DİFERANSİYEL DENKLEMLER

Euler yöntemi

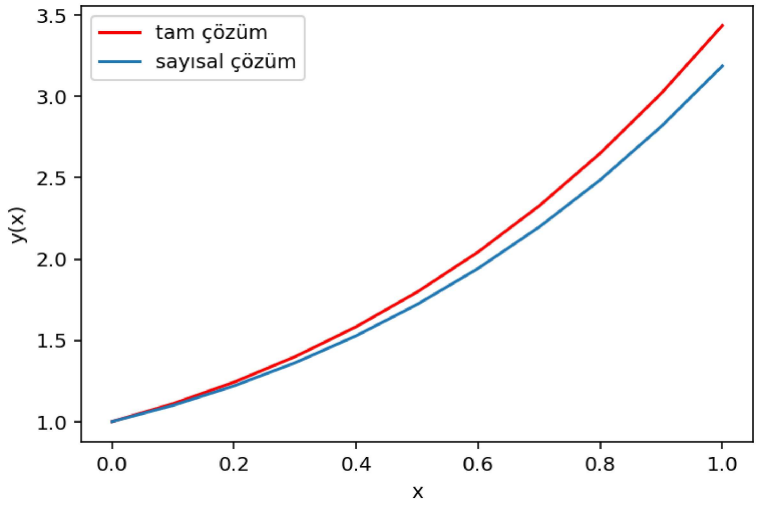
In [1]:
```
def euler(x0, y0, h, n):
    x=x0; y=y0; xd=[x0]; yd=[y0];
    for i in range(n):
        y = y + h*f(x,y)
        yd.append(y)
        x = x + h
        xd.append(x)
    return xd, yd
```

In [2]:
```
from math import *

def f(x,y):
    return x+y

n = 10
h = 0.1
x0 = 0.0
y0 = 1.0
x, y = euler(x0,y0,h,n)
y_tam=[2*exp(xi) - xi - 1.0 for xi in x]

print "x   y(euler)   y(tam)"
for i,xi in enumerate(x):
    print "%.2f" % xi,"%.6f" % y[i],"%.6f" % y_tam[i]
```

```
x   y(euler)   y(tam)
0.00 1.000000 1.000000
0.10 1.100000 1.110342
0.20 1.220000 1.242806
0.30 1.362000 1.399718
0.40 1.528200 1.583649
0.50 1.721020 1.797443
0.60 1.943122 2.044238
0.70 2.197434 2.327505
0.80 2.487178 2.651082
0.90 2.815895 3.019206
1.00 3.187485 3.436564
```

In [3]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(x, y_tam, color='red', label=u'tam çözüm')
plt.plot(x, y, label=u'sayısal çözüm')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.show()
```

Out[3]:

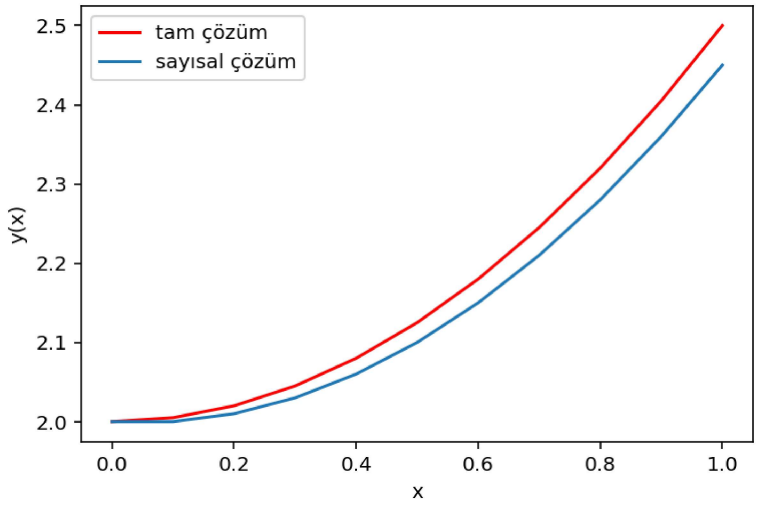


In [4]:
```
from math import *

def f(x,y):
    return x

n = 10
h = 0.1
x0 = 0.0
y0 = 2.0
x, y = euler(x0,y0,h,n)
y_tam=[xi**2./2. + 2. for xi in x]

print "x   y(euler)   y(tam)"
for i,xi in enumerate(x):
    print "%.2f" % xi,"%.6f" % y[i],"%.6f" % y_tam[i]
```

```
x   y(euler)   y(tam)
0.00 2.000000 2.000000
0.10 2.000000 2.005000
0.20 2.010000 2.020000
0.30 2.030000 2.045000
0.40 2.060000 2.080000
0.50 2.100000 2.125000
0.60 2.150000 2.180000
0.70 2.210000 2.245000
0.80 2.280000 2.320000
0.90 2.360000 2.405000
1.00 2.450000 2.500000
```

In [5]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(x, y_tam, color='red', label=u'tam çözüm')
plt.plot(x, y, label=u'sayısal çözüm')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.show()
```

