Hanyijie(4)

June 23, 2024

1

1.1

For the equations:

$$\begin{cases} \frac{dx}{dt} = \sigma(y-x) \\ \frac{dy}{dt} = -xz + rx - y \\ \frac{dz}{dt} = xy - bz \end{cases}$$

1. $\sigma = 10, b = 3/8, r = 25$ z-x 2. $\sigma = 10, b = 3/8$ r z-x 3. r 4. r

. ([0.5,0.5,0.5]) np.linspace(0,1000,50000)

. r 50,683,684,

. x=a;y=b;z=c x=a;y=b;z=c

r=50; r=683

. r 0-683, r FFT FFT X

r x

r-F

1. r=684
2. 5min ## ### #### 1 ##### a.

[4]:
```
# Labrary
import numpy as np
import matplotlib.pyplot as plt
# Graph
## Library
from matplotlib.font_manager import FontProperties
## Set the font globally
font_path = '/usr/share/fonts/opentype/noto/NotoSansCJK-Regular.ttc'
plt.rcParams['font.family'] = 'sans-serif'
```

```
plt.rcParams['font.sans-serif'] = [FontProperties(fname=font_path).get_name()]
plt.rcParams['axes.unicode_minus'] = False  # Ensure the minus sign is␣
↪displayed correctly

# Graph in pack
def␣
↪graph(ax,x,y,x_label="x0",y_label="y0",title="Title0",legend='legend0',loc='upper␣
↪left',\
          color="r",linestyle='-',linewidth=1,marker='o'):
    ## Parameter
    ### rangle
    #x_range=np.array([np.min(x)-0.1*(np.max(x)-np.min(x)),np.max(x)+0.1*(np.
↪max(x)-np.min(x))])
    #y_range=np.array([np.min(y)-0.1*(np.max(y)-np.min(y)),np.max(y)+0.1*(np.
↪max(y)-np.min(y))])

    ## figure
    ## plot
    ax.
↪plot(x,y,marker=marker,color=color,linestyle=linestyle,linewidth=linewidth,label=legend)
    ## title
    ax.set_title(title,fontsize=20,weight='bold',x=0.5,y=1)
    ## legend
    ax.legend(loc=loc,prop = {'size':8})
    ## Axis
    ### label
    ax.set_xlabel(x_label,fontsize=14,labelpad=0)
    ax.set_ylabel(y_label,fontsize=14)
    ### limit
    #ax.set_xlim(x_range)
    #ax.set_ylim(y_range)
    ### tick
    #axn.set_xticks(np.linspace(0,10,4)) #x axis scale. range: 0-10 points: 4
    #axn.set_xticklabels(['you','are','so','nice'])
    ax.tick_params(axis='both',direction='in',color='black',length=5,width=1)␣
↪#axis='x'or'y'or'both'

    return 0
```

b. 1. Fourth order Runge-Kutta method diff_right_func diff_right_func scipy.solve_ivp

```
def diff_right_func(dvariable_value_array,t):#([y0,y1,y2...],x)
    x,y,z = dvariable_value_array
    diff_group_value = np.array([sigma*(y-x),-x*z+r*x-y,x*y-b*z]) #the sequence: dy_0/dx=...;dy_
    return diff_group_value
```

[5]:
```python
# Library
import numpy as np
import matplotlib.pyplot as plt

def Integral_runge_kutta4_func(dvariable_value_array, x, dx, diff_right_func):
    k1_ndarray = dx * diff_right_func(dvariable_value_array, x) #Format :ndarray
    k2_ndarray = dx * diff_right_func(dvariable_value_array + 0.5 * k1_ndarray, x + 0.5 * dx)
    k3_ndarray = dx * diff_right_func(dvariable_value_array + 0.5 * k2_ndarray, x + 0.5 * dx)
    k4_ndarray = dx * diff_right_func(dvariable_value_array + k3_ndarray, x + dx)
    return dvariable_value_array + (k1_ndarray + 2 * k2_ndarray+ 2 * k3_ndarray + k4_ndarray) / 6.
def self_integral(xdata_array,yinitial_array,Integral_func,diff_right_func):
    y=yinitial_array
    ydata=np.zeros((np.size(yinitial_array),np.size(xdata_array)))
    ydata[:,0]=y
    for i in range(np.size(xdata_array)-1):
        #intergal
        x=xdata_array[i]
        dx=xdata_array[i+1]-xdata_array[i]
        y=Integral_func(y, x, dx, diff_right_func)
        #store
        ydata[:,i+1]=y
    return ydata
```

2.

[6]:
```python
def parameter_vary(r=25,yinitial_array=np.array([0.1,0.1,0.1]),t_array=np.linspace(0,100,1000),sigma=10,b=3/8):
    ## equation groups
    def diff_right_func(dvariable_value_array,t):#([y0,y1,y2...],x)
        x,y,z = dvariable_value_array
        diff_group_value = np.array([sigma*(y-x),-x*z+r*x-y,x*y-b*z]) #the sequence: dy_0/dx=...;dy_1/dx=...
        return diff_group_value
    sol=self_integral(t_array,yinitial_array,Integral_runge_kutta4_func,diff_right_func)
    return sol
```

3.

[7]:
```python
#
def half_down(endpoint_y):
    if endpoint_y[0]*endpoint_y[1]<=0 and endpoint_y[0]>0:
        return 1
    else:
```

```
        return 0
```

4.

```
[8]: #   [x;y;z]
def Poincare_plane(sol_cut,median_value):
    informationx=np.array([np.array([0]),np.array([0])])
    informationy=np.array([np.array([0]),np.array([0])])
    informationz=np.array([np.array([0]),np.array([0])])

    for j in range((np.shape(sol_cut))[1]-1):
        if half_down([sol_cut[0,j],sol_cut[0,j+1]]-median_value[0]):#x=0    y z
            informationx=np.concatenate((informationx,np.array([np.
 ↪array([sol_cut[1,j]]),np.array([sol_cut[2,j]])])),axis=1)
    for j in range((np.shape(sol_cut))[1]-1):
        if half_down([sol_cut[1,j],sol_cut[1,j+1]]-median_value[1]):#x=0    x z
            informationy=np.concatenate((informationy,np.array([np.
 ↪array([sol_cut[0,j]]),np.array([sol_cut[2,j]])])),axis=1)
    for j in range((np.shape(sol_cut))[1]-1):
        if half_down([sol_cut[2,j],sol_cut[2,j+1]]-median_value[2]):#x=0    x y
            informationz=np.concatenate((informationz,np.array([np.
 ↪array([sol_cut[0,j]]),np.array([sol_cut[1,j]])])),axis=1)

    return informationx[:,1:-1],informationy[:,1:-1],informationz[:,1:-1]
```

