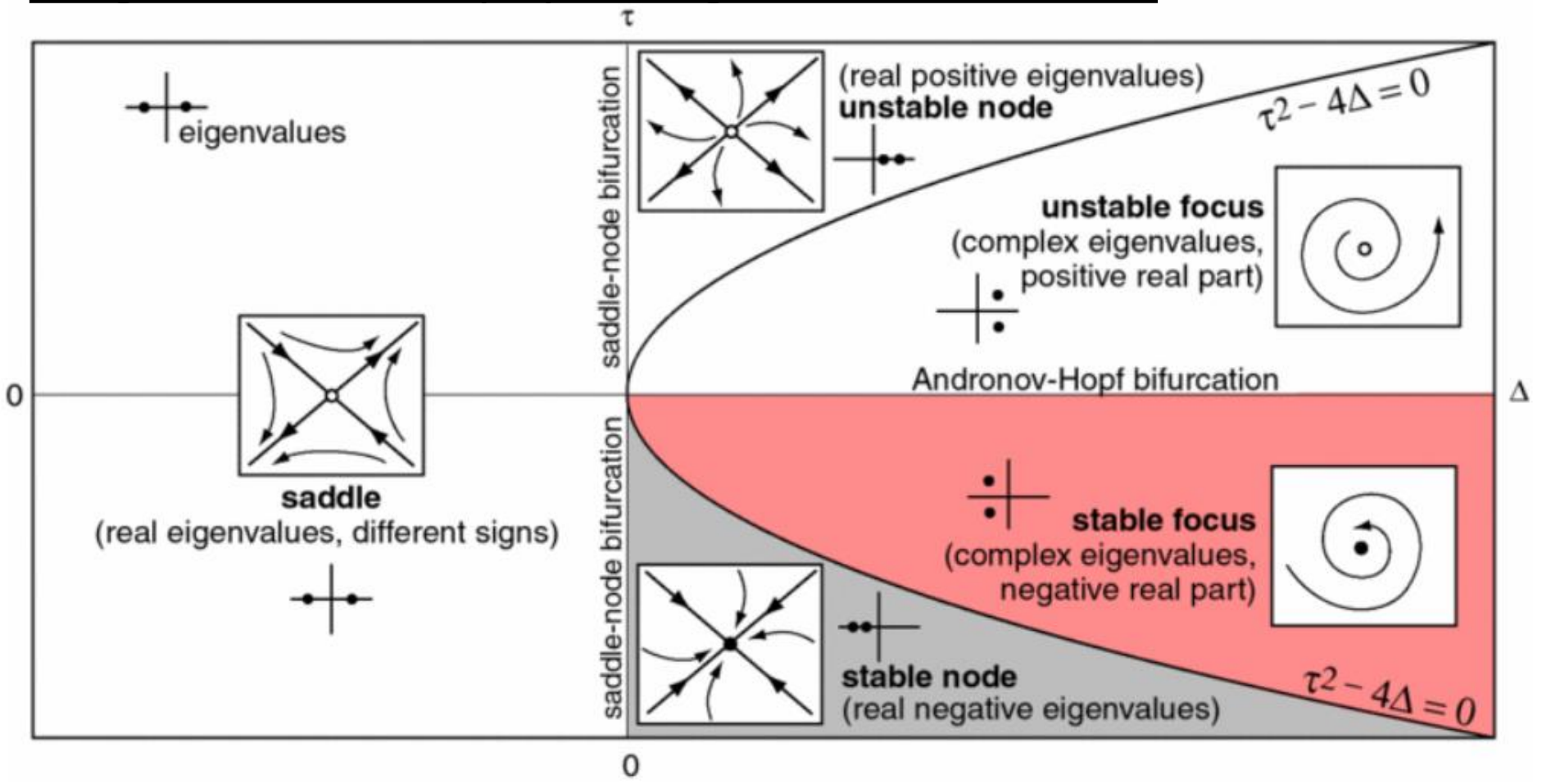
$$\mathbf{J}_{f\begin{pmatrix}0\\0\end{pmatrix}} = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & \varepsilon \end{pmatrix}$$

$$Notons\ J_0 = \mathbf{J}_{f\begin{pmatrix}0\\0\end{pmatrix}} \Rightarrow\ \chi_{J_0} = X^2 - \varepsilon X + \omega_0^2$$

$$\Delta_{\chi_{J_0}} = \tau^2 - 4\Delta\ \ en\ posant\ \tau = \varepsilon\ et\ \Delta = \omega_0^2$$

$$\lambda_{J_0} = \frac{\varepsilon}{2} \pm \frac{\sqrt{\varepsilon^2 - 4\omega_0^2}}{2}$$

Diagramme de Hopf par Eugene M. Izhikevich



Conclusion: $\varepsilon \geq 2\omega_0$

II) Modèle de Zeeman

$$\begin{cases} \dot{x} = \frac{-x^3 - Tx + y}{\varepsilon} \\ \\ \dot{y} = x - x_d + u(x_d - x_s) \end{cases}$$

Où:

- x = longueur des cellules myocardiques;
- y = activité électrochimique globale du cœur;
- T = tension musculaire appliquée (T > 0);
- u = fonction temporelle;
- ϵ = paramètre intrinsèque au système

$$f \quad : \quad (x,y) \quad \mapsto (\tfrac{1}{\epsilon}(x^3 - \mathrm{T}x + y), x - x_d + \mathrm{u}(x_d - x_0))$$

$$\mathbf{J}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} -\frac{1}{\epsilon}(3x^2 - T) & -\frac{1}{\epsilon} \\ 1 & 0 \end{bmatrix}$$

$$\lambda = \tfrac{1}{2\epsilon}((-3x_1^2 - T) \pm \sqrt{(-3x_1^2 - T)^2 - 4\epsilon}))$$

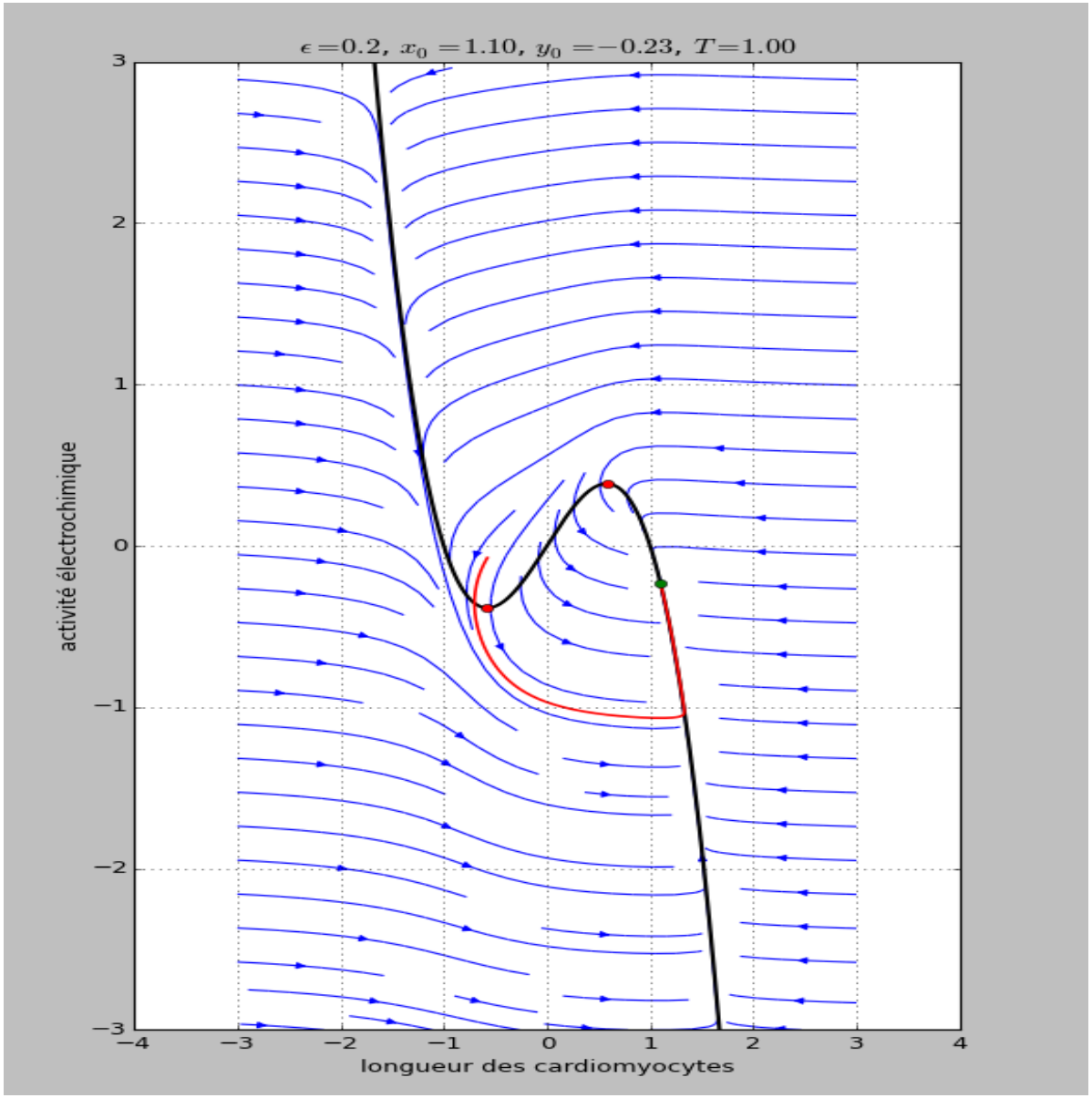
Équilibre: $x^3 - \mathrm{T}x + y = 0 \Rightarrow y = -x^3 + Tx$

