Working with Correlation Functions

In [1]:
```
import numpy as np
import pyPRISM
import matplotlib.pyplot as plt
```

Combining correlation functions

Explanation

First we note the expression for a pair-correlation function in real-space:

$$g_{i,j}(r) = \frac{1}{\rho_{i,j}} \frac{n_{i,j}(r)}{4\pi r^2 dr}$$

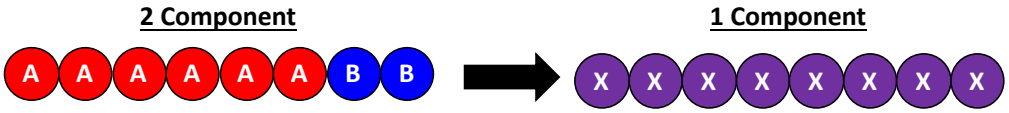
where $g_{i,j}(r)$ is the value of the pair correlation function at distance r, $\rho_{i,j} = \rho_i \rho_j V$ is the bulk pair density of site-types i and j, ρ_i is the number density of site-type i, and $n_{i,j}(r)$ is the average number of i-j site-type pairs separated by a distance r.

By recongizing that $n_{i,j}(r)$ is additive between different site-type pairs and assuming the direct correlation function, $\hat{c}_{i,j}(k)$, has a similar form, we can inuit how to combine the direct correlation functions together. Let's consider a case where we want to calculate the overall direct correlations between two groups of site-types, $X = i_1, i_2, i_3, \ldots, i_n$ and $Y = j_1, j_2, j_3, \ldots, j_n$. Based on the expression above for $g_{i,j}(r)$, the overall direct correlation function between the two super site-types X and Y would be.

$$\rho_X \rho_Y \hat{c}_{X,Y}(k) = \sum_{i \in X, j \in Y} \rho_i \rho_j \hat{c}_{i,j}(k)$$

The expressions for $\hat{c}_{X,X}(k)$ and $\hat{c}_{Y,Y}(k)$ follow similarly.

2-Component to 1-Component Example and Verification



Let's consider the asymmetric diblock copolymer shown in the image being "coarsened" from two components to one component. Effectively, we want to combine the direct correlation functions so that we have a single super site type:

$$X = A + B$$

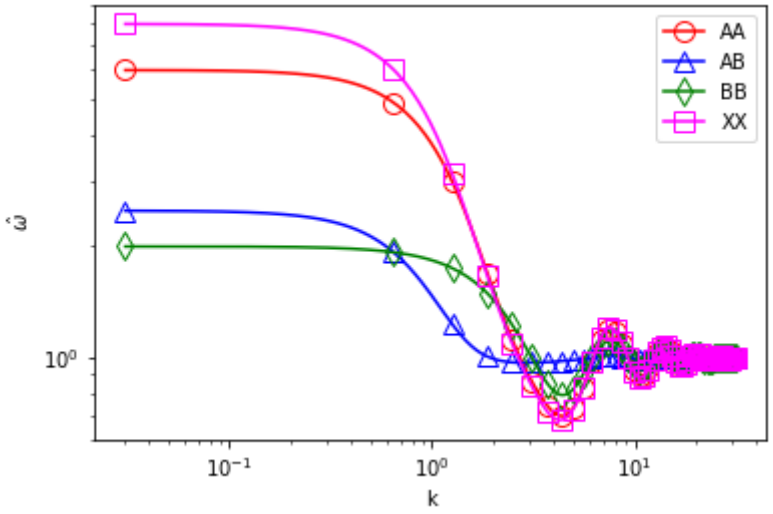
Based on the arguments presented above, the combined direct correlation function would be

$$\rho_X \rho_X \hat{c}_{X,X} = \rho_A \rho_A \hat{c}_{A,A}(k) + 2\rho_A \rho_B \hat{c}_{A,B}(k) + \rho_B \rho_B \hat{c}_{B,B}(k)$$

To verify this expression, we will carry out two PRISM calculations: one for the original and one for the coarsened system. If our expression is correct, we should be able to reproduce the direct correlation functions of the coarsened system using those from the original.