Out[5]:



In [6]:
```
log(e)
```

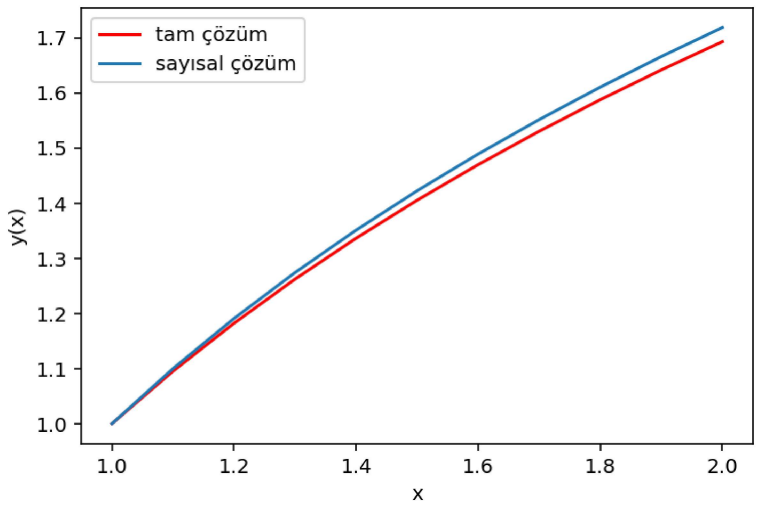
Out[6]:
```
1.0
```

In [7]:
```
from math import *

def f(x,y):
    return 1./x

n = 10
h = 0.1

x0 = 1.0
y0 = 1.0

x, y = euler(x0,y0,h,n)
y_tam=[log(xi) + 1. for xi in x]

print "x   y(euler)   y(tam)"
for i,xi in enumerate(x):
    print "%.2f" % xi,"%.6f" % y[i],"%.6f" % y_tam[i]
```

```
x   y(euler)   y(tam)
1.00 1.000000 1.000000
1.10 1.100000 1.095310
1.20 1.190909 1.182322
1.30 1.274242 1.262364
1.40 1.351166 1.336472
1.50 1.422594 1.405465
1.60 1.489261 1.470004
1.70 1.551761 1.530628
1.80 1.610584 1.587787
1.90 1.666140 1.641854
2.00 1.718771 1.693147
```

In [8]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(x, y_tam, color='red', label=u'tam çözüm')
plt.plot(x, y, label=u'sayısal çözüm')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.show()
```

Out[8]:



Runge-Kutta yöntemi

http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html (http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html)

https://youtu.be/X-_qCcYDbuY (https://youtu.be/X-_qCcYDbuY)
https://youtu.be/hhgG8KL_pCk (https://youtu.be/hhgG8KL_pCk)

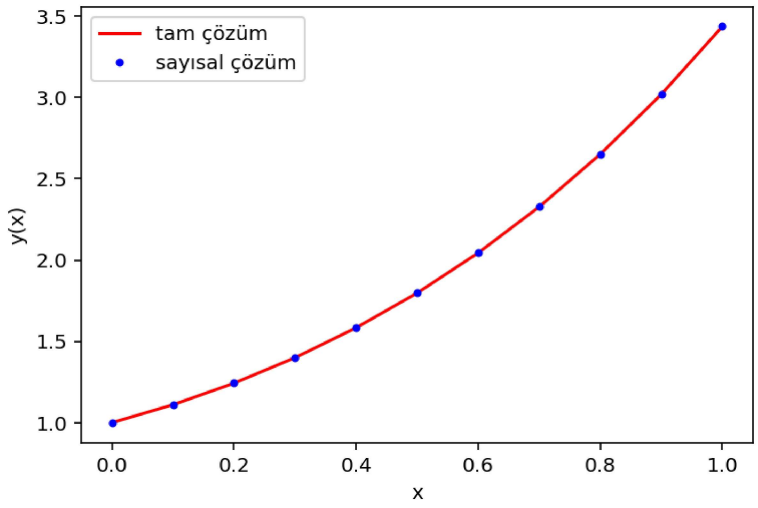
In [9]:
```
def rk4(x0,y0,h,n):
    x=x0;y=y0;xd=[x0];yd=[y0]
    for i in range(n):
        k1=h*f(x,y)
        k2=h*f(x+0.5*h, y+0.5*k1)
        k3=h*f(x+0.5*h, y+0.5*k2)
        k4=h*f(x+h, y+k3)
        # yeni x ve y değerlerini hesapla
        y=y+(k1+2*(k2+k3)+k4)/6.0
        x=x+h
        # hesaplanan x ve y değerlerini
        # xd, yd listelerine ekle
        yd.append(y)
        xd.append(x)
    return (xd,yd)
```

In [10]:
```
from math import *
def f(x,y):
    return x+y

n=10
h=0.1
x0=0.0
y0=1.0
x,y=rk4(x0,y0,h,n)
y_tam=[2*exp(xi) - xi - 1.0 for xi in x]

print "x   y(runge-kutta)   y(tam)"
for i,xi in enumerate(x):
    print "%.2f" % xi,"%.6f" % y[i],"%.6f" % y_tam[i]
```

