PRACTICAL 2

1: Solver selection

Solve y**2=1 with APOPT solver. See APMonitor documentation or GEKKO documentation for additional solver options.

```
pip install gekko
```

```
Collecting gekko
  Downloading gekko-1.0.2-py3-none-any.whl (12.4 MB)
     |████████████████████████████████| 12.4 MB 8.9 MB/s
Requirement already satisfied: numpy>=1.8 in /usr/local/lib/python3.7/dist-packages (from gekko) (1.21.6)
Installing collected packages: gekko
Successfully installed gekko-1.0.2
```

4: Linear and Polynomial Regression

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

xm = np.array([0,1,2,3,4,5])
ym = np.array([0.1,0.2,0.3,0.5,0.8,2.0])

#### Solution
from gekko import Gekko
m = GEKKO()
m.options.IMODE=2
# coefficients
c = [m.FV(value=0) for i in range(4)]
x = m.Param(value=xm)
y = m.CV(value=ym)
y.FSTATUS = 1
# polynomial model
m.Equation(y==c[0]+c[1]*x+c[2]*x**2+c[3]*x**3)

# linear regression
c[0].STATUS=1
c[1].STATUS=1
```

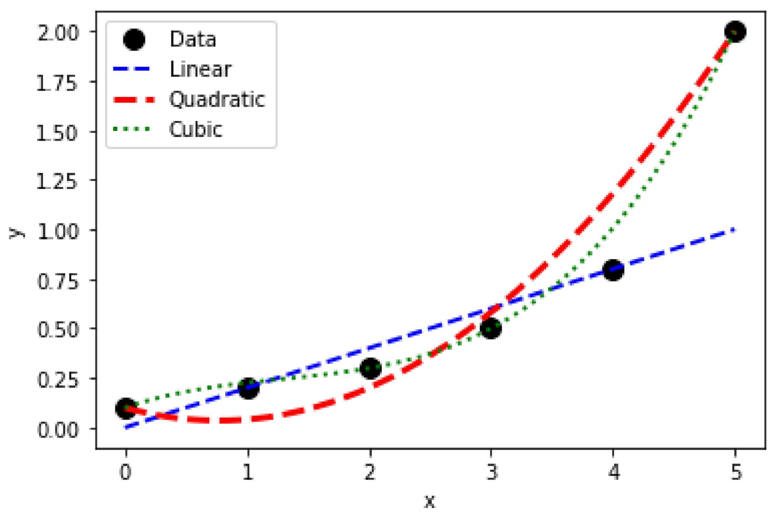
```
m.solve(disp=False)
p1 = [c[1].value[0],c[0].value[0]]

# quadratic
c[2].STATUS=1
m.solve(disp=False)
p2 = [c[2].value[0],c[1].value[0],c[0].value[0]]

# cubic
c[3].STATUS=1
m.solve(disp=False)
p3 = [c[3].value[0],c[2].value[0],c[1].value[0],c[0].value[0]]

# plot fit
plt.plot(xm,ym,'ko',markersize=10)
xp = np.linspace(0,5,100)
plt.plot(xp,np.polyval(p1,xp),'b--',linewidth=2)
plt.plot(xp,np.polyval(p2,xp),'r--',linewidth=3)
plt.plot(xp,np.polyval(p3,xp),'g:',linewidth=2)
plt.legend(['Data','Linear','Quadratic','Cubic'],loc='best')
plt.xlabel('x')
plt.ylabel('y')
```

Text(0, 0.5, 'y')




```
from gekko import GEKKO
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# generate training data
x = np.linspace(0.0,2*np.pi,20)
y = np.sin(x)
```

```
# option for fitting function
select = False # True / False
if select:
    # Size with cosine function
    nin = 1  # inputs
    n1 = 1   # hidden layer 1 (linear)
    n2 = 1   # hidden layer 2 (nonlinear)
    n3 = 1  # hidden layer 3 (linear)
    nout = 1 # outputs
else:
    # Size with hyperbolic tangent function
    nin = 1  # inputs
    n1 = 2   # hidden layer 1 (linear)
    n2 = 3   # hidden layer 2 (nonlinear)
    n3 = 2   # hidden layer 3 (linear)
    nout = 1 # outputs

# Initialize gekko
train = GEKKO()
test = GEKKO()

model = [train,test]

for m in model:
    # input(s)
    m.inpt = m.Param()

    # layer 1
    m.w1 = m.Array(m.FV, (nin,n1))
    m.l1 = [m.Intermediate(m.w1[0,i]*m.inpt) for i in range(n1)]

    # layer 2
    m.w2a = m.Array(m.FV, (n1,n2))
    m.w2b = m.Array(m.FV, (n1,n2))
    if select:
        m.l2 = [m.Intermediate(sum([m.cos(m.w2a[j,i]+m.w2b[j,i]*m.l1[j]) \
                            for j in range(n1)])) for i in range(n2)]
    else:
        m.l2 = [m.Intermediate(sum([m.tanh(m.w2a[j,i]+m.w2b[j,i]*m.l1[j]) \
                            for j in range(n1)])) for i in range(n2)]

    # layer 3
    m.w3 = m.Array(m.FV, (n2,n3))
    m.l3 = [m.Intermediate(sum([m.w3[j,i]*m.l2[j] \
            for j in range(n2)])) for i in range(n3)]

    # output(s)
    m.outpt = m.CV()
    m.Equation(m.outpt==sum([m.l3[i] for i in range(n3)]))

    # flatten matrices
```

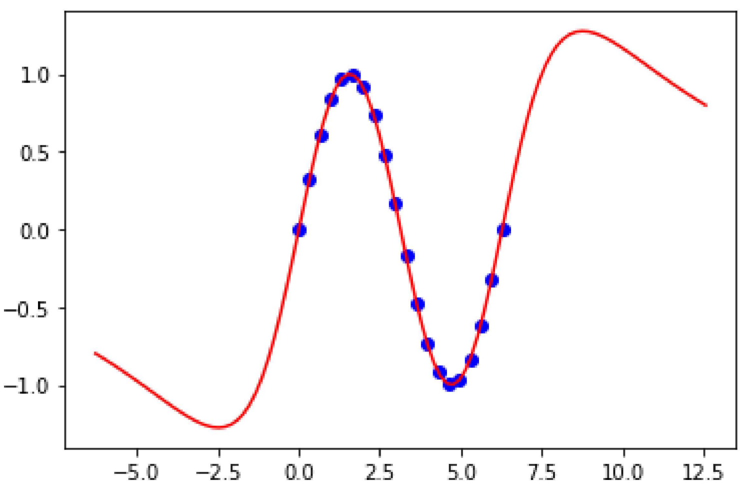
```
    m.w1 = m.w1.flatten()
    m.w2a = m.w2a.flatten()
    m.w2b = m.w2b.flatten()
    m.w3 = m.w3.flatten()

# Fit parameter weights
m = train
m.inpt.value=x
m.outpt.value=y
m.outpt.FSTATUS = 1
for i in range(len(m.w1)):
    m.w1[i].FSTATUS=1
    m.w1[i].STATUS=1
    m.w1[i].MEAS=1.0
for i in range(len(m.w2a)):
    m.w2a[i].STATUS=1
    m.w2b[i].STATUS=1
    m.w2a[i].FSTATUS=1
    m.w2b[i].FSTATUS=1
    m.w2a[i].MEAS=1.0
    m.w2b[i].MEAS=0.5
for i in range(len(m.w3)):
    m.w3[i].FSTATUS=1
    m.w3[i].STATUS=1
    m.w3[i].MEAS=1.0
m.options.IMODE = 2
m.options.SOLVER = 3
m.options.EV_TYPE = 2
m.solve(disp=False)

# Test sample points
m = test
for i in range(len(m.w1)):
    m.w1[i].MEAS=train.w1[i].NEWVAL
    m.w1[i].FSTATUS = 1
    print('w1['+str(i)+']: '+str(m.w1[i].MEAS))
for i in range(len(m.w2a)):
    m.w2a[i].MEAS=train.w2a[i].NEWVAL
    m.w2b[i].MEAS=train.w2b[i].NEWVAL
    m.w2a[i].FSTATUS = 1
    m.w2b[i].FSTATUS = 1
    print('w2a['+str(i)+']: '+str(m.w2a[i].MEAS))
    print('w2b['+str(i)+']: '+str(m.w2b[i].MEAS))
for i in range(len(m.w3)):
    m.w3[i].MEAS=train.w3[i].NEWVAL
    m.w3[i].FSTATUS = 1
    print('w3['+str(i)+']: '+str(m.w3[i].MEAS))
m.inpt.value=np.linspace(-2*np.pi,4*np.pi,100)
m.options.IMODE = 2
m.options.SOLVER = 3
m.solve(disp=False)
```

```
plt.figure()
plt.plot(x,y,'bo')
plt.plot(test.inpt.value,test.outpt.value,'r-')
plt.show()
```

```
w1[0]: 0.72039948257
w1[1]: 0.97020409045
w2a[0]: -2.9352864777
w2b[0]: 0.75977201785
w2a[1]: -0.50374820216
w2b[1]: 0.759772019
w2a[2]: -0.34909522023
w2b[2]: 0.15424839056
w2a[3]: -3.9587870618
w2b[3]: 0.63464219684
w2a[4]: 0.090025946126
w2b[4]: 0.63464219772
w2a[5]: -1.3759710865
w2b[5]: 0.45143615048
w3[0]: 0.75546187561
w3[1]: 0.75546255577
w3[2]: 0.75547569468
w3[3]: 0.75544873479
w3[4]: -1.4721970635
w3[5]: -1.4721970366
```



6s completed at 10:15 PM