Seuil de l'action: $3x^2 - T = 0 \Leftrightarrow x = \pm\sqrt{\frac{T}{3}}$

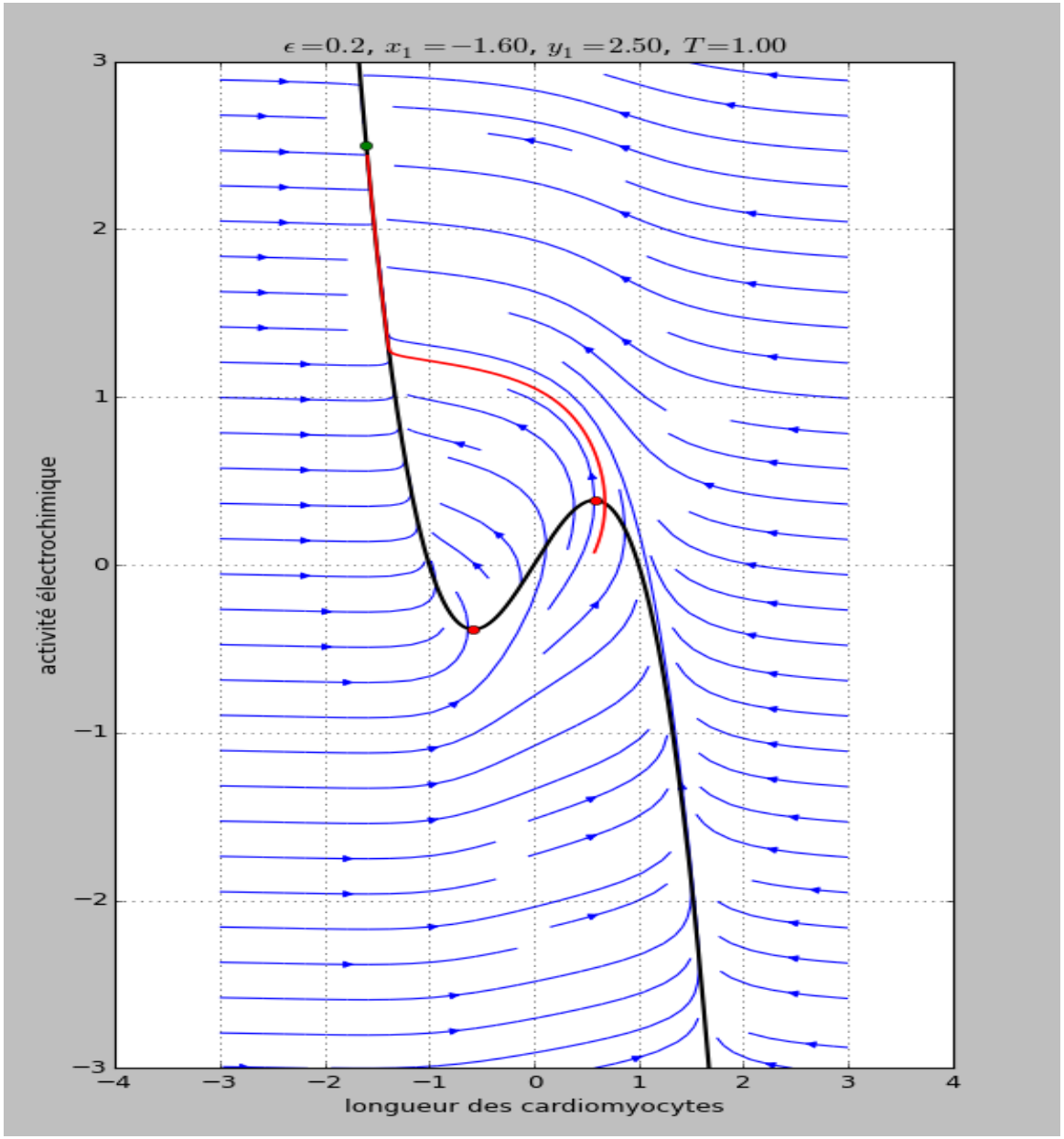
$$\boxed{x_a = -x_b = \sqrt{\frac{T}{3}}}$$