5.FFT

```
[9]: # Library
import numpy as np
from scipy.fftpack import fft,ifft #core
import matplotlib.pyplot as plt
from matplotlib.pylab import mpl

def FFT(t_array_cut,y_vector):
    # Date
    ## parameter
    N=np.size(t_array_cut) #
    ##
    x=t_array_cut
    y=y_vector   #        200 400 600
    ## amplitude
    N_amplitude=np.arange(N)

    # FFT
    fft_y=fft(y)
    abs_y=np.abs(fft_y)                 #         (  )
    angle_y=np.arange(0,N)               #
    ##
```

```
    normalization_y=abs_y/N
    ##
    half_x = N_amplitude[range(int(N/2))]                             #
    normalization_half_y = normalization_y[range(int(N/2))]
    return half_x,normalization_half_y
```

6. FFT

[10]:
```
def FFT_parameter_vary(t_array,r=25,yinitial_array=np.array([0.5,0.5,0.5])):
    sol25=parameter_vary(r,yinitial_array=np.
 ↪array(yinitial_array),t_array=t_array)
    t_array_cut=t_array[int(1/3*(np.shape(t_array))[0]):-1]
    sol25_cut=sol25[:,int(1/3*(np.shape(sol25))[1]):-1]
    ##
    t_array_cut=t_array_cut[1:-1:5]
    sol25_cut=sol25_cut[:,1:-1:5]

    ## FFT
    half_x,normalization_half_y=FFT(t_array_cut,sol25[1,:])
    return t_array_cut,sol25_cut,half_x,normalization_half_y
```

7.

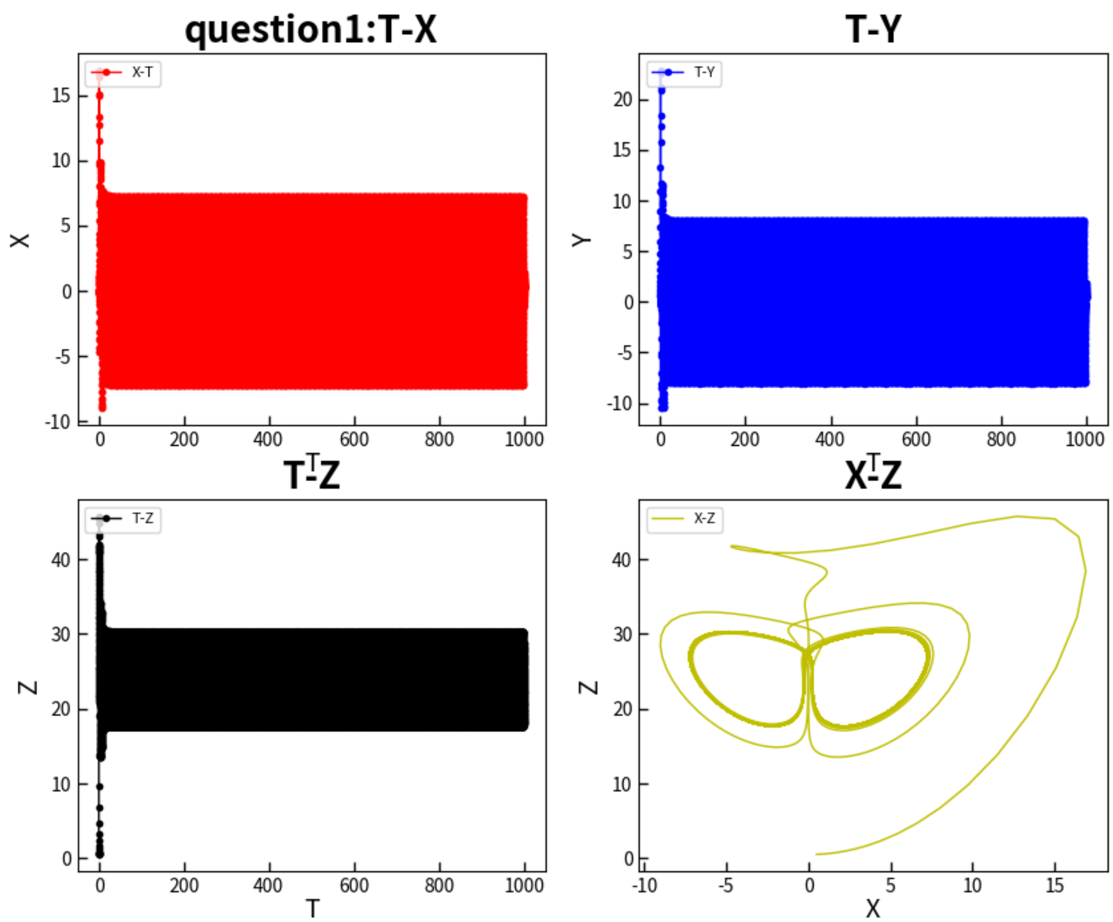
[11]:
```
# Konwn half_x0,normalization_half_y0
def periodicity(half_x1,normalization_half_y1):
    standard=10*(np.sort(normalization_half_y1,axis=0))[-int(1/80*np.
 ↪size(normalization_half_y1))]
    #     ,    0
    if not(all(normalization_half_y1<standard)):
        return half_x1[np.argmax(normalization_half_y1)]
    else:
        return 0
```

1.1.1

x,y,z 0.5,r 25

[12]:
```
# Data
## parameter
t_array=np.linspace(0,1000,50000)
yinitial_array=np.array([0.5,0.5,0.5])

# Conpution
sol=parameter_vary(r=25,yinitial_array=yinitial_array,t_array=t_array)
# Graph
# fig
ax=[0,0,0,0]
fig = plt.figure(figsize=(10,8))
```

```
ax[0] = fig.add_subplot(2,2,1)
ax[1] = fig.add_subplot(2,2,2)
ax[2]= fig.add_subplot(2,2,3)
ax[3]= fig.add_subplot(2,2,4)
# plot
graph(ax[0],t_array,sol[0,:],x_label="T",y_label="X",title="question1:
↪T-X",legend='X-T',color='r',linestyle='-',marker='.')
graph(ax[1],t_array,sol[1,:
↪],x_label="T",y_label="Y",title="T-Y",legend='T-Y',color='b',linestyle='-',marker='.
↪')
graph(ax[2],t_array,sol[2,:
↪],x_label="T",y_label="Z",title="T-Z",legend='T-Z',color='k',linestyle='-',marker='.
↪')
graph(ax[3],sol[0,:],sol[2,:
↪],x_label="X",y_label="Z",title="X-Z",legend='X-Z',color='y',linestyle='-',marker='')
# output
plt.show()
#plt.savefig('savefig_example.eps') #eps piture
```



