Rossler Model

عنوان واحد : شبیه سازی عددی

گردآورنده : سینا سلیمی

فهرست

راهنمای اجرای برنامه ها

تمامی کد ها ارائه شده همراه این گزارش به زبان python 3.8 است . برای رسم نمودارها در پایتون از کتابخانه ای به اسم matplotlib استفاده شده است که به راحتی با خطفرمان سیستم عامل یا IDE می توان این پکیج را به پکیج ها پایتون اضافه کرد در ادامه روش اضافه کردن این پکیج به کمک خط فرمان سیستم‌عامل های ویندوز و لینوکس به اختصار توضیح داده شده است:

در linux :

وارد ترمینال شده و دستور "pip3.8 install matplotlib" را وارد کرده , پس از اتمام کار می‌توانید با دستور "python3.8 ProgramName.py" , اقدام به اجرای برنامه کنید.

در windows :

در محل نصب پایتون (پیشفرض "C:\Python\Scripts\") خط فرمان ویندوز را باز کرده ("cmd") و دستور "pip3.8.exe install matplotlib" را وارد کرده , پس از اتمام کار به محل فایل برنامه بروید و در خطفرمان با وارد کردن دستور "python3.8 ProgramName.py" اقدام به اجرای برنامه کنید.

** بجای ProgramName نام برنامه پایتون را وارد کنید

** حتما به ورژن پایتون خود دقت کنید ← 3.8

مقدمه

مدل راسلر , یک الگو برای مدل کردن سیستم های غیر خطی و هرج و مرج (chaotic) است . Rössler attractor رفتاری مشابه مدل لورنز دارد با این تفاوت تجزیه و تحلیل راحت تری دارد.

معادلات سیستم راسلر به صورت زیر تعریف میشود :

$$\frac{dx}{dt} = -y - z$$

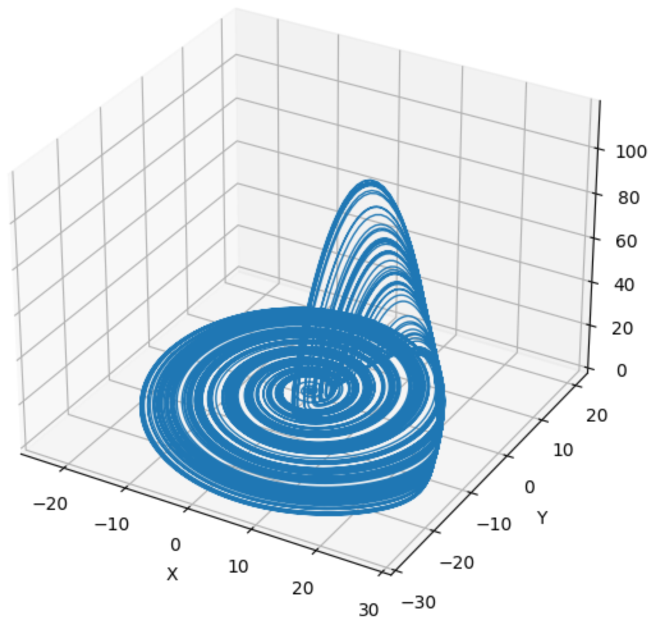
$$\frac{dy}{dt} = x + ay$$

$$\frac{dz}{dt} = b + z\left(x - c\right)$$

فضای فاز

در زیر 2 نمونه از فضای فاز 3 بعدی مدل راسلر آورده شده است :

