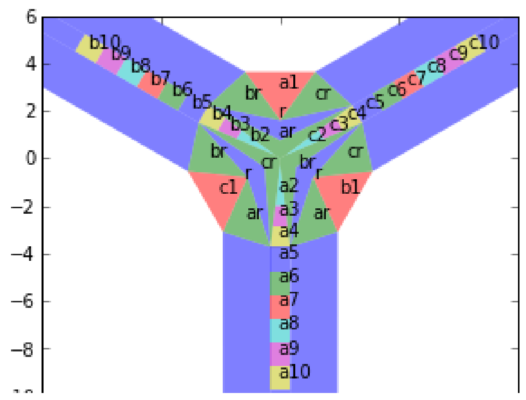
Demo of the `electrode` package

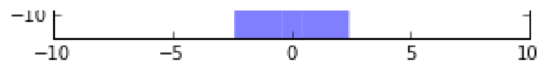
This demo is similar to the "SurfacePattern demo: forward - finite - electric" of Roman Schmied, see http://atom.physik.unibas.ch/people/romanschmied/code/SurfacePattern.php

In [1]:
```
from numpy import *
from matplotlib import pyplot as plt
from scipy import constants
from electrode.transformations import euler_matrix
from electrode import (System, CoverElectrode,
        PolygonPixelElectrode)

set_printoptions(precision=2)
```

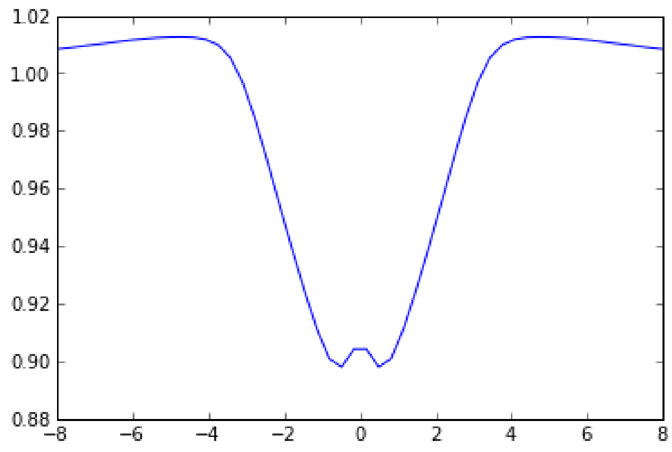
In [2]:
```
def rfjunction(l1, l2, a1, a2, a3):
    a0 = 2*(2**.5-1)
    b0 = 2.
    rmax = 5000
    J = 6
    D = 1
    nmax = 0
    cover_height = 150
    infp = lambda t: matrix([cos(t)*rmax, sin(t)*rmax])
    rot = lambda t: matrix([[cos(t), -sin(t)], [sin(t), cos(t)]])
    rot3 = lambda p: reduce(lambda a,b: a+b, ([[i*rot(t) for i in q] for q in p]
        (0., 2*pi/3, 4*pi/3)), [])
    els = []
    els.append(["r", rot3([
        [infp(3*pi/2), (a0/2+b0, -l2), (a0/2, -l1)],
        [infp(3*pi/2), (-a0/2, -l1), (-a0/2-b0, -l2)],
        [(a0/2, -l1), (a1*3**.5/2, a1*-.5),
            (.5*(a0/2+3**.5*l1), .5*(-3**.5*a0/2+l1)),
            (a2*3**.5/2, a2*-.5)],
        ])])
    sect = []
    sect.append(["r", [
        [[[a0/2, -l1]], [[a0/2+b0, -l2]], [[a1*3**.5/2, -.5*a1]]],
        [[[-a0/2, -l1]], [[-a1*3**.5/2, -.5*a1]], [[-a0/2-b0, -l2]]],
        [(a0/2, -l1)*rot(4*pi/3), (a2*3**.5/2, -.5*a2)*rot(4*pi/3),
        (-1+2**.5+3**.5*l1, 3**.5-6**.5+l1)*rot(4*pi/3)*.5,
        (3**.5/2, .5)*rot(4*pi/3)*a3, (0, 0), (0,
            -a3)*rot(4*pi/3)]]])
    sect.append(["1", [
        [(0, a1), (.25*(-a0-2*b0+2*3**.5*l2), .5*(3**.5*(a0/2+b0)+l2)),
            (-.25*(-a0-2*b0+2*3**.5*l2),
                .5*(3**.5*(a0/2+b0)+l2))]]])
    sect.append(["2", [
        [(a0/4, -(l1+a3)/2), (0, -a3), (-a0/4, -(l1+a3)/2)]]])
    sect.append(["3", [
        [(3*a0/8, -(3*l1+a3)/4), (a0/4, -(l1+a3)/2),
            (-a0/4, -(l1+a3)/2), (-3*a0/8, -(3*l1+a3)/4)]]])
    sect.append(["4", [
        [(a0/2, -l1), (3*a0/8, -(3*l1+a3)/4),
            (-3*a0/8, -(3*l1+a3)/4), (-a0/2, -l1)]]])
    for j in range(J):
        sect.append(["%s" % (j+5), [
            [(-a0/2, -(l1+j*D)), (-a0/2, -(l1+(j+1)*D)),
                (a0/2, -(l1+(j+1)*D)), (a0/2, -(l1+j*D))]]])
    sect.append(["%s" % (J+5), [
        [(-a0/2, -(l1+J*D)), infp(3*pi/2), (a0/2, -(l1+J*D))]]])
    for n, t in zip("abc", (0, 2*pi/3, 4*pi/3)):
        for m, p in sect:
            els.append([n+m, [[r*rot(t) for r in q] for q in p]])
    els = [(n, [c_[array(p)[:,0,:], zeros((len(p),))]
            for p in el]) for n, el in els]
    s = System()
    for n, paths in els:
        s.electrodes.append(PolygonPixelElectrode(name=n, paths=paths,
                nmax=nmax, cover_height=cover_height))
    s.electrodes.append(CoverElectrode(name="m",
```

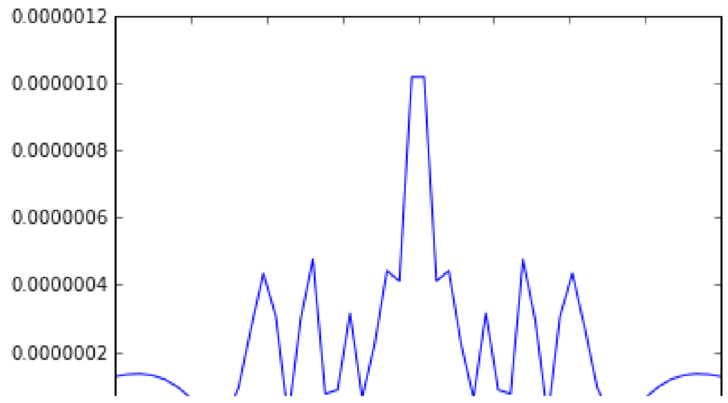
In [3]:
```
s = rfjunction(l1=3.7321851829486046, l2=3.0921935283018334,
               a1=1.5968461727578884, a2=0.7563967699214798,
               a3=0.2630201367283588)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, aspect="equal")
ax.set_xlim((-10,10))
ax.set_ylim((-11,6))
s.plot(ax)
```

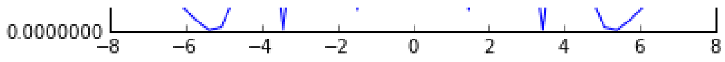


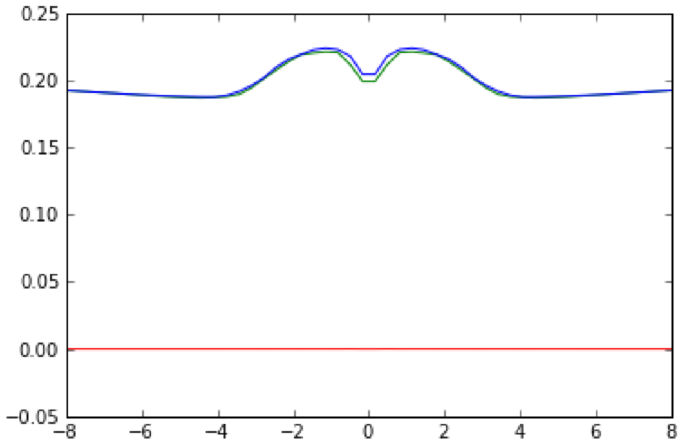


In [4]:
```
# connect b rf, c rf and main rf
s.electrode("r").voltage_rf = 1
s.electrode("br").voltage_rf = 1
s.electrode("cr").voltage_rf = 1

def channel(x):
    """return channel minimum at x=x"""
    x1 = s.minimum((x, tan(pi/6)*abs(x), 1.), axis=(1,))
    x0 = s.minimum(x1, axis=(1, 2))
    return x0

xc = array(map(channel, linspace(-8, 8, 50)))

plt.figure()
_ = plt.plot(xc[:, 0], xc[:, 2])

plt.figure()
_ = plt.plot(xc[:, 0], s.potential(xc))

plt.figure()
m = map(s.modes, xc)
om = array([mi[0] for mi in m])
mm = array([mi[1] for mi in m])
 = plt.plot(xc[:, 0], om[:, 0], "r-", xc[:, 0], om[:, 1], "g-", xc[:, 0], om[:, 2]
```

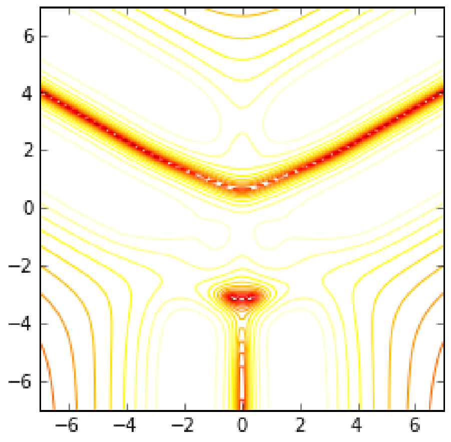






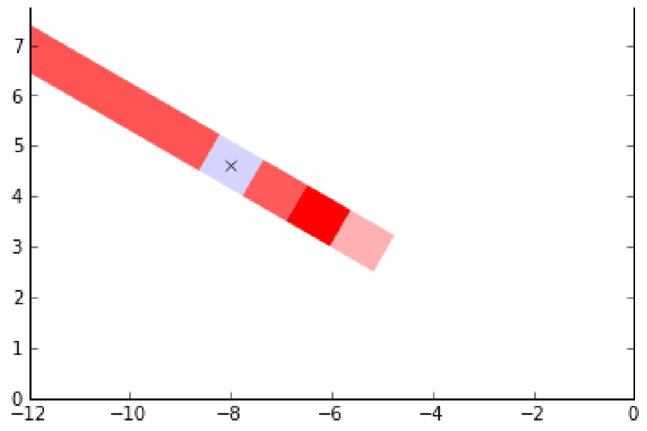


In [5]:
```
n = 100
r = 7
xyz = mgrid[-r:r:1j*n, -r:r:1j*n, 1:2]
xyzt = xyz.transpose((1, 2, 3, 0)).reshape((-1, 3))
p = s.potential(xyzt)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, aspect="equal")
_ = ax.contour(xyz[0].reshape((n,n)), xyz[1].reshape((n,n)),
           log(p).reshape((n,n)),
           20, cmap=plt.cm.hot)
```



In [6]:
```
x0a10, x0b10, x0c10 = (channel(s.electrode(e).paths[0].mean(axis=0)[0]) for e in "
x0 = x0b10
els = "b7 b8 b9 b10 b11".split()
for line in s.analyze_shims([x0], electrodes=els, use_modes=True,
    forces=["x z".split()], curvatures=["xx xz".split()]):
    if type(line) == type(""):
        print line
forces, curvatures, us = line

# add some yy curvature
for eli, ui in zip(els, us[:, 2]):
    s.electrode(eli).voltage_dc = .02*ui

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, aspect="equal")
ax.set_xlim((-12,0))
ax.set_ylim((0,8))
s.plot_voltages(ax)
ax.plot(x0[0], x0[1], "kx")

l = 40e-6
u = 27.
m = 24*constants.atomic_mass
q = 1*constants.elementary_charge
o = 2*pi*55e6
scale = (u*q/l/o)**2/(4*m) # rf pseudopotential energy scale
dc_scale = scale/q # dc energy scale

for line in s.analyze_static(x0, l=l, u=u, o=o, m=m, q=q):
    print line

# undo yy curvature at x0
for eli, ui in zip(els, us[:, 2]):
```

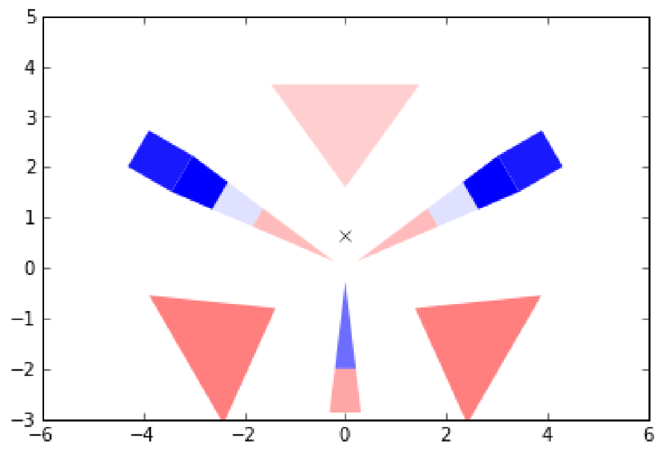
```
shim analysis for points: [[-8.    4.62  1.01]]
 forces: [[&apos;x&apos;, &apos;z&apos;]]
 curvatures: [[&apos;xx&apos;, &apos;xz&apos;]]
 matrix shape: (5, 4), rank: 4
 electrodes: [&apos;b7&apos; &apos;b8&apos; &apos;b9&apos; &apos;b10&apos;
&apos;b11&apos;]
 sh_0x : [ 19.49  57.06  -5.05   3.05  -4.39]
 sh_0z : [-2.07 -6.71 -4.41 -4.64 -4.55]
 sh_0xx: [ 1.83   5.9    3.79 -1.     3.95]
 sh_0xz: [ 13.47  39.1   -8.34   2.49   1.17]
u = 27 V, f = 55 MHz, m = 24 amu, q = 1 qe, l = 40 µm, axis