```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy as ipw
from scipy import integrate
```

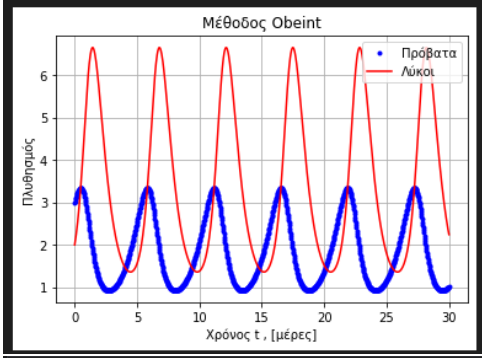
✓ 8.1s

```python
alpha = 1 #Αύξηση θηραμάτων όταν οι θηρευτές απουσιάζουν
beta = 0.3 #Πως πεθαίνουν τα θηράματα λόγω των θηρευτών
delta = 0.8 #Μετά απο πόσα θηράματα που έχουν πιαστεί θα έχουμε νεό θηρευτή
gamma = 1.5 #Πως μειόνονται οι θηρευτές όταν απουσιάζουν τα θηράματα
x0=3
y0=2
#Σε αυτό το σημείο ορίζουμε την συνάρτηση μας με όποια απο τα a,b,g,d έχουμε επιλέξει να χρησιμοποιήσουμε.
def derivative(X, t, alpha , beta, delta , gamma):
        x,y = X
        dotx=x*(alpha -beta*y)
        doty=y*(-gamma + delta*x)
        return np.array([dotx,doty])
```

```python
Nt=1000
tmax=30
t=np.linspace(0., tmax, Nt) #Αυτή την εντολή την χρησιμοποιούμε για να ορίσουμε τον αριθμό των δειγμάτων που στην συγκεκριμένη περίπτωση ξεκινάει απο 0
                            # φτάνει έως το tmax και παράγει Nt δείγματα , όσο λιγότερα δείγματα τόσο 'χειρότερα' θα βγούνε τα γραφήματα
X0=[x0,y0]#Η αρχική συνθήκη αν θέλουμε να υπάρχει
res= integrate.odeint(derivative, X0, t, args=(alpha, beta , delta, gamma))#Μέσω αυτής της συνάρτησης μπορούμε να λύσουμε την διαφορική που έχουμε ορίσει παραπάνω
x,y=res.T                   #όπου args είναι αυτά που χρειαζόμαστε για να λειτουργήσει η συνάρτηση όπως την έχουμε ορίσει
#και η εντολή .T μας αντιστέφει του x,y άξονες για να φτιάξουμε το διάγραμμα μέτα.
```

```python
plt.figure()
plt.grid() #για να έχει το γράφημα γραμμές
plt.title('Μέθοδος Obeint')
plt.plot(t, x , '.b', label='Πρόβατα') #τα x,y  που έχουμε απο την λύση της διαφορικής
plt.plot(t, y, '-r', label='Λύκοι')
plt.xlabel('Χρόνος t , [μέρες]')
plt.ylabel('Πλυθησμός')
plt.legend() #μας δείχενι τις κατηγορίες των δεδομένων που παρουσιάζουμε στο γράφημα στο κουτάκι πάνω δεξία
plt.show()
```

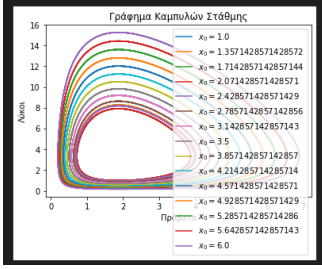
✓ 0.4s




```python
plt.figure()
I=np.linspace(1.0, 6.0, 15) #αρχικές συνθήκες για τον πληθυσμό των προβάτων , αρχική τιμή , τελική τιμή και πλήθος δειγμάτων αντίστοιχα
for sheep in I :
    X0=[sheep, 1.0]
    Xf=integrate.odeint(derivative, X0, t, args=(alpha, beta , delta, gamma)) #η ένταξη της διαφορικής εξίσωσης ενω το βήμα αλλάζει κάθε φορα X0
    plt.plot(Xf[:,0], Xf[:,1], "-", label = "$x_0 =$"+str(X0[0]))       #οπου XF οι λύσεις της διαφορικής για ξεχωριστά x0 κάθε φορά που αλλάζουν μέσα στην ενταλή επανάληψης
plt.xlabel('Πρόβατα')
plt.ylabel('Λύκοι')
plt.legend()
plt.title('Γράφημα Καμπυλών Στάθμης')    #οπου το κέντρο ειναι (γ/δ,α/β) που έχουμε ορίσει στην αρχή του κώδικα
```