To start, we load the and plot all necessary $\hat{\omega}_{i,j}$ needed for this calculation

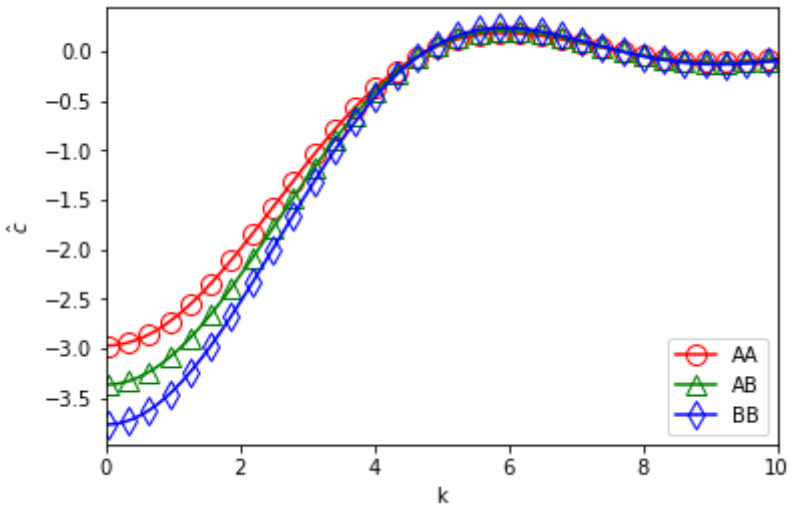
In [2]:
```
N_A = 6 # number of A beads in the chain
N_B = 2 # number of B beads in the chain

# load \omega from simulation
omega_AA_k, omega_AA = np.loadtxt('omega-{}{}-AA.txt'.format(N_A,N_B)).T
omega_AB_k, omega_AB = np.loadtxt('omega-{}{}-AB.txt'.format(N_A,N_B)).T
omega_BB_k, omega_BB = np.loadtxt('omega-{}{}-BB.txt'.format(N_A,N_B)).T
omega_XX_k, omega_XX = np.loadtxt('omega-{}{}-PP.txt'.format(N_A,N_B)).T

kw = {}
kw['fillstyle'] = 'none'
kw['markevery'] = 20
kw['ms'] = 10

plt.loglog(omega_AA_k,omega_AA,label='AA',marker='o',color='red',**kw)
plt.loglog(omega_AB_k,omega_AB+1.0,label='AB',marker='^',color='blue',**
kw)
plt.loglog(omega_BB_k,omega_BB,label='BB',marker='d',color='green',**kw)
plt.loglog(omega_XX_k,omega_XX,label='XX',marker='s',color='magenta',**k
w)

plt.legend()
plt.xlabel('k')
plt.ylabel('$\hat{\omega}$')
plt.show()
```



Next we use pyPRISM to calculate the direct correlation functions, $\hat{c}_{i,j}(k)$ for the original, two component system.

In [3]:
```
sys1 = pyPRISM.System(['A','B'],kT=1.0)
sys1.domain                           = pyPRISM.Domain(length=1024,dr=0.1)
sys1.diameter[sys1.types]              = 1.0
sys1.potential[sys1.types,sys1.types] = pyPRISM.potential.HardSphere()
sys1.density['A']                     = 0.3*N_A/(N_A+N_B)
sys1.density['B']                     = 0.3*N_B/(N_A+N_B)
sys1.omega['A','A']                   = pyPRISM.omega.FromArray(omega = om
ega_AA,k = omega_AA_k)
sys1.omega['A','B']                   = pyPRISM.omega.FromArray(omega = om
ega_AB,k = omega_AB_k)
sys1.omega['B','B']                   = pyPRISM.omega.FromArray(omega = om
ega_BB,k = omega_BB_k)
sys1.closure[sys1.types,sys1.types]   = pyPRISM.closure.PY()
PRISM = sys1.solve()

ck1 = PRISM.directCorr

kw['markevery'] = 10
plt.plot(sys1.domain.k,ck1['A','A'],label='AA',marker='o',color='red',**
kw)
plt.plot(sys1.domain.k,ck1['A','B'],label='AB',marker='^',color='green',
**kw)
plt.plot(sys1.domain.k,ck1['B','B'],label='BB',marker='d',color='blue',*
*kw)
plt.xlim(0,10)
plt.legend(loc='lower right')
plt.xlabel('k')
plt.ylabel('$\hat{c}$')
plt.show()
```

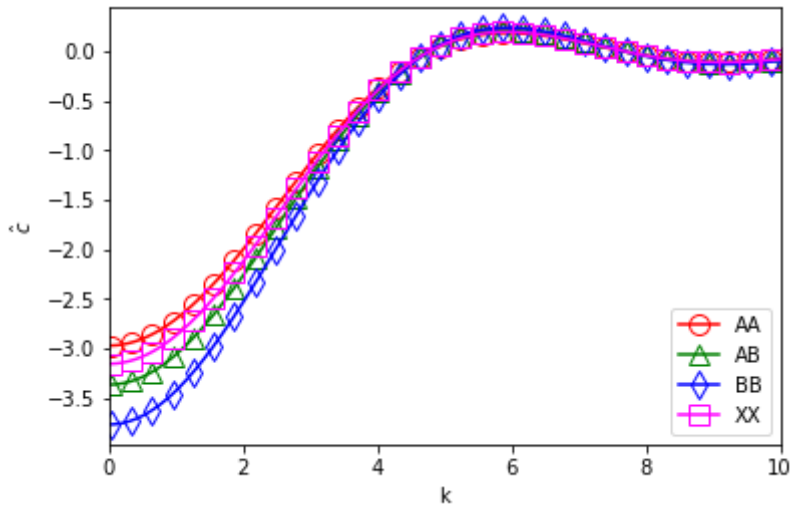
```
0:  |F(x)| = 0.109134; step 1; tol 0.00105024
1:  |F(x)| = 0.00272055; step 1; tol 0.000559286
2:  |F(x)| = 2.04581e-06; step 1; tol 5.08928e-07
```



We repeat this process for the coarsened, 1-component system.

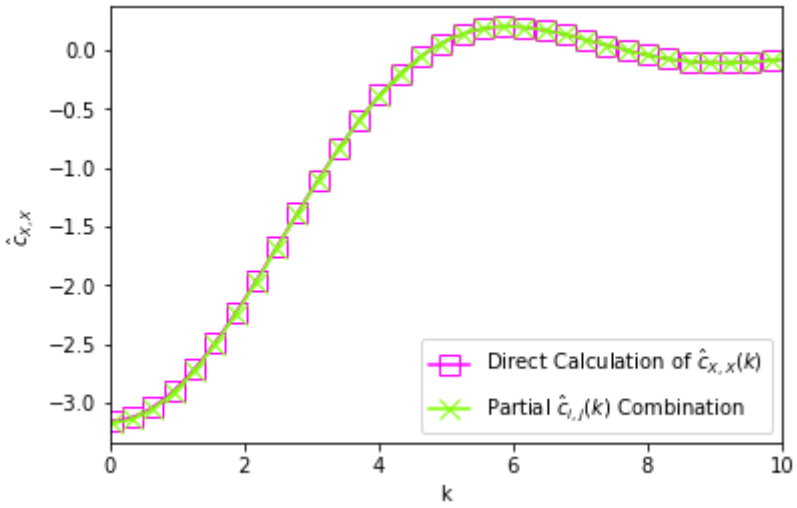
In [4]:
```
sys2 = pyPRISM.System(['X'],kT=1.0)
sys2.domain                                = pyPRISM.Domain(length=1024,dr=0.
1)
sys2.density[sys2.types]                   = sys1.density.total
sys2.diameter[sys2.types]                  = 1.0
sys2.potential[sys2.types,sys2.types] = pyPRISM.potential.HardSphere()
sys2.closure[sys2.types,sys2.types]    = pyPRISM.closure.PY()
sys2.omega[sys2.types,sys2.types]      = pyPRISM.omega.FromArray(omega =
omega_XX,k = omega_XX_k)
PRISM2 = sys2.solve()

ck2 = PRISM2.directCorr

kw['markevery'] = 10
plt.plot(sys1.domain.k,ck1['A','A'],label='AA',marker='o',color='red',**
kw)
plt.plot(sys1.domain.k,ck1['A','B'],label='AB',marker='^',color='green',
**kw)
plt.plot(sys1.domain.k,ck1['B','B'],label='BB',marker='d',color='blue',*
*kw)
plt.plot(sys2.domain.k,ck2['X','X'],label='XX',marker='s',color='magent
a',**kw)
plt.xlim(0,10)
plt.legend(loc='lower right')
plt.xlabel('k')
plt.ylabel('$\hat{c}$')
plt.show()
```

```
0:  |F(x)| = 0.0623092; step 1; tol 0.00118424
1:  |F(x)| = 0.00134571; step 1; tol 0.000419801
2:  |F(x)| = 7.60538e-07; step 1; tol 2.87461e-07
```