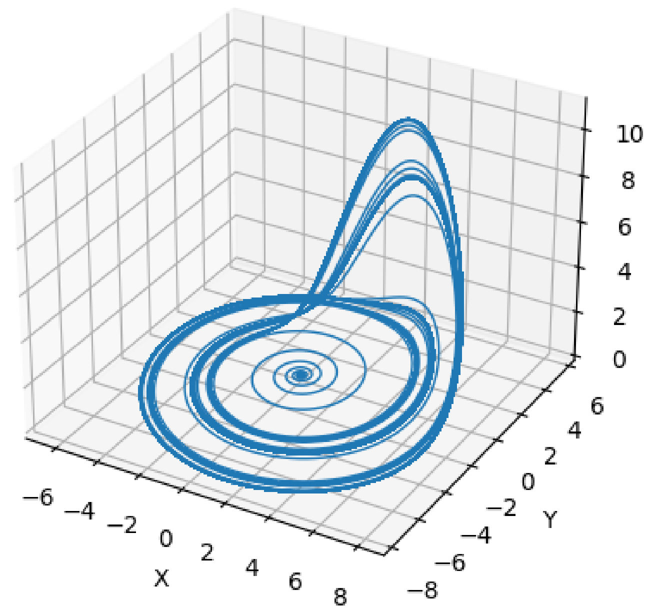
$$X_0 = Y_0 = Z_0 = 0$$



در تناوب زمانی مداوم و همراه با آشوب

a = b = 0.2

c = 15



در تناوب زمانی مداوم و بدون آشوب

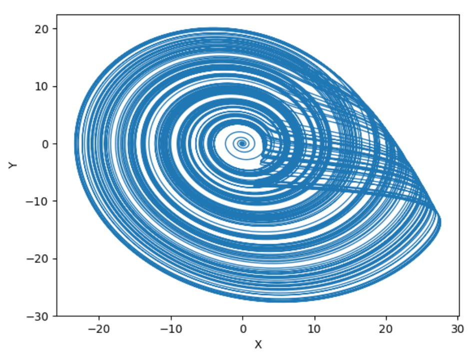
a = b = 0.2

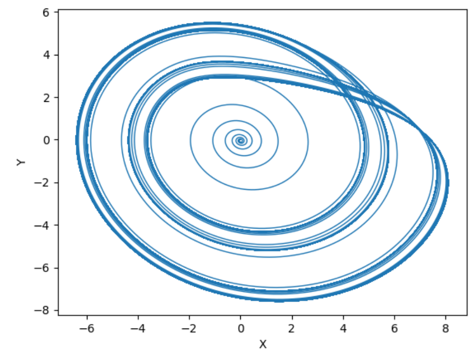
c = 5.7

در ادامه کد برنامه رسم فضای فاز 3 بعدی مدل راسلر آورده شده است (مربوط به تناوب زمانی غیر آشوب) .

```python
def rungeKutta(n: int, dt):  # calculate loop
   X = [0]
   Y = [0]
   Z = [0]

   for i in range(n):
       # runge kutta
       Xprime = X[i] + ((-Y[i] - Z[i]) * dt * 0.5)
       Yprime = Y[i] + ((X[i] + (a * Y[i])) * dt * 0.5)
       Zprime = Z[i] + (b + (Z[i] * (X[i] - c))) * dt * 0.5

       Y.insert(i + 1, Y[i] + ((Xprime + (a * Yprime)) * dt))
       X.insert(i + 1, X[i] + ((-Yprime - Zprime) * dt))
       Z.insert(i + 1, Z[i] + (b + (Zprime * (Xprime - c))) * dt)

   return X, Y, Z


def createPlot(X, Y, Z):  # create plot by matplotlib package
   plt.figure().gca(projection='3d')
   plt.xlabel("X")
   plt.ylabel("Y")
   plt.plot(X, Y, Z, linewidth=1)

   plt.show()


def saveLists(X, Y, Z): # save lists to file
   fileName = input("Please enter filename like this -> name.format ")

   file = open(fileName, "w")

   for i in range(len(X)):
       file.write(str(Z[i]) + "," + str(Y[i]) + "," + str(X[i]) + "\n")


if __name__ == '__main__': # start program here

   # def constants
   a = 0.2
   b = 0.2
   c = 5.7

   # get time values from user
   T = float(input("Please enter total time (s):\n"))
   dT = float(input("Please enter time step (s):\n"))

   n = int(T / dT)  # find loop range

   X, Y, Z = rungeKutta(n, dT)  # start calculate

   try:  # try to create plot by matplotlib
       import matplotlib.pyplot as plt
       createPlot(X, Y, Z)
   except ImportError as e:  # if can't find matplotlib run this
       saveLists(X, Y, Z)
```

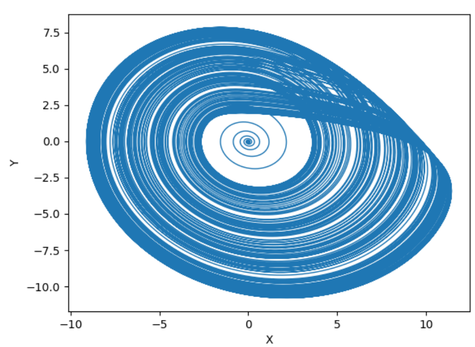
در ادامه تمام فضای فاز ها به صورت 2 بعدی در صفحه‌ی XOY رسم می‌شود (تصویر 3 بعد روی صفحه‌ی XOY) . نمودار ها زیر مربوط به تصویر شده‌ی نمودار های 3 بعدی بالا هستند (تغییرات صورت گرفته در کد برنامه فقط به این صورت است که دیگر مقادیر Z رسم نمیشود) :



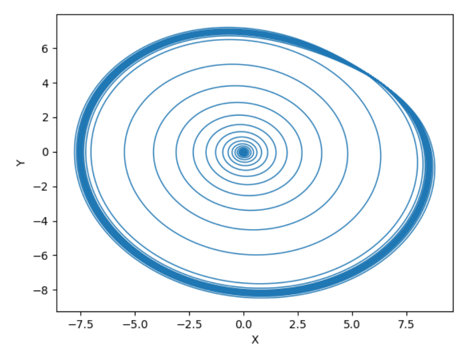


در این نمودار ها عامل آشوب پارامتر ثابت c است . میبینیم که با تغییر مقدار از 4 به 15 سیستم از نوسان با 2 فرکانس به آشوب رسید . اما آنطور که از معادلات مدل راسلر پیدا است غیر از پارامتر ثابت c دو پارامتر ثابت دیگر a و b وجود دارد . (خطوط مارپیچی که در مرکز دیده میشود , تلاش سیستم برای گذار از مقادیر اولیه به fixed points است)

در دو نمودار زیر c=5.7 و b=0.2 است و پارامتر a متغیر . میبینیم که با تغییر مقدار a از 0.1 به 0.2 , سیستم از نوسان با یک فرکانس , به آشوب رسید :

