ZelCa (SAGE worksheet)

Values for the growth factor and rate of growth at desired redshift.

Those can be read off from the ZA output for xi: Xi.dat, line 2, columns 4 and 5.

```
growth=7.62410522e-01
fFactor=7.44321155e-01
```

```
dir='/media/work/PW3/ZelCa/executable/'
outDir='/media/work/PW3/ZelCa/executable/'
```

```
import matplotlib as mpl
mpl.rcParams['font.family'] = 'serif'
```

Tree level results

The file "linear.dat" contains the tree-level results in SPT for the real space ξ, the redshift space multipoles, as well as several funcionals of ξ needed for the calculation of the 3-point function, ζ. The lines below illustrate how one extracts those n-point functions from the file and plots them.

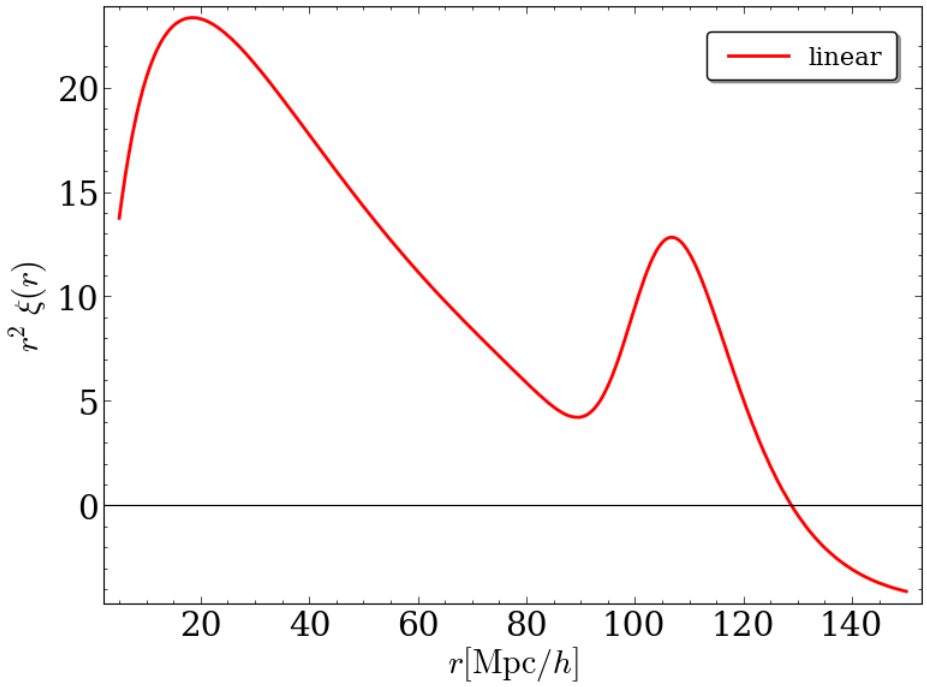
```
f = open(dir+'linear.dat', 'r')
qArr=[]
xiLArr=[]
xiL0Arr=[]
xiL2Arr=[]
xiL4Arr=[]
gradNablaXiArr=[]
gradXiArr=[]
HArr=[]
GArr=[]

line=f.readline()
line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
    qArr.append(float(xy[0]))
    xiLArr.append(float(xy[3]))
    xiL2Arr.append(float(xy[4]))
    xiL4Arr.append(float(xy[5]))
    gradNablaXiArr.append(float(xy[6]))
    gradXiArr.append(float(xy[7]))
    HArr.append(float(xy[8]))
    GArr.append(float(xy[9]))
    line=f.readline()
```

-- 2-pt functions at tree level

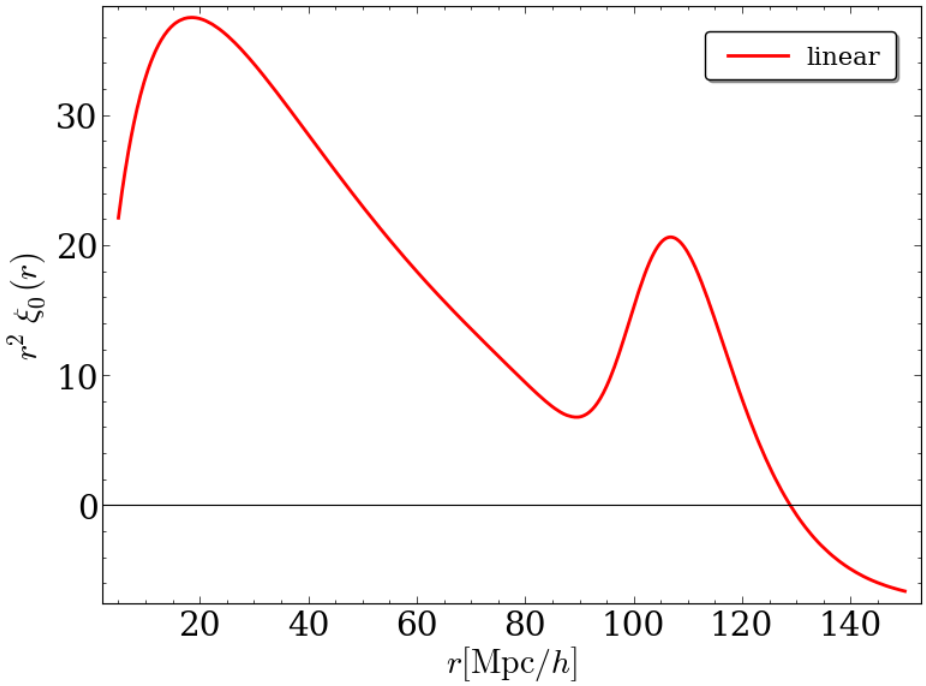
```
from sage.gsl.all import spline

data=zip(qArr,xiLArr)
xiL = spline([(d[0],growth*growth*d[1]) for d in data]);

data=zip(qArr,xiLArr)
xiL0 = spline([(d[0],growth*growth*d[1]*(1 + 2/3*fFactor + 1/5*fFactor^2)) for d in data]);

data=zip(qArr,xiL2Arr)
```

```
xiL2 = spline([(d[0],growth*growth*d[1]*(4/3*fFactor + 4/7*fFactor^2)) for d in data]);

data=zip(qArr,xiL4Arr)
xiL4 = spline([(d[0],growth*growth*d[1]*(8/35*fFactor^2)) for d in data]);



data=zip(qArr,gradNablaXiArr)
gradNablaXi = spline([(d[0],growth*growth*d[1]*(-2)*(-1)*(3/2)) for d in data]);

data=zip(qArr,gradXiArr)
gradXi = spline([(d[0],growth*growth*d[1]*(-2)*(3/2)) for d in data]);

data=zip(qArr,HArr)
H = spline([(d[0],growth*growth*d[1]*(-3)) for d in data]);

data=zip(qArr,GArr)
G = spline([(d[0],growth*growth*d[1]) for d in data]);
```