Notons A(x_a, y_a) et B(x_b, y_b) les points délimitant l'action.

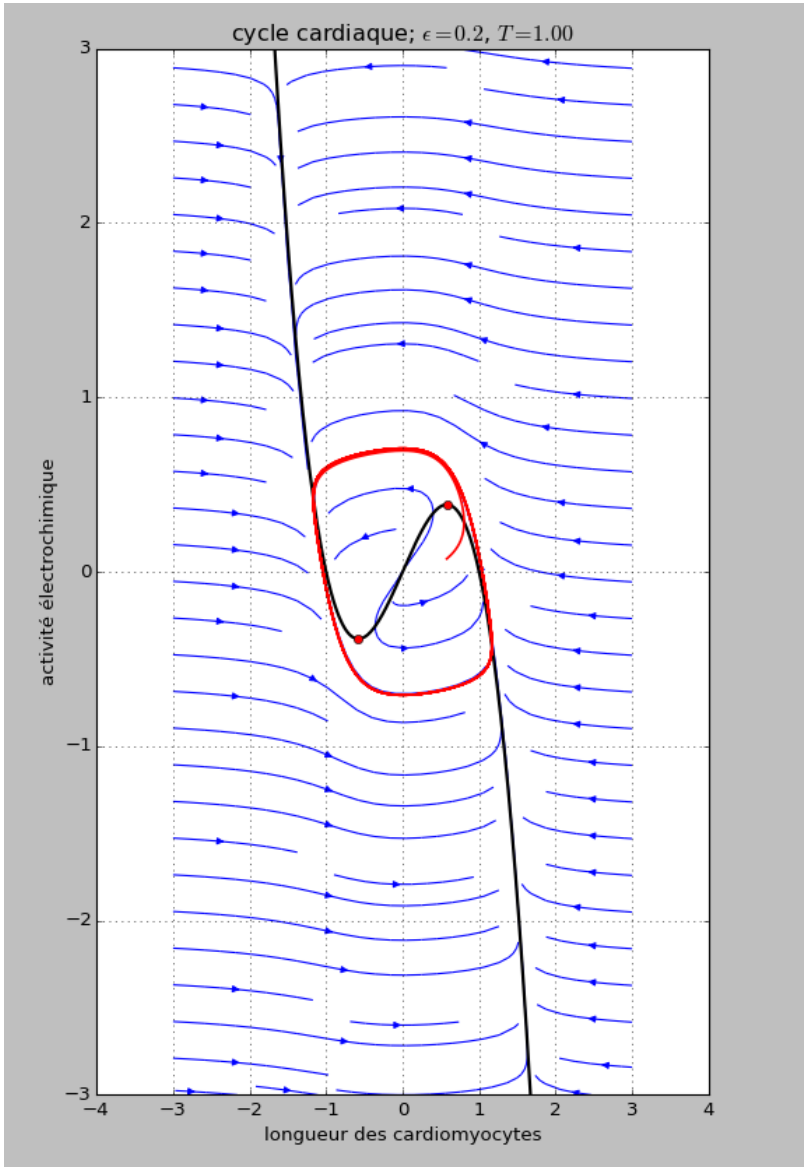
Simulation de la diastole

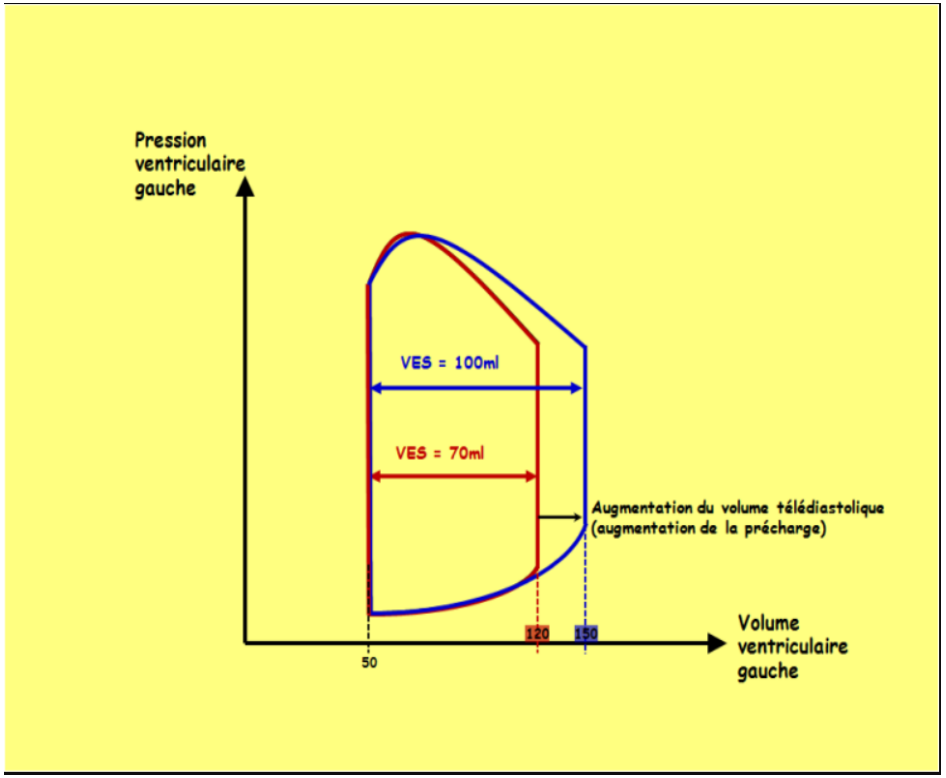


Simulation de la systole



Lien avec la loi de Frank-Starling





Crédit: Unisciel

Fin de la présentation

Merci de votre écoute.