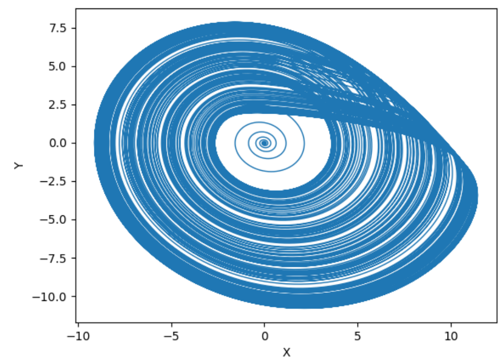


a = 0.2

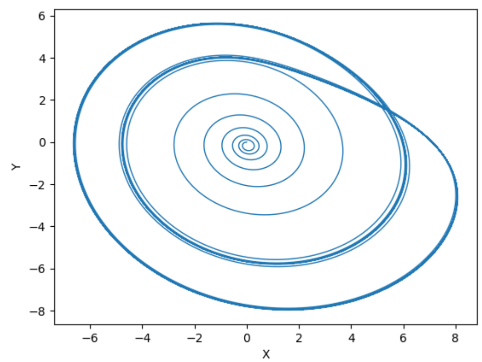


a = 0.1

همچنین در دو نمودار زیر c=5.7 و a=0.2 است و پارامتر b متغیر . میبینیم که با تغییر مقدار b از 0.2 به 1 , سیستم از آشوبی که در دو نمودار قبلی به وجود آمده بود , به نوسان با دو فرکانس رسید :



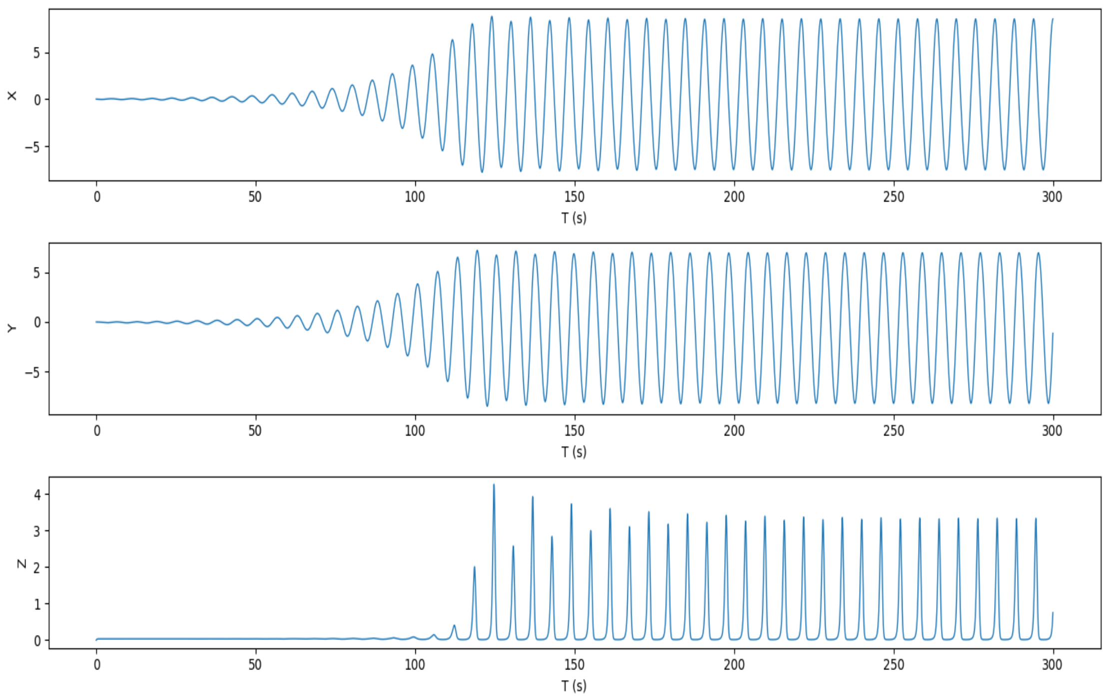
b = 0.2

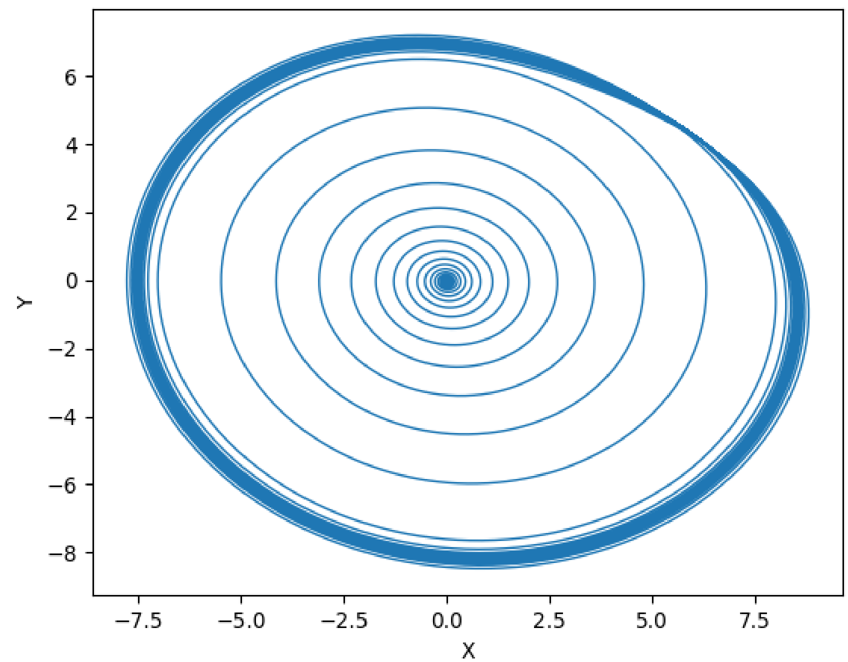


b = 1

** با رسم شش نمودار بالا دیدیم که مدل راسلر که دارای سه پارامتر ثابت است , می‌تواند با تغییر در هر یک از این پارامتر ها به آشوب برسد . تفاوت در این سه پارامتر در به آشوب رساندن این این است که با افزایش دو پارامتر a , c تعداد فرکانس های سیستم افزایش می‌یابد و به آشوب میرسد اما در مورد پارامتر b , افزایش مقدار باعث کاهش تعداد فرکانس ها سیستم میشود .

