Q1) Final answer page 10 – 12
Q2) Final answer Page 13 -14
Q3) Final answer Page 15 - 24

In [1]:
```
import pandas as pd
from pandas import DataFrame
from sets import Set
import numpy as np
import random
import matplotlib.pyplot as plt
from numpy import linalg as LA
import statsmodels.formula.api as smf
%matplotlib inline

#read in
data = pd.read_csv("D3.csv", index_col=None, header=None)
```

In [2]:
```
data
```

	0	1	2	3
0	0.4583	0.6838	0.7048	0.6163
1	0.2960	0.4277	0.4469	0.3412
2	0.7317	0.8338	0.7625	1.2294
3	0.7521	0.9045	0.9324	1.1811
4	0.2971	0.4406	0.3523	0.4627
5	0.5690	0.7744	0.7080	0.8496
6	0.5560	0.6993	0.7811	0.8134
7	0.8944	0.9993	0.8953	1.7522
8	0.9532	0.9891	1.0171	1.8413
9	0.5852	0.6853	0.7713	0.9571
10	0.2460	0.3076	0.3997	0.4428
11	0.7274	0.8168	0.8868	1.1470
12	0.6883	0.8762	0.8019	1.0934
13	0.0299	0.0536	0.2728	0.1728
14	0.5465	0.6068	0.6054	0.8482
15	0.3535	0.5987	0.4213	0.4845
16	0.5786	0.7974	0.6678	0.9648

17	0.9023	1.0985	1.0148	1.8765
18	0.0744	0.0961	0.1261	0.2215
19	0.2444	0.4324	0.3031	0.3703
20	0.8804	1.1202	1.0502	1.6452
21	0.4072	0.4359	0.4780	0.4926
22	0.2974	0.4120	0.4351	0.3366
23	0.9637	0.9747	1.1227	2.1088
24	0.9354	1.1511	1.0917	1.8354
25	0.7469	0.8616	0.7649	1.2555
26	0.5351	0.5608	0.6869	0.8578
27	0.9431	1.1252	0.9453	2.0060
28	0.9929	1.0581	1.0494	2.2145
29	0.7302	0.8999	0.8349	1.2542
…	…	…	…	…
70	0.5770	0.8100	0.6721	0.9221
71	0.3503	0.3505	0.5038	0.6314
72	0.2180	0.2646	0.2760	0.4502
73	0.4936	0.5679	0.6447	0.8384
74	0.9606	1.0949	1.1772	1.8519
75	0.0479	0.1713	0.1343	0.1089
76	0.3572	0.3614	0.3608	0.4457
77	0.8382	1.0503	0.9573	1.6205
78	0.4292	0.5745	0.5343	0.7358
79	0.1792	0.3630	0.2189	0.3386
80	0.1824	0.3248	0.3863	0.3530
81	0.3378	0.3779	0.4452	0.3914
82	0.4073	0.4644	0.4095	0.5683
83	0.7805	1.0157	0.9044	1.3310
84	0.6432	0.7363	0.6515	1.1121
85	0.8862	0.9477	0.9715	1.6006

86	0.6251	0.7523	0.7606	0.9568
87	0.4605	0.5385	0.6499	0.6222
88	0.4725	0.7163	0.6730	0.6312
89	0.1406	0.1990	0.2983	0.1853
90	0.1904	0.2297	0.3409	0.2029
91	0.9607	1.1051	0.9952	1.9249
92	0.3330	0.4887	0.4263	0.4096
93	0.2491	0.3914	0.4092	0.4393
94	0.3888	0.6010	0.5129	0.4680
95	0.6411	0.7634	0.7966	0.9876
96	0.3298	0.4415	0.3770	0.5578
97	0.7259	0.7414	0.9487	1.2807
98	0.8300	0.9700	0.9045	1.6492
99	0.9617	1.1835	1.1587	1.8941

100 rows × 4 columns

In [3]:
```
#cross validation data
sel = 10
samp = Set()
while len(samp)<sel:
    samp.add(random.randint(0,99))

l = list(samp)

test = data.iloc[l ,:]
tdata = data.drop(data.index[l])
```

In [4]:
```
data
#full data
```

	0	1	2	3
0	0.4583	0.6838	0.7048	0.6163
1	0.2960	0.4277	0.4469	0.3412

2	0.7317	0.8338	0.7625	1.2294
3	0.7521	0.9045	0.9324	1.1811
4	0.2971	0.4406	0.3523	0.4627
5	0.5690	0.7744	0.7080	0.8496
6	0.5560	0.6993	0.7811	0.8134
7	0.8944	0.9993	0.8953	1.7522
8	0.9532	0.9891	1.0171	1.8413
9	0.5852	0.6853	0.7713	0.9571
10	0.2460	0.3076	0.3997	0.4428
11	0.7274	0.8168	0.8868	1.1470
12	0.6883	0.8762	0.8019	1.0934
13	0.0299	0.0536	0.2728	0.1728
14	0.5465	0.6068	0.6054	0.8482
15	0.3535	0.5987	0.4213	0.4845
16	0.5786	0.7974	0.6678	0.9648
17	0.9023	1.0985	1.0148	1.8765
18	0.0744	0.0961	0.1261	0.2215
19	0.2444	0.4324	0.3031	0.3703
20	0.8804	1.1202	1.0502	1.6452
21	0.4072	0.4359	0.4780	0.4926
22	0.2974	0.4120	0.4351	0.3366
23	0.9637	0.9747	1.1227	2.1088
24	0.9354	1.1511	1.0917	1.8354
25	0.7469	0.8616	0.7649	1.2555
26	0.5351	0.5608	0.6869	0.8578
27	0.9431	1.1252	0.9453	2.0060
28	0.9929	1.0581	1.0494	2.2145
29	0.7302	0.8999	0.8349	1.2542
...	...	...	...	...
70	0.5770	0.8100	0.6721	0.9221
71	0.3503	0.3505	0.5038	0.6314