```
x   y(runge-kutta)   y(tam)
0.00 1.000000 1.000000
0.10 1.110342 1.110342
0.20 1.242805 1.242806
0.30 1.399717 1.399718
0.40 1.583648 1.583649
0.50 1.797441 1.797443
0.60 2.044236 2.044238
0.70 2.327503 2.327505
0.80 2.651079 2.651082
0.90 3.019203 3.019206
1.00 3.436559 3.436564
```

In [11]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(x, y_tam, color='red', label=u'tam çözüm')
plt.plot(x, y, 'b.', label=u'sayısal çözüm')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.show()
```

Out[11]:



İkinci dereceden denklemler ve lineer denklem sistemleri

https://www.youtube.com/watch?v=_HDHh-ArpzQ (https://www.youtube.com/watch?v=_HDHh-ArpzQ)

Runge-Kutta yöntemiyle m bilinmeyenli lineer denklem sistemi

In [12]:
```
def rk4m(x,y,h,n):
    Xlist = [x]
    Ylist = [y]
    for i in range(n):
        k0=f(x,y)
        k1=f(x+h/2.0,y+(h/2)*k0)
        k2=f(x+h/2.0,y+(h/2)*k1)
        k3=f(x+h, y+h*k2)
        # yeni x ve y değerlerini hesapla
        y=y+(h/6)*(k0+2.0*k1+2.0*k2+k3)
        x=x+h
        # hesaplanan x ve y değerlerini
        # xd, yd listelerine ekle
        Xlist.append(x)
        Ylist.append(y)
    #print "k0 = %s"%k0
    #print "k1 = %s"%k1
    #print "k2 = %s"%k2
    #print "k3 = %s"%k3
    return Xlist, Ylist
```

UYGULAMA

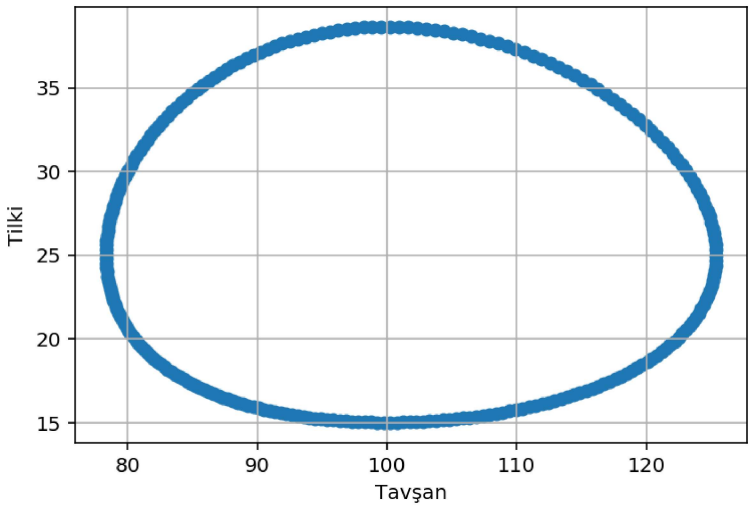
Av-avcı Modeli

In [13]:

```
# Bekir Karaoğlu'nun kitabındaki program
from numpy import *
from pylab import *

def f(x,y, a=0.25, b=0.01, c= 1.0, d=0.01):
    f = zeros(2)
    f[0] = a*y[0] - b*y[0]*y[1]
    f[1] = -c*y[1] + d*y[0]*y[1]
    return f

h = 0.05
n = 255
x0 = 0.0
y0 = array([80., 30.])

X, Y = rk4m(x0,y0,h,n)

y1 = [y1 for y1,y2 in Y]
y2 = [z2 for z1,z2 in Y]

print "t", "   y[1]", " y[2]"
for i in range(0, n, 10):
    print "%10.3f"%X[i], "%12.3f"%y1[i], "%12.3f"%y2[i]

scatter(y1, y2)
grid()
xlabel(u'Tavşan')
ylabel(u'Tilki')
show()
```

t	y[1]	y[2]
0.000	80.000	30.000
0.500	78.608	27.037
1.000	78.361	24.269
1.500	79.144	21.814
2.000	80.848	19.730
2.500	83.372	18.036
3.000	86.622	16.728
3.500	90.505	15.795
4.000	94.917	15.226
4.500	99.737	15.022
5.000	104.811	15.193
5.500	109.941	15.764
6.000	114.868	16.775
6.500	119.268	18.275
7.000	122.750	20.309
7.500	124.882	22.892
8.000	125.261	25.970
8.500	123.609	29.370
9.000	119.903	32.772
9.500	114.449	35.733
10.000	107.862	37.795
10.500	100.911	38.633
11.000	94.322	38.164
11.500	88.633	36.556
12.000	84.143	34.132
12.500	80.957	31.263