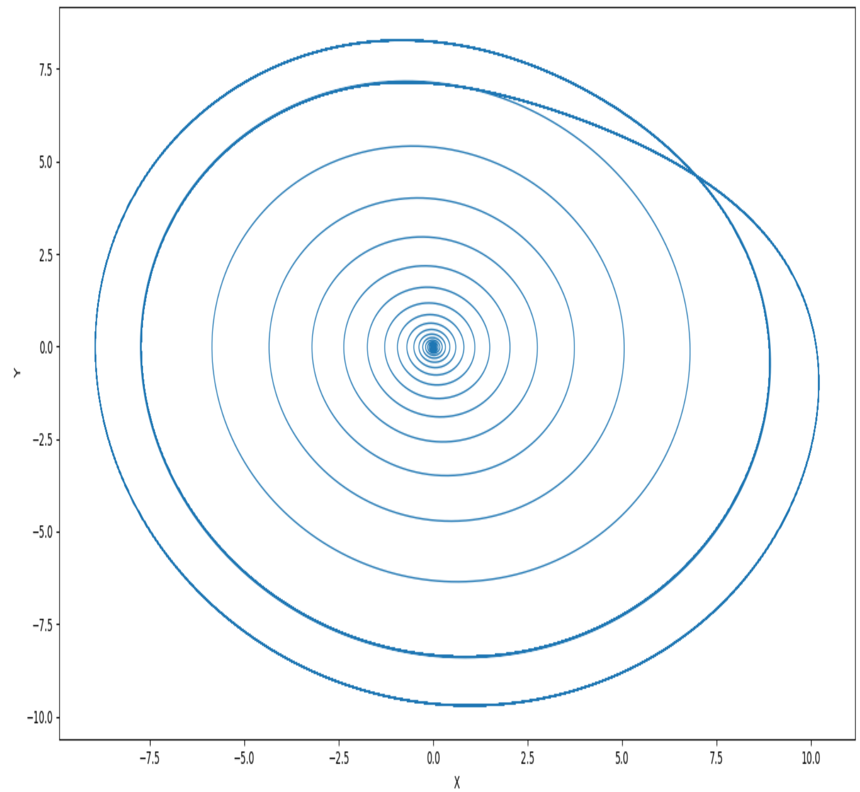
نوسانات زمانی

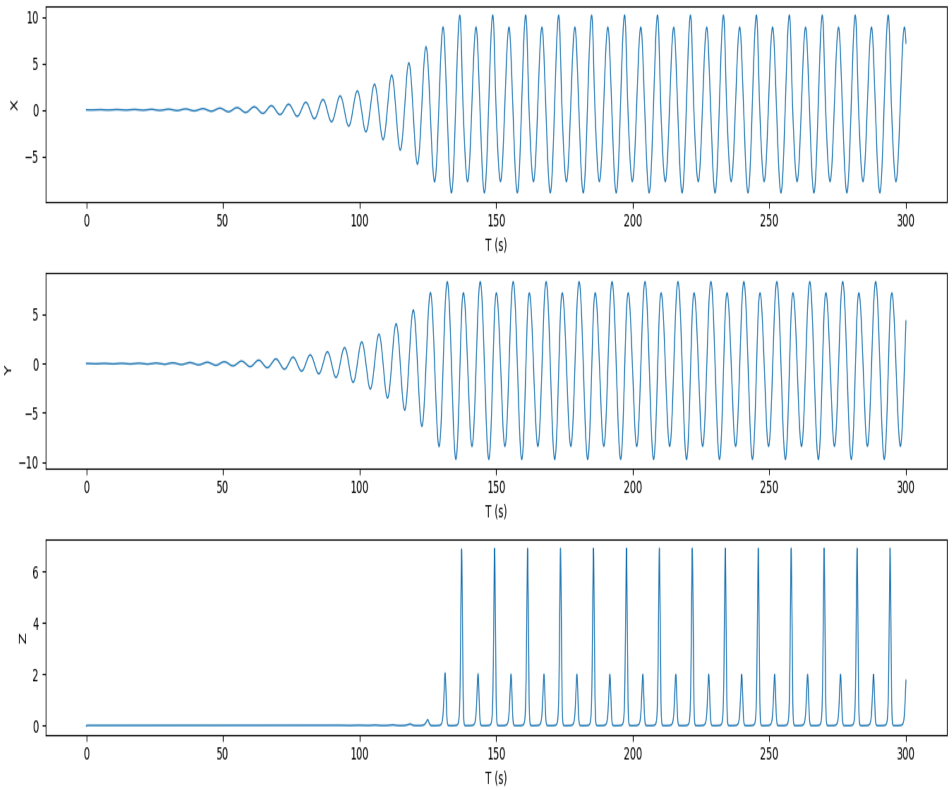




این نمودار ها برای مقادیر $a=0.1$, $b=0.2$, $c=5.7$ رسم شده است . همان طور که گفته شده بود در این مقادیر از پارامتر های ثابت , سیستم با یک فرکانس نوسان میکند

حال نمودار زمانی و فضای فاز را برای مقادیر a=0.1 , b=0.1 , c=6 رسم میکنیم :





در این مقادیر سیستم در دو فرکانس نوسان می‌کند که می‌توان به راحتی از روی نمودار زمانی و فضای فاز تشخیص داد.

