In [16]:

```
import numpy as np
import mltools as ml
import matplotlib.pyplot as plt
data = np.genfromtxt("data/curve80.txt",delimiter=None)
```

problem 1 1

In [40]:

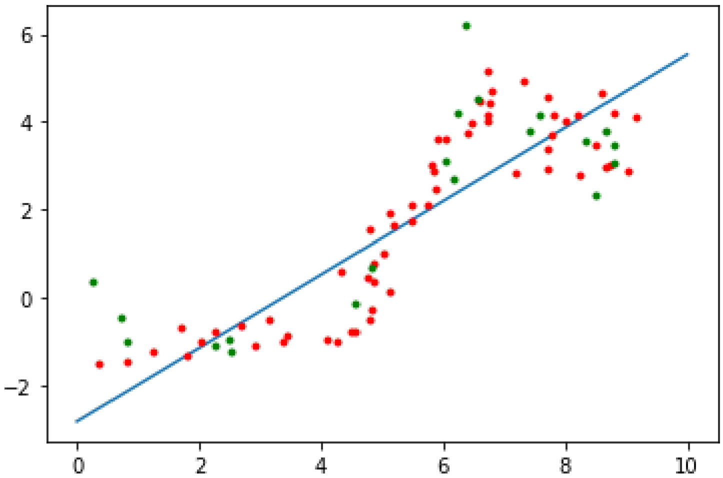
```
#problem 1  1
X = data[:,0]
X = np.atleast_2d(X).T
Y = data[:,1]
Xtr,Xte,Ytr,Yte = ml.splitData(X,Y,0.75)


print(Xtr.shape, Xte.shape, Ytr.shape, Yte.shape)
```

```
(60, 1) (20, 1) (60,) (20,)
```

problem 1 2

In [51]:

```
#problem 1  2a

lr = ml.linear.linearRegress( Xtr, Ytr )
xs = np.linspace(0,10,200)
xs = xs[:,np.newaxis]
ys = lr.predict( xs )

plt.plot(xs,ys,Xtr,Ytr,'r.',Xte,Yte,'g.')
plt.show()

#red dots Xtr,Ytr ,blue line prediction function, green dots Xte,Yte
```



In [42]:

```
print(lr.theta)
# 2b   they match the plot because y intercept is below -2 and slope upward is positiv
```

```
[[-2.82765049  0.83606916]]
```

In [43]:

```
print(lr.mse(Xtr,Ytr))
print(lr.mse(Xte,Yte))
# 2c    mean squared error of the predictions on the training is 1.127711955609391,  c
```

```
1.127711955609391
2.242349203010126
```

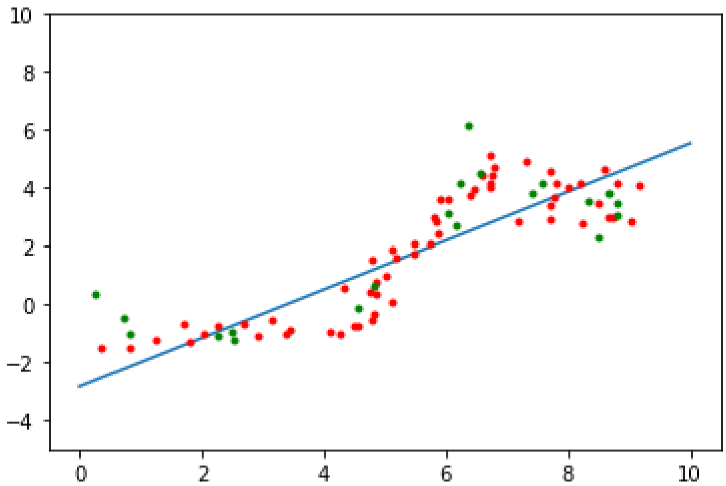
problem 1 3

3c I would recommand the 10 as ploynomial degree because it has the lowest green line, in the data I print err2[4] which is 0.609060074952002 is lower than others.