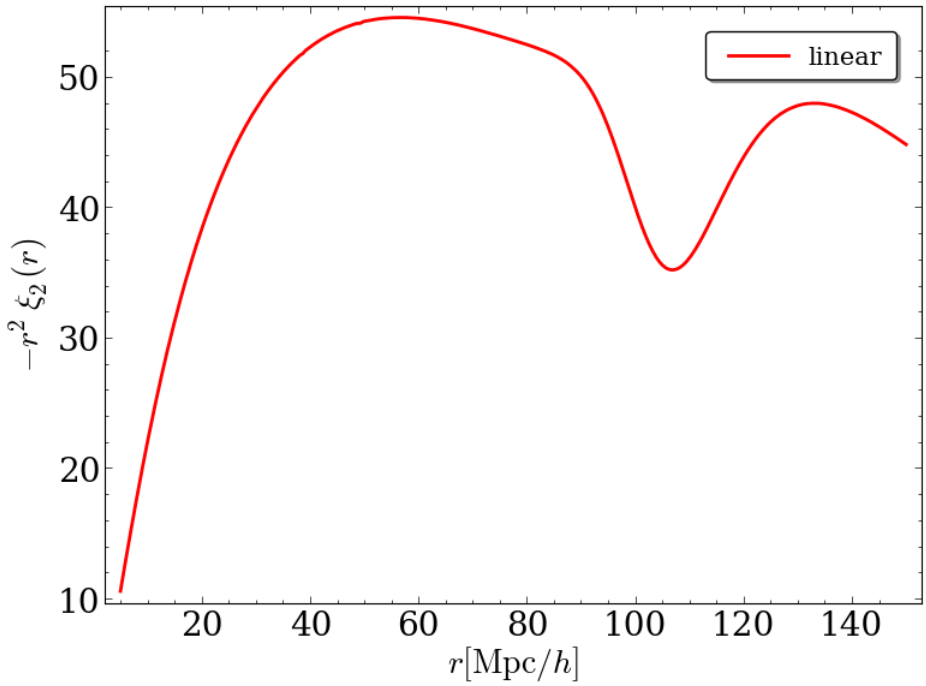
```
def toPlot(x): return xiL(x)*x^2

plotXiL=plot(toPlot,(r,5,150),frame=True,axes_labels=[u"$r [\mathrm{Mpc}/h]$",u"$r^2\\xi(r)$"],fontsize=20,
color='red',legend_label='linear',thickness=2)
plotXiL.xmin(5)
plotXiL.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3,
       fancybox='True',shadow='True',font_weight='medium')

plotXiL
```




```
def toPlot(x): return xiL0(x)*x^2

plotXiL0=plot(toPlot,(r,5,150),frame=True,axes_labels=[u"$r [\mathrm{Mpc}/h]$",u"$r^2
\\xi_0(r)$"],fontsize=20, color='red',legend_label='linear',thickness=2)
plotXiL0.xmin(5)
plotXiL0.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3
    ,fancybox='True',shadow='True',font_weight='medium')
plotXiL0
```