✓ 1.2s

```
Text(0.5, 1.0, 'Γράφημα Καμπυλών Στάθμης')
```

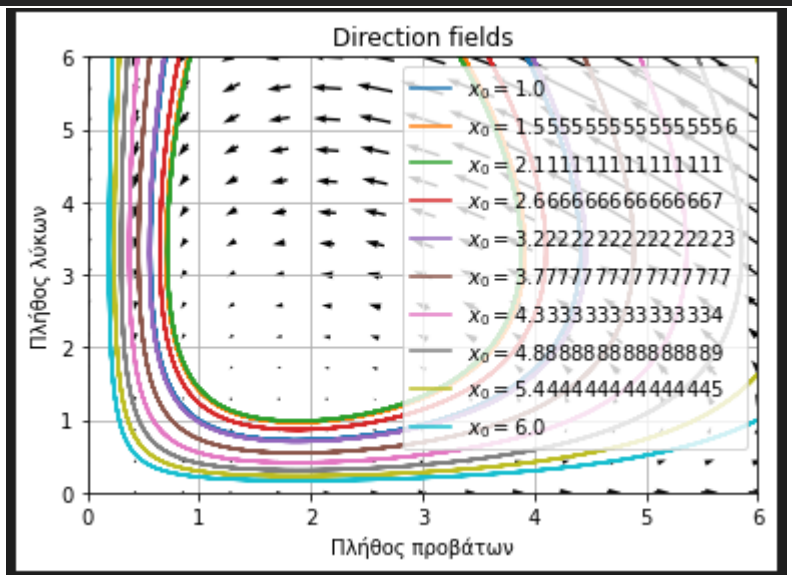


```
n=15
x=np.linspace(0,6, n) #κανει generate 15 αριθμούς απο 0-6
y=np.linspace(0,6, n)
X1 , Y1 =np.meshgrid(x,y) #Με αυτή την εντολή φτιάχνουμε έναν πίνακα με διαστάσεις τα x*y (π.χ για n=15 η διάσταση του πίνακα ειναι 15x15) τα στοιχεία του πίνακα θα τα χρησιμοποιήσουμε για την
                          #δημιουργία βελακίων


D=[X1,Y1]
Dx1 , Dy1 =derivative(D ,t=0 , alpha=1, beta=1 , delta=0.75 , gamma=1.5)   #η εντολή meshgrib μας βοηθάει έτσι ωστε τα X1,Y1,Dx1,Dy1 να έχουν την ίδια διάσταση γιατι αλλιώς θα είχαμε πρόβλημα
                                                                           #στο γράφημα


plt.figure()
plt.title('Direction fields')
Q = plt.quiver(X1, Y1, Dx1, Dy1,  pivot='mid', cmap=plt.cm.jet) #με αυτή την εντολή φτιάχνουμε τα βελάκια.Όπου τα X1,Y1 είναι οι συντεταγμένες για το που βρίσκονται τα βελάκια και τα Dx1,Dy1
                                                                #για το
I=np.linspace(1.0, 6.0, 10)         #εδώ γράφουμε πάλι οτι και στο παραπάνω γράφημα έτσι ώστε στο γράφημα που φτιάχνουμε εκτός απο τα βελάκια να έχει και το γράφημα καμπυλών στάθμης
for sheep in I :
    X0=[sheep, 1.0]
    Xf=integrate.odeint(derivative, X0, t, args=(alpha, beta , delta, gamma))
    plt.plot(Xf[:,0], Xf[:,1], "-", label = "$x_0 =$"+str(X0[0]))

plt.xlabel('Πλήθος προβάτων')
plt.ylabel('Πλήθος λύκων')
plt.legend()
plt.grid()
plt.xlim(0, 6)
plt.ylim(0, 6)
```



```python
#Σε περίπτωση συγκομιδής το γραφημα θα αλλάξει
μ1=0.5
μ2=1
def derivative2(X, t, alpha , beta, delta , gamma , μ1 , μ2): #Αντίστοιχα αλλάζουμε και την συνάρτηση που θα χρησιμοποιήσουμε προσθέτωντας μέσα σε αυτή τους όρους μ1 , μ2
        x,y = X
        dotx=x*((alpha-μ1) -beta*y)
        doty=y*(-(gamma+μ2) + delta*x)
        return np.array([dotx,doty])



plt.figure()
I=np.linspace(1.0, 6.0, 1) #εδω βάζουμε 1 στον δείκτη των δειγμάτων για να δούμε καθαρά την διαφορά στη μέση τιμή ανά περίοδο λύσης
for sheep in I :
    X0=[sheep, 1.0]
    Xf=integrate.odeint(derivative, X0, t, args=(alpha, beta , delta, gamma))
    plt.plot(Xf[:,0], Xf[:,1], "-", label = "$x_0 =$"+str(X0[0]))  #Αυτές τις εντολές τις χρησιμοποιήσαμε και παραπάνω για το διάγραμμα καμπυλών στάθμης
    Xz=integrate.odeint(derivative2, X0, t, args=(alpha, beta , delta, gamma, μ1 , μ2))
    plt.plot(Xz[:,0], Xz[:,1], "-", label = "$x_0 =$"+str(X0[0]))  #Εδώ προσθέτουμε στο γράφημα και την περίπτωση της συγκομιδής για να δούμε την διαφορά στη μέση τιμή ανά περίοδο αντίστοιχης
plt.xlabel('Πρόβατα')                          #λύσης , που στην περίπτωση συγκομιδής ειναι ((γ+μ2)/δ, (α-μ1)/β))
plt.ylabel('Λύκοι')
plt.title('Γράφημα Καμπυλών Στάθμης με και χωρις συγκομιδή')
```

✓ 0.2s