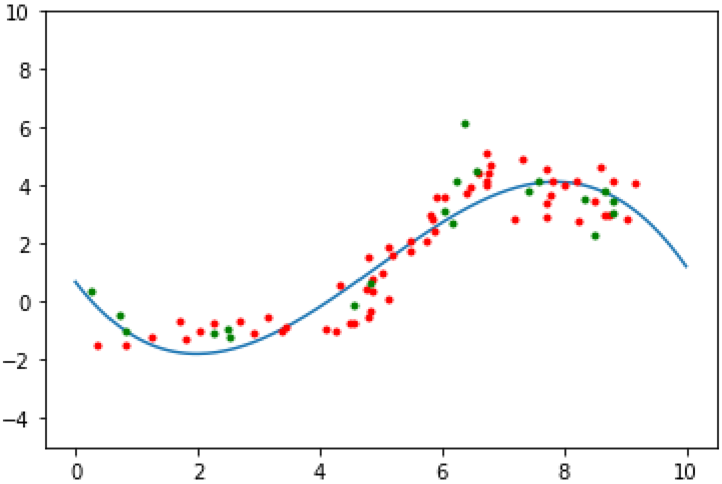
In [53]:

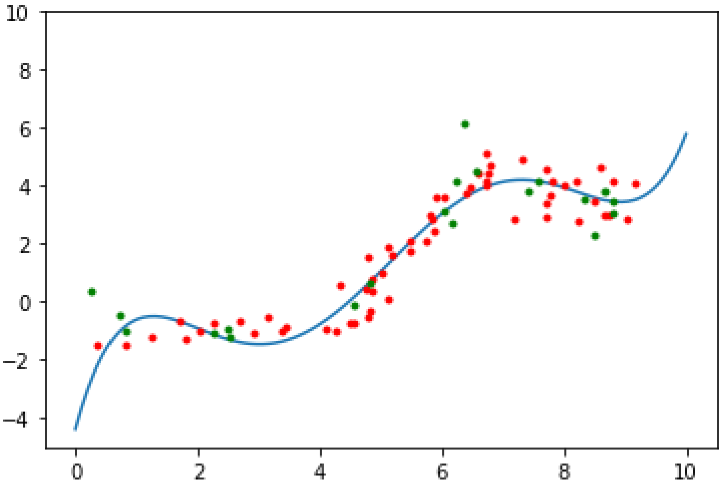
```
err1 = np.zeros((6,))
err2 = np.zeros((6,))

count=0
for i in [1,3,5,7,10,18]:

    XtrP = ml.transforms.fpoly(Xtr, i, bias=False)
    XtrP,params = ml.transforms.rescale(XtrP)
    print('----------')


    lr = ml.linear.linearRegress(XtrP,Ytr)
    xsP = ml.transforms.fpoly(xs,i, bias=False)
    xsP,params = ml.transforms.rescale(xsP,params)

    #print(xsP)
    ys = lr.predict(xsP)
    plt.plot(xs,ys,Xtr,Ytr,'r.',Xte,Yte,'g.')
    plt.ylim(-5,10)
    plt.show()

    XteP = ml.transforms.fpoly(Xte, i, bias=False)

    XteP,params = ml.transforms.rescale(XteP,params)

    err1[count] = lr.mse(XtrP,Ytr)
    err2[count] = lr.mse(XteP,Yte)
    count+=1



plt.semilogy([1,3,5,7,10,18],err1,'y')
plt.semilogy([1,3,5,7,10,18],err2,'b')
plt.show()
print(err2[4])

print(err2[3])
#red dots Xtr,Ytr ,blue line prediction function, green dots Xte,Yte
#yellow line is training , blue line is test error
```