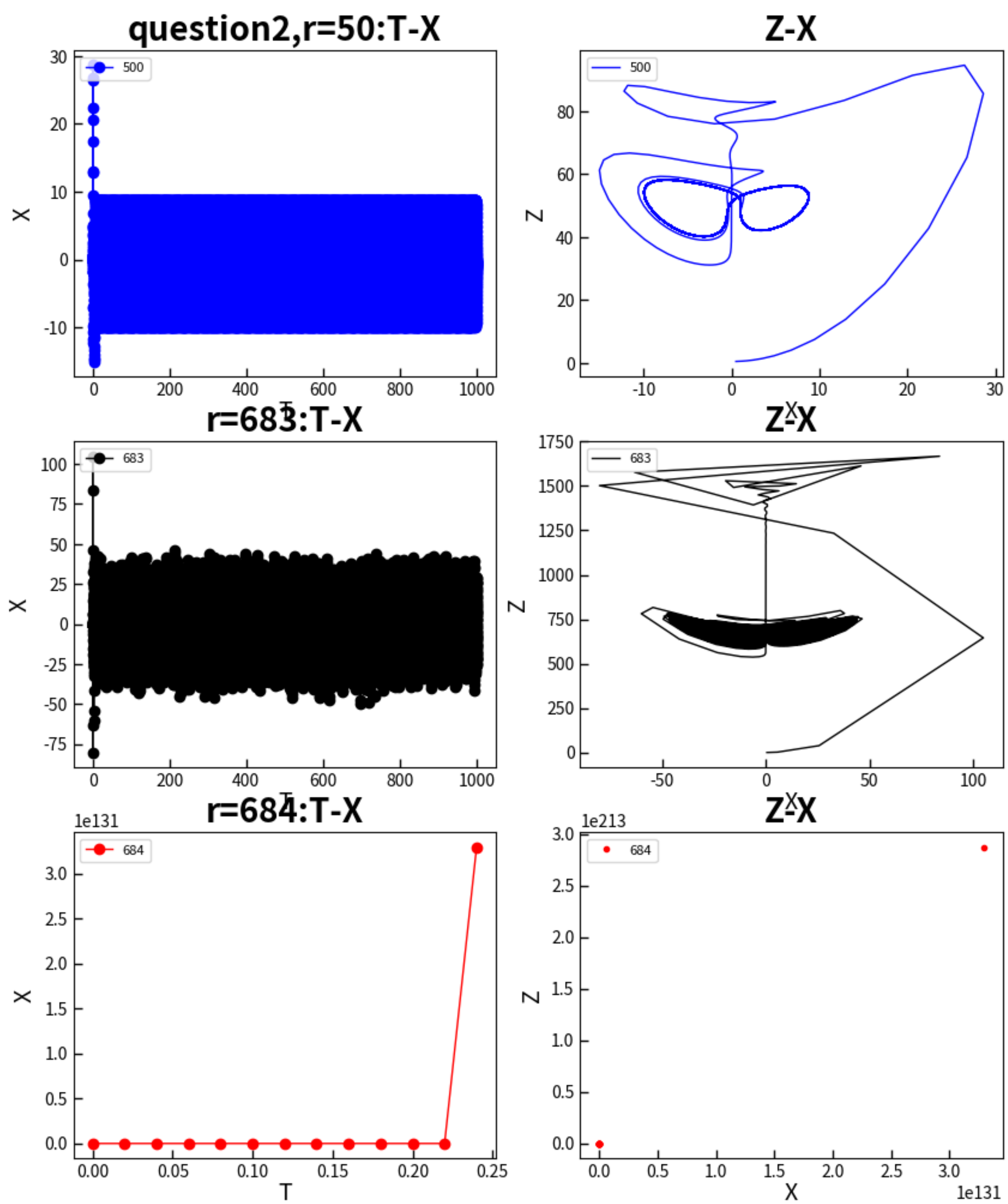
1.1.2

r 50 683 684 ( )

[13]:
```
sol50=parameter_vary(r=50,yinitial_array=np.
  ↪array(yinitial_array),t_array=t_array)
sol683=parameter_vary(r=683,yinitial_array=np.
  ↪array(yinitial_array),t_array=t_array)
sol684=parameter_vary(r=684,yinitial_array=np.
  ↪array(yinitial_array),t_array=t_array)

# Graph
# fig
ax=[0,0,0,0,0,0]
fig = plt.figure(figsize=(10,12))
ax[0] = fig.add_subplot(3,2,1)
ax[1] = fig.add_subplot(3,2,2)
ax[2] = fig.add_subplot(3,2,3)
ax[3] = fig.add_subplot(3,2,4)
ax[4] = fig.add_subplot(3,2,5)
ax[5] = fig.add_subplot(3,2,6)

# plot
graph(ax[4],t_array,sol684[0,:],x_label="T",y_label="X",title="r=684:
  ↪T-X",legend='684',color='r',linestyle='-')
graph(ax[5],sol684[0,:],sol684[2,:
  ↪],x_label="X",y_label="Z",title="Z-X",legend='684',color='r',linestyle='',marker='.
  ↪')

graph(ax[2],t_array,sol683[0,:],x_label="T",y_label="X",title="r=683:
  ↪T-X",legend='683',color='k',linestyle='-')
graph(ax[3],sol683[0,:],sol683[2,:
  ↪],x_label="X",y_label="Z",title="Z-X",legend='683',color='k',linestyle='-',marker='')

graph(ax[0],t_array,sol50[0,:],x_label="T",y_label="X",title="question2,r=50:
  ↪T-X",legend='500',color='b',linestyle='-')
graph(ax[1],sol50[0,:],sol50[2,:
  ↪],x_label="X",y_label="Z",title="Z-X",legend='500',color='b',linestyle='-',marker='')
# output
plt.show()
#plt.savefig('savefig_example.eps') #eps piture
```

```
/tmp/ipykernel_71154/887059201.py:5: RuntimeWarning: overflow encountered in
scalar multiply
  diff_group_value = np.array([sigma*(y-x),-x*z+r*x-y,x*y-b*z]) #the sequence:
dy_0/dx=…;dy_1/dx=…
/tmp/ipykernel_71154/887059201.py:5: RuntimeWarning: invalid value encountered
in scalar subtract
```

```
  diff_group_value = np.array([sigma*(y-x),-x*z+r*x-y,x*y-b*z]) #the sequence:
dy_0/dx=…;dy_1/dx=…
/tmp/ipykernel_71154/887059201.py:5: RuntimeWarning: invalid value encountered
in scalar add
  diff_group_value = np.array([sigma*(y-x),-x*z+r*x-y,x*y-b*z]) #the sequence:
dy_0/dx=…;dy_1/dx=…
/tmp/ipykernel_71154/4231725843.py:10: RuntimeWarning: invalid value encountered
in add
  return dvariable_value_array + (k1_ndarray + 2 * k2_ndarray+ 2 * k3_ndarray +
k4_ndarray) / 6.
```



1.1.3

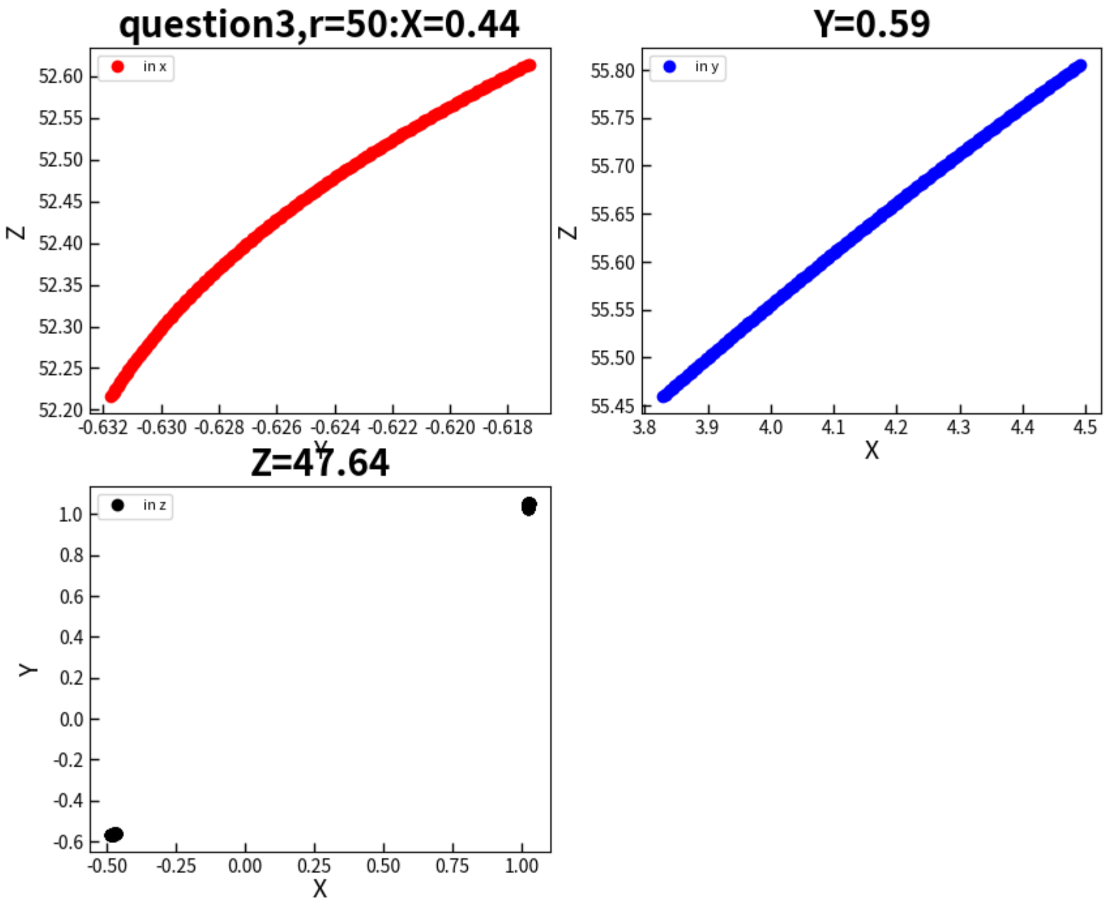
1. r=25 50
2. r=683
3. r=684

1. 2/3 2. z=c y=b,x=a

1 r=50: r=25

800 ±0.5 z=22.64

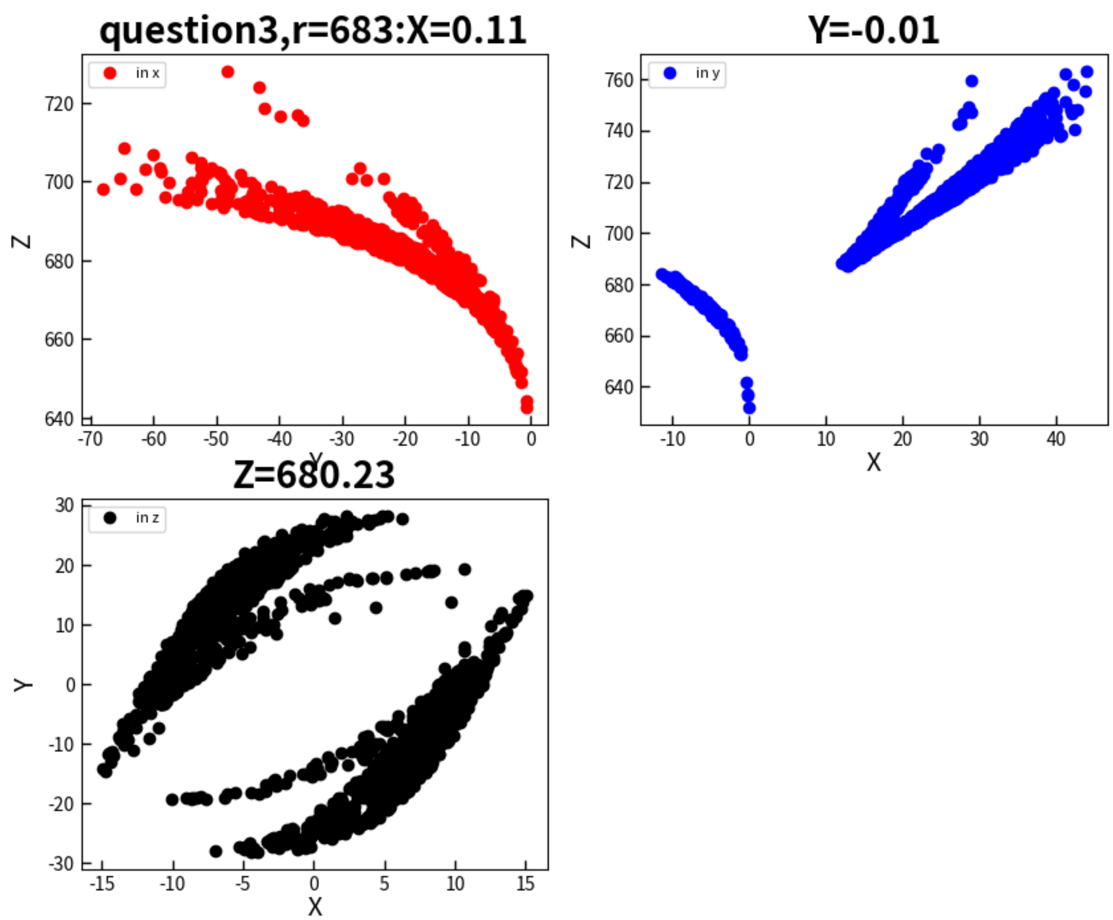
[14]:
```
# Data
##  2/3

t_array_cut=t_array[int(1/3*np.size(t_array)):-1]
sol_cut=sol50[:,int(1/3*(np.shape(sol))[1]):-1]
median_value=np.median(sol_cut,axis=1)
##
infox,infoy,infoz=Poincare_plane(sol_cut,median_value)

# Graph
# fig
ax=[0,0,0,0,0,0]
fig = plt.figure(figsize=(10,8))
ax[0] = fig.add_subplot(2,2,1)
ax[1] = fig.add_subplot(2,2,2)
ax[2] = fig.add_subplot(2,2,3)
# plot

title_1="question3:X={}".format(median_value[0])
graph(ax[0],infox[0,:],infox[1,:],x_label="Y",y_label="Z",title="question3,r=50:
 ↪X={:.2f}".format(median_value[0]),legend='in x',color='r',linestyle='')
graph(ax[1],infoy[0,:],infoy[1,:],x_label="X",y_label="Z",title="Y={:.2f}".
 ↪format(median_value[1]),legend='in y',color='b',linestyle='')
graph(ax[2],infoz[0,:],infoz[1,:],x_label="X",y_label="Y",title="Z={:.2f}".
 ↪format(median_value[2]),legend='in z',color='k',linestyle='')
# output
plt.show()
#plt.savefig('savefig_example.eps') #eps piture
```



2 r=683

[15]:
```
# Data
##  2/3

t_array_cut=t_array[int(1/3*np.size(t_array)):-1]
sol_cut=sol683[:,int(1/3*(np.shape(sol))[1]):-1]
median_value=np.median(sol_cut,axis=1)
##
infox,infoy,infoz=Poincare_plane(sol_cut,median_value)

# Graph
# fig
ax=[0,0,0,0,0,0]
fig = plt.figure(figsize=(10,8))
ax[0] = fig.add_subplot(2,2,1)
ax[1] = fig.add_subplot(2,2,2)
ax[2] = fig.add_subplot(2,2,3)
```

```
# plot

title_1="question3:X={}".format(median_value[0])
graph(ax[0],infox[0,:],infox[1,:
↪],x_label="Y",y_label="Z",title="question3,r=683:X={:.2f}".
↪format(median_value[0]),legend='in x',color='r',linestyle='')
graph(ax[1],infoy[0,:],infoy[1,:],x_label="X",y_label="Z",title="Y={:.2f}".
↪format(median_value[1]),legend='in y',color='b',linestyle='')
graph(ax[2],infoz[0,:],infoz[1,:],x_label="X",y_label="Y",title="Z={:.2f}".
↪format(median_value[2]),legend='in z',color='k',linestyle='')
# output
plt.show()
#plt.savefig('savefig_example.eps') #eps piture
```

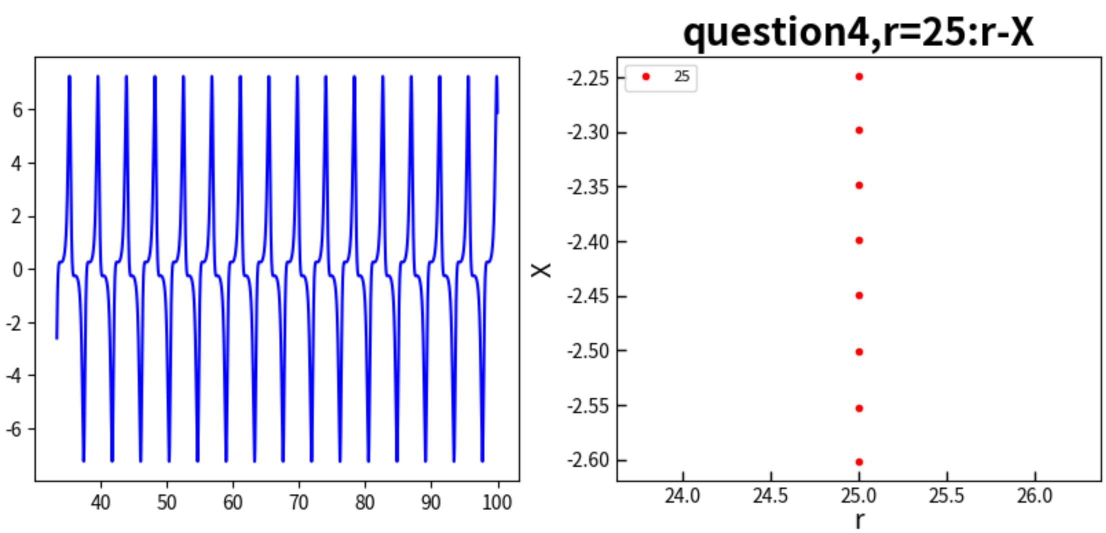


1.1.4 :

t_array=np.linspace(0,100,10000), 2/3

r=18 r=25 , 4 ,

[16]:
```
# Data
##
t_array=np.linspace(0,100,10000)
start_seq=int(1/3*np.size(t_array))
t_array_cut=t_array[start_seq:-1]
# Conpution
sol=parameter_vary(r=25,yinitial_array=yinitial_array,t_array=t_array)
sol_cut=sol[0,start_seq:-1]

# period
sol_period=sol_cut[0:-1:862]
n=np.size(sol_period)
r_period=np.zeros(n)+25

# Graph
# fig
ax=[0,0,0]
fig = plt.figure(figsize=(10,4))
ax[0] = fig.add_subplot(1,2,1)
ax[1] = fig.add_subplot(1,2,2)
ax[0].plot(t_array_cut,sol_cut,color='b',linestyle='-')
graph(ax[1],r_period,sol_period,x_label="r",y_label="X",title="question4,r=25:
↪r-X",legend='25',color='r',linestyle='',marker='.')
plt.show()
```



sol_period=sol_cut[:,1:-1:space] ,

r=25 space 862 862/10000*100=8.62

[17]:
```
# period
space_group=np.arange(1,1000,1)
para=100000
period=0

for space in space_group:
    sol_period=sol_cut[0:-1:space]
    if para>np.var(sol_period):
        period=space
        para=np.var(sol_period)
print("r=25   :{}".format(period/10000*100))
```

```
r=25   :8.62
```

r=18 5.36

[18]:
```
# Calculation
sol=parameter_vary(r=18,yinitial_array=yinitial_array,t_array=t_array)
sol_cut=sol[0,start_seq:-1]

for space in space_group:
    sol_period=sol_cut[0:-1:space]
    if para>np.var(sol_period):
        period=space
        para=np.var(sol_period)
print("r=18   :{}".format(period/10000*100))
```

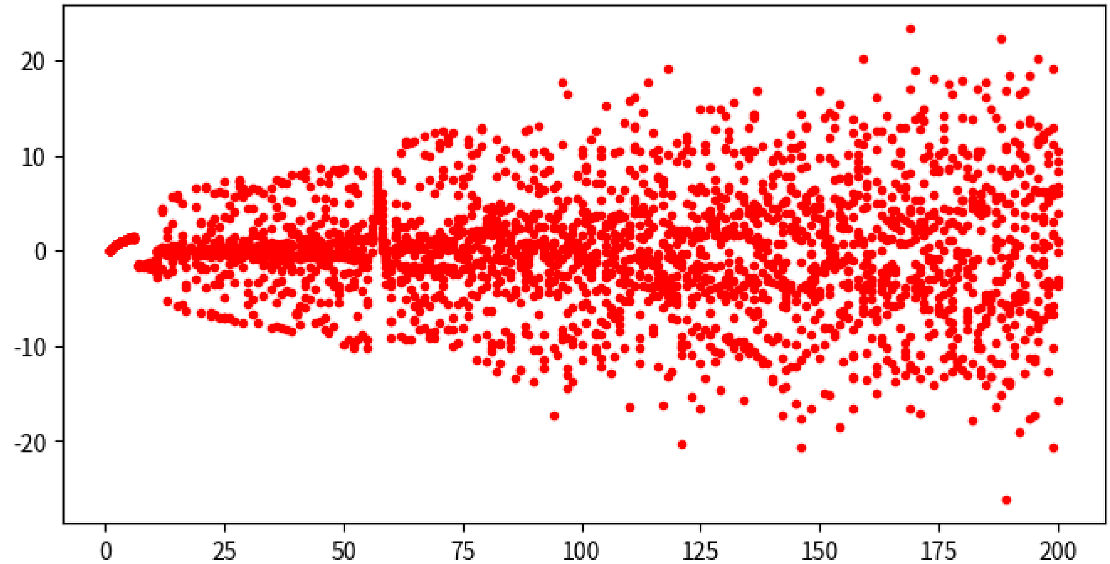
```
r=18   :5.36
```

r=18 5.36

, , r

[19]:
```
r_period=np.linspace(1,200,200)
num=np.size(r_period)

# plot
ax=[0,0,0,0,0,0]
fig = plt.figure(figsize=(8,4))
ax[0] = fig.add_subplot(1,1,1)

for i in range(num):
    # Conpution
    sol=parameter_vary(r=r_period[i],yinitial_array=yinitial_array,t_array=t_array)
    sol_cut=sol[:,start_seq:-1]
    # period
    sol_period=sol_cut[0,1:-1:536]
    n=np.size(sol_period)
```

```
    r_period_i=np.zeros(n)+r_period[i]
    # plot
    ax[0].plot(r_period_i,sol_period,linestyle='',marker='.',color='r')
```



1.1.5 FFT

x #### r 684 r 0 683

t_array=np.linspace(0,250,15000), 2/3

r parameter_vary(r=) r x-y-z

t $\frac{2\pi}{W}$ F θ $\frac{2\pi}{W}$ F$np.sin(W$t)