Out[13]:

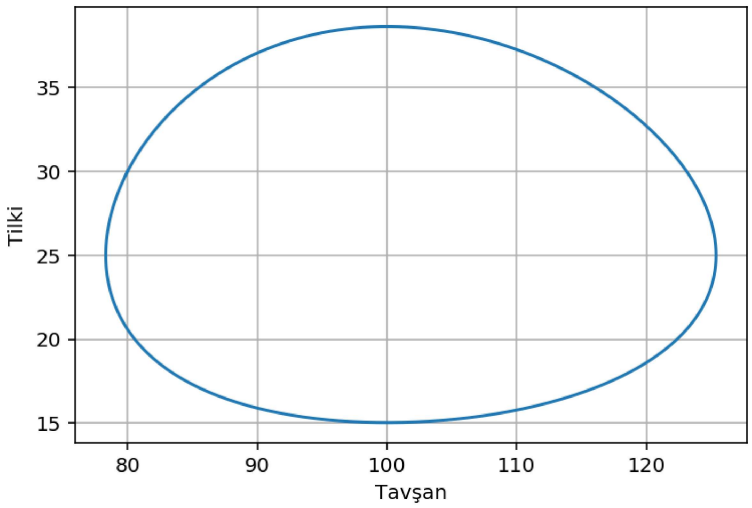


In [14]:

```
# Aynı programın yeniden düzenlenmiş hali
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

def f(x, y, a=0.25, b=0.01, c= 1.0, d=0.01):
    dydt1 =  a*y[0] - b*y[0]*y[1]
    dydt2 = -c*y[1] + d*y[0]*y[1]
    return np.array([dydt1, dydt2])

h = 0.05
n = 255
x0 = 0.0
y0 = np.array([80., 30.])

X, Y = rk4m(x0, y0, h, n)

Y = np.array(Y)
y1 = Y[0:,0]
y2 = Y[0:,1]

print "%10s %12s %12s"%("t","y[1]","y[2]")
for i in range(0, n, 10):
    print "%10.3f %12.3f %12.3f"%(X[i], y1[i], y2[i])

plt.plot(y1,y2)
plt.xlabel(u'Tavşan')
plt.ylabel(u'Tilki')
plt.grid()
plt.show()
```

t	y[1]	y[2]
0.000	80.000	30.000
0.500	78.608	27.037
1.000	78.361	24.269
1.500	79.144	21.814
2.000	80.848	19.730
2.500	83.372	18.036
3.000	86.622	16.728
3.500	90.505	15.795
4.000	94.917	15.226
4.500	99.737	15.022
5.000	104.811	15.193
5.500	109.941	15.764
6.000	114.868	16.775
6.500	119.268	18.275
7.000	122.750	20.309
7.500	124.882	22.892
8.000	125.261	25.970
8.500	123.609	29.370
9.000	119.903	32.772
9.500	114.449	35.733
10.000	107.862	37.795
10.500	100.911	38.633
11.000	94.322	38.164
11.500	88.633	36.556
12.000	84.143	34.132
12.500	80.957	31.263

Out[14]:



Numpy kullanmadan yapılan çözüm

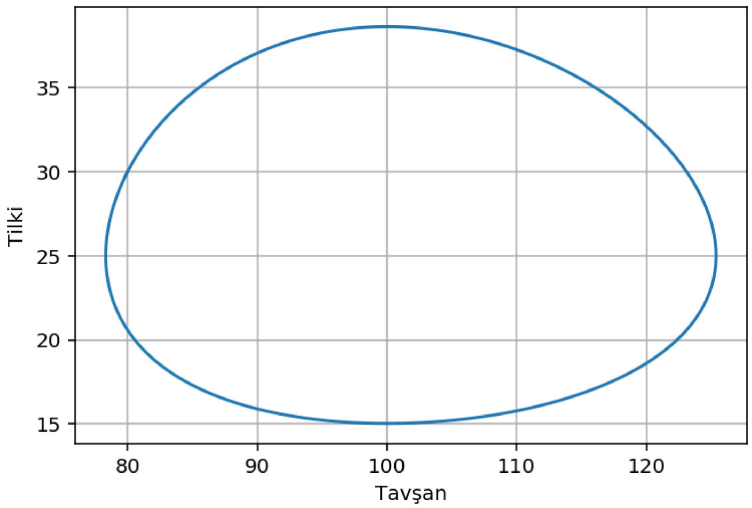
In [15]:

```
sumList = lambda l1,l2: [sum(i) for i in zip(l1, l2)]

def rk4f(x, y, h, n):
    Xlist = [x]
    Y1list = [y[0]]
    Y2list = [y[1]]
    for i in range(n):
        k00=f1(x      , y)
        k10=f2(x      , y)

        k01 = f1(x+h/2.0, sumList(y, [(h/2)*k00, (h/2)*k10]))
        k11 = f2(x+h/2.0, sumList(y, [(h/2)*k00, (h/2)*k10]))

        k02 = f1(x+h/2.0, sumList(y, [(h/2)*k01, (h/2)*k11]))
        k12 = f2(x+h/2.0, sumList(y, [(h/2)*k01, (h/2)*k11]))

        k03 = f1(x+h/2.0, sumList(y, [h*k02, h*k12]))
        k13 = f2(x+h/2.0, sumList(y, [h*k02, h*k12]))

        y[0] = y[0] + (h/6)*(k00+2.0*k01+2.0*k02+k03)
        y[1] = y[1] + (h/6)*(k10+2.0*k11+2.0*k12+k13)

        x=x+h
        Xlist.append(x)
        Y1list.append(y[0])
        Y2list.append(y[1])

    return Xlist, Y1list, Y2list
```