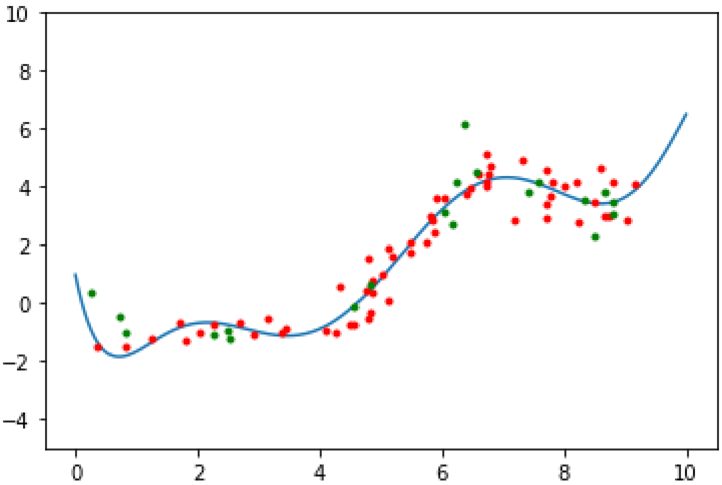
```
----------
```

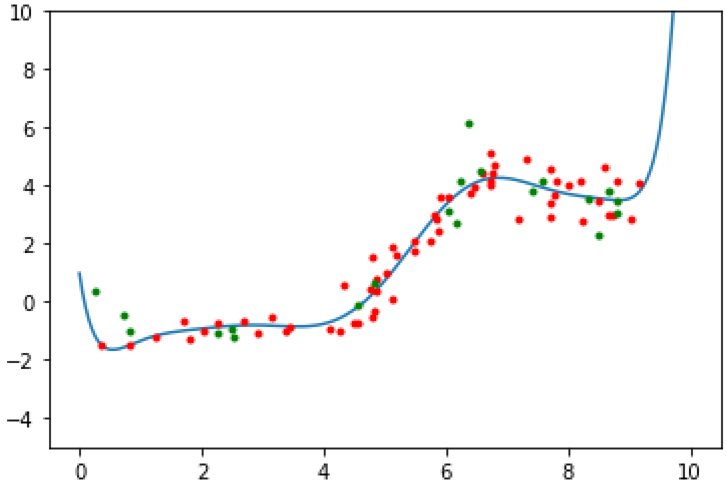


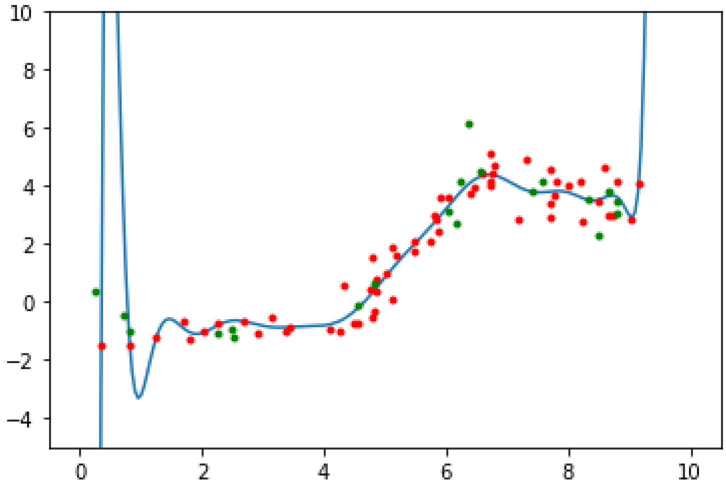

```
----------
```

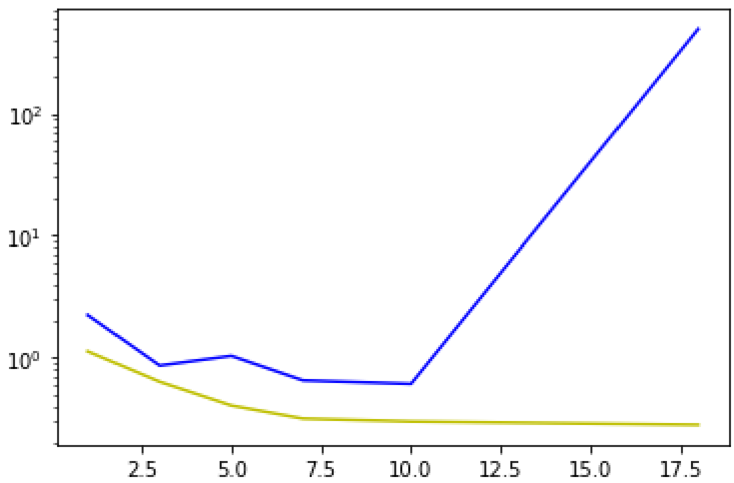

10
8
6
4
2
0
−2
−4
0 2 4 6 8 10


10
8
6
4
2
0
−2
−4
0 2 4 6 8 10


10
8
6
4
2
0
−2
−4
0 2 4 6 8 10








```
0.609060074952002
0.6502246079664455
```

