In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df = pd.read_csv('ANISO2.CSV', sep = ',')
df
```

Out[2]:

	A	B	C	B1
0	0.00000	-0.08194	NaN	--
1	0.01221	-0.08194	NaN	--
2	0.02441	1.00000	NaN	--
3	0.03662	1.00000	NaN	--
4	0.04883	1.00000	NaN	--
5	0.06103	-0.50000	NaN	--
6	0.07324	0.15388	NaN	--
7	0.08545	1.00000	NaN	--
8	0.09766	1.00000	NaN	--
9	0.10986	1.00000	NaN	--
10	0.12207	-0.08194	NaN	--
11	0.13428	1.00000	NaN	--
12	0.14648	1.00000	NaN	--
13	0.15869	1.00000	NaN	--
14	0.17090	1.00000	NaN	--
15	0.18310	1.00000	NaN	--
16	0.19531	1.00000	NaN	--
17	0.20752	1.00000	NaN	--
18	0.21973	1.00000	NaN	--
19	0.23193	-0.08194	NaN	--
20	0.24414	1.00000	NaN	--
21	0.25635	1.00000	NaN	--
22	0.26855	0.15388	NaN	--
23	0.28076	1.00000	NaN	--
24	0.29297	-0.08194	NaN	--
25	0.30518	1.00000	NaN	--
26	0.31738	-0.50000	NaN	--
27	0.32959	-0.50000	NaN	--
28	0.34180	1.00000	NaN	--
29	0.35400	1.00000	NaN	--
...	...	...	...	...
4066	49.63379	1.00000	NaN	--
4067	49.64600	1.00000	NaN	--

	A	B	C	B1
4068	49.65820	1.00000	NaN	--
4069	49.67041	1.00000	NaN	--
4070	49.68262	-0.08194	NaN	--
4071	49.69482	1.00000	NaN	--
4072	49.70703	1.00000	NaN	--
4073	49.71924	0.30530	NaN	--
4074	49.73145	1.00000	NaN	--
4075	49.74365	0.30530	NaN	--
4076	49.75586	-0.50000	NaN	--
4077	49.76807	1.00000	NaN	--
4078	49.78027	0.30530	NaN	--
4079	49.79248	-0.08194	NaN	--
4080	49.80469	0.30530	NaN	--
4081	49.81690	-0.50000	NaN	--
4082	49.82910	-0.50000	NaN	--
4083	49.84131	1.00000	NaN	--
4084	49.85352	-0.50000	NaN	--
4085	49.86572	-0.50000	NaN	--
4086	49.87793	1.00000	NaN	--
4087	49.89014	1.00000	NaN	--
4088	49.90234	1.00000	NaN	--
4089	49.91455	-0.50000	NaN	--
4090	49.92676	1.00000	NaN	--
4091	49.93897	1.00000	NaN	--
4092	49.95117	-0.08194	NaN	--
4093	49.96338	1.00000	NaN	--
4094	49.97559	-0.08194	NaN	--
4095	49.98779	1.00000	NaN	--

4096 rows × 4 columns

In [3]:

```
df1 = df.iloc[221:1230,:]
df1
```

Out[3]:

	A	B	C	B1
221	2.69775	0.34441	NaN	--
222	2.70996	0.30530	NaN	--
223	2.72217	0.48840	NaN	--
224	2.73438	0.54797	NaN	--
225	2.74658	0.43166	NaN	--
226	2.75879	0.33730	NaN	--
227	2.77100	0.41401	NaN	--
228	2.78320	0.42793	NaN	--
229	2.79541	0.40007	NaN	--
230	2.80762	0.39090	NaN	0.34411
231	2.81982	0.36942	NaN	0.34397
232	2.83203	0.37640	NaN	0.34383
233	2.84424	0.35615	NaN	0.34369
234	2.85644	0.36995	NaN	0.34355
235	2.86865	0.35605	NaN	0.34341
236	2.88086	0.34820	NaN	0.34327
237	2.89307	0.35122	NaN	0.34313
238	2.90527	0.34927	NaN	0.34299
239	2.91748	0.34319	NaN	0.34285
240	2.92969	0.35043	NaN	0.34272
241	2.94190	0.34782	NaN	0.34258
242	2.95410	0.32477	NaN	0.34244
243	2.96631	0.34704	NaN	0.3423
244	2.97852	0.34094	NaN	0.34217
245	2.99072	0.34143	NaN	0.34203
246	3.00293	0.34312	NaN	0.34189
247	3.01514	0.33800	NaN	0.34175
248	3.02734	0.30860	NaN	0.34162
249	3.03955	0.32003	NaN	0.34148
250	3.05176	0.34967	NaN	0.34135
...	...	...	...	...
1200	14.64844	0.33951	NaN	0.26183
1201	14.66065	0.28808	NaN	0.26178

	A	B	C	B1
1202	14.67285	0.26999	NaN	0.26172
1203	14.68506	0.26008	NaN	0.26166
1204	14.69727	0.25314	NaN	0.2616
1205	14.70947	0.22315	NaN	0.26155
1206	14.72168	0.27951	NaN	0.26149
1207	14.73389	0.24606	NaN	0.26143
1208	14.74609	0.28025	NaN	0.26137
1209	14.75830	0.32170	NaN	0.26131
1210	14.77051	0.34899	NaN	0.26126
1211	14.78272	0.36472	NaN	0.2612
1212	14.79492	0.32657	NaN	0.26114
1213	14.80713	0.28016	NaN	0.26108
1214	14.81934	0.22778	NaN	0.26103
1215	14.83154	0.11214	NaN	0.26097
1216	14.84375	0.21735	NaN	0.26091
1217	14.85596	0.27810	NaN	0.26085
1218	14.86816	0.27026	NaN	0.2608
1219	14.88037	0.24652	NaN	0.26074
1220	14.89258	0.18354	NaN	0.26068
1221	14.90479	0.18761	NaN	0.26062
1222	14.91699	0.30183	NaN	0.26057
1223	14.92920	0.27882	NaN	0.26051
1224	14.94141	0.22463	NaN	0.26045
1225	14.95361	0.36883	NaN	0.2604
1226	14.96582	0.19080	NaN	0.26034
1227	14.97803	0.29507	NaN	0.26028
1228	14.99023	0.24759	NaN	0.26022
1229	15.00244	0.09393	NaN	0.26017

1009 rows × 4 columns

In [4]:

```
df1.reset_index(drop = True, inplace = True)
df1
```

Out[4]:

	A	B	C	B1
0	2.69775	0.34441	NaN	--
1	2.70996	0.30530	NaN	--
2	2.72217	0.48840	NaN	--
3	2.73438	0.54797	NaN	--
4	2.74658	0.43166	NaN	--
5	2.75879	0.33730	NaN	--
6	2.77100	0.41401	NaN	--
7	2.78320	0.42793	NaN	--
8	2.79541	0.40007	NaN	--
9	2.80762	0.39090	NaN	0.34411
10	2.81982	0.36942	NaN	0.34397
11	2.83203	0.37640	NaN	0.34383
12	2.84424	0.35615	NaN	0.34369
13	2.85644	0.36995	NaN	0.34355
14	2.86865	0.35605	NaN	0.34341
15	2.88086	0.34820	NaN	0.34327
16	2.89307	0.35122	NaN	0.34313
17	2.90527	0.34927	NaN	0.34299
18	2.91748	0.34319	NaN	0.34285
19	2.92969	0.35043	NaN	0.34272
20	2.94190	0.34782	NaN	0.34258
21	2.95410	0.32477	NaN	0.34244
22	2.96631	0.34704	NaN	0.3423
23	2.97852	0.34094	NaN	0.34217
24	2.99072	0.34143	NaN	0.34203
25	3.00293	0.34312	NaN	0.34189
26	3.01514	0.33800	NaN	0.34175
27	3.02734	0.30860	NaN	0.34162
28	3.03955	0.32003	NaN	0.34148
29	3.05176	0.34967	NaN	0.34135
...	...	...	...	...
979	14.64844	0.33951	NaN	0.26183
980	14.66065	0.28808	NaN	0.26178

	A	B	C	B1
981	14.67285	0.26999	NaN	0.26172
982	14.68506	0.26008	NaN	0.26166
983	14.69727	0.25314	NaN	0.2616
984	14.70947	0.22315	NaN	0.26155
985	14.72168	0.27951	NaN	0.26149
986	14.73389	0.24606	NaN	0.26143
987	14.74609	0.28025	NaN	0.26137
988	14.75830	0.32170	NaN	0.26131
989	14.77051	0.34899	NaN	0.26126
990	14.78272	0.36472	NaN	0.2612
991	14.79492	0.32657	NaN	0.26114
992	14.80713	0.28016	NaN	0.26108
993	14.81934	0.22778	NaN	0.26103
994	14.83154	0.11214	NaN	0.26097
995	14.84375	0.21735	NaN	0.26091
996	14.85596	0.27810	NaN	0.26085
997	14.86816	0.27026	NaN	0.2608
998	14.88037	0.24652	NaN	0.26074
999	14.89258	0.18354	NaN	0.26068
1000	14.90479	0.18761	NaN	0.26062
1001	14.91699	0.30183	NaN	0.26057
1002	14.92920	0.27882	NaN	0.26051
1003	14.94141	0.22463	NaN	0.26045
1004	14.95361	0.36883	NaN	0.2604
1005	14.96582	0.19080	NaN	0.26034
1006	14.97803	0.29507	NaN	0.26028
1007	14.99023	0.24759	NaN	0.26022
1008	15.00244	0.09393	NaN	0.26017

1009 rows × 4 columns

In [5]:

```
cols = ['B']
df1[cols] = df1[df1[cols]>0][cols]
df1[cols] = df1[df1[cols]<0.4][cols]
df1.dropna()
df1
```

```
C:\Users\sony\Anaconda3 new\lib\site-packages\pandas\core\frame.py:3137: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  self[k1] = value[k2]
```

Out[5]:

	A	B	C	B1
0	2.69775	0.34441	NaN	--
1	2.70996	0.30530	NaN	--
2	2.72217	NaN	NaN	--
3	2.73438	NaN	NaN	--
4	2.74658	NaN	NaN	--
5	2.75879	0.33730	NaN	--
6	2.77100	NaN	NaN	--
7	2.78320	NaN	NaN	--
8	2.79541	NaN	NaN	--
9	2.80762	0.39090	NaN	0.34411
10	2.81982	0.36942	NaN	0.34397
11	2.83203	0.37640	NaN	0.34383
12	2.84424	0.35615	NaN	0.34369
13	2.85644	0.36995	NaN	0.34355
14	2.86865	0.35605	NaN	0.34341
15	2.88086	0.34820	NaN	0.34327
16	2.89307	0.35122	NaN	0.34313
17	2.90527	0.34927	NaN	0.34299
18	2.91748	0.34319	NaN	0.34285
19	2.92969	0.35043	NaN	0.34272
20	2.94190	0.34782	NaN	0.34258
21	2.95410	0.32477	NaN	0.34244
22	2.96631	0.34704	NaN	0.3423
23	2.97852	0.34094	NaN	0.34217
24	2.99072	0.34143	NaN	0.34203
25	3.00293	0.34312	NaN	0.34189
26	3.01514	0.33800	NaN	0.34175
27	3.02734	0.30860	NaN	0.34162
28	3.03955	0.32003	NaN	0.34148
29	3.05176	0.34967	NaN	0.34135
...	...	...	...	...
979	14.64844	0.33951	NaN	0.26183
980	14.66065	0.28808	NaN	0.26178

	A	B	C	B1
981	14.67285	0.26999	NaN	0.26172
982	14.68506	0.26008	NaN	0.26166
983	14.69727	0.25314	NaN	0.2616
984	14.70947	0.22315	NaN	0.26155
985	14.72168	0.27951	NaN	0.26149
986	14.73389	0.24606	NaN	0.26143
987	14.74609	0.28025	NaN	0.26137
988	14.75830	0.32170	NaN	0.26131
989	14.77051	0.34899	NaN	0.26126
990	14.78272	0.36472	NaN	0.2612
991	14.79492	0.32657	NaN	0.26114
992	14.80713	0.28016	NaN	0.26108
993	14.81934	0.22778	NaN	0.26103
994	14.83154	0.11214	NaN	0.26097
995	14.84375	0.21735	NaN	0.26091
996	14.85596	0.27810	NaN	0.26085
997	14.86816	0.27026	NaN	0.2608
998	14.88037	0.24652	NaN	0.26074
999	14.89258	0.18354	NaN	0.26068
1000	14.90479	0.18761	NaN	0.26062
1001	14.91699	0.30183	NaN	0.26057
1002	14.92920	0.27882	NaN	0.26051
1003	14.94141	0.22463	NaN	0.26045
1004	14.95361	0.36883	NaN	0.2604
1005	14.96582	0.19080	NaN	0.26034
1006	14.97803	0.29507	NaN	0.26028
1007	14.99023	0.24759	NaN	0.26022
1008	15.00244	0.09393	NaN	0.26017

1009 rows × 4 columns

In [6]:

```
df1 = df1[np.isfinite(df1['B'])]
df1
```

Out[6]:

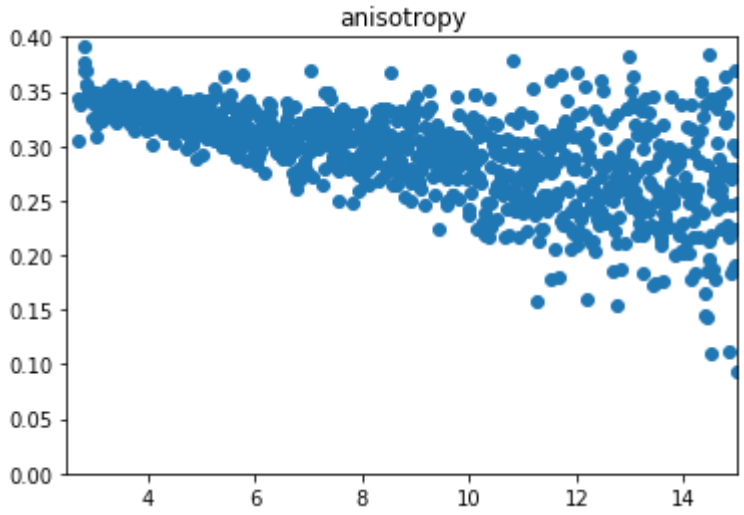
	A	B	C	B1
0	2.69775	0.34441	NaN	--
1	2.70996	0.30530	NaN	--
5	2.75879	0.33730	NaN	--
9	2.80762	0.39090	NaN	0.34411
10	2.81982	0.36942	NaN	0.34397
11	2.83203	0.37640	NaN	0.34383
12	2.84424	0.35615	NaN	0.34369
13	2.85644	0.36995	NaN	0.34355
14	2.86865	0.35605	NaN	0.34341
15	2.88086	0.34820	NaN	0.34327
16	2.89307	0.35122	NaN	0.34313
17	2.90527	0.34927	NaN	0.34299
18	2.91748	0.34319	NaN	0.34285
19	2.92969	0.35043	NaN	0.34272
20	2.94190	0.34782	NaN	0.34258
21	2.95410	0.32477	NaN	0.34244
22	2.96631	0.34704	NaN	0.3423
23	2.97852	0.34094	NaN	0.34217
24	2.99072	0.34143	NaN	0.34203
25	3.00293	0.34312	NaN	0.34189
26	3.01514	0.33800	NaN	0.34175
27	3.02734	0.30860	NaN	0.34162
28	3.03955	0.32003	NaN	0.34148
29	3.05176	0.34967	NaN	0.34135
30	3.06397	0.34610	NaN	0.34121
31	3.07617	0.33665	NaN	0.34108
32	3.08838	0.34282	NaN	0.34094
33	3.10059	0.32724	NaN	0.34081
34	3.11279	0.34405	NaN	0.34067
35	3.12500	0.32869	NaN	0.34054
...	...	...	...	...
979	14.64844	0.33951	NaN	0.26183
980	14.66065	0.28808	NaN	0.26178

	A	B	C	B1
981	14.67285	0.26999	NaN	0.26172
982	14.68506	0.26008	NaN	0.26166
983	14.69727	0.25314	NaN	0.2616
984	14.70947	0.22315	NaN	0.26155
985	14.72168	0.27951	NaN	0.26149
986	14.73389	0.24606	NaN	0.26143
987	14.74609	0.28025	NaN	0.26137
988	14.75830	0.32170	NaN	0.26131
989	14.77051	0.34899	NaN	0.26126
990	14.78272	0.36472	NaN	0.2612
991	14.79492	0.32657	NaN	0.26114
992	14.80713	0.28016	NaN	0.26108
993	14.81934	0.22778	NaN	0.26103
994	14.83154	0.11214	NaN	0.26097
995	14.84375	0.21735	NaN	0.26091
996	14.85596	0.27810	NaN	0.26085
997	14.86816	0.27026	NaN	0.2608
998	14.88037	0.24652	NaN	0.26074
999	14.89258	0.18354	NaN	0.26068
1000	14.90479	0.18761	NaN	0.26062
1001	14.91699	0.30183	NaN	0.26057
1002	14.92920	0.27882	NaN	0.26051
1003	14.94141	0.22463	NaN	0.26045
1004	14.95361	0.36883	NaN	0.2604
1005	14.96582	0.19080	NaN	0.26034
1006	14.97803	0.29507	NaN	0.26028
1007	14.99023	0.24759	NaN	0.26022
1008	15.00244	0.09393	NaN	0.26017

997 rows × 4 columns

In [8]:

```
import matplotlib.pyplot as plt
plt.plot(df1.A, df1.B,'o')
plt.title("anisotropy")
plt.xlim(2.5,15)
plt.ylim(0,0.4)
plt.show()
```



In [9]:

```
from math import *
%matplotlib inline
import scipy.optimize
from lmfit import Model
from lmfit import minimize, Parameters, Parameter, report_fit
```

In [10]:

```
t = df1['A'].values
rt = df1['B'].values
noisy = rt + 0.001*np.random.normal(size=len(rt))
```

In [13]:

```
# define objective function: returns the array to be minimized
def fcn2min(params, t, noisy):
    c0 = params['c0'].value
    c1 = params['c1'].value
    c2 = params['c2'].value
    c3 = params['c3'].value
    c4 = params['c4'].value
    c5 = params['c5'].value
    c6 = params['c6'].value
    c7 = params['c7'].value
    c8 = params['c8'].value
    c9 = params['c9'].value
    c10 = params['c10'].value
    c11 = params['c11'].value
    model = ((c0*np.exp(-t/c1)*c2*np.exp(-t/c3))+ (c4*np.exp(-t/c5)*c6*np.exp(-t/c7))+(
c8*np.exp(-t/c9)*c10*np.exp(-t/c11)))/((c0*np.exp(-t/c1))+ (c4*np.exp(-t/c5))+ (c8*np.
exp(-t/c9)))
    return model - noisy
```

In [120]:

```
# create a set of Parameters
params = Parameters()
params.add('c0', value= 100, min=0 )
params.add('c1', value= 0.110, min=0)
params.add('c2', value= 0.35, min=0, max=0.4)
params.add('c3', value= 0.100, min=0)
params.add('c4', value= 100, min=0 )
params.add('c5', value= 1.5, min=0)
params.add('c6', value= 0.35, min=0, max=0.4)
params.add('c7', value= 20, min=0, max=200)
params.add('c8', value= 100, min=0 )
params.add('c9', value= 3.2)
params.add('c10', value= 0.35, min=0, max=0.4)
params.add('c11', value= 50, min=0, max=200)
```

In [121]:

```
params['c1'].vary = False
params['c5'].vary = False
params['c9'].vary = False
```

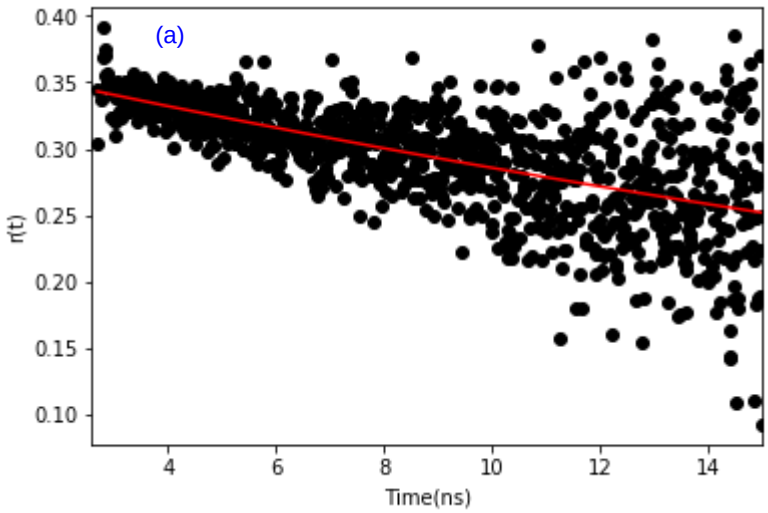
In [122]:

```
# do fit, here with leastsq model
result = minimize(fcn2min, params, args=(t, noisy))
# calculate final result
final = noisy + result.residual
# write error report
report_fit(result.params)
```

```
[[Variables]]
    c0:   100.000000 (init = 100)
    c1:   0.11 (fixed)
    c2:   0.35000000 (init = 0.35)
    c3:   0.10000000 (init = 0.1)
    c4:   2.3866e+10 (init = 100)
    c5:   1.5 (fixed)
    c6:   0.36707551 (init = 0.35)
    c7:   39.8414444 (init = 20)
    c8:   100.000000 (init = 100)
    c9:   3.2 (fixed)
    c10:  0.35000000 (init = 0.35)
    c11:  50.0000000 (init = 50)
```

In [124]:

```
# try to plot results
try:
    import pylab
    pylab.plot(t, noisy, 'ko')
    pylab.plot(t, final, 'r')
    pylab.xlim(2.6,15)
    pylab.xlabel('Time(ns)')
    pylab.ylabel('r(t)')
    pylab.show()
except:
    pass
```



In [125]:

```
lmfit_Rsquared = 1 - result.residual.var()/np.var(noisy)
print('Fit R-squared:', lmfit_Rsquared, '\n')
print(result.params)
```

```
Fit R-squared: 0.37964749918350826

Parameters([('c0', <Parameter 'c0', 100.0, bounds=[0:inf]>), ('c1', <Param
eter 'c1', value=0.11 (fixed), bounds=[0:inf]>), ('c2', <Parameter 'c2',
0.35, bounds=[0:0.4]>), ('c3', <Parameter 'c3', 0.10000000000000009, bound
s=[0:inf]>), ('c4', <Parameter 'c4', 23865524345.58093, bounds=[0:inf]>),
('c5', <Parameter 'c5', value=1.5 (fixed), bounds=[0:inf]>), ('c6', <Param
eter 'c6', 0.3670755103664445, bounds=[0:0.4]>), ('c7', <Parameter 'c7', 3
9.84144435424659, bounds=[0:200]>), ('c8', <Parameter 'c8', 100.0, bounds=
[0:inf]>), ('c9', <Parameter 'c9', value=3.2 (fixed), bounds=[-inf:inf]>),
('c10', <Parameter 'c10', 0.35, bounds=[0:0.4]>), ('c11', <Parameter 'c1
1', 50.0, bounds=[0:200]>)])
```

In [126]:

```
print('Fit X^2: ', result.chisqr)
print('Fit reduced-X^2:', result.redchi)
```

```
Fit X^2:  1.1057577180026508
Fit reduced-X^2: 0.0011191879736868935
```

In [127]:

```
plt.plot(t,result.residual,'r')
```

Out[127]:

```
[<matplotlib.lines.Line2D at 0xcc747f0>]
```