72	0.2180	0.2646	0.2760	0.4502
73	0.4936	0.5679	0.6447	0.8384
74	0.9606	1.0949	1.1772	1.8519
75	0.0479	0.1713	0.1343	0.1089
76	0.3572	0.3614	0.3608	0.4457
77	0.8382	1.0503	0.9573	1.6205
78	0.4292	0.5745	0.5343	0.7358
79	0.1792	0.3630	0.2189	0.3386
80	0.1824	0.3248	0.3863	0.3530
81	0.3378	0.3779	0.4452	0.3914
82	0.4073	0.4644	0.4095	0.5683
83	0.7805	1.0157	0.9044	1.3310
84	0.6432	0.7363	0.6515	1.1121
85	0.8862	0.9477	0.9715	1.6006
86	0.6251	0.7523	0.7606	0.9568
87	0.4605	0.5385	0.6499	0.6222
88	0.4725	0.7163	0.6730	0.6312
89	0.1406	0.1990	0.2983	0.1853
90	0.1904	0.2297	0.3409	0.2029
91	0.9607	1.1051	0.9952	1.9249
92	0.3330	0.4887	0.4263	0.4096
93	0.2491	0.3914	0.4092	0.4393
94	0.3888	0.6010	0.5129	0.4680
95	0.6411	0.7634	0.7966	0.9876
96	0.3298	0.4415	0.3770	0.5578
97	0.7259	0.7414	0.9487	1.2807
98	0.8300	0.9700	0.9045	1.6492
99	0.9617	1.1835	1.1587	1.8941

100 rows × 4 columns

```
In [5]: test
        #Test Data
```

	0	1	2	3
99	0.9617	1.1835	1.1587	1.8941
68	0.9968	1.1830	1.1533	2.0711
38	0.2851	0.5135	0.4872	0.4692
39	0.6396	0.6811	0.6868	0.9443
11	0.7274	0.8168	0.8868	1.1470
77	0.8382	1.0503	0.9573	1.6205
46	0.6331	0.8574	0.6413	0.9542
16	0.5786	0.7974	0.6678	0.9648
81	0.3378	0.3779	0.4452	0.3914
30	0.6795	0.8793	0.6882	1.1521

```
In [6]: tdata
        #Training Data
```

0	1	2	3	
0	0.4583	0.6838	0.7048	0.6163
1	0.2960	0.4277	0.4469	0.3412
2	0.7317	0.8338	0.7625	1.2294
3	0.7521	0.9045	0.9324	1.1811
4	0.2971	0.4406	0.3523	0.4627
5	0.5690	0.7744	0.7080	0.8496
6	0.5560	0.6993	0.7811	0.8134
7	0.8944	0.9993	0.8953	1.7522
8	0.9532	0.9891	1.0171	1.8413
9	0.5852	0.6853	0.7713	0.9571
10	0.2460	0.3076	0.3997	0.4428

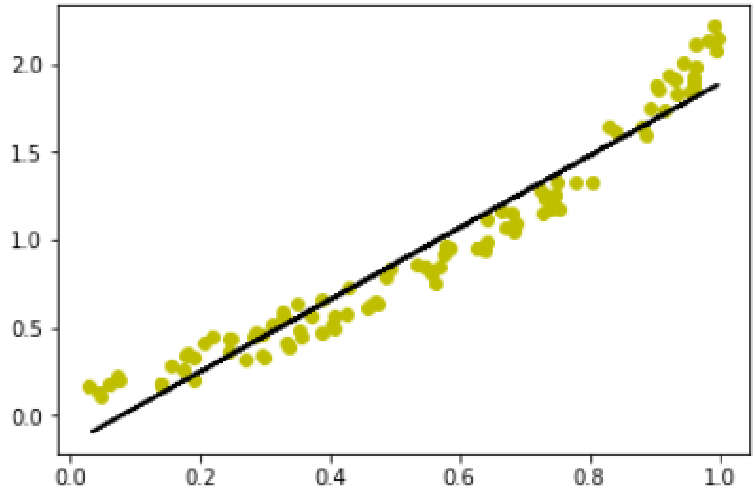
12	0.6883	0.8762	0.8019	1.0934
13	0.0299	0.0536	0.2728	0.1728
14	0.5465	0.6068	0.6054	0.8482
15	0.3535	0.5987	0.4213	0.4845
17	0.9023	1.0985	1.0148	1.8765
18	0.0744	0.0961	0.1261	0.2215
19	0.2444	0.4324	0.3031	0.3703
20	0.8804	1.1202	1.0502	1.6452
21	0.4072	0.4359	0.4780	0.4926
22	0.2974	0.4120	0.4351	0.3366
23	0.9637	0.9747	1.1227	2.1088
24	0.9354	1.1511	1.0917	1.8354
25	0.7469	0.8616	0.7649	1.2555
26	0.5351	0.5608	0.6869	0.8578
27	0.9431	1.1252	0.9453	2.0060
28	0.9929	1.0581	1.0494	2.2145
29	0.7302	0.8999	0.8349	1.2542
31	0.3889	0.5070	0.6155	0.6616
32	0.2702	0.5009	0.3327	0.3264
...	...	...	...	...
66	0.5621	0.7441	0.6466	0.7546
67	0.9143	1.1580	1.0035	1.7364
69	0.3725	0.6067	0.6148	0.5720
70	0.5770	0.8100	0.6721	0.9221
71	0.3503	0.3505	0.5038	0.6314
72	0.2180	0.2646	0.2760	0.4502
73	0.4936	0.5679	0.6447	0.8384
74	0.9606	1.0949	1.1772	1.8519
75	0.0479	0.1713	0.1343	0.1089
76	0.3572	0.3614	0.3608	0.4457
78	0.4292	0.5745	0.5343	0.7358

79	0.1792	0.3630	0.2189	0.3386
80	0.1824	0.3248	0.3863	0.3530
82	0.4073	0.4644	0.4095	0.5683
83	0.7805	1.0157	0.9044	1.3310
84	0.6432	0.7363	0.6515	1.1121
85	0.8862	0.9477	0.9715	1.6006
86	0.6251	0.7523	0.7606	0.9568
87	0.4605	0.5385	0.6499	0.6222
88	0.4725	0.7163	0.6730	0.6312
89	0.1406	0.1990	0.2983	0.1853
90	0.1904	0.2297	0.3409	0.2029
91	0.9607	1.1051	0.9952	1.9249
92	0.3330	0.4887	0.4263	0.4096
93	0.2491	0.3914	0.4092	0.4393
94	0.3888	0.6010	0.5129	0.4680
95	0.6411	0.7634	0.7966	0.9876
96	0.3298	0.4415	0.3770	0.5578
97	0.7259	0.7414	0.9487	1.2807
98	0.8300	0.9700	0.9045	1.6492

90 rows × 4 columns

In [7]:
```
#linear regression scatter
fit = np.polyfit(data[0],data[3],1)
fit_fn = np.poly1d(fit)
plt.plot(data[0],data[3], 'yo', data[0], fit_fn(data[0]), '--k')
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0xe0a8fd0>,
 <matplotlib.lines.Line2D at 0xe0be0b8>]
```



Problem 1 A)

Y= 2.048 x - 0.1612

X	Y
0.3	1.88713478225
0.5	3.93542391976
0.8	5.98371305727

Problem 1 Part b)

I randomly selected 10 points for test data and left the other 90 for training data. After calculating using regression on the training data, I was left with M= 2.039x – 0.1512 and I found the generalization error of **0.0192996468815**

Problem 1 C)

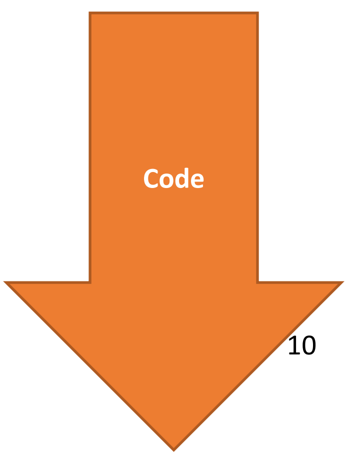
Model	0.3	0.5	0.8
$1.655\ x^2 + 0.2487\ x + 0.1958$	2.0993594371	7.31257151504	15.8354446974
$0.1758\ x^4 + 1.077\ x^3 - 0.3761\ x^2 + 1.177\ x + 0.09765$	2.15127918332	12.3749967934	43.5580092174
$-0.9501\ x^5 + 2.631\ x^4 - 1.209\ x^3 + 0.5445\ x^2 + 1.028\ x + 0.104$	2.14860875989	6.35831777391	-42.3161845583

Problem 1 D)

For the cross validation I used the same test and training data from 1b. However, this time I calculated the models using the polynomial regressions. The generalization error was then calculated using these models.

Polynomial	Model	Error
2	$1.651\ x^2 + 0.2603\ x + 0.1934$	0.00489796576083
3	$1.607\ x^3 - 0.8535\ x^2 + 1.324\ x + 0.08702$	0.00568684383895
4	$0.9041\ x^4 - 0.2457\ x^3 + 0.3792\ x^2 + 1.032\ x + 0.1044$	0.00593551475353
5	$1.484\ x^5 - 2.895\ x^4 + 3.253\ x^3 - 1.01\ x^2 + 1.253\ x + 0.09508$	0.00603435414003

Problem 1 Part a and c



In [8]:
```
#Problem 1a+1c
#polynomial regression 1-5
for i in range(1,6):
    fit = np.polyfit(data[0],data[3],i)
    fit_fn = np.poly1d(fit)
    print "polynomial: ",i
    print fit_fn

    print "0.3: ", fit_fn(1)
    print "0.5: ", fit_fn(2)
    print "0.8: ", fit_fn(3)
```

```
polynomial:  1

2.048 x - 0.1612
0.3:  1.88713478225
0.5:  3.93542391976
0.8:  5.98371305727
polynomial:  2
       2
1.655 x + 0.2487 x + 0.1958
0.3:  2.0993594371
0.5:  7.31257151504
0.8:  15.8354446974
polynomial:  3
       3          2
1.439 x - 0.6187 x + 1.235 x + 0.09416
0.3:  2.14957133621
0.5:  11.60261107
0.8:  37.0882251144
polynomial:  4
        4         3          2
0.1758 x + 1.077 x - 0.3761 x + 1.177 x + 0.09765
0.3:  2.15127918332
0.5:  12.3749967934
0.8:  43.5580092174
polynomial:  5
         5         4         3          2
-0.9501 x + 2.631 x - 1.209 x + 0.5445 x + 1.028 x + 0.104
0.3:  2.14860875989
0.5:  6.35831777391
0.8:  -42.3161845583
```

Problem 1 Part b and d

Code

Part b)

In [9]:
```
#Problem 1b+1d
#polynomial regression 1-5 on Training Data and Test Data
for i in range(1,6):
    fit = np.polyfit(tdata[0],tdata[3],i)
    fit_fn = np.poly1d(fit)
    print "polynomial: ",i
    print fit_fn
    print ""

    M = 0;
    for row in test.iterrows():
        x = fit_fn(row[1][0])-row[1][3]
        M += x*x

    print "(M(x)-y)^2: ", M/sel
    print ""
```

```
polynomial:  1

2.039 x - 0.1512

(M(x)-y)^2:  0.0192966468815

polynomial:  2
       2
1.651 x + 0.2603 x + 0.1934

(M(x)-y)^2:  0.00489796576083

polynomial:  3
       3          2
1.607 x - 0.8535 x + 1.324 x + 0.08702

(M(x)-y)^2:  0.00568684383895

polynomial:  4
        4          3          2
0.9041 x - 0.2457 x + 0.3792 x + 1.032 x + 0.1044

(M(x)-y)^2:  0.00593551475353

polynomial:  5
       5         4         3        2
1.484 x - 2.895 x + 3.253 x - 1.01 x + 1.253 x + 0.09508

(M(x)-y)^2:  0.00603435414003
```

Problem 2 Part a)

$Y = 2.2850x_1 - 0.2913x_2 + 0.0667x_3 - 0.1360$

(x_1, x_2, x_3)	y
(0.3,0.4,0.1)	0.439632732168
(0.5,0.2,0.4)	0.974889919898
(0.8,0.2,0.7)	1.68039532756

Problem 2 Part b)

For the cross validation I used the same test and training data from part 1. However this time I used all the variables from the training data to acquire the linear regression model. The generalization error was then calculated using all of the variables from the testing data.

$Y = 2.529848\, x_1 - 0.411109\, x_2 - 0.068430\, x_3 - 0.094904$

$(M(x)-y)^2$: 0.0241688387617

In [10]:
```
#basic multivariable regression
lm = smf.ols(formula='data[3] ~ data[0] + data[1] + data[2]', data = data).fit()
lm.summary()
```

Out[10]:

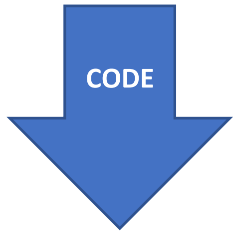
OLS Regression Results

Dep. Variable:	data[3]	**R-squared:**	0.945
Model:	OLS	**Adj. R-squared:**	0.943
Method:	Least Squares	**F-statistic:**	547.8
Date:	Sun, 29 Oct 2017	**Prob (F-statistic):**	3.12e-60
Time:	20:07:55	**Log-Likelihood:**	54.682
No. Observations:	100	**AIC:**	-101.4
Df Residuals:	96	**BIC:**	-90.94
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-0.1360	0.046	-2.935	0.004	-0.228	-0.044
data[0]	2.2850	0.297	7.702	0.000	1.696	2.874
data[1]	-0.2913	0.200	-1.454	0.149	-0.689	0.106
data[2]	0.0667	0.212	0.314	0.754	-0.355	0.488

Omnibus:	19.445	**Durbin-Watson:**	1.869
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5.991
Skew:	0.281	**Prob(JB):**	0.0500
Kurtosis:	1.940	**Cond. No.**	38.4

Problem 2 Part a and b



In [11]:
```
#Least squares helper function uses current lm params
def leastSquares(x1, x2, x3):
    return  lm.params[0] + lm.params[1] * x1 + lm.params[2] * x2 + lm.params[3] * x3
```

In [12]:
```
#Problem 2a
print leastSquares(0.3,0.4,0.1)
print leastSquares(0.5,0.2,0.4)
print leastSquares(0.8,0.2,0.7)
```

```
0.439632732168
0.974889919898
1.68039532756
```

In [13]:
```
#Problem 2b
lm = smf.ols(formula='tdata[3] ~ tdata[0] + tdata[1] + tdata[2]', data = tdata).fit()
M = 0.0;
print "parameters:"
print lm.params
print ""
for row in test.iterrows():
    x = leastSquares(row[1][0], row[1][1], row[1][2]) - row[1][3]
    M += x*x
print "(M(x)-y)^2: ", M/sel
```

```
parameters:
Intercept   -0.094904
tdata[0]     2.529848
tdata[1]    -0.411109
tdata[2]    -0.068430
dtype: float64

(M(x)-y)^2:  0.0241688387617
```

Problem 3 Function1:

$$F1(x,y) = (x-5)^2+(y+2)^2$$

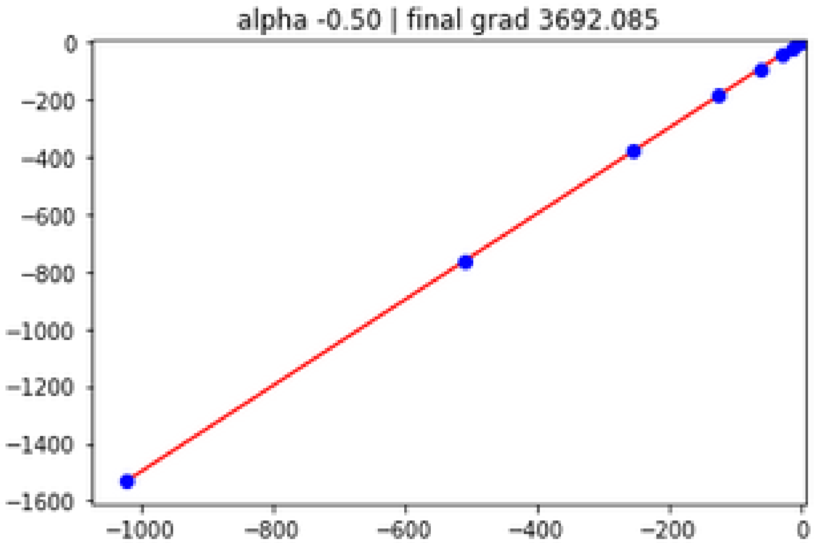
In [14]:
```
#Problem 3
#Function 1
def func(x,y):
    return (x-5)**2 + (y+2)**2
def func_grad(vx,vy):
    dfdx = 2.0*vx - 4.0
    dfdy = 2.0*vy - 6.0
    return np.array([dfdx,dfdy])

#prepare for contour plot
xlist = np.linspace(0, 5, 26)
ylist = np.linspace(0, 5, 26)
x, y = np.meshgrid(xlist, ylist)
z = func(x,y)
lev = np.linspace(0,20,21)

#iterate location
v_init = np.array([0,0])
num_iter = 10
values = np.zeros([num_iter,2])
for alpha in [.5,-.5]:
    values[0,:] = v_init
    v = v_init
    print "iter: ", 1, "value: ", v, "func: ", func(v[0],v[1])
    # actual gradient descent algorithm
    for i in range(1,num_iter):
        v = v - alpha * func_grad(v[0],v[1])
        values[i,:] = v
        print "iter: ", i+1, "v: ", v, "func: ", func(v[0],v[1])

    #plotting
    plt.contour(x,y,z,levels=lev)
    plt.plot(values[:,0],values[:,1],'r-')
    plt.plot(values[:,0],values[:,1],'bo')
    grad_norm = LA.norm(func_grad(v[0],v[1]))
    title = "alpha %0.2f | final grad %0.3f" % (alpha,grad_norm)
    plt.title(title)
```

```
iter:  1 value:  [0 0] func:  29
iter:  2 v:  [ 2.  3.] func:  34.0
iter:  3 v:  [ 2.  3.] func:  34.0
iter:  4 v:  [ 2.  3.] func:  34.0
iter:  5 v:  [ 2.  3.] func:  34.0
iter:  6 v:  [ 2.  3.] func:  34.0
iter:  7 v:  [ 2.  3.] func:  34.0
iter:  8 v:  [ 2.  3.] func:  34.0
iter:  9 v:  [ 2.  3.] func:  34.0
iter:  10 v:  [ 2.  3.] func:  34.0
iter:  1 value:  [0 0] func:  29
iter:  2 v:  [-2. -3.] func:  50.0
iter:  3 v:  [-6. -9.] func:  170.0
iter:  4 v:  [-14. -21.] func:  722.0
iter:  5 v:  [-30. -45.] func:  3074.0
iter:  6 v:  [-62. -93.] func:  12770.0
iter:  7 v:  [-126. -189.] func:  52130.0
iter:  8 v:  [-254. -381.] func:  210722.0
iter:  9 v:  [-510. -765.] func:  847394.0
iter:  10 v:  [-1022. -1533.] func:  3398690.0
```



Problem 3 Function2:

F2(x,y) = (1-(y-4))2+35((x+6)-(y-4)2)2

In [17]:
```
#Problem 3
#Function 1

def func(x,y):
    return (1-(y-4))**2 + 35*((x+6)-(y-4)**2)**2
def func_grad(vx,vy):
    dfdx = 40*vx - 40*(vy - 3)**2 + 120
    dfdy = 2*vy + 20*(-4*vy + 12)*(vx - (vy - 3)**2 + 3) - 8
    return np.array([dfdx,dfdy])

#prepare for contour plot
xlist = np.linspace(0, 5, 26)
ylist = np.linspace(0, 5, 26)
x, y = np.meshgrid(xlist, ylist)
z = func(x,y)
lev = np.linspace(0,20,21)



_init = np.array([0,0])
num_iter = 100
values = np.zeros([num_iter,2])
for alpha in [1.0/8592,1.0/481.69,.5]:
    print alpha
    values[0,:] = v_init
    v = v_init
    print "iter: ", 1, "value: ", v, "func: ", func(v[0],v[1])
    # actual gradient descent algorithm
    for i in range(1,num_iter):
        v = v - alpha * func_grad(v[0],v[1])
        values[i,:] = v
        print "iter: ", i+1, "v: ", v, "func: ", func(v[0],v[1])

    #plotting
    plt.contour(x,y,z,levels=lev)
    plt.plot(values[:,0],values[:,1],'r-')
    plt.plot(values[:,0],values[:,1],'bo')
    grad_norm = LA.norm(func_grad(v[0],v[1]))
    title = "alpha %0.2f | final grad %0.3f" % (alpha,grad_norm)
    plt.title(title)
    file = "gd-%2.0f.pdf" % (alpha*100)
    plt.savefig(file, bbox_inches='tight')
    plt.clf()
    plt.cla()
```

```
0.000116387337058


iter:  1 value:  [0 0] func:  3525
iter:  2 v:  [ 0.02793296  0.16852886] func:  2643.48596102
iter:  3 v:  [ 0.0511606   0.30095749] func:  2060.60991683
iter:  4 v:  [ 0.07087042  0.40821383] func:  1653.82348808
iter:  5 v:  [ 0.08784661  0.4970472 ] func:  1358.23585821
iter:  6 v:  [ 0.10263676  0.57190074] func:  1136.59017022
iter:  7 v:  [ 0.11563969  0.63584352] func:  966.140909989
iter:  8 v:  [ 0.1271555   0.69107697] func:  832.313692687
iter:  9 v:  [ 0.13741607  0.73922893] func:  725.39672674
iter:  10 v:  [ 0.14660448  0.78153372] func:  638.705121322
iter:  11 v:  [ 0.15486793  0.81894725] func:  567.510017194
iter:  12 v:  [ 0.1623266   0.85222324] func:  508.387195633
iter:  13 v:  [ 0.16907994  0.88196532] func:  458.806532969
iter:  14 v:  [ 0.17521119  0.90866349] func:  416.865231055
```