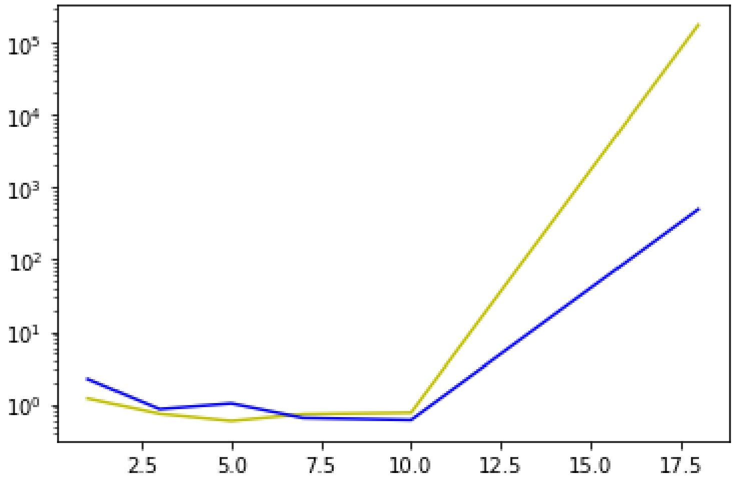
problem 2 .1

In [48]:

```
nFolds = 5;
d=1

def crossv(nFolds,d,Xtr,Ytr):
    J = np.zeros(nFolds)
    for iFold in range(nFolds):
        Xti,Xvi,Yti,Yvi = ml.crossValidate(Xtr,Ytr,nFolds,iFold)


        Xtip = ml.transforms.fpoly(Xti, d, bias=False)
        Xtip,params = ml.transforms.rescale(Xtip)
```

```python
        lr = ml.linear.linearRegress(Xtip,Yti)
        xviP = ml.transforms.fpoly(Xvi,d, bias=False)
        xviP,params = ml.transforms.rescale(xviP,params)
        J[iFold] = lr.mse(xviP,Yvi)

    print(np.mean(J))
    return np.mean(J)


# crossv(5,10)

err3 = np.zeros((6,))
count=0
for i in [1,3,5,7,10,18]:
    err3[count] = crossv(nFolds,i,Xtr,Ytr)
    count+=1


plt.semilogy([1,3,5,7,10,18],err3,'y')
plt.semilogy([1,3,5,7,10,18],err2,'b')
plt.show()

#yellow line is the cross validation and blue line is test error from previous
```

```
1.2118626629641986
0.7429005752051646
0.5910703726407939
0.7335637831327386
0.7677056830803561
176227.52604921878
```



problem 2 .2

It has samiliar trend between five-fold cross validation compare to mse test error, it was relatively accurate at low degree, and create high degree polynomial, then become over fitting.We can also see has a higher error rate in high degree.

problem 2 .3

I would choose 5 degree with 0.5910703726407939 which is the lowest of yellow line.

problem 2 .4

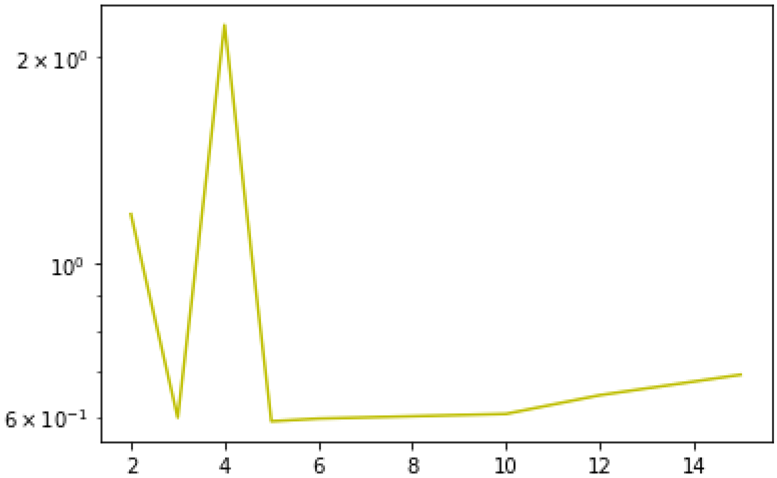
It was first high and fluctuates in the begining, a significant drop on five then it starts gradually increase error.The reason is because n folds divede data into parts, when n is small, we have less data to train and compare, it would have bad results. As n grows, the training set is growing larger so it would have better result.

In [49]:

```python
err4 = np.zeros((8,))
count=0
```

```
for i in [2,3,4,5,6,10,12,15]:
    err4[count] = crossv(i,5,Xtr,Ytr)
    count+=1

plt.semilogy([2,3,4,5,6,10,12,15],err4,'y')

plt.show()
#yellow line is cross validation error
```

```
1.1795458641319012
0.5984555010979001
2.219526156064467
0.5910703726407939
0.5963380050012014
0.6058256908836851
0.6448758386950425
0.690566966174373
```



Problem 3

I have done this homework by myself.