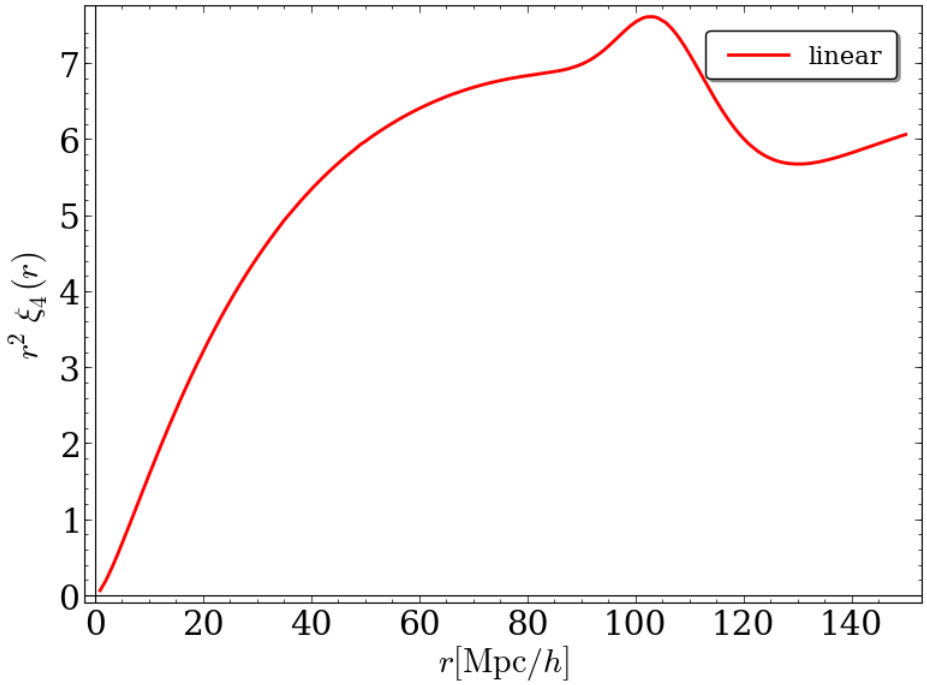



```
def toPlot(x): return -xiL2(x)*x^2

plotXiL2=plot(toPlot,(r,5,150),frame=True,axes_labels=[u"$r [\mathrm{Mpc}/h]$",u"$-r^2
\\xi_2(r)$"],fontsize=20, color='red',legend_label='linear',thickness=2)
plotXiL2.xmin(5)
plotXiL2.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3
    ,fancybox='True',shadow='True',font_weight='medium')
plotXiL2
```




```
def toPlot(x): return xiL4(x)*x^2

plotXiL4=plot(toPlot,(r,1,150),frame=True,axes_labels=[u"$r [\mathrm{Mpc}/h]$",u"$r^2
\\xi_4(r)$"],fontsize=20, color='red',legend_label='linear',thickness=2)
plotXiL4.xmin(1)
plotXiL4.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3
```

```
    ,fancybox='True',shadow='True',font_weight='medium')
plotXiL4
```



-- 3-point function at tree level

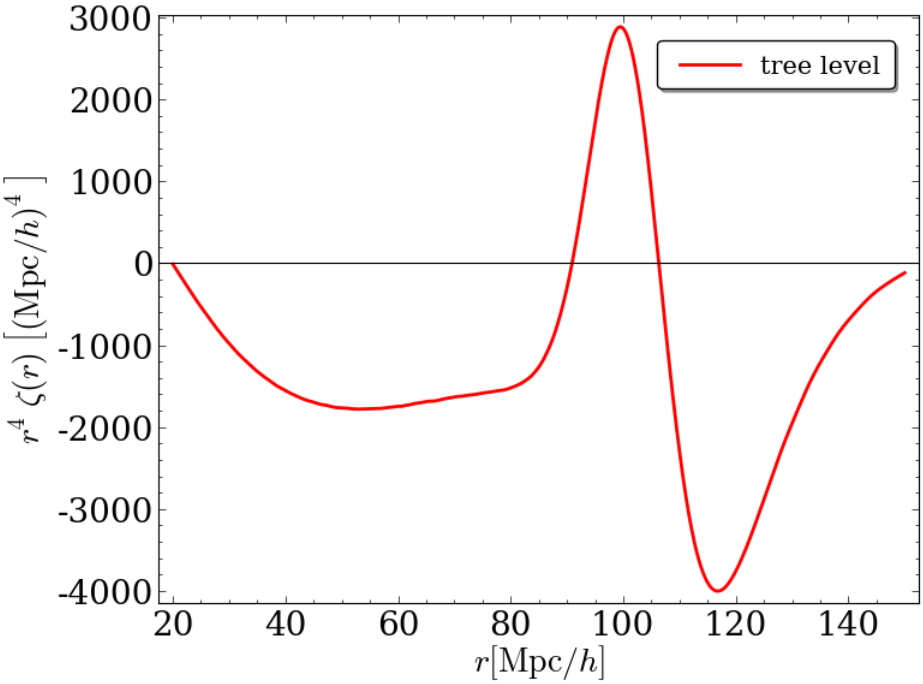
```
%cython
cargs -O3

from __main__ import xiL,gradXi,gradNablaXi,H,G

def dot(x,y): return sum([d[0]*d[1] for d in zip(x,y)])
def DIAG(i,j):
    if (i==j):
        return 1.0
    else:
        return 0.0

def bracket(x13, X13, x23, X23):
    return (10.0/7.0*xiL(x13)*xiL(x23) +
            (gradXi(x13)*gradNablaXi(x23) + gradXi(x23)*gradNablaXi(x13))*
            (dot(X13,X23)/x13/x23) +
             4.0/7.0*sum(
                      [  (H(x13)*X13[i]*X13[j]/x13/x13 + G(x13)*DIAG(i,j))*
                         (H(x23)*X23[i]*X23[j]/x23/x23 + G(x23)*DIAG(i,j))

                         for i in range(0,3)  for j in range(0,3)
                      ]
                    )
           )

def fullBracket(x13, X13, x23, X23, x12, X12):
    return (bracket(x13, X13, x23, X23) + bracket(x13, X13, x12, X12) +
                bracket(x12, X12,x23, [-d for d in X23]))

def subtract(x,y): return x-y

def zeta(r1, r2, mu):
    v3 = map(subtract,[0, 0, r1],[0, sqrt(1.0 - mu**2)* r2, mu* r2])
    return fullBracket(r1, [0, 0, r1], r2, [0,sqrt(1.0 - mu**2)* r2, mu*r2], sqrt(dot(v3,v3)), v3)

def bracketZA(x13, X13, x23, X23):
    return   (3.0/7.0*xiL(x13)*xiL(x23) -
              3.0/7.0*sum(
```

```
                        [  (H(x13)*X13[i]* X13[j]/x13/x13 + G(x13)*DIAG(i,j))*
                           (H(x23)*X23[i] *X23[j]/x23/x23 + G(x23)*DIAG(i,j))

                           for i in range(0,3) for j in range(0,3)
                        ]
                    )
             )


def fullBracketZA(x13, X13, x23, X23, x12, X12):
    return  ( bracketZA(x13, X13, x23, X23) +
              bracketZA(x13, X13, x12, X12) +
              bracketZA(x12, X12, x23, [-d for d in X23])
            )


def zetaZAres(r1, r2, mu):
    v3 =map(subtract,[0, 0, r1],[0, sqrt(1.0 - mu**2)*r2, mu*r2])
    return fullBracketZA(r1, [0, 0, r1], r2, [0, sqrt(1.0 - mu**2)* r2, mu*r2], sqrt(dot(v3,v3)), v3)
```

__home_use...9_code_sage11_spyx.c __home_use...ode_sage11_spyx.html

```
def toPlot(r):
    return zeta(r,r,1/2)/(r^(-4))

plot3ptTree=plot(toPlot,(r,20,150),frame=True,axes_labels=[u"$r [\mathrm{Mpc}/h]$",
    r"$r^4\zeta(r)\ \left[(\mathrm{Mpc}/h)^4\right]$"],fontsize=20, color='red',legend_label='tree level'
    ,thickness=2)


plot3ptTree.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,
    handlelength=3,fancybox='True',shadow='True',font_weight='medium')
```

```
plot3ptTree
```



N-point functions in the Zeldovich Approximation (ZA)

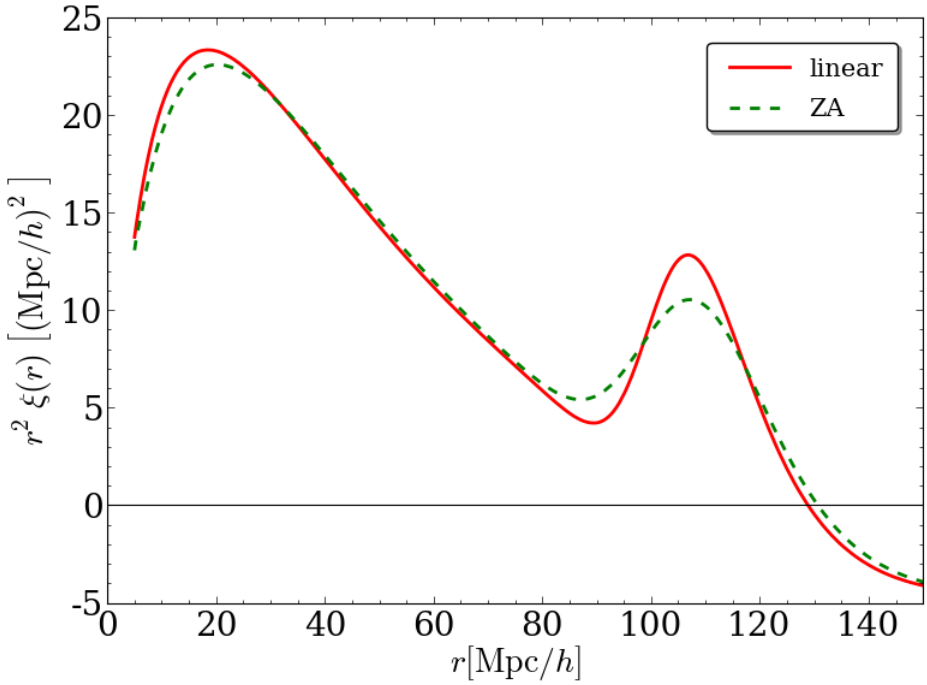
-- 2-point functions

real space

```
f = open(dir+'Xi.dat', 'r')
```

```
rArr=[]
xiZAArr=[]

line=f.readline()
line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
    if ( (abs(float(xy[3]))<0.05) | (abs(float(xy[2])/float(xy[1]))<0.03) ):
        rArr.append(float(xy[0]))
        xiZAArr.append(float(xy[1]))
    line=f.readline()

from sage.gsl.all import spline

data=zip(rArr,xiZAArr)
xiZA = spline(data);
```

```
def toPlot(x): return xiZA(x)*x^2

plotXiZA=plot(toPlot,(r,5,150),frame=True,
     axes_labels=[u"$r [\mathrm{Mpc}/h]$",r"$r^2\xi(r)\ \left[(\mathrm{Mpc}/h)^2\right]$"],fontsize=20,
color='green',legend_label='ZA',linestyle="--",axes=True,thickness=2)
plotXiZA.xmin(5)
plotXiZA.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3,
      fancybox='True',shadow='True',font_weight='medium')
show(plotXiL+plotXiZA,xmin=0,xmax=150,ymin=-5,ymax=25,axes_pad=0,legend_labelspacing=0.7)
```



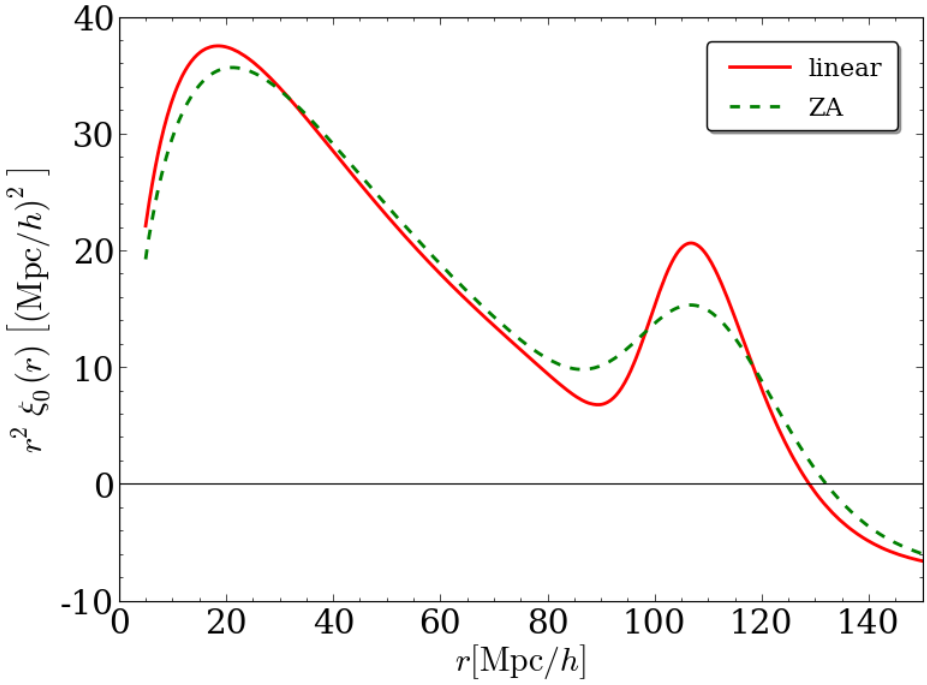

```
show(plotXiL+plotXiZA,xmin=0,xmax=150,ymin=-5,ymax=25,axes_pad=0,legend_labelspacing=0.7,filename=outDir+
   'xi.pdf')
```

Monopole

```
f = open(dir+'XiRS0.dat', 'r')
rArr=[]
xiZA0Arr=[]

line=f.readline()
line=f.readline()
line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
```

```
    if ( (abs(float(xy[3]))<0.05) | (abs(float(xy[2])/float(xy[1]))<0.03) ):
        rArr.append(float(xy[0]))
        xiZA0Arr.append(float(xy[1]))
    line=f.readline()

from sage.gsl.all import spline

data=zip(rArr,xiZA0Arr)
xiZA0 = spline(data);
```

```
def toPlot(x): return xiZA0(x)*x^2

plotXiZA0=plot(toPlot,(r,5,150),frame=True,
    axes_labels=[u"$r [\mathrm{Mpc}/h]$",r"$r^2\xi_0(r)\ \left[(\mathrm{Mpc}/h)^2\right]$"],fontsize=20,
    color='green',legend_label='ZA',linestyle="--",axes=True,thickness=2)
plotXiZA0.xmin(5)

plotXiZA0.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3,
    fancybox='True',shadow='True',font_weight='medium')
show(plotXiL0+plotXiZA0,xmin=0,xmax=150,ymin=-10,ymax=40,axes_pad=0,legend_labelspacing=0.7)
```



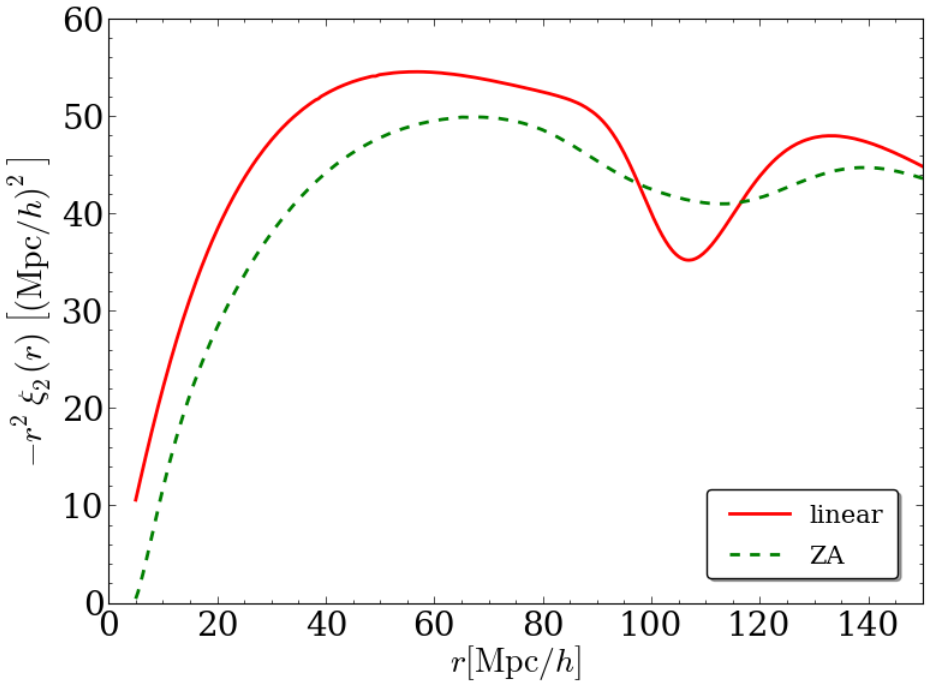

```
show(plotXiL0+plotXiZA0,xmin=0,xmax=150,ymin=-10,ymax=40,axes_pad=0,legend_labelspacing=0.7,filename=outDir
   +'xiRS0.pdf')
```

Quadrupole

```
f = open(dir+'XiRS2.dat', 'r')
rArr=[]
xiZA2Arr=[]

line=f.readline()
line=f.readline()
line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
    if ( (abs(float(xy[3]))<0.05) | (abs(float(xy[2])/float(xy[1]))<0.03) ):
        rArr.append(float(xy[0]))
        xiZA2Arr.append(float(xy[1]))
    line=f.readline()

from sage.gsl.all import spline
```

```
data=zip(rArr,xiZA2Arr)
xiZA2 = spline(data);
```

```
def toPlot(x): return -xiZA2(x)*x^2

plotXiZA2=plot(toPlot,(r,5,150),frame=True,
   axes_labels=[u"$r [\mathrm{Mpc}/h]$",r"$-r^2\xi_2(r)\ \left[(\mathrm{Mpc}/h)^2\right]$"],fontsize=20,
    color='green',legend_label='ZA',linestyle="--",axes=False,thickness=2)
plotXiZA2.xmin(5)

plotXiZA2.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3
    ,fancybox='True',shadow='True',font_weight='medium')

show(plotXiL2+plotXiZA2,xmin=0,xmax=150,ymin=0,ymax=60,axes_pad=0,legend_labelspacing=0.7,
   legend_loc='lower right')
```



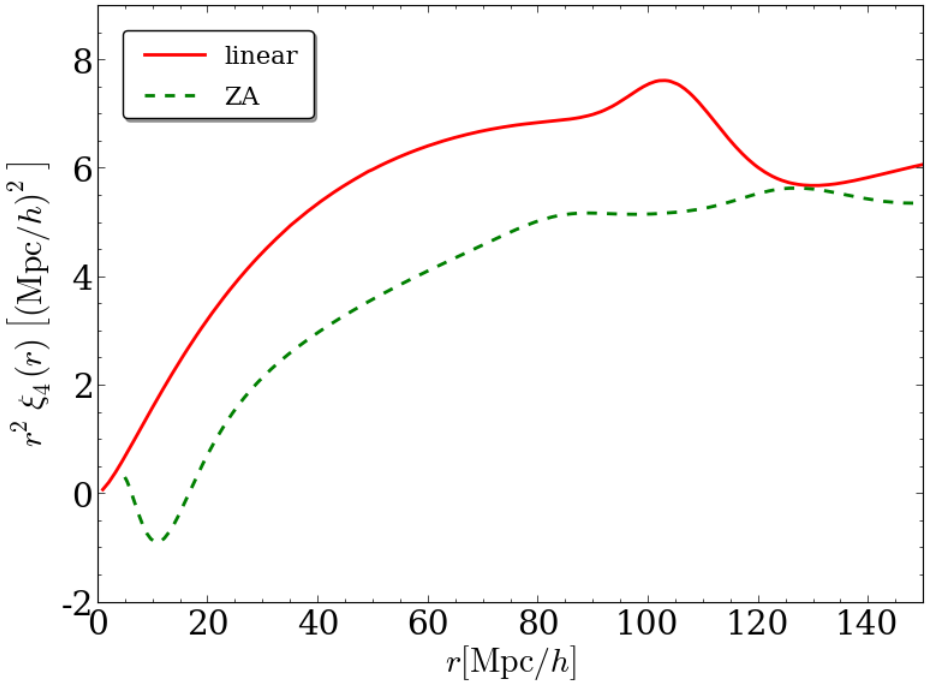

```
show(plotXiL2+plotXiZA2,xmin=0,xmax=150,ymin=0,ymax=60,axes_pad=0,legend_labelspacing=0.7,filename=outDir
    +'xiRS2.pdf',legend_loc='lower right')
```

Hexadecapole

```
f = open(dir+'XiRS4.dat', 'r')
rArr=[]
xiZA4Arr=[]

line=f.readline()
line=f.readline()
line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
    if ( (abs(float(xy[3]))<0.05) | (abs(float(xy[2])/float(xy[1]))<0.03) ):
        rArr.append(float(xy[0]))
        xiZA4Arr.append(float(xy[1]))
    line=f.readline()

from sage.gsl.all import spline

data=zip(rArr,xiZA4Arr)
xiZA4 = spline(data);
```

```
def toPlot(x): return xiZA4(x)*x^2
```

```
plotXiZA4=plot(toPlot,(r,5,150),frame=True,
    axes_labels=[u"$r [\mathrm{Mpc}/h]$",r"$r^2\xi_4(r)\ \left[(\mathrm{Mpc}/h)^2\right]$"],fontsize=20,
    color='green',legend_label='ZA',linestyle="--",axes=False,thickness=2)
plotXiZA4.xmin(5)

plotXiZA4.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3
    ,fancybox='True',shadow='True',font_weight='medium')
show(plotXiL4+plotXiZA4,xmin=0,xmax=150,ymin=-2,ymax=9,axes_pad=0,legend_labelspacing=0.7)
```



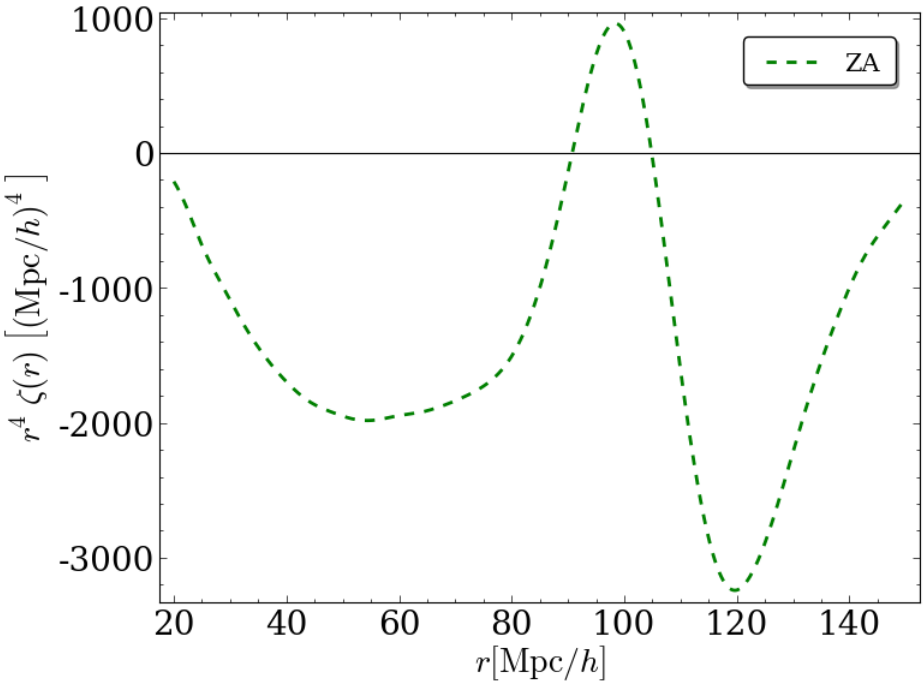

```
show(plotXiL4+plotXiZA4,xmin=0,xmax=150,ymin=-2,ymax=9,axes_pad=0,legend_labelspacing=0.7,filename=outDir
   +'xiRS4.pdf')
```

-- 3-point function

```
f = open(dir+'Zeta.dat', 'r')
rArr=[]
threePtArr=[]

line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
    if ( (abs(float(xy[5]))<0.05) | (abs(float(xy[4])/float(xy[3]))<0.03) ):
        rArr.append(float(xy[0]))
        threePtArr.append(float(xy[3]))
    line=f.readline()

from sage.gsl.all import spline

data=zip(rArr,threePtArr)
threePt = spline(data);
```

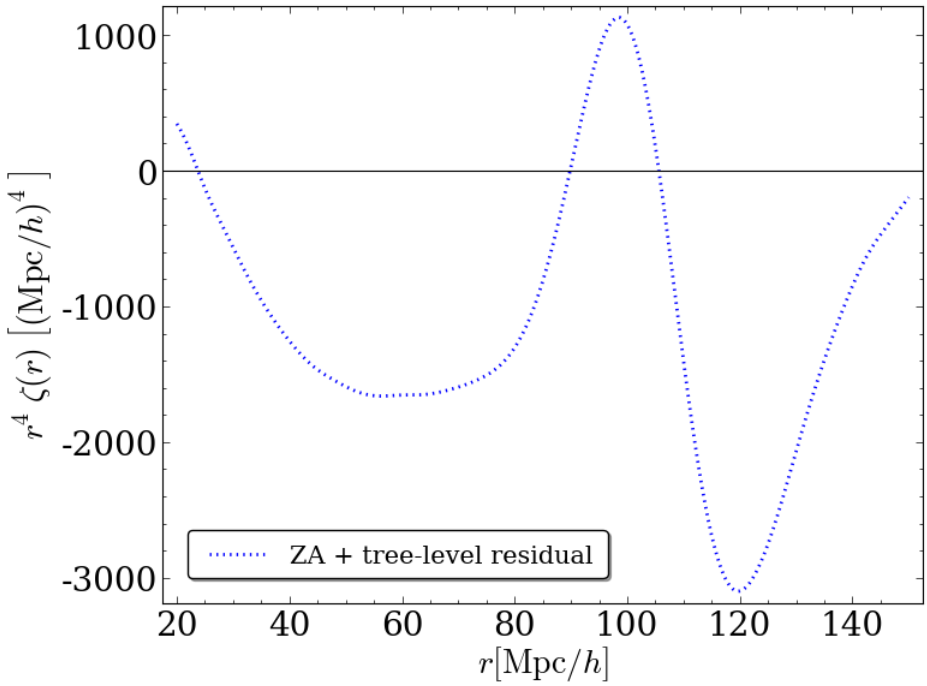
```
def toPlot(r):
    return threePt(r)/(r^(-4))

plot3ptZA=plot(toPlot,(r,20,150),frame=True,
  axes_labels=["$r[\mathrm{Mpc}/h]$",r"$r^4\zeta(r)\ \left[(\mathrm{Mpc}/h)^4\right]$"],fontsize=20,
  color='green',linestyle='--',legend_label='ZA',thickness=2)

plot3ptZA.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1,handlelength=3
    ,fancybox='True',shadow='True',font_weight='medium')
```

```
plot3ptZA
```

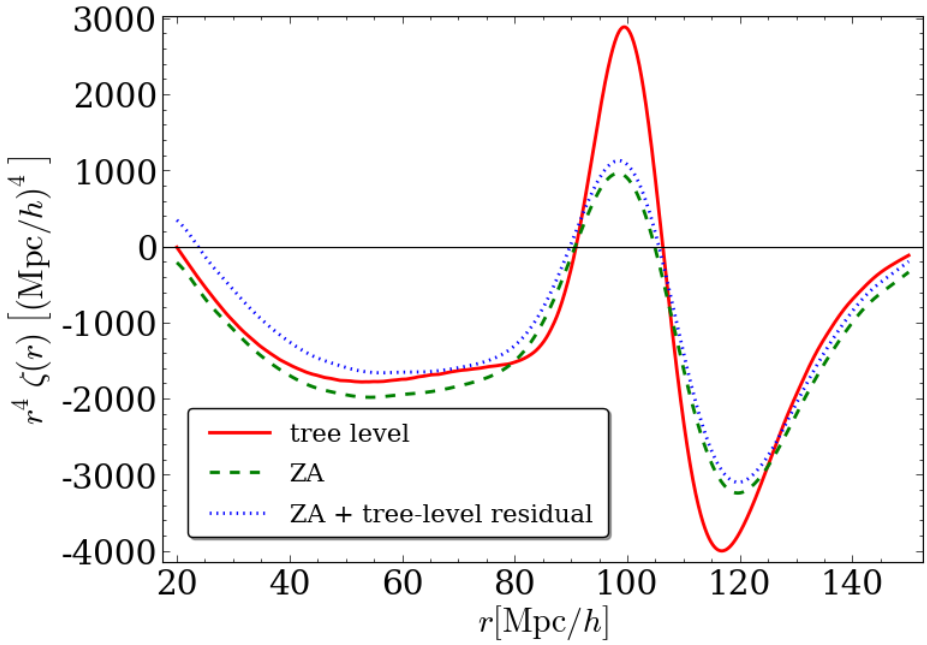



```
def toPlot(r):
    return (threePt(r)+zetaZAres(r,r,1/2))/(r^(-4))

plot3ptZAplusResidual=plot(toPlot,(r,20,150),frame=True,
axes_labels=[u"$r [\mathrm{Mpc}/h]$",r"$r^4\zeta(r)\ \left[(\mathrm{Mpc}/h)^4\right]$"],fontsize=20,
color='blue',linestyle=':',legend_label='ZA + tree-level residual',thickness=2)


plot3ptZAplusResidual.set_legend_options(font_size=15,font_family='serif',back_color='white',borderaxespad=1
    ,handlelength=3,fancybox='True',shadow='True',font_weight='medium')
```

```
plot3ptZAplusResidual
```




```
show(plot3ptTree+plot3ptZA+plot3ptZAplusResidual,legend_labelspacing=0.7,legend_borderaxespad=0.005
    ,aspect_ratio=0.0135)
```




```
show(plot3ptTree+plot3ptZA+plot3ptZAplusResidual,legend_labelspacing=0.7,legend_borderaxespad=0.005
   ,filename=outDir+'Zeta.pdf',aspect_ratio=0.0135)
```

```

```

2-dim RS ξ

```
import matplotlib.pyplot as plt
from pylab import *
from sage.plot.matrix_plot import MatrixPlot
```

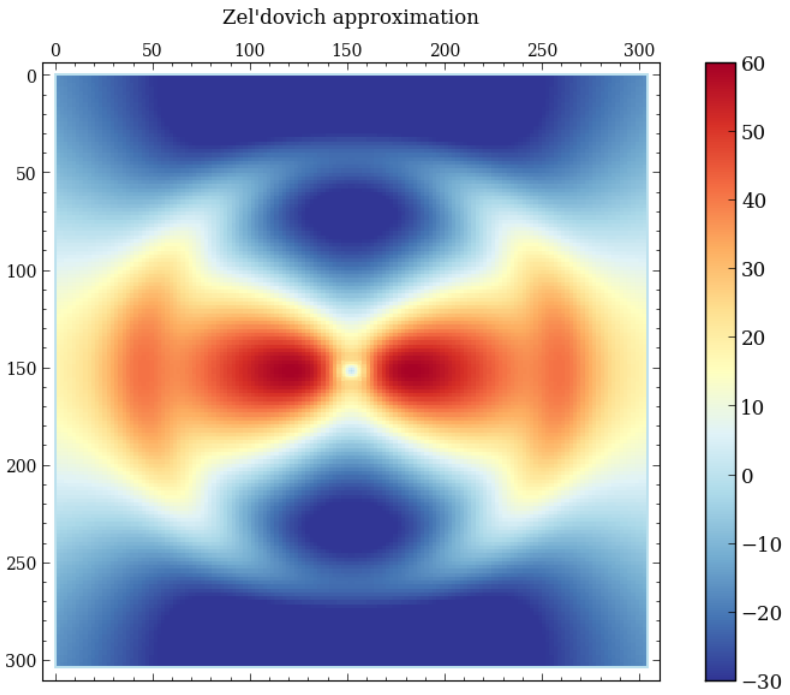
```
X=[]
Y=[]
Z=[]
f = open(dir+'rs2d.dat', 'r')
line=f.readline()
line=f.readline()
line=f.readline()
line=f.readline()
while(line !=''):
    xy=line.split(' ')
    xx=float(xy[0])
    yy=float(xy[1])
    X.append(int(xx))
    Y.append(int(yy))
    Z.append(float(xy[2])*(xx*xx+yy*yy))

    line=f.readline()
```

```
arr=[[0 for i in range(-152,153)] for j in range(-152,153)]
```

```
for i in range(len(X)):
        arr[X[i]+152][Y[i]+152]=Z[i]
        arr[X[i]+152+1][Y[i]+152]=Z[i]
        arr[X[i]+152][Y[i]+152+1]=Z[i]
        arr[X[i]+152+1][Y[i]+152+1]=Z[i]
for i in range(-152,153):
    for j in range(-152,153):
        arr[i+152][j+152]=arr[abs(i)+152][abs(j)+152]
```

```
arr=transpose(arr)
```

```
matrix_plot(arr, axes=False,cmap='RdYlBu_r',colorbar=True,vmin=-30,vmax=60,title="Zel'dovich approximation")
```




```
arrL=[[0 for i in range(-152,153)] for j in range(-152,153)]
```

```
%cython
cargs -O3
from __main__ import xiL0,xiL2,xiL4,arrL
cdef extern from "gsl/gsl_sf_legendre.h":
    double  gsl_sf_legendre_Pl(int l, double x)
from sage.gsl import *
```

```
from numpy import mod

for i in range(-152,153):
        for j in range(-152,153):
                x=float(i)
                y=float(j)
                r=sqrt(x**2+y**2)
                if (r>0):
                    mu=y/r
                    arrL[i+152][j+152]=(gsl_sf_legendre_Pl(0,mu)*xiL0(r)+
                         gsl_sf_legendre_Pl(2,mu)*xiL2(r)+gsl_sf_legendre_Pl(4,mu)*xiL4(r))*r**2
```

__home_use...9_code_sage43_spyx.c __home_use...ode_sage43_spyx.html

```
matrix_plot(transpose(arrL), axes=False,cmap='RdYlBu_r',colorbar=True,vmin=-30)
```