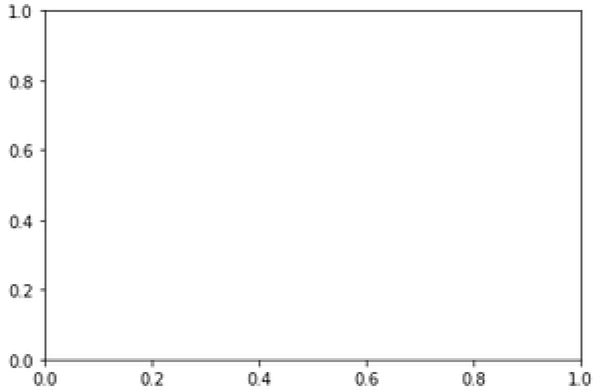
```
iter:  15 v:  [ 0.18079069  0.93272031] func:  381.10980871
iter:  16 v:  [ 0.18587847  0.95447001] func:  350.414519043
iter:  17 v:  [ 0.19052611  0.97419273] func:  323.896573888
iter:  18 v:  [ 0.1947783   0.99212527] func:  300.855933585
iter:  19 v:  [ 0.19867393  1.00846933] func:  280.731830614
iter:  20 v:  [ 0.20224712  1.0233979 ] func:  263.070903959
iter:  21 v:  [ 0.20552789  1.03706032] func:  247.503524715
iter:  22 v:  [ 0.2085428   1.04958622] func:  233.7259886
iter:  23 v:  [ 0.21131548  1.06108874] func:  221.486969051
iter:  24 v:  [ 0.21386698  1.07166709] func:  210.577103802
iter:  25 v:  [ 0.21621614  1.08140868] func:  200.820913042
iter:  26 v:  [ 0.21837991  1.09039081] func:  192.070471208
iter:  27 v:  [ 0.22037352  1.09868213] func:  184.200410979
iter:  28 v:  [ 0.22221074  1.10634378] func:  177.103948692
iter:  29 v:  [ 0.22390405  1.11343045] func:  170.689699695
iter:  30 v:  [ 0.22546476  1.11999115] func:  164.879109556
iter:  31 v:  [ 0.22690317  1.12606995] func:  159.604369013
iter:  32 v:  [ 0.22822863  1.1317066 ] func:  154.806711616
iter:  33 v:  [ 0.22944973  1.136937  ] func:  150.435016099
iter:  34 v:  [ 0.23057428  1.14179368] func:  146.444652948
iter:  35 v:  [ 0.23160946  1.14630615] func:  142.796527786
iter:  36 v:  [ 0.23256184  1.15050126] func:  139.456284297
iter:  37 v:  [ 0.23343747  1.15440348] func:  136.393637124
iter:  38 v:  [ 0.23424188  1.15803512] func:  133.581811209
iter:  39 v:  [ 0.23498021  1.16141661] func:  130.997068694
iter:  40 v:  [ 0.23565716  1.16456661] func:  128.618308185
iter:  41 v:  [ 0.23627708  1.16750227] func:  126.426724079
iter:  42 v:  [ 0.23684398  1.1702393 ] func:  124.405515928
iter:  43 v:  [ 0.23736158  1.17279216] func:  122.539639673
iter:  44 v:  [ 0.23783331  1.17517415] func:  120.815594027
iter:  45 v:  [ 0.23826234  1.17739752] func:  119.221236472
iter:  46 v:  [ 0.23865162  1.17947355] func:  117.745624304
iter:  47 v:  [ 0.23900388  1.18141269] func:  116.378876903
iter:  48 v:  [ 0.23932164  1.18322455] func:  115.112056089
iter:  49 v:  [ 0.23960727  1.18491804] func:  113.937061887
iter:  50 v:  [ 0.23986292  1.1865014 ] func:  112.846541499
iter:  51 v:  [ 0.24009064  1.18798225] func:  111.833809596
iter:  52 v:  [ 0.24029231  1.18936766] func:  110.892778366
iter:  53 v:  [ 0.24046967  1.19066418] func:  110.017895966
iter:  54 v:  [ 0.24062435  1.19187788] func:  109.20409225
iter:  55 v:  [ 0.24075788  1.1930144 ] func:  108.44673081
iter:  56 v:  [ 0.24087166  1.19407897] func:  107.741566492
iter:  57 v:  [ 0.240967    1.19507648] func:  107.084707693
iter:  58 v:  [ 0.24104512  1.19601142] func:  106.472582829
iter:  59 v:  [ 0.24110718  1.19688802] func:  105.901910446
iter:  60 v:  [ 0.24115423  1.19771017] func:  105.369672542
iter:  61 v:  [ 0.24118726  1.19848152] func:  104.873090685
iter:  62 v:  [ 0.24120719  1.19920547] func:  104.409604626
iter:  63 v:  [ 0.24121489  1.19988515] func:  103.976853081
iter:  64 v:  [ 0.24121116  1.20052351] func:  103.572656455
iter:  65 v:  [ 0.24119675  1.20112329] func:  103.195001267
iter:  66 v:  [ 0.24117235  1.20168705] func:  102.842026106
iter:  67 v:  [ 0.24113863  1.20221715] func:  102.512008926
iter:  68 v:  [ 0.2410962   1.20271582] func:  102.203355547
iter:  69 v:  [ 0.24104561  1.20318512] func:  101.91458923
iter:  70 v:  [ 0.24098741  1.20362699] func:  101.644341213
iter:  71 v:  [ 0.24092209  1.20404322] func:  101.391342096
iter:  72 v:  [ 0.24085011  1.20443551] func:  101.154414009
```

```
iter:  73 v:  [ 0.2407719  1.2048054] func:  100.932463465
iter:  74 v:  [ 0.24068788  1.20515437] func:  100.72447483
iter:  75 v:  [ 0.24059841  1.20548379] func:  100.529504371
iter:  76 v:  [ 0.24050386  1.20579493] func:  100.346674787
iter:  77 v:  [ 0.24040455  1.20608898] func:  100.175170222
iter:  78 v:  [ 0.24030079  1.20636706] func:  100.014231671
iter:  79 v:  [ 0.24019287  1.2066302 ] func:  99.8631527777
iter:  80 v:  [ 0.24008105  1.20687938] func:  99.7212759634
iter:  81 v:  [ 0.2399656  1.2071155] func:  99.5879888702
iter:  82 v:  [ 0.23984674  1.20733942] func:  99.4627210855
iter:  83 v:  [ 0.2397247   1.20755191] func:  99.3449411233
iter:  84 v:  [ 0.23959968  1.20775374] func:  99.2341536403
iter:  85 v:  [ 0.23947187  1.20794558] func:  99.129896867
iter:  86 v:  [ 0.23934146  1.20812808] func:  99.0317402339
iter:  87 v:  [ 0.2392086   1.20830185] func:  98.939282178
iter:  88 v:  [ 0.23907347  1.20846746] func:  98.8521481134
iter:  89 v:  [ 0.23893621  1.20862542] func:  98.7699885534
iter:  90 v:  [ 0.23879695  1.20877625] func:  98.692477371
iter:  91 v:  [ 0.23865582  1.20892039] func:  98.6193101869
iter:  92 v:  [ 0.23851294  1.20905828] func:  98.5502028748
iter:  93 v:  [ 0.23836843  1.20919032] func:  98.4848901753
iter:  94 v:  [ 0.23822239  1.2093169 ] func:  98.4231244086
iter:  95 v:  [ 0.23807492  1.20943835] func:  98.3646742796
iter:  96 v:  [ 0.23792612  1.20955503] func:  98.3093237682
iter:  97 v:  [ 0.23777606  1.20966722] func:  98.256871097
iter:  98 v:  [ 0.23762482  1.20977523] func:  98.2071277724
iter:  99 v:  [ 0.2374725   1.20987932] func:  98.1599176922
iter:  100 v:  [ 0.23731914  1.20997975] func:  98.1150763157
0.00207602399884
iter:  1 value:  [0 0] func:  3525
iter:  2 v:  [ 0.49824576  3.00608275] func:  1066.72356318
iter:  3 v:  [ 0.20775115  3.01374355] func:  963.1461816
iter:  4 v:  [-0.0586079   3.02515998] func:  875.780439411
iter:  5 v:  [-0.30281136  3.04149587] func:  803.014026616
iter:  6 v:  [-0.5266455   3.06405201] func:  743.496724775
iter:  7 v:  [-0.73169455  3.09420572] func:  696.047550983
iter:  8 v:  [-0.91931984  3.13331737] func:  659.55700221
iter:  9 v:  [-1.09062559  3.18259194] func:  632.88066733
iter:  10 v:  [-1.24641331  3.24287702] func:  614.724331591
iter:  11 v:  [-1.3871343   3.31437643] func:  603.534984709
iter:  12 v:  [-1.51286107  3.39627423] func:  597.441632926
iter:  13 v:  [-1.62331432  3.48632048] func:  594.326039877
iter:  14 v:  [-1.7179958   3.58054439] func:  592.10570218
iter:  15 v:  [-1.79646722  3.67339827] func:  589.210211557
iter:  16 v:  [-1.85875355  3.75864135] func:  585.019930342
iter:  17 v:  [-1.90573063  3.830921  ] func:  579.90860431
iter:  18 v:  [-1.93926586  3.88735328] func:  574.771333069
iter:  19 v:  [-1.96196414  3.92810377] func:  570.389472618
iter:  20 v:  [-1.97663409  3.95563277] func:  567.098016361
iter:  21 v:  [-1.98577955  3.97329639] func:  564.842568833
iter:  22 v:  [-1.9913362  3.9842238] func:  563.390454347
iter:  23 v:  [-1.99465513  3.99082243] func:  562.492389017
iter:  24 v:  [-1.9966162  3.9947467] func:  561.950664388
iter:  25 v:  [-1.99776738  3.99705877] func:  561.628774495
iter:  26 v:  [-1.99844055  3.99841336] func:  561.439202193
iter:  27 v:  [-1.99883335  3.9992043 ] func:  561.328124156
iter:  28 v:  [-1.99906233  3.9996652 ] func:  561.263217018
iter:  29 v:  [-1.99919579  3.99993342] func:  561.225333559
```

```
iter:  30 v:  [-1.99927363  4.00008937] func:  561.203221323
iter:  31 v:  [-1.99931911  4.00017995] func:  561.190297836
iter:  32 v:  [-1.99934576  4.00023252] func:  561.182722746
iter:  33 v:  [-1.99936146  4.00026298] func:  561.17825899
iter:  34 v:  [-1.99937081  4.00028058] func:  561.175604617
iter:  35 v:  [-1.99937645  4.0002907 ] func:  561.174002303
iter:  36 v:  [-1.99937994  4.00029648] func:  561.173011655
iter:  37 v:  [-1.99938219  4.00029974] func:  561.172376628
iter:  38 v:  [-1.9993837   4.00030152] func:  561.17194837
iter:  39 v:  [-1.99938479  4.00030246] func:  561.171640351
iter:  40 v:  [-1.99938564  4.0003029 ] func:  561.171402273
iter:  41 v:  [-1.99938634  4.00030305] func:  561.171204898
iter:  42 v:  [-1.99938697  4.00030303] func:  561.171031231
iter:  43 v:  [-1.99938754  4.00030291] func:  561.170871393
iter:  44 v:  [-1.99938808  4.00030274] func:  561.170719642
iter:  45 v:  [-1.99938861  4.00030254] func:  561.17057264
iter:  46 v:  [-1.99938912  4.00030232] func:  561.170428447
iter:  47 v:  [-1.99938963  4.00030208] func:  561.170285935
iter:  48 v:  [-1.99939014  4.00030184] func:  561.170144448
iter:  49 v:  [-1.99939065  4.0003016 ] func:  561.170003606
iter:  50 v:  [-1.99939115  4.00030136] func:  561.169863186
iter:  51 v:  [-1.99939165  4.00030111] func:  561.169723061
iter:  52 v:  [-1.99939215  4.00030086] func:  561.169583154
iter:  53 v:  [-1.99939265  4.00030062] func:  561.169443423
iter:  54 v:  [-1.99939315  4.00030037] func:  561.169303842
iter:  55 v:  [-1.99939365  4.00030012] func:  561.169164396
iter:  56 v:  [-1.99939415  4.00029987] func:  561.169025077
iter:  57 v:  [-1.99939465  4.00029963] func:  561.168885879
iter:  58 v:  [-1.99939515  4.00029938] func:  561.1687468
iter:  59 v:  [-1.99939565  4.00029913] func:  561.168607838
iter:  60 v:  [-1.99939615  4.00029889] func:  561.168468991
iter:  61 v:  [-1.99939664  4.00029864] func:  561.16833026
iter:  62 v:  [-1.99939714  4.0002984 ] func:  561.168191643
iter:  63 v:  [-1.99939764  4.00029815] func:  561.168053141
iter:  64 v:  [-1.99939813  4.0002979 ] func:  561.167914752
iter:  65 v:  [-1.99939863  4.00029766] func:  561.167776478
iter:  66 v:  [-1.99939912  4.00029741] func:  561.167638318
iter:  67 v:  [-1.99939962  4.00029717] func:  561.167500271
iter:  68 v:  [-1.99940011  4.00029692] func:  561.167362338
iter:  69 v:  [-1.99940061  4.00029668] func:  561.167224519
iter:  70 v:  [-1.9994011   4.00029644] func:  561.167086813
iter:  71 v:  [-1.99940159  4.00029619] func:  561.16694922
iter:  72 v:  [-1.99940209  4.00029595] func:  561.166811741
iter:  73 v:  [-1.99940258  4.0002957 ] func:  561.166674375
iter:  74 v:  [-1.99940307  4.00029546] func:  561.166537121
iter:  75 v:  [-1.99940356  4.00029522] func:  561.166399981
iter:  76 v:  [-1.99940405  4.00029497] func:  561.166262954
iter:  77 v:  [-1.99940455  4.00029473] func:  561.16612604
iter:  78 v:  [-1.99940504  4.00029449] func:  561.165989238
iter:  79 v:  [-1.99940553  4.00029425] func:  561.165852549
iter:  80 v:  [-1.99940602  4.000294  ] func:  561.165715972
iter:  81 v:  [-1.9994065   4.00029376] func:  561.165579508
iter:  82 v:  [-1.99940699  4.00029352] func:  561.165443156
iter:  83 v:  [-1.99940748  4.00029328] func:  561.165306916
iter:  84 v:  [-1.99940797  4.00029304] func:  561.165170789
iter:  85 v:  [-1.99940846  4.0002928 ] func:  561.165034773
iter:  86 v:  [-1.99940894  4.00029255] func:  561.16489887
iter:  87 v:  [-1.99940943  4.00029231] func:  561.164763078
```

```
iter:  88 v:  [-1.99940992  4.00029207] func:  561.164627399
iter:  89 v:  [-1.9994104   4.00029183] func:  561.16449183
iter:  90 v:  [-1.99941089  4.00029159] func:  561.164356374
iter:  91 v:  [-1.99941137  4.00029135] func:  561.164221029
iter:  92 v:  [-1.99941186  4.00029111] func:  561.164085795
iter:  93 v:  [-1.99941234  4.00029087] func:  561.163950673
iter:  94 v:  [-1.99941283  4.00029063] func:  561.163815662
iter:  95 v:  [-1.99941331  4.00029039] func:  561.163680762
iter:  96 v:  [-1.99941379  4.00029015] func:  561.163545973
iter:  97 v:  [-1.99941428  4.00028992] func:  561.163411295
iter:  98 v:  [-1.99941476  4.00028968] func:  561.163276728
iter:  99 v:  [-1.99941524  4.00028944] func:  561.163142272
iter:  100 v:  [-1.99941572  4.0002892 ] func:  561.163007926
0.5
iter:  1 value:  [0 0] func:  3525
iter:  2 v:  [ 120.  724.] func:  9.40127838462e+12
iter:  3 v:  [  1.03944800e+07  -1.49886671e+10] func:  1.76652627966e+42
iter:  4 v:  [  4.49320284e+21   1.34694243e+32] func:  1.15203104865e+130
iter:  5 v:  [  3.62850784e+65  -9.77478235e+97] func:  inf
iter:  6 v:  [  1.91092740e+197   3.73577989e+295] func:  inf
iter:  7 v:  [ inf -inf] func:  nan
iter:  8 v:  [ nan  nan] func:  nan
iter:  9 v:  [ nan  nan] func:  nan
iter:  10 v:  [ nan  nan] func:  nan
iter:  11 v:  [ nan  nan] func:  nan
iter:  12 v:  [ nan  nan] func:  nan
iter:  13 v:  [ nan  nan] func:  nan
iter:  14 v:  [ nan  nan] func:  nan
iter:  15 v:  [ nan  nan] func:  nan
iter:  16 v:  [ nan  nan] func:  nan
iter:  17 v:  [ nan  nan] func:  nan
iter:  18 v:  [ nan  nan] func:  nan
iter:  19 v:  [ nan  nan] func:  nan
iter:  20 v:  [ nan  nan] func:  nan
iter:  21 v:  [ nan  nan] func:  nan
iter:  22 v:  [ nan  nan] func:  nan
iter:  23 v:  [ nan  nan] func:  nan
iter:  24 v:  [ nan  nan] func:  nan
iter:  25 v:  [ nan  nan] func:  nan
iter:  26 v:  [ nan  nan] func:  nan
iter:  27 v:  [ nan  nan] func:  nan
iter:  28 v:  [ nan  nan] func:  nan
iter:  29 v:  [ nan  nan] func:  nan
iter:  30 v:  [ nan  nan] func:  nan
iter:  31 v:  [ nan  nan] func:  nan
iter:  32 v:  [ nan  nan] func:  nan
iter:  33 v:  [ nan  nan] func:  nan
iter:  34 v:  [ nan  nan] func:  nan
iter:  35 v:  [ nan  nan] func:  nan
iter:  36 v:  [ nan  nan] func:  nan
iter:  37 v:  [ nan  nan] func:  nan
iter:  38 v:  [ nan  nan] func:  nan
iter:  39 v:  [ nan  nan] func:  nan
iter:  40 v:  [ nan  nan] func:  nan
iter:  41 v:  [ nan  nan] func:  nan
iter:  42 v:  [ nan  nan] func:  nan
iter:  43 v:  [ nan  nan] func:  nan
iter:  44 v:  [ nan  nan] func:  nan
```

```
iter:  45 v:  [ nan  nan] func:  nan
iter:  46 v:  [ nan  nan] func:  nan
iter:  47 v:  [ nan  nan] func:  nan
iter:  48 v:  [ nan  nan] func:  nan
iter:  49 v:  [ nan  nan] func:  nan
iter:  50 v:  [ nan  nan] func:  nan
iter:  51 v:  [ nan  nan] func:  nan
iter:  52 v:  [ nan  nan] func:  nan
iter:  53 v:  [ nan  nan] func:  nan
iter:  54 v:  [ nan  nan] func:  nan
iter:  55 v:  [ nan  nan] func:  nan
iter:  56 v:  [ nan  nan] func:  nan
iter:  57 v:  [ nan  nan] func:  nan
iter:  58 v:  [ nan  nan] func:  nan
iter:  59 v:  [ nan  nan] func:  nan
iter:  60 v:  [ nan  nan] func:  nan
iter:  61 v:  [ nan  nan] func:  nan
iter:  62 v:  [ nan  nan] func:  nan
iter:  63 v:  [ nan  nan] func:  nan
iter:  64 v:  [ nan  nan] func:  nan
iter:  65 v:  [ nan  nan] func:  nan
iter:  66 v:  [ nan  nan] func:  nan
iter:  67 v:  [ nan  nan] func:  nan
iter:  68 v:  [ nan  nan] func:  nan
iter:  69 v:  [ nan  nan] func:  nan
iter:  70 v:  [ nan  nan] func:  nan
iter:  71 v:  [ nan  nan] func:  nan
iter:  72 v:  [ nan  nan] func:  nan
iter:  73 v:  [ nan  nan] func:  nan
iter:  74 v:  [ nan  nan] func:  nan
iter:  75 v:  [ nan  nan] func:  nan
iter:  76 v:  [ nan  nan] func:  nan
iter:  77 v:  [ nan  nan] func:  nan
iter:  78 v:  [ nan  nan] func:  nan
iter:  79 v:  [ nan  nan] func:  nan
iter:  80 v:  [ nan  nan] func:  nan
iter:  81 v:  [ nan  nan] func:  nan
iter:  82 v:  [ nan  nan] func:  nan
iter:  83 v:  [ nan  nan] func:  nan
iter:  84 v:  [ nan  nan] func:  nan
iter:  85 v:  [ nan  nan] func:  nan
iter:  86 v:  [ nan  nan] func:  nan
iter:  87 v:  [ nan  nan] func:  nan
iter:  88 v:  [ nan  nan] func:  nan
iter:  89 v:  [ nan  nan] func:  nan
iter:  90 v:  [ nan  nan] func:  nan
iter:  91 v:  [ nan  nan] func:  nan
iter:  92 v:  [ nan  nan] func:  nan
iter:  93 v:  [ nan  nan] func:  nan
iter:  94 v:  [ nan  nan] func:  nan
iter:  95 v:  [ nan  nan] func:  nan
iter:  96 v:  [ nan  nan] func:  nan
iter:  97 v:  [ nan  nan] func:  nan
iter:  98 v:  [ nan  nan] func:  nan
iter:  99 v:  [ nan  nan] func:  nan
iter:  100 v:  [ nan  nan] func:  nan
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0