در فضای فاز , دو حلقه پررنگ اخر(پررنگ تر به این معنی که سیستم از این مسیرها بیشتر عبور کرده است) .

در نمودارهای زمانی , می‌توان دید که سیستم در هر یک از نمودارها دارای دو دامنه متفاوت است که یک درمیان تکرار میشوند.

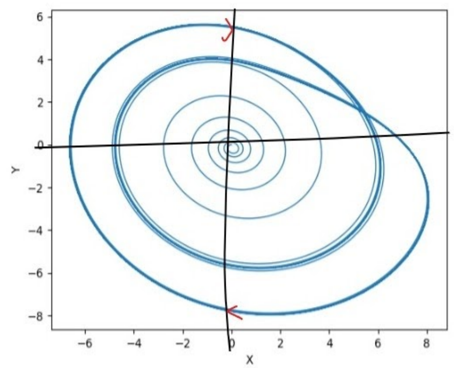
در ادامه کد مربوط به شبیه سازی زمانی مدل راسلر آمده است (مربوط به سیستم دو فرکانسی) .

```python
def rungeKutta(n: int, dt):  # calculate loop
    X = [0]
    Y = [0]
    Z = [0]
    T = [0]

    for i in range(n):
        # runge kutta
        Xprime = X[i] + ((-Y[i] - Z[i]) * dt * 0.5)
        Yprime = Y[i] + ((X[i] + (a * Y[i])) * dt * 0.5)
        Zprime = Z[i] + (b + (Z[i] * (X[i] - c))) * dt * 0.5

        Y.insert(i + 1, Y[i] + ((Xprime + (a * Yprime)) * dt))
        X.insert(i + 1, X[i] + ((-Yprime - Zprime) * dt))
        Z.insert(i + 1, Z[i] + (b + (Zprime * (Xprime - c))) * dt)
        T.insert(i + 1, T[i] + dT)
    return X, Y, Z, T


def createPlot(X, Y, Z, T):  # create plot by matplotlib package
    # plot X
    plt.subplot(3, 1, 1)
    plt.xlabel("T (s)")
    plt.ylabel("X")
    plt.plot(T, X, linewidth=1)
    # plot Y
    plt.subplot(3, 1, 2)
    plt.xlabel("T (s)")
    plt.ylabel("Y")
    plt.plot(T, Y, linewidth=1)
    # plot Z
    plt.subplot(3, 1, 3)
    plt.xlabel("T (s)")
    plt.ylabel("Z")
    plt.plot(T, Z, linewidth=1)
    plt.show()


def saveLists(X, Y, Z, T):  # save lists to file
    fileName = input("Please enter filename like this -> name.format ")
    file = open(fileName, "w")
    for i in range(len(X)):
        file.write(str(Z[i]) + "," + str(Y[i]) + "," + str(X[i]) + "," + str(T) + "\n")


if __name__ == '__main__':
    # def constants
    a = 0.1
    b = 0.1
    c = 6

    # get time values from user
    totalT = float(input("Please enter total time (s):\n"))
    dT = float(input("Please enter time step (s):\n"))

    n = int(totalT / dT)  # find loop range

    X, Y, Z, T = rungeKutta(n, dT)  # start calculate

    try:  # try to create plot by matplotlib
        import matplotlib.pyplot as plt
        createPlot(X, Y, Z, T)

    except ImportError as e:  # if can't find matplotlib run this
        saveLists(X, Y, Z, T)
```

مقطع پوانکاره

در بخش های قبل مقادیری برای پارامتر های ثابت مدل راسلر پیدا کردیم , اکنون می‌خواهیم برای این مقادیر poincare section را رسم کنیم . شرط فیلتر کردن متغیر های سیستم به شرح زیر است :

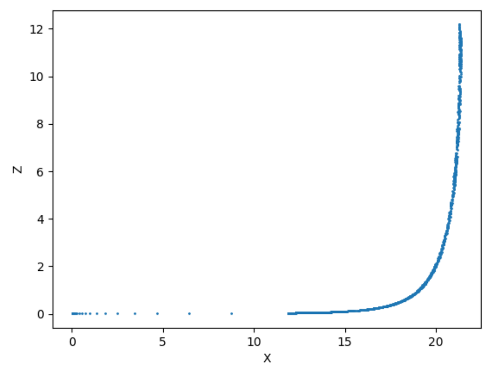
شرط گذر از فیلتر , X=0 در هر step از حلقه محاسباتی است ,برای اینکه مقادیر ورودی هر step گسسته است , ممکن است که دقیقا روی X = 0 مقداری را ضبط نکند بنابراین شرط قبلی با شرط " تغییر علامت مقدار X "جایگزین می‌شود , اما آنطور که ار نمودار زیر پیدا است یک مسیر مشخص دوبار از محور X=0 عبور میکند (به عبارتی دوبار تغییر علامت می‌دهد) بنابراین بار دیگر شرط قبلی با شرط " تغییر علامت X , از + به - " جایگزین میشود (مسیرهای هم جهت با فلش قرمز رنگ پایینی).

