```python
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 22 13:12:43 2022

@author: cleme
"""

from numpy import *
import sympy as sp
import matplotlib.pyplot as plt


class ML():
    """modelisation of Moris Lecar model.
    """
    def __init__(self):
        self.V = linspace(-65,20,1000)
        #----initialisation of all parameters
        self.EL=-60
        self.EK=-84
        self.ECa=120
        #--
        self.V1=-1.2
        self.V2=18
        self.V3=2
        self.V4=30
        #--
        self.gL=2
        self.gK=8
        self.gCa=4.4
        #--
        self.C=20
        self.gamma=0.04
        self.W=0.5
        self.I=0

    #==========================================================================
    # Definition of all model's functions
    def m_inf(self, V):
        return (1+tanh((V-self.V1)/self.V2))/2

    def w_inf(self, V):
        return (1+tanh((V-self.V2)/self.V4))/2

    def to_inf(self, V):
        return 1/(cosh((V-self.V3)/(2*self.V4)))

    def W_null(self, V):
        return self.gamma*(self.w_inf(V)-self.W/self.to_inf(V))

    def V_null(self, V, I):
        #return(-gCa*((1+tanh((V-V1)/V2))/2)*(V-ECa)-gL*(V-EL)+I)/gK*(V-EK)
        return (I - self.gL*(V-self.EL)-self.gCa*((1 + tanh((V -self.V1)/self.V2))/2)*(V-se

    #==========================================================================
    def isoclines(self, Imin, Imax, nI):
        """show impact of gamma variation
```

```
        """
        V=self.V
        color=['r-', 'y-', 'g-', 'purple', 'black']
        h=(Imax-Imin)//nI
        i_c=0
        legend=[]
        for I in range(Imin, Imax, h ):
            plt.plot(V, self.V_null(V, I), color[i_c%len(color)])
            i_c+=1
            legend.append(f"V(t) : I={I}")
        plt.plot(V, self.w_inf(V), "b-")
        legend.append("W(t)")
        plt.legend(legend)
        plt.title("Tracé des isoclines de W et de V \n pour différentes valeurs de I")
        plt.show()

    def middle_branche(self, Gmin, Gmax, nG):
        """Tracé de W pour différentes valeurs de gamma
        """
        V=self.V
        legend=[]#initialisation légende
        color=['r-', 'y-', 'g-', 'purple', 'black']
        i_c=0#compteur pour indice couleur
        for G in linspace(Gmin, Gmax, nG):
            self.gamma = G #changement valeur de gamma
            plt.plot(V, self.W_null(V), color[i_c%len(color)])
            i_c+=1
            legend.append(f"W(t) : gamma={G}")
        #plt.plot(V, self.w_inf(V), "b-")
        plt.legend(legend)
        plt.title("Tracé de W pour différentes valeurs de gamma")
        plt.show()
        self.gamma=0.04 #réinitialisation de gamma

    def V_intersept(self, iso1, iso2):
        """return nulcline intersection coordinates. Can be use to get intersection point c
        any two fuctions iso1 and iso2
        """
        #https://askcodez.com/intersection-de-deux-graphes-en-python-trouvez-la-valeur-x.ht
        V = linspace(-65, 20, 100) #abscisse
        f = iso2(V, 0)
        g = iso1(V)

        plt.plot(V, f, '-')
        plt.plot(V, g, '-')

        idx = argwhere(diff(sign(f - g)) != 0).reshape(-1) + 0 #get intersection abscisses

        plt.plot(V[idx], f[idx], 'ro') #draw the red dots
        plt.legend(['V(t) : I=0', 'W(t)', "points intersection"])
        plt.title("Tracé des points d'intersection de V et W")
        plt.show()

        intersections=[]
        for i in range(len(V[idx])):
            xai = V[idx][i]
            yai = f[idx][i]
```

```python
            intersections.append((xai, yai))
        return intersections

    def ML_pros(self):
        """draw the V and W processus (integration of V' and W' in the ML model)
        """
        v = Symbol("v")
        w = Symbol("w")
        dw = self.gamma * (((1 + tanh((v - self.V2)/self.V4))/2)- w)/((cosh((v - self.V3)/(
        dv = -self.gCa * ((1 + tanh((v - self.V1)/self.V2))/2) * (v - self.ECa) - self.gK *
        return [Integral(dw, v)]

    def jacobienne(self, I):
        """Returne the symbolic expression of the ML system's Jacobienne
        """
        v = Symbol("v")
        w = Symbol("w")
        dw = self.gamma * (((1 + tanh((v - self.V2)/self.V4))/2)- w)/((cosh((v - self.V3)/(
        dv = -self.gCa * ((1 + tanh((v - self.V1)/self.V2))/2) * (v - self.ECa) - self.gK *
        J = [[dv.diff(v), "-------------", dv.diff(w)],[dw.diff(v), "-------------", dw.dif
        return J

    def J(self, v, w):
        """Calculate the Jacobienne of ML system, based on ML.jacobienne() output expressio
        """
        return array([[-8*w + (v - 120)*(0.122222222222222*tanh(v/18 + 0.0666666666666667)*
          960 - 8*v],
         [(0.000666666666666667 - 0.000666666666666667*tanh(v/30 - 3/5)**2)*cosh(v/60 - 1/3
          -0.04*cosh(v/60 - 1/30)]],dtype=float)

    def steady_point(self):
        """Determinates the nature of steady states
        """
        #----
        det = [] #déterminants
        vp = [] #valeur propres
        tr = [] #trace
        intersept = self.V_intersept(self.w_inf, self.V_null)
        #----
        for dot in intersept:
            xai=dot[0]
            yai=dot[1]
            jac = ml.J(xai, yai) #jacobienne au point dot
            #print(dot, jac)
            det.append(linalg.det(jac)) #déterminant
            vp.append(linalg.eigvals(jac)) #valeur propre
            tr.append(sum(vp[-1])) #trace

        #----interprétation des résultats
        for i in range (len(intersept)): #pour chaque points
            #----
            print("TRACE", tr[i])
            if all([vpi > 0 for vpi in vp[i]]): print(f" {intersept[i]} minimum, instable")
            elif all([vpi < 0 for vpi in vp[i]]): print(f" {intersept[i]} maximum, stable")
            else : print(f" {intersept[i]} point selle, instable")
            #----
        if tr[i]**2 < 4*det[i]: print(f"{intersept[i]} est une spirale")
```

```python
        else: print(f"{intersept[i]} est indéterminé (un neud ?)")



if __name__ == '__main__':

    ml = ML()
    #ml.isoclines(0, 100, 4)
    #ml.steady_point()
    ml.middle_branche(0.001, 0.04, 4)
    ml.isoclines(0, 100, 4)
    #print(ml.V_intersept(ml.w_inf, ml.V_null))
    ml.steady_point()
```