Annexe 1: Estimation de l'erreur de la méthode d'Euler

$$x(t+h) = x(t) + hf(x(t), \varepsilon)$$

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + O(h^2)$$

$$\Rightarrow f(x(t), \varepsilon) = \frac{x(t+h) - x(t)}{h} = x'(t) + \frac{h}{2}x''(t) + O(h)$$

$$Posons\ X = \begin{pmatrix} x_0 \\ x_1 \\ \dots \\ x_{N-1} \end{pmatrix}$$

$$Pour\ j \in [\![1, N-1]\!], \begin{cases} x(t_j)\ valeur\ exacte\ au\ temps\ t_j \\ f_1(t_j) = \frac{x_{j+1}-x_j}{h} \end{cases}$$

$$\begin{aligned} \tau_j &= \frac{x(t_{j+1}) - x(t_j)}{h} - f_1(t_j) \\ &= [x'(t_j) + \frac{h}{2}x''(t_j) + O(h)] - f_1(t_j) \\ &\simeq \frac{h}{2}x''(t_j) + O(h) \end{aligned}$$

$$Posons\ X^* = \begin{pmatrix} x(t_0) \\ x(t_1) \\ \cdots \\ x(t_{N-1}) \end{pmatrix}\ et\ E = X - X^* = \begin{pmatrix} x_0 - x(t_0) \\ x_1 - x(t_1) \\ \cdots \\ x_{N-1} - x(t_{N-1}) \end{pmatrix}$$

$$En\ notant\ A = \frac{1}{h}\begin{pmatrix} 1 & & & & \\ 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & & \\ & & & 1 & -1 \end{pmatrix},\ E\ v\acute{e}rifie$$

$$AE = T = \begin{pmatrix} \tau_0 \\ \tau_1 \\ \cdots \\ \tau_{N-1} \end{pmatrix} \Leftrightarrow e'(t) = \tau(t)\ et\ e(0) = 0$$

$$\Rightarrow e(t) = \frac{h}{2}x'(t) + C$$

$$\Rightarrow \boxed{e(t) = \frac{h}{2}x'(t) + x_0 - x(t_0)(\frac{h}{2} + 1) + O(h)}$$

Annexe 2: Validité du modèle de Zeeman

$$x^3 - \mathrm{T}x + y = 0 \Rightarrow \boxed{y = -x^3 + Tx}$$

Seuil de l'action: $3x^2 - T = 0 \Leftrightarrow x = \pm\sqrt{\frac{T}{3}}$

$$\boxed{x_a = -x_b = \sqrt{\frac{T}{3}}}$$

Notons A(x_a, y_a) et B(x_b, y_b) les points délimitant l'action.

Notons $D(x_d, y_d)$ (diastole)

Diastole valide:

$$x_d > x_a$$

$$\Leftrightarrow x_d > \sqrt{\frac{T}{3}}$$

$$\Leftrightarrow \boxed{T < 3x_d^2}$$

Notons $S(x_s, y_s)$ (systole)

Systole valide:

$$y_s > y_b$$
$$\Leftrightarrow y_s > Tx_b - x_b^3$$
$$\Leftrightarrow y_s > \frac{T^{\frac{3}{2}}}{3\sqrt{3}} - \frac{T^{\frac{3}{2}}}{\sqrt{3}}$$
$$\Leftrightarrow y_s > -\frac{2T^{\frac{3}{2}}}{3\sqrt{3}}$$
$$\Leftrightarrow y_s^2 > \frac{4T^3}{27}$$