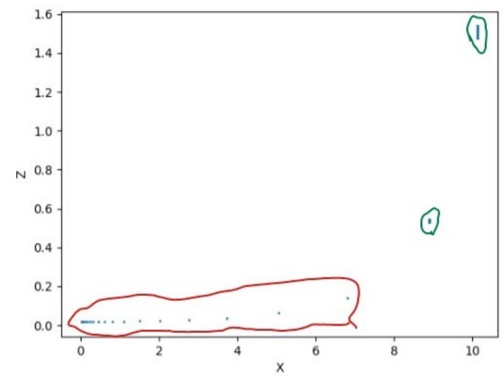


در این شرط می توان به جای X از Y هم استفاده کر و همچنین جهت دیگری را انتخاب کرد (- به +)

در کد معادل است با :

```
if ( Y[step+1]>0  and  Y[step] <0)
```

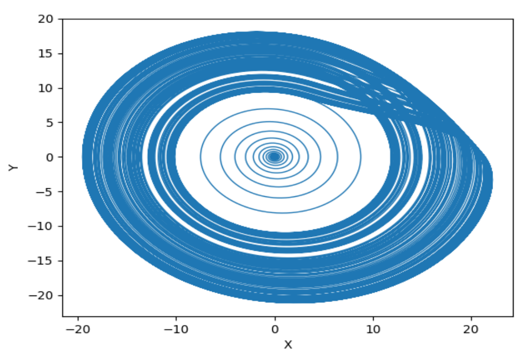


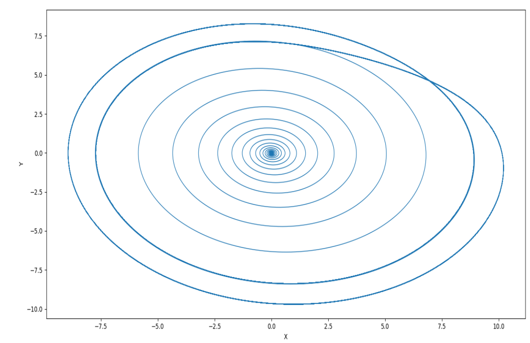


* نمودار سمت چپ برای $a = 0.1$, $b = 0.1$, $c = 14$ است در زمانی که سیستم در آشوب است و مسیر در محورِ Y از - به + تغییر علامت می‌دهد.

* نمودار سمت راست برای $a = 0.1$, $b = 0.1$, $c = 6$ است در زمانی که سیستم در آشوب نیست و مسیر در محور Y از - به + تغییر علامت می‌دهد ; محدوده مشخص شده با رنگ قرمز متشکل از مسیر هایی است که سیستم برای رسیدن به fixed points (محدوده ها سبز) طی کرده است . محدوده مشخص شده با رنگ سبز fixed points سیستم در مقدار گفته شده در a,b,c هستند که سیستم از این دو محدود به صورت یکی درمیان عبور میکند (به عبارت دیگر سیستم دارای دو فرکانس است) .

در زیر فضای فاز مربوط به نمودار ها بالا به ترتیب آورده شده است :





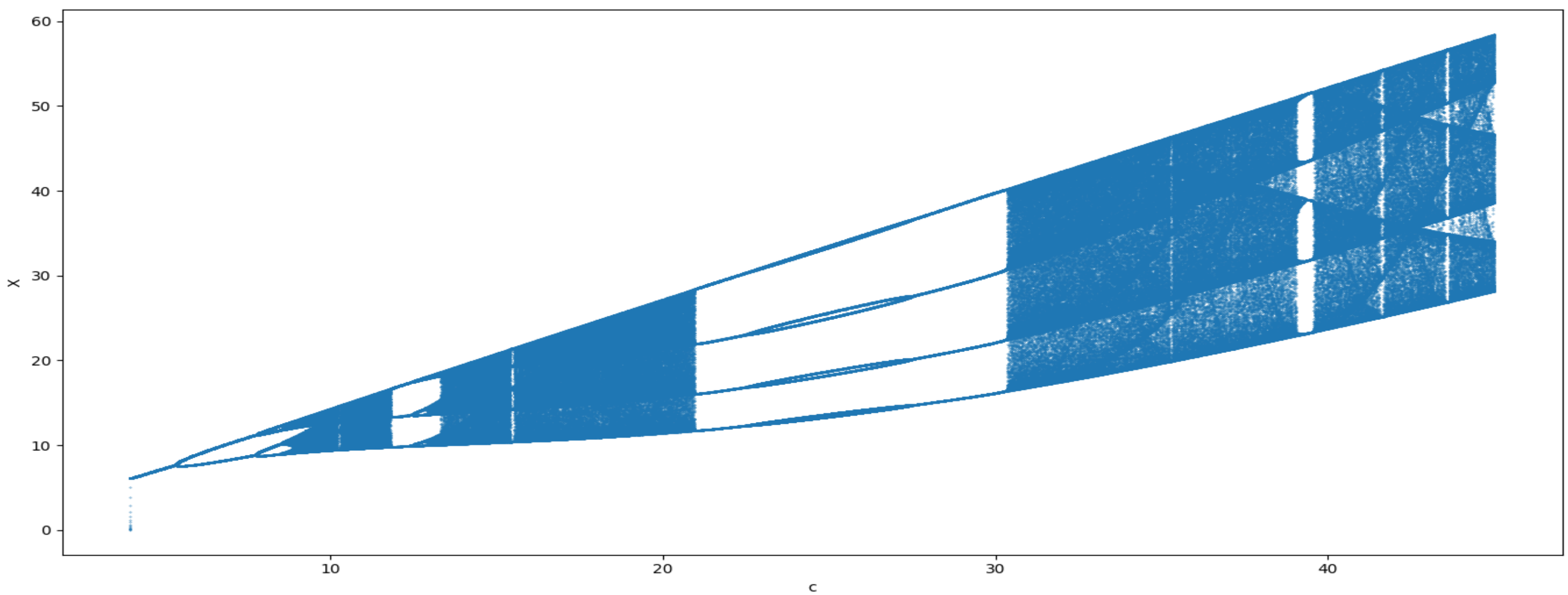
در ادامه کد برنامه شبیه سازی مقطع پوانکاره مدل راسلر آورده شده است (مربوط به سیستم دارای آشوب):

```python
def rungeKutta(n: int, dt):  # calculate loop
   X = 0
   Y = 0
   Z = 0
   PZ = []
   PX = []

   for i in range(n):
       # runge kutta
       Xprime = X + ((-Y - Z) * dt * 0.5)
       Yprime = Y + ((X + (a * Y)) * dt * 0.5)
       Zprime = Z + (b + (Z * (X - c))) * dt * 0.5

       newY = Y + ((Xprime + (a * Yprime)) * dt)
       newX = X + ((-Yprime - Zprime) * dt)
       newZ = Z + (b + (Zprime * (Xprime - c))) * dt

       if newY > 0 and Y < 0:
           PX.append(abs(newX))
           PZ.append(abs(newZ))

       Y = newY
       X = newX
       Z = newZ
   return PX, PZ


def createPlot(X, Z):  # create plot by matplotlib package
   plt.xlabel("X")
   plt.ylabel("Z")
   plt.scatter(X, Z, s=1)
   plt.show()

def saveLists(X, Z):  # save lists to file
   fileName = input("Please enter filename like this -> name.format ")
   file = open(fileName, "w")
   for i in range(len(X)):
       file.write(str(Z[i]) + "," + str(X[i]) + "\n")


if __name__ == '__main__':
   # def constants
   a = 0.1
   b = 0.1
   c = 14

   # get time values from user
   totalT = float(input("Please enter total time (s):\n"))
   dT = float(input("Please enter time step (s):\n"))

   n = int(totalT / dT)  # find loop range

   X, Z = rungeKutta(n, dT)  # start calculate

   try:  # try to create plot by matplotlib

       import matplotlib.pyplot as plt

       createPlot(X, Z)

   except ImportError as e:  # if can't find matplotlib run this

       saveLists(X, Z)
```

نمودار دوشاخگی

برای رسم نمودار های دوشاخگی , به علت وجود چند پارامتر آشوب ناک , سیستم دارای سه نمودار دوشاخگی است و برای رسم هر یک از این نمودار ها پارامتر مربوط به عنوان متغیر تعریف می‌شود دو پارامتر دیگر ثابت می‌مانند (برای نمودار دوشاخگی از شرط poincare section استفاده میکنیم) .

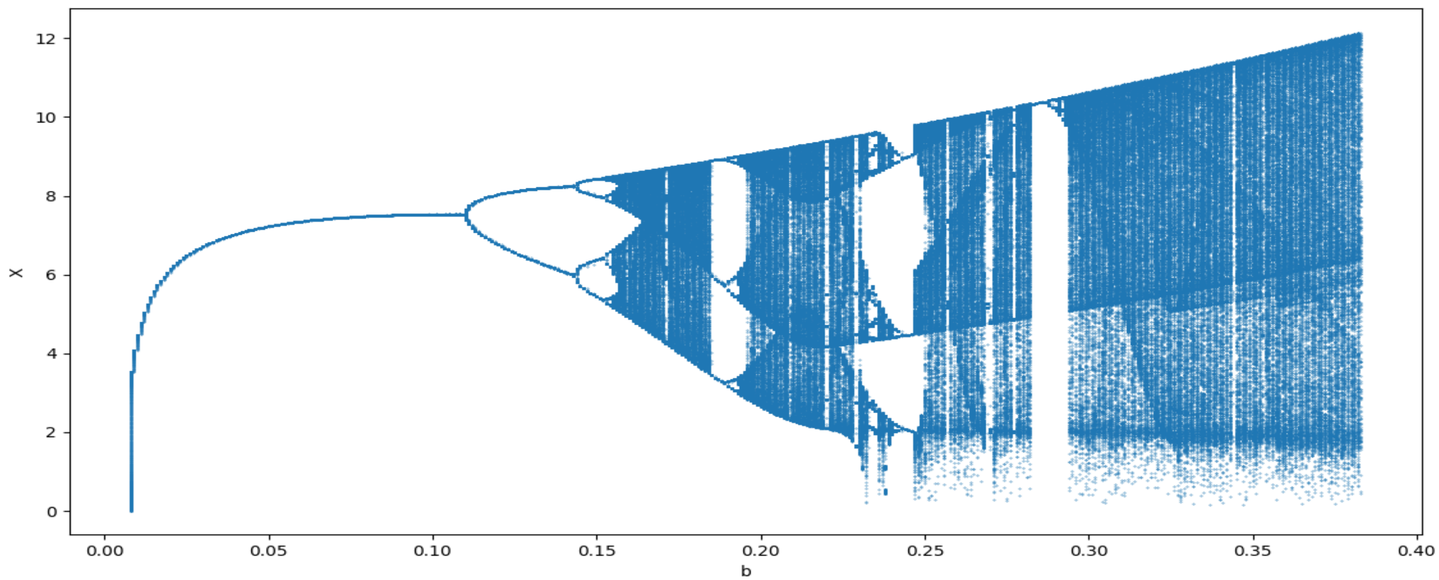
برای شروع نمودار دوشاخگی پارامتر c را رسم میکنیم (a = 0.1 , b = 0.1) :