$O(n^2/2)$ r $O(n^4)$

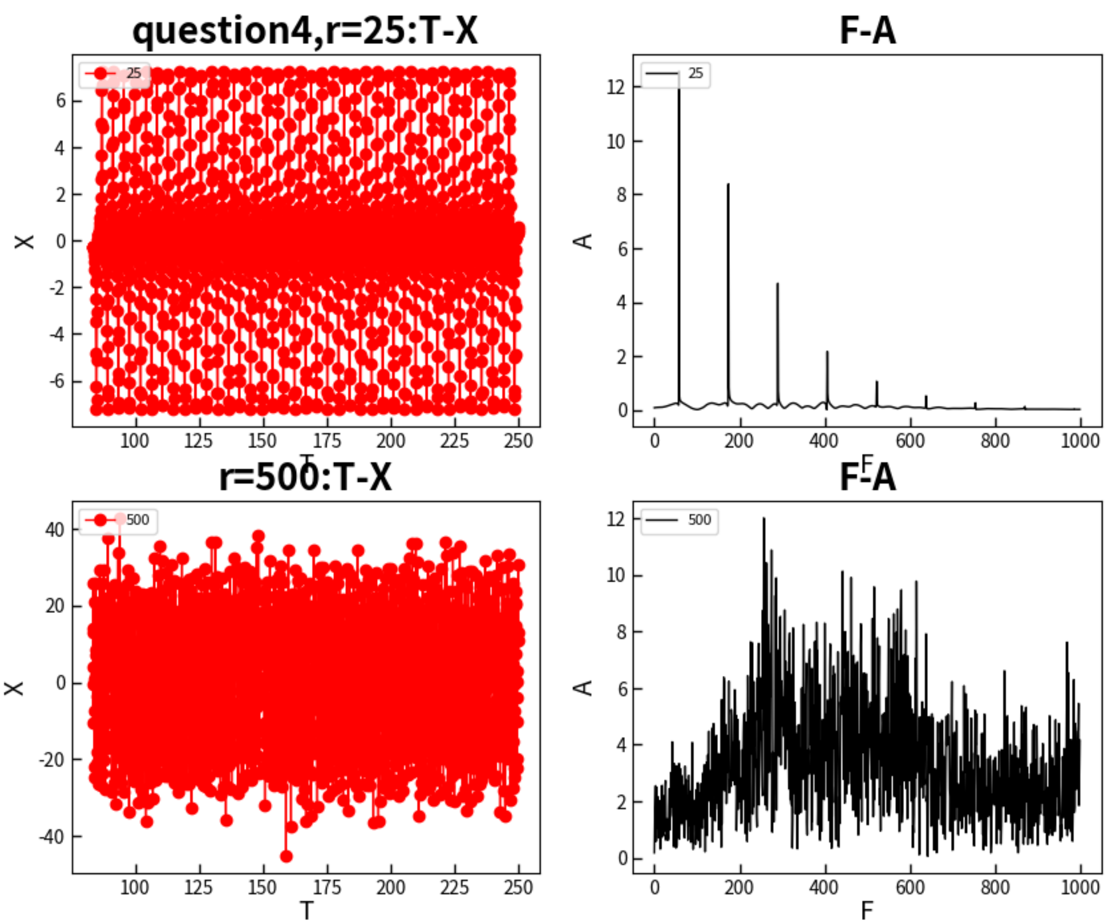
FFT FFT

1. 700 15000*2/3/5 t_array_cut[0:-1:5] FFT

r=25 r=500

[20]:
```
# Data
##
t_array=np.linspace(0,250,15000)
##  r
t_array_cut0,sol_cut0,half_x0,normalization_half_y0=FFT_parameter_vary(t_array,r=25)
t_array_cut1,sol_cut1,half_x1,normalization_half_y1=FFT_parameter_vary(t_array,r=683)

# Graph
# fig
ax=[0,0,0,0,0,0]
```

```
fig = plt.figure(figsize=((10,8)))
ax[0]=fig.add_subplot(2,2,1)
ax[1]=fig.add_subplot(2,2,2)
ax[2]=fig.add_subplot(2,2,3)
ax[3]=fig.add_subplot(2,2,4)
graph(ax[0],t_array_cut0,sol_cut0[0,:
  ↪],x_label="T",y_label="X",title="question4,r=25:
  ↪T-X",legend='25',color='r',linestyle='-')
graph(ax[1],half_x0,normalization_half_y0,x_label="F",y_label="A",title="F-A",legend='25',colo
graph(ax[2],t_array_cut1,sol_cut1[0,:],x_label="T",y_label="X",title="r=500:
  ↪T-X",legend='500',color='r',linestyle='-')
graph(ax[3],half_x1,normalization_half_y1,x_label="F",y_label="A",title="F-A",legend='500',col
plt.show()
```

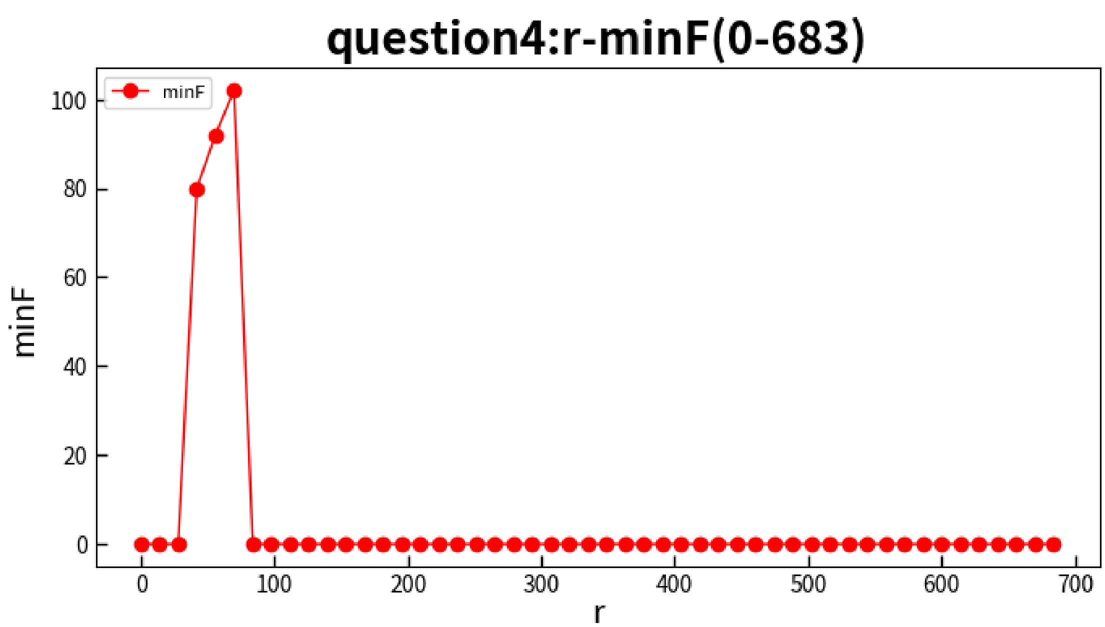


2. “ A-F A 1/80 10 , 0 ”

3. r linspace(0,684)

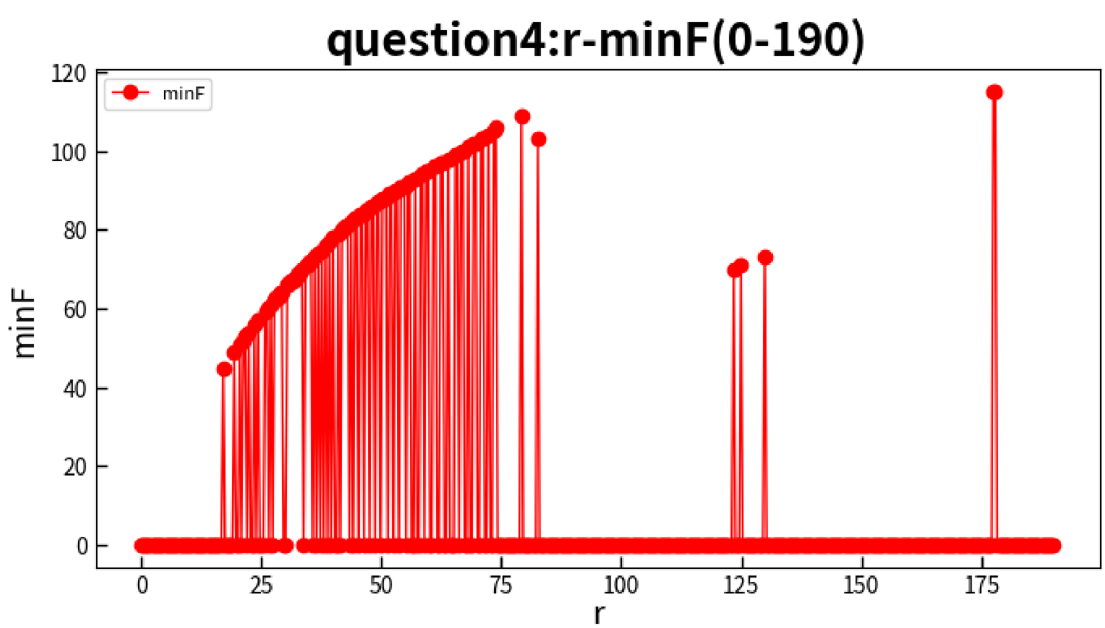
[21]:

```
# Konwn half_x0,normalization_half_y0
# Data
##
t_array=np.linspace(0,250,15000)
##  r
r=np.linspace(0,684,50)
n=np.size(r)
##
F=np.zeros(n)


for i in range(n):
    # FFT
    ␣
 ↪t_array_cut0,sol_cut0,half_x0,normalization_half_y0=FFT_parameter_vary(t_array,r=r[i])
    F[i]=periodicity(half_x0,normalization_half_y0)

#plot
fig = plt.figure(figsize=((8,4)))
ax=fig.add_subplot(1,1,1)
graph(ax,r,F,x_label="r",y_label="minF",title="question4:
 ↪r-minF(0-683)",legend='minF',loc='upper left',\
          color="r",linestyle='-',linewidth=1,marker='o')
plt.show()
```



