13+0
=>13/20

Tutorial09

June 26, 2020

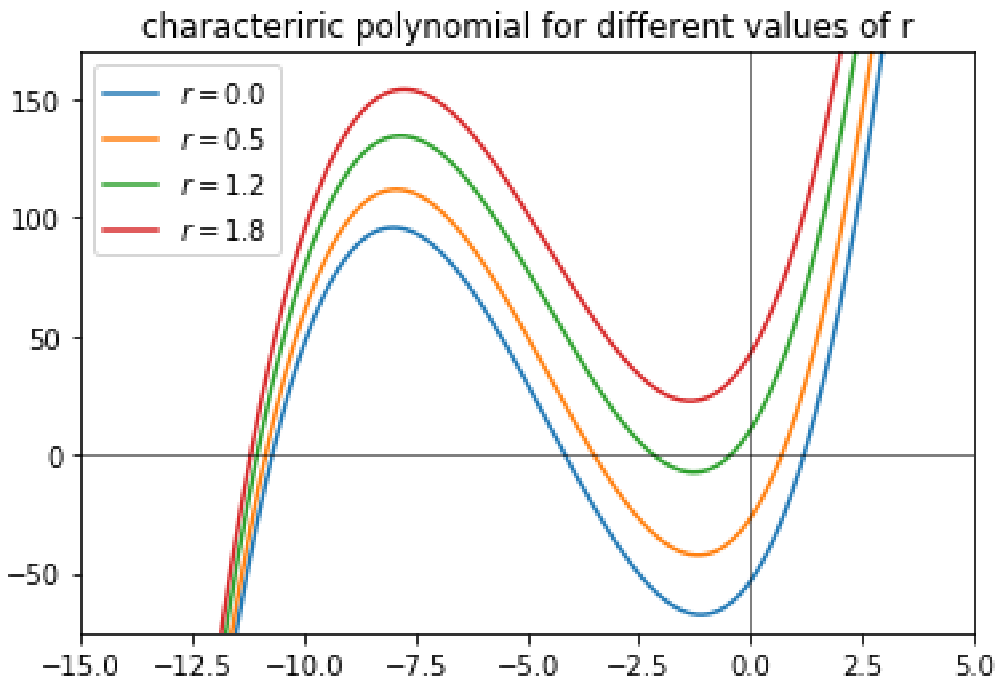
1 Tutorial 9: Fixed Points of the Lorenz dynamical system

Group Members: Julius Franke (el442, juliusttf@gmail.com), Erik Meister (kd400, erik.meister@me.com), Eugen Dizer (qo452, eugen9898@web.de)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize
from mpl_toolkits import mplot3d
```

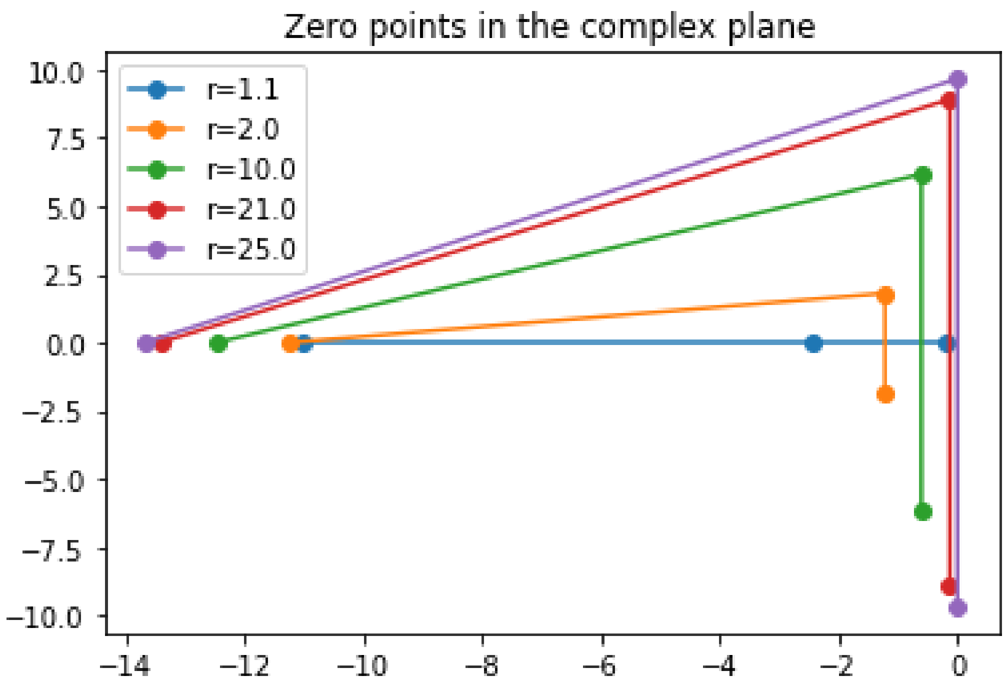
[2]:
```
def polynomial(sig,b,lamb,r):
    return lamb**3+(1+b+sig)*lamb**2+b*(sig+r)*lamb+2*sig*b*(r-1)
```

[3]:
```
sig=10
b=8/3
r=np.array([0,0.5,1.2,1.8])
lamb=np.linspace(-20,10,500)