همان طور که از نمودار پیداست و در بخش های اولیه گفته شد,افزایش مقدار c باعث افزایش تعداد فرکانس ها سیستم می‌شود (در بعضی مقاطع به صورت period doubling برای مثال در c=5 سیستم دارای یک فرکانس است و افزایش به مقدار c=6 سیستم دارای دو فرکانس است و با افزایش دوباره به مقدار c=8.5 سیستم دارای چهار فرکانس است)

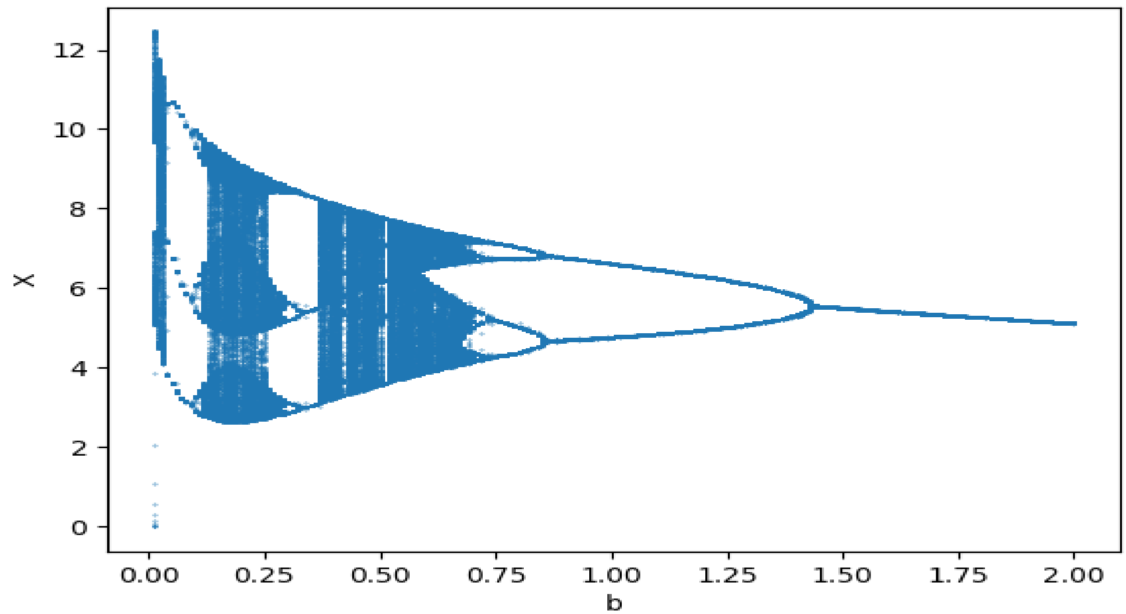
به این صورت دو برابر شدن تعداد فرکانس های سیستم دیده میشود.اما در تمام مقادیر اینگونه نیست برای مثال در $c = 7.5$ تعداد فرکانس ها نصف شده است.محدوده های توپر به معنی آشوب در سیستم است برای مثال در محدوده c=[14,20]c سیستم در آشوب است.

نمودار دوشاخگی پارامتر a با ($c = 5.7$, $b = 0.2$) :



در این نمودار همانند c همانطور که انتظار می‌رفت افزایش a باعث افزایش تعداد فرکانس های سیستم به صورت period doubling میشود و به ترتیب در $a = 0.1$ و $a = 0.2$ سیستم دارای یک فرکانس و چندین فرکانس (آشوب) است .

نمودار دوشاخگی پارامتر b با (c = 5.7 , a = 0.2):



و دوباره همانطور که در بخش های ابتدایی پیش بینی شد , افزایش b باعث کاهش تعداد فرکانس های سیستم می‌شود (نصف شدن تعداد فرکانس ها). با افزایش مکرر b سیستم در انتها به یک فرکانس می‌رسد.

در ادامه کد شبیه سازی bifurcation diagram پارامتر ثابت a مدل راسلر آورده شده است (برای رسم نمودار های c و b همانند a عمل میشود):

```python
# def constant
a = 0.2
b = 0.2
c = 5.7

sa = 0
ea = 0.4
da = 0.001
na = int((ea - sa) / da)

# time
T = 10000
dT = 0.01
n = int(T / dT)

X = 0
Y = 0
Z = 0

# poincare section lists
PX = []
Pa = []

for j in range(na):  # loop of "a" values
   printProgressBar(j + 1, na, prefix='Progress:', suffix='Complete', length=50)
   sa = sa + da

   for i in range(n):  # time loop
       Xprime = X + ((-Y - Z) * dT * 0.5)
       Yprime = Y + ((X + (sa * Y)) * dT * 0.5)
       Zprime = Z + (b + (Z * (X - c))) * dT * 0.5

       newY = Y + ((Xprime + (sa * Yprime)) * dT)
       newX = X + ((-Yprime - Zprime) * dT)
       newZ = Z + (b + (Zprime * (Xprime - c))) * dT

       if newY < 0 and Y > 0:
           Pa.append(sa)
           PX.append(abs(newX))

       X = newX
       Y = newY
       Z = newZ


try:
   import matplotlib.pyplot as plt
   plt.xlabel("a")
   plt.ylabel("X")
   plt.scatter(Pa, PX, s=0.1)

   plt.show()

except ImportError as e:
   fileName = input("Please enter filename like this -> name.format ")
   file = open(fileName, "w")
   for i in range(len(X)):
       file.write(str(PX[i]) + "," + str(Pa[i]) + "\n")
```