4. 190

[22]:
```
# Konwn half_x0,normalization_half_y0
# Data
##
t_array=np.linspace(0,250,15000)
##  r
r=np.linspace(0,190,500)
n=np.size(r)
##
F=np.zeros(n)


for i in range(n):
    # FFT
    ␣
 ↪t_array_cut0,sol_cut0,half_x0,normalization_half_y0=FFT_parameter_vary(t_array,r=r[i])
    F[i]=periodicity(half_x0,normalization_half_y0)

#plot
fig = plt.figure(figsize=((8,4)))
ax=fig.add_subplot(1,1,1)
graph(ax,r,F,x_label="r",y_label="minF",title="question4:
 ↪r-minF(0-190)",legend='minF',loc='upper left',\
          color="r",linestyle='-',linewidth=1,marker='o')
plt.show()
print("  ")
```

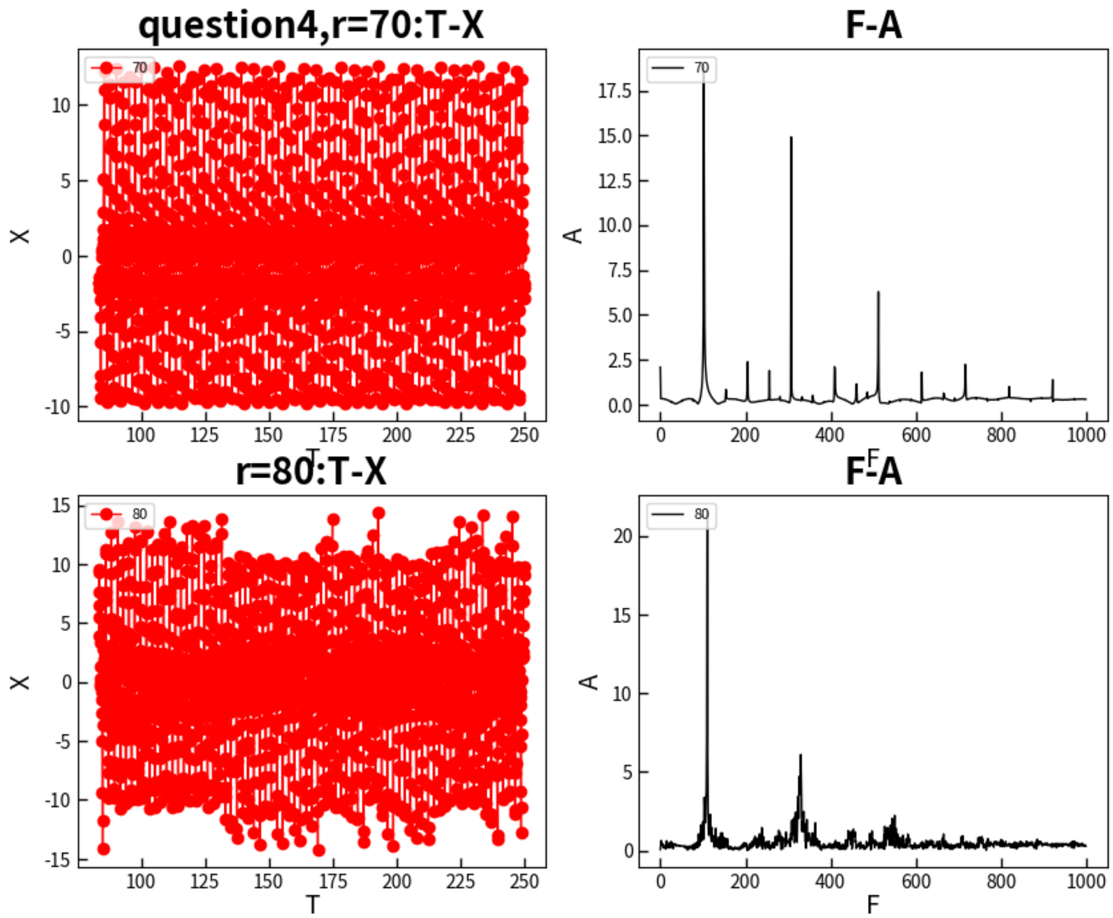


75 80 " A-F A 1/80 10 , 0 "

r 11 x x

5. r=[70,80] r=80 r=80 FFT

[23]:
```
# Data
##
t_array=np.linspace(0,250,15000)
##  r
t_array_cut0,sol_cut0,half_x0,normalization_half_y0=FFT_parameter_vary(t_array,r=70)
t_array_cut1,sol_cut1,half_x1,normalization_half_y1=FFT_parameter_vary(t_array,r=80)

# Graph
# fig
ax=[0,0,0,0,0,0]
fig = plt.figure(figsize=((10,8)))
ax[0]=fig.add_subplot(2,2,1)
ax[1]=fig.add_subplot(2,2,2)
ax[2]=fig.add_subplot(2,2,3)
ax[3]=fig.add_subplot(2,2,4)
graph(ax[0],t_array_cut0,sol_cut0[0,:
 ↪],x_label="T",y_label="X",title="question4,r=70:
 ↪T-X",legend='70',color='r',linestyle='-')
graph(ax[1],half_x0,normalization_half_y0,x_label="F",y_label="A",title="F-A",legend='70',colo
graph(ax[2],t_array_cut1,sol_cut1[0,:],x_label="T",y_label="X",title="r=80:
 ↪T-X",legend='80',color='r',linestyle='-')
graph(ax[3],half_x1,normalization_half_y1,x_label="F",y_label="A",title="F-A",legend='80',colo
plt.show()
```



1.2

[0.5,0.5,0.5] r 70 80

r r r ( )

r=24.74 r>24.74

r