$$\Leftrightarrow \boxed{T^3 < \frac{27}{4} y_s^2}$$

Listing Van der Pol:

```
import numpy as np
from matplotlib.pyplot import *

w_0 = 1                                                 # pulsation du système

def f(X,epsilon):
    x_1,x_2 = X                                         # changement de variable matriciel
    next_X = np.array([x_2,epsilon*(1-x_1**2)*x_2-w_0*x_1])
    return next_X

def derivee(x0,epsilon,tmax,N):
    h = tmax/(N-1)
    X = [x0]
    T = [0]
    for i in range(N):
        x_i = X[-1]                                     # on utilise les dernières valeurs ajoutées dans X et T
        t_i = T[-1]
        X += [x_i+h*f(x_i,epsilon)]                     # taux d'accroissement avec Euler
        T += [t_i+h]
    return np.array(X),np.array(T)


X1,T1 = derivee(np.array([2,1]),2,50,10000)             # exemple de condition initiale


def extrait(X,T):
    x = [elem[0] for elem in X]                         # courbe représentative de la solution de l'équation différentielle
    return x,T

def extrait_valeurs(X):                                 # portrait de phase
    x = [elem[0] for elem in X]                         # valeurs de x
    v = [elem[1] for elem in X]                         # valeurs de x_point
    return x,v


subplot(221)
x_1,t_1 = extrait(X1,T1)
plot(t_1,x_1)
title(" évolution de x(t); epsilon = 2, w_0 = 1, x_0 = (2,1), t_max = 50, N = 10000")
xlabel("t")
ylabel("x(t)")
subplot(222)
x_1,v_1 = extrait_valeurs(X1)
plot(x_1,v_1)
title(" portrait de phase correspondant; epsilon = 2, t_max = 50, N = 10000 ")
xlabel("x")
ylabel("x_point")
plot(x_1[0],v_1[0],'og')                                # coordonées de la condition initiale
```

```python
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
import subprocess


epsilon, w_0 = 2, 1

def f(x, y):                                                    # changement de variable matriciel
  return y, epsilon*(1-x**2)*y-w_0*x                            # changement y = x_point


def f2(P,t):                                                    # fonction qui definit le flot sur R2
    return V(P[0],P[1])


xmin, xmax, dx = -3, 3, 0.1                                     # réseau de points en x,y
ymin, ymax, dy =  -3, 3, 0.1
X,Y = np.meshgrid(np.arange(xmin, xmax, dx), np.arange(ymin, ymax, dy) )
f1, f2 = f(X,Y)                                                 # calcule le champ de vecteur f sur le réseau qui correspond à l'espace de phases (y,y_point)

# Dessin des lignes de champ
plt.streamplot(X,Y, f1,f2)
plt.title('champ de vecteur en (x,y)')
plt.xlabel('x')
plt.ylabel('y')
plt.title("Espace de phase; epsilon = 2, w_0 = 1")
plt.plot(0,0,'og')
```

Listing Zeeman:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import odeint

#%%
eps=0.2; T=1; x0 = 1.1; u = 1; x1 = -1.6                                    # paramètres; diastole: u = 0; systole: u = 1

#Résolution du système
def zeeman(y):
    x,b = y
    return [ -(b+x**3-T*x)/eps, x-x0 + u*(x0-x1) ];


#Création du flot
x_lim = [-3,3]
y_lim = [-3,3]
X, B = np.meshgrid(np.arange(x_lim[0], x_lim[1]+0.01, 0.1), np.arange(y_lim[0], y_lim[1]+0.01, 0.1))

x_cub = np.linspace(x_lim[0], x_lim[1], 501)
b_cub = T*x_cub-x_cub**3
t_sol = np.linspace(0, 20, 2501)



a = (T/3.0)**0.5                                                            # abcisse du point A délimitant le début de l'action
plt.ylim(y_lim)
plt.plot(x_cub, b_cub,'k', lw=2)                                            # équilibre du système
dXdt, dBdt = zeeman([X,B])
plt.streamplot(X, B, dXdt, dBdt)
y_sol = odeint(lambda y,t: zeeman(y), [(T/3)**0.5, (1/(27)**0.5)*2*(T/3)*(T/3)**0.5], t_sol)            # solution du système
x_sol, b_sol = y_sol.T
plt.xlabel('longueur des cardiomyocytes')
plt.ylabel('activité électrochimique')
plt.plot(x_sol, b_sol, 'r', lw=1.5)
plt.grid()
plt.plot(x1,T*x1-x1**3,'og')                                                # repérer la systole
plt.plot(a,T*a-a**3,'or')                                                   # dèlimitation de l'action
plt.plot(-a,-T*a+a**3,'or')
plt.title('$\epsilon=%.2g$, $x_1=%.2f$, $y_1=%.2f$, $T=%.2f$'%(eps,x1,T*x1-x1**3,T))
plt.show()
```