In [16]:
```
# Aynı programın yeniden düzenlenmiş hali
import matplotlib.pyplot as plt
%matplotlib inline

def f1(x, y, a=0.25, b=0.01):
    dydt1 =  a*y[0] - b*y[0]*y[1]
    return dydt1

def f2(x, y, c= 1.0, d=0.01):
    dydt2 = -c*y[1] + d*y[0]*y[1]
    return dydt2

h = 0.05
n = 255
x0 = 0.0
y0 = [80., 30.]

X, y1, y2 = rk4f(x0, y0, h, n)

print "%10s %12s %12s"%("t","y[1]","y[2]")
for i in range(0, n, 100):
    print "%10.3f %12.3f %12.3f"%(X[i], y1[i], y2[i])

plt.plot(y1,y2)
plt.xlabel(u'Tavşan')
plt.ylabel(u'Tilki')
plt.grid()
plt.show()
```

```
         t         y[1]         y[2]
     0.000       80.000       30.000
     5.000      104.811       15.193
    10.000      107.862       37.795
```

Out[16]:



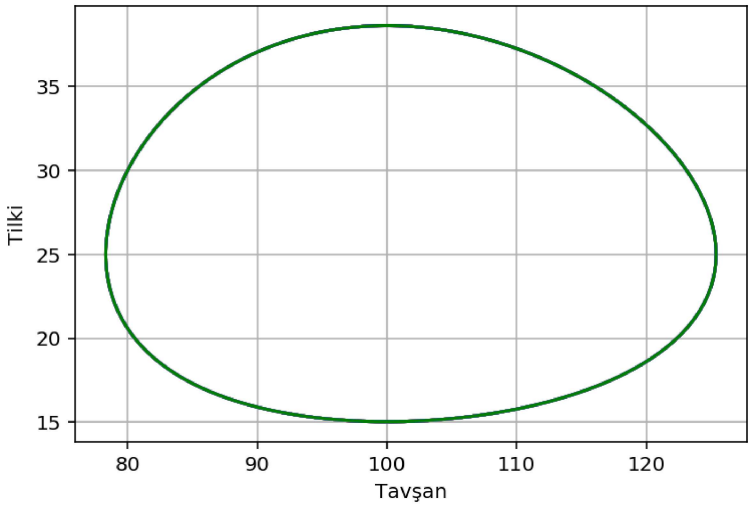
Rengi değişen çizim

In [17]:

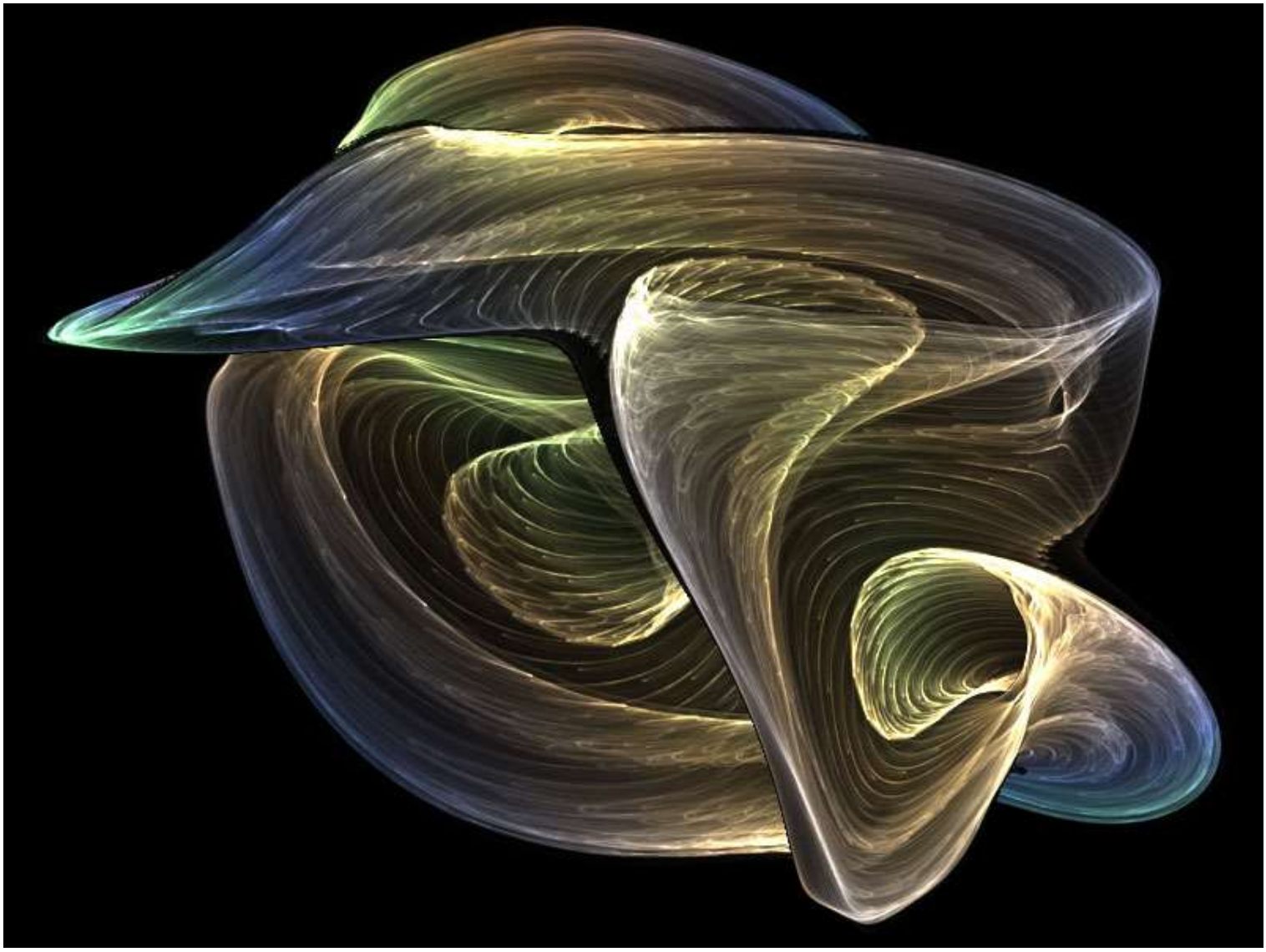
```
# Aynı programın yeniden düzenlenmiş hali
import matplotlib.pyplot as plt
%matplotlib inline

def f1(x, y, a=0.25, b=0.01):
    dydt1 =  a*y[0] - b*y[0]*y[1]
    return dydt1

def f2(x, y, c= 1.0, d=0.01):
    dydt2 = -c*y[1] + d*y[0]*y[1]
    return dydt2

h = 0.05
nc = 10
ns = 255
n = ns*nc

x0 = 0.0
y0 = [80., 30.]

X, y1, y2 = rk4f(x0, y0, h, n)

print "%10s %12s %12s"%("t","y[1]","y[2]")
for i in range(0, n, ns):
    print "%10.3f %12.3f %12.3f"%(X[i], y1[i], y2[i])

colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']


for i in range(nc):
    imin = ns*i
    imax = ns*(i+1)
    plt.plot(y1[imin:imax], y2[imin:imax], colors[i%8])


plt.xlabel(u'Tavşan')
plt.ylabel(u'Tilki')
plt.grid()
plt.show()
```

t	y[1]	y[2]
0.000	80.000	30.000
12.750	79.848	29.767
25.500	79.704	29.533
38.250	79.567	29.300
51.000	79.438	29.068
63.750	79.316	28.835
76.500	79.201	28.604
89.250	79.094	28.372
102.000	78.993	28.142
114.750	78.900	27.913

Out[17]:



Lorenz modelinde Kaos



http://www.chaoscope.org/gallery.htm (http://www.chaoscope.org/gallery.htm)

https://matplotlib.org/examples/mplot3d/lorenz_attractor.html
(https://matplotlib.org/examples/mplot3d/lorenz_attractor.html)

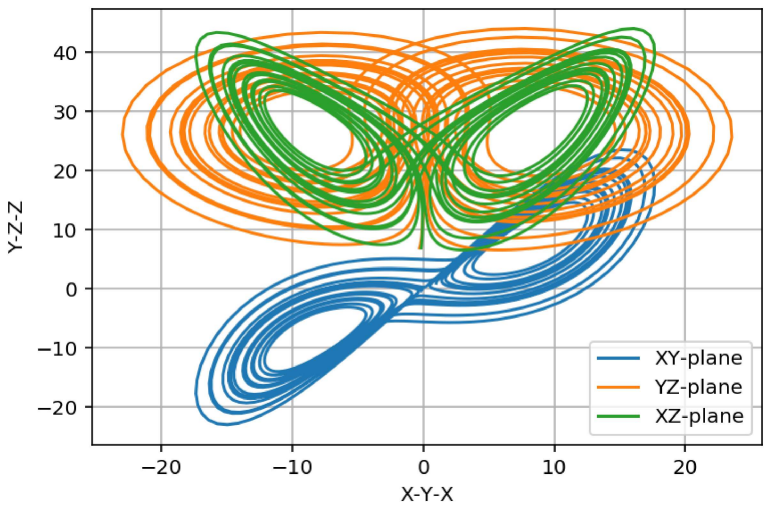
http://nbviewer.jupyter.org/github/mkarakoc/plotExamples/blob/master/Lorenz_Attractor.ipynb
(http://nbviewer.jupyter.org/github/mkarakoc/plotExamples/blob/master/Lorenz_Attractor.ipynb)

launch binder
(http://mybinder.org:/repo/mkarakoc/plotexamples)

In [18]:
```
# Aynı programın yeniden düzenlenmiş hali
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#%matplotlib notebook

def f(x, y):
    # y[0]-->x, y[1]-->y, y[2]-->z
    dxdt =  sigma*(y[1] - y[0])
    dydt = -y[0]*y[2] + rr*y[0] - y[1]
    dzdt = y[0]*y[1] - bb*y[2]
    return np.array([dxdt, dydt, dzdt])

h = 0.01
n = 2000
dn=500
x0 = 0.0
y0 = np.array([1., 1., 20.])

sigma, bb, rr = 10., 8./3., 28.
Xlist, Ylist = rk4m(x0, y0, h, n)

Ylist = np.array(Ylist)
Xs = Ylist[0:,0]
Ys = Ylist[0:,1]
Zs = Ylist[0:,2]

print "%10s %12s %12s %12s"%("t","X","Y","Z")
for i in range(0, n, dn):
    print "%10.3f %12.3f %12.3f %12.3f"%(Xlist[i], Xs[i], Ys[i], Zs[i])

plt.plot(Xs, Ys, label='XY-plane')
plt.plot(Ys, Zs, label='YZ-plane')
plt.plot(Xs, Zs, label='XZ-plane')
plt.xlabel('X-Y-X')
plt.ylabel('Y-Z-Z')
plt.grid()
plt.legend(loc='best')
plt.show()
```

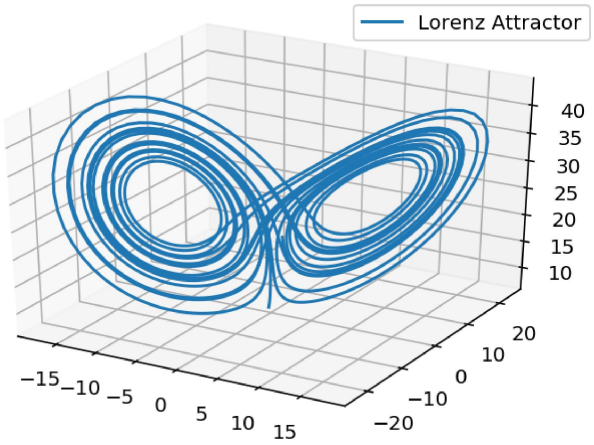
```
         t            X            Y            Z
     0.000        1.000        1.000       20.000
     5.000       -8.968       -2.812       33.958
    10.000        3.253        5.626       12.418
    15.000        6.693        2.041       30.482
```

Out[18]:



In [19]:
```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib notebook
%matplotlib inline

mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(Xs, Ys, Zs, label='Lorenz Attractor')
ax.legend()

plt.show()
```

Out[19]:



DİFERANSİYEL DENKLEMLER II

Sınır Değer ve Öz Değer problemleri

Sonsuz Kuyu

In [20]:
```
# Dönüm noktalarını sayan fonksiyon
def NodeCounter(ylist):
    nodes = 0
    for i, y in enumerate(ylist[2:-1]):
        if ylist[i+1]*y<0: nodes += 1
    return nodes
```