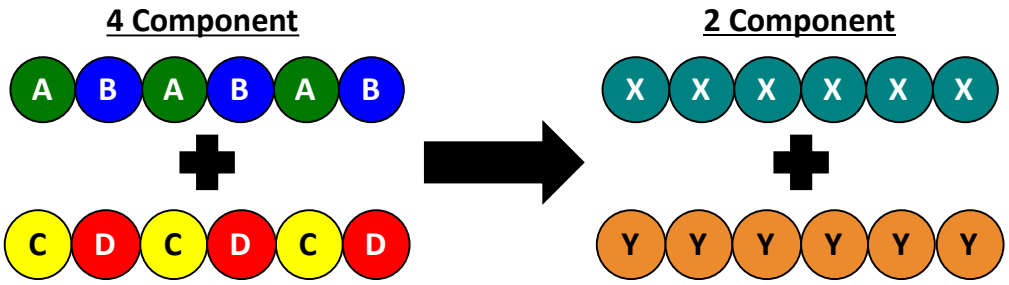


Now, we will test our ability to combine correlation functions.

In [5]:
```
rho1 = sys1.density.pair
rho1_total = sys1.density.total

ck3  = 0
ck3 +=     rho1['A','A']*ck1['A','A']
ck3 += 2.0*rho1['A','B']*ck1['A','B']
ck3 +=     rho1['B','B']*ck1['B','B']
ck3 /= (rho1_total*rho1_total)

plt.plot(sys1.domain.k,ck2['X','X'],label='Direct Calculation of $\hat
{c}_{X,X}(k)$',marker='s',color='magenta',**kw)
plt.plot(sys1.domain.k,ck3,label='Partial $\hat{c}_{i,j}(k)$ Combinatio
n',marker='x',color='chartreuse',**kw)

plt.xlim(0,10)
plt.xlabel('k')
plt.ylabel('$\hat{c}_{X,X}$')
plt.legend(loc='best')
plt.show()
```



4-Component Example



Considering the system shown above, we define the following super-site types:

$$X = A + B$$

$$Y = C + D$$

Based on this, the combined direct correlation functions would be

$$\rho_X \rho_X \hat{c}_{X,X} = \rho_A \rho_A \hat{c}_{A,A}(k) + 2\rho_A \rho_B \hat{c}_{A,B}(k) + \rho_B \rho_B \hat{c}_{B,B}(k)$$

$$\rho_Y \rho_Y \hat{c}_{Y,Y} = \rho_C \rho_C \hat{c}_{C,C}(k) + 2\rho_C \rho_D \hat{c}_{C,D}(k) + \rho_D \rho_D \hat{c}_{D,D}(k)$$

$$\rho_X \rho_Y \hat{c}_{X,Y} = \rho_A \rho_C \hat{c}_{A,C}(k) + \rho_A \rho_D \hat{c}_{A,D}(k) + \rho_B \rho_C \hat{c}_{B,C}(k) + \rho_B \rho_D \hat{c}_{B,D}(k)$$

Application of combining correlation functions: Overall $\hat{\chi}_{i,j}(k)$

Consider the second, four component example above. One motivation for combining the correlation functions this way is so that and overall $\hat{\chi}(k)$ between the two chains can be calculated. Assuming that all sites have equal diameters, the definition of the non-dimensionalized PRISM chi [1] is

$$\chi_{i,j}^{eff} = \hat{\chi}_{i,j}(k \to 0) = \frac{1}{2}\rho \left[\hat{c}_{i,i}(k) + \hat{c}_{j,j}(k) - 2\hat{c}_{i,j}(k)\right]$$

where i and j represent site-types, $\rho = \sum_i \rho_i$ is the total density, and $\hat{c}_{i,j}(k)$ is the value of the direct correlation function between site-types i and j at wavenumber k. There is an expression for systems with unequal site-type diameters, but we show this expression here for simplicity.

While it might seem intuitive to combine the $\hat{\chi}(k)$ values directly, it is more rigorous to combine the individual correlation functions. The $\chi_{X,Y}^{eff}$ for the four component example would be defined as

$$\chi_{X,Y}^{eff} = \hat{\chi}_{X,Y}(k \to 0) = \frac{1}{2}\rho \left[\hat{c}_{X,X}(k) + \hat{c}_{Y,Y}(k) - 2\hat{c}_{X,Y}(k)\right]$$

with the $\hat{c}_{X,X}(k)$, $\hat{c}_{Y,Y}(k)$, and $\hat{c}_{X,Y}(k)$ as defined above.

Citations

[1] Kenneth S. Schweizer and John G. Curro, Integral equation theory of the structure and thermodynamics of polymer blends, The Journal of Chemical Physics 91, 5059 (1989); doi: 10.1063/1.457598