for i in r:
    plt.plot(lamb,polynomial(sig,b,lamb,i),label=r'$r={}$'.format(i))
    plt.ylim(-75,170)
    plt.xlim(-15,5)


plt.plot([0,0],[-100,200],'k-',lw=0.5)
plt.plot([-20,10],[0,0],'k-',lw=0.5)
plt.title('characteriric polynomial for different values of r')
plt.legend()
plt.show()
```



```
[4]: def p(lamb):
         return lamb**3+(1+b+sig)*lamb**2+b*(sig+r)*lamb+2*sig*b*(r-1)
```

```
[5]: for r in np.array([1.1,2,10,21,25]):
         roots = np.roots([1,(1+b+sig),b*(sig+r),2*sig*b*(r-1)])
         x = []
         y = []
         for i in range(len(roots)):
             x.append(roots[i].real)
             y.append(roots[i].imag)
         plt.plot(x,y, marker="o", label="r={}".format(r))
     plt.title('Zero points in the complex plane')
     plt.legend()
     plt.show()
```



2 Homework 9: The Lorenz attractor

1. Solve numerically, using rk4, the above coupled set of equations for the values r = 0.5, 1.17, 1.3456, 25.0 and 29.0. Choose the initial conditions near one of the fixed points: $C\pm$ for $r > 1$ and $(0, 0, 0)$ for $r < 1$. Explain the behavior, as much as possible, with the stability properties of the fixed points.

[6]:
```
# use given rk4 implementation
def rk4_step(y0, x0, f, h):

    k1 = h * f(y0)
    k2 = h * f(y0 + k1/2)
    k3 = h * f(y0 + k2/2)
    k4 = h * f(y0 + k3)

    xp1 = x0 + h
    yp1 = y0 + 1/6*(k1 + 2*k2 + 2*k3 + k4)

    return(yp1,xp1)

def rk4(y0, x0, f, h, n):

    yn = np.zeros((n+1, y0.shape[0]))
    xn = np.zeros(n+1)
    yn[0,:] = y0
    xn[0] = x0

    for n in np.arange(1,n+1,1):
        yn[n,:], xn[n] = rk4_step(y0 = yn[n-1,:], x0 = xn[n-1], f = f, h = h)

    return(yn, xn)
```

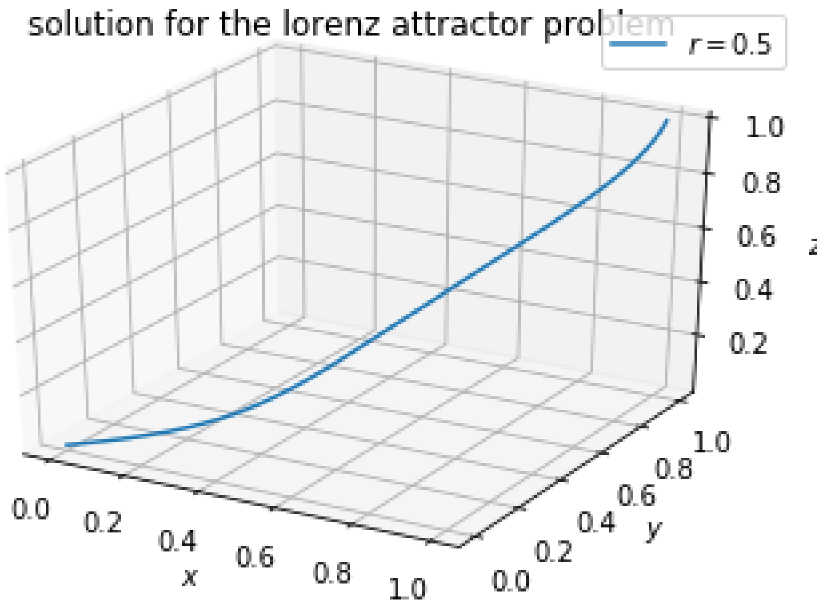
[7]:
```
#Define derivatives of all variables, when getting the M x 1 array of all␣
↪variables a
def derivative_of_x_pos(a):
    x_pos=a[0:1]
    y_pos=a[1:2]
    z_pos=a[2:3]
    return list(-sig*(x_pos-y_pos))

def derivative_of_y_pos(a):
    x_pos=a[0:1]
    y_pos=a[1:2]
    z_pos=a[2:3]
    return list(r*x_pos-y_pos-x_pos*z_pos)

def derivative_of_z_pos(a):
    x_pos=a[0:1]
    y_pos=a[1:2]
    z_pos=a[2:3]
    return list(x_pos*y_pos-b*z_pos)

#Define function that calculates the derivates of all variables of the ODE
def deriv(a):
    return np.array(derivative_of_x_pos(a) + derivative_of_y_pos(a) +␣
↪derivative_of_z_pos(a))
```

[8]:
```
# define function, which takes an initial condition and gives back a 3D-plot of␣
↪x,y,z
def traj(initial):

    #Choose steps size h
    stepsize = 0.005

    #use rk4 to numerically calculate x,y,z over time
    pos, time = rk4(initial, 0, deriv, stepsize, 10000)

    pos_x = []
    pos_y = []
    pos_z = []
    for i in range(len(pos)):
        pos_x.append(pos[i][0])
        pos_y.append(pos[i][1])
        pos_z.append(pos[i][2])

    fig = plt.figure()
    ax = plt.axes(projection="3d")

    _=ax.plot3D(pos_x, pos_y, pos_z, label=r'$r={}$'.format(r))
    ax.set_title('solution for the lorenz attractor problem')
    ax.legend(bbox_to_anchor=(1, 1))
    ax.set_xlabel(r'$x$')
    ax.set_ylabel(r'$y$')
    ax.set_zlabel(r'$z$')
```

First we choose $r = 0.5$ Since in this case $r < 1$, we choose the initial condition near the trivial fixpoint.

[9]:
```
r=0.5
init = np.array([1,1,1])
traj(init)
```

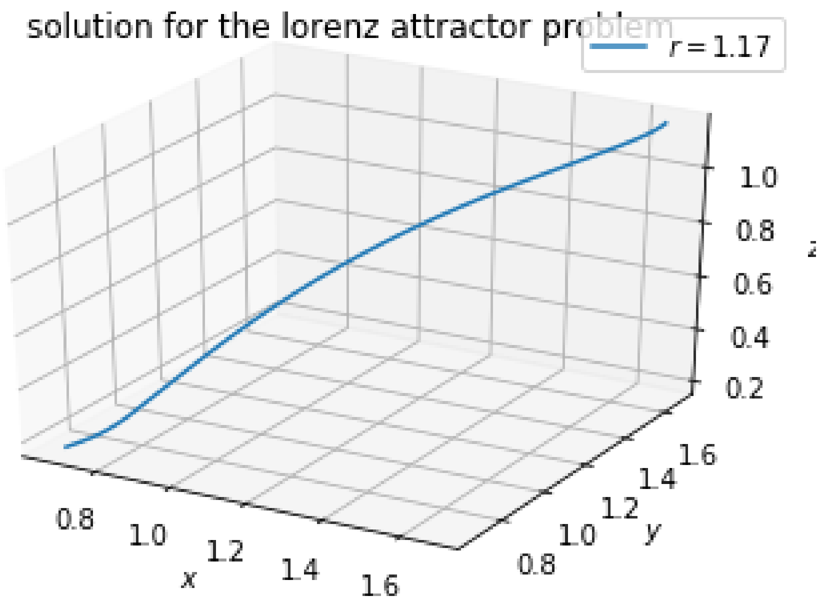
✓



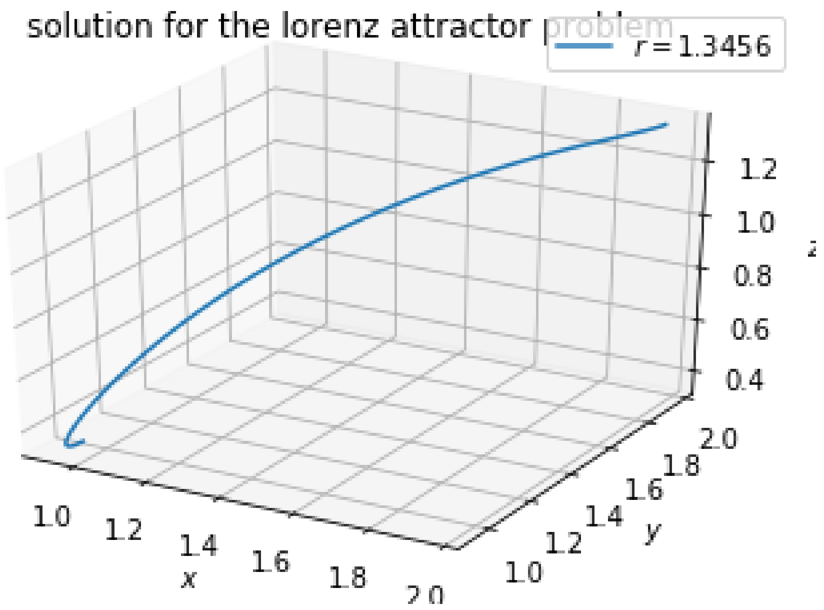
One can clearly see, that the solution approaches $(0, 0, 0)$ directly. This is expected, because the trivial fixpoint is stable for $r < 1$, since all eigenvalues are real and negative.

The following values for r all satisfy $r > 1$, so we choose the initial condition near the 2nd/3rd fixpoint.

[10]:
```
r_array =np.array([1.17, 1.3456, 25.0, 29.0])

for r in r_array:
    #initial conditions (choose near fix points)
    init = np.array([np.sqrt(b*(r-1)) + 1, np.sqrt(b*(r-1)) + 1, r-1 + 1])
    traj(init)
```

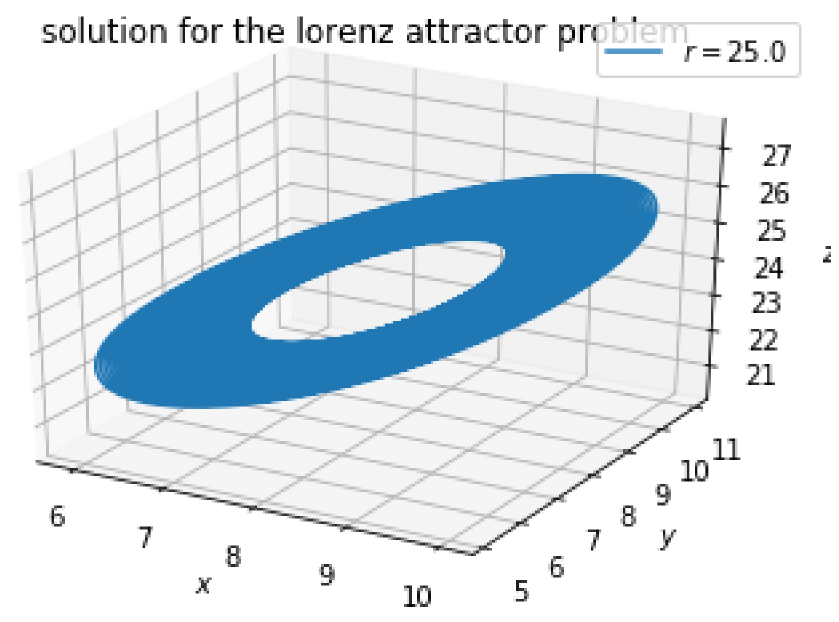


✓

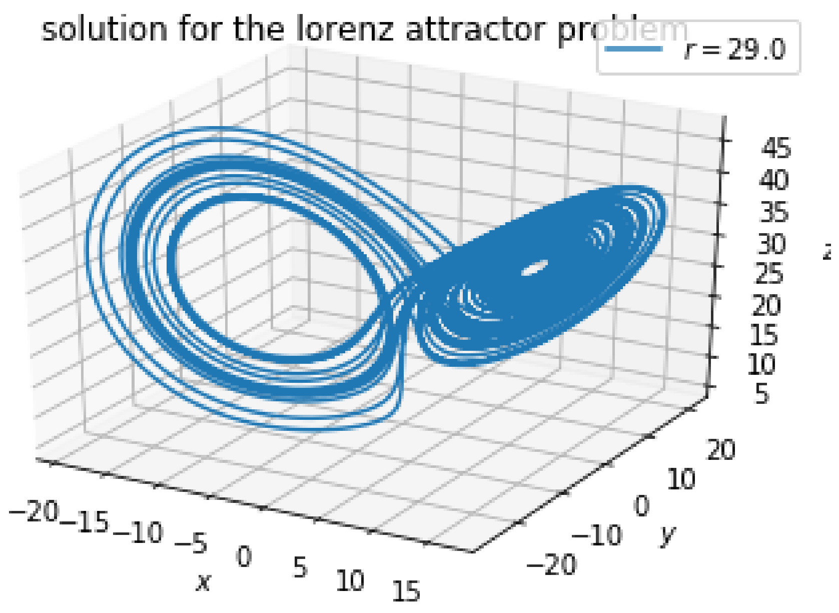


✓

✓



✓



plot 1: For $1 < r < 1.346$ the eigenvalues are still real and negative as one can see in the plot from the tutorial "zero points in the complex plane". Therefore our solution is fully stable.

plot 2: For $1.346 < r < 24.74$ one eigenvalue remains real and smaller tha zero and the other two are complex conjugates with a negative realpart. Therefore our solution circulates towards the fixpoint.

plot 3: We discover that for $r = 25$ our solution is a "stable limit cycle". The solution never reaches the fixpoint, but circulates around it infinitely. With our mathematical tools we can not say something about the existence of a "stable limit cycle".

plot 4: For $r = 29$ the realpart of the two complex eigenvalues is positive and our fixpoints become unstable. The solution circulates away from the first fixpoint and then jumps over to the second fixpoint. But because this fixpoint is also unstable it again circulates away and then jumps back to the first fixpoint and so on. The solution "oscillates" between the two fixpoints, but never reaches them as they are unstable. They can not escape as they are trapped because of the "volume contraction". We can see the shape of a strange attractor!

```
[ ]:
```

13/13