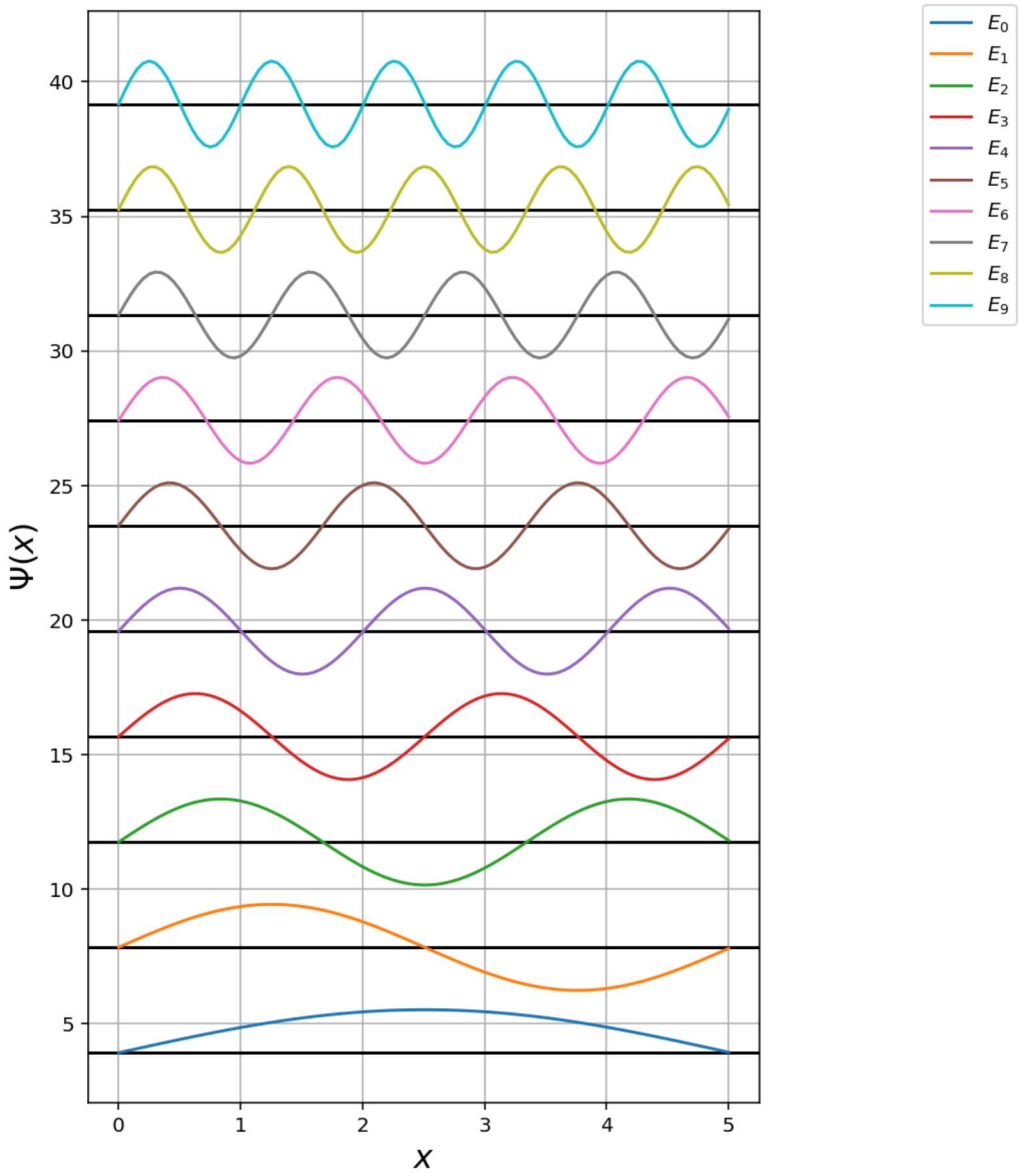
In [21]:

```
# Aynı programın yeniden düzenlenmiş hali
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#%matplotlib notebook

def f(x, y):
    dPsi1dx =  y[1]
    dPsi2dx =  -k2(x)*y[0]
    return np.array([dPsi1dx, dPsi2dx])

def k2(x):
    if x>=0 and x<=5:
        k2V = V0+E
    else:
        k2V = 1/0
    return k2V

h = 0.05
n = 100
x0 = 0.0
y0 = np.array([0., h])
yn = 0.
ytol = 1e-3

V0 = 0

Ens = []
Waves = []
E = 0.
dE = .001
nodeCount = 0
for i in range(10):
    while True:
        Xlist, Ylist = rk4m(x0, y0, h, n)
        # transpose transpose([[a,b], [c,d], [e, f]]) --> [[a,c,e], [b, d, f]]
        nodes = NodeCounter(map(list, zip(*Ylist))[0])
        if abs(Ylist[-1][0] - yn)<ytol and nodes == nodeCount:
            break
        E += dE
    print nodes
    Ens += [E]
    nodeCount +=1
    Ylist = np.array(Ylist).transpose()
    Waves += [Ylist]

print "Öz değerler: %s"%Ens
```

```
0
1
2
3
4
5
6
7
8
9
Öz değerler: [0.3920000000000003, 1.5669999999999382, 3.524999999999723, 6.267000000
000428, 9.792000000000012, 14.099999999997625, 19.192000000000473, 25.06700000000765
3, 31.72600000001579, 39.16999999999941]
```

In [22]:
```
plt.figure(figsize=(6, 10))
for i, W in enumerate(Waves):
    plt.axhline(Ens[i]*10/(i+1), color="black")
    plt.plot(Xlist, W[0]*20*(i+1) + Ens[i]*10/(i+1), label="$E_{%s}$"%i) # Dalga Fonk
siyonları
    #plt.plot(Xlist, W[1]*20*(i+1) + Ens[i]*10/(i+1)) # Dalga Fonksiyonlarının Türevl
eri

plt.legend(bbox_to_anchor=(1.4, 1.015))
plt.grid()
plt.xlabel('$x$', fontsize=16)
plt.ylabel('$\Psi(x)$', fontsize=16)
plt.show()
```

Out[22]:



In [ ]: