Analyzing Amber simulations of Parvalbumin using MDAnalysis and Ptraj

Step 0. File prep

FYI, but not needed for example. (ran as user 'guest' to test)

- Use ptraj to generate a concatenated cdf file on DLX, as well as a pdb (ptraj-generated dcd lack header?)

```
cd ~/storage/WT_apo_amber_1/
```

- Convert to dcd

```
catdcd -o WT_holo_1.dcd -s WT_holo_1.pdb -stype pdb -netcdf WT_holo_1.cdf
```

- scp locally from kafka

```
scp $DLX:~/storage/WT_apo_amber_1/WT_holo_1.pdb .
scp $DLX:~/storage/WT_apo_amber_1/WT_holo_1.dcd .

scp $DLX:~/Work/WT_apo_amber_1/prmtop .
```

Step 1. Setting up notebook

- Load sources (Must be done prior to opening notebook!) `export MYPATH=/home/huskeypm/sources/mypython/`
  `export PYTHONPATH=$PYTHONPATH:$MYPATH/lib/python2.7/site-packages/`
  `python -c "import MDAnalysis"`

- Launch ipython notebook (see wiki)

Step 2. Reading and analyzing a trajectory with MDAnalysis

In [9]:
```
import MDAnalysis
```

In [10]:
```
#%cd /net/home/huskeypm/localTemp/parv/amber_holo/
prefix = "/u1/shared/parvanalysisAmber/WT_holo_1"
dcdFile = prefix+".dcd"
pdbFile = prefix+".pdb"
```

Load trajectory

In [8]:
```
u = MDAnalysis.Universe(pdbFile, dcdFile)
print u.trajectory
```

```
< DCDReader '/u1/shared/parvanalysisAmber/WT_holo_1.dcd' with 15601 frames of 1716 atoms (0 fixed) >
```

Define function for analyzing each frame of a trajectory.

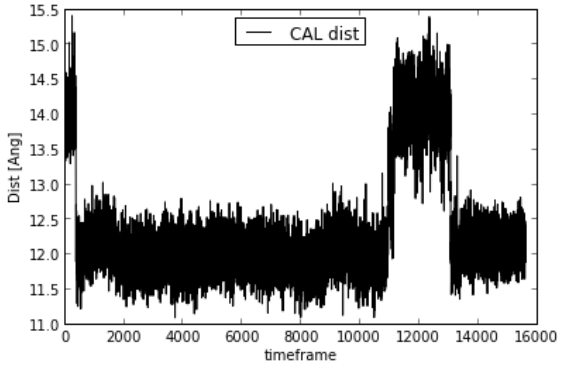
In [13]:
```
# u - the universe object
# bb - an MDAnalysis selection (defined below)
def trajdist(u,bb):
  ds = np.zeros( u.trajectory.numframes )
  time = np.arange(u.trajectory.numframes )
  for i,ts in enumerate(u.trajectory):
    r = bb[1].pos - bb[0].pos
    d = numpy.linalg.norm(r)
    ds[i]=d
    #print d

  return time,ds
```

Define a selection object for CAL (will have two in this system), then use the previous function to analysis distance over all frames (time slices)

In [19]:
```
bb = u.selectAtoms('name CAL')
time,ds = trajdist(u,bb)
plot(time,ds,'k',label="CAL dist")
plt.xlabel("timeframe")
plt.ylabel("Dist [Ang]")
plt.legend(loc=0)
```

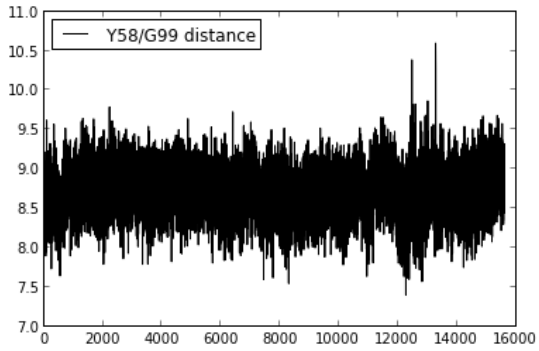
Out[19]: <matplotlib.legend.Legend at 0x3cca550>



Plot distance between C-alpha atoms of Y58 and G99

In [20]:
```
bb = u.selectAtoms('name CA and (resid 58 or resid 99)')
time,ds = trajdist(u,bb)
plot(time,ds,'k',label="Y58/G99 distance")
plt.legend(loc=0)
```

Out[20]: <matplotlib.legend.Legend at 0x3bbb250>



Step 3. Dynamic cross correlation matrix with ptraj

This matrix shows collective motion between residues. Highly correlated residues have correlation values of 1. (red), while uncorrelated have values approaching 0 (blue)

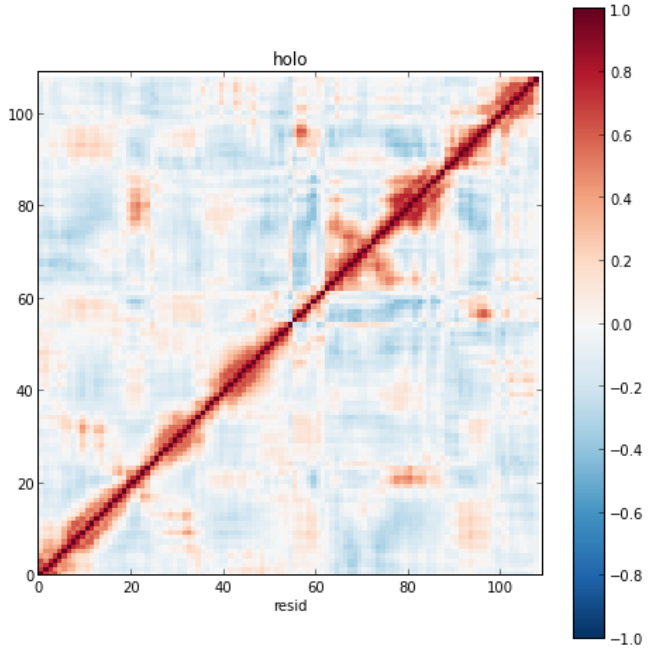
To generate the files needed, I ran the following (but you need not)

- Compute cross-correlation matrix using ptraj (also rmsf in the same dccm.in file)
  ```
  source ~/bin/amber.bash
  PRM=/u1/shared/parvanalysisAmber/WT_holo_1.pdb
  $AMBERHOME/bin/ptraj $PRM  < dccm.in > dccm.out
  ```
- This command will generate `matrix_correl_CA.dat`

Define function

In [24]:
```
def plotcorr(dccmName,title=""):
  v=np.loadtxt(dccmName)
  plt.figure(figsize=(8,8))
  #pcolormesh(np.arange(109),np.arange(109),v,cmap="RdBu_r")
  pcolormesh(v,cmap="RdBu_r")
  plt.colorbar()
  plt.xlim([0,109])
  plt.ylim([0,109])
  plt.clim([-1,1])
  plt.title(title)
  plt.xlabel("resid")
  axes().set_aspect('equal')
  return v
```

In [26]:
```
name = "matrix_correl_CA.dat"
#apoName="/net/home/huskeypm/localTemp/parv/amber_apo/"+name
holoRoot="/u1/shared/parvanalysisAmber/"
holoName=holoRoot+name
#apo=plotcorr(apoName,title="apo")
holo=plotcorr(holoName,title="holo")
```



Sites I and II (where Ca2+ bind) are located approximately at 52-63 and 92-99. The pairing of Y58-G99 marks the beta sheet formed between sites I and II .

Step 4. RMSF with ptraj

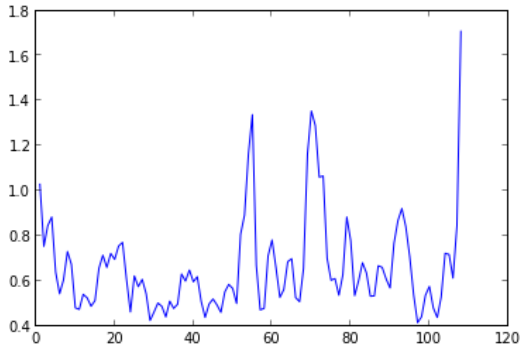
Atomic fluctuations RMSF also computed using ptraj/dccm.in sscript

In [32]:
```
def plotrmsf(rmsfName,title="",plot=True):
  v=np.loadtxt(rmsfName)
  if(plot==False):
        return v
  plt.figure(figsize=(8,8))
  #pcolormesh(np.arange(109),np.arange(109),v,cmap="RdBu_r")
  plt.plot(v[:,0],v[:,1])
  plt.title(title)
  plt.xlabel("resid")
  #axes().set_aspect('equal')
  return v
```

In [33]:
```
name = "rmsf.dat"
#apoName="/net/home/huskeypm/localTemp/parv/amber_apo/"+name
holoName=holoRoot+name
#apo=plotrmsf(apoName,title="apo",plot=False)
holo=plotrmsf(holoName,title="holo",plot=False)
```

In [34]:
```
#plot(apo[:,0],apo[:,1],'r',label="apo")
plot(holo[:,0],holo[:,1],'b',label="holo")
```

Out[34]:
```
[<matplotlib.lines.Line2D at 0x44d33d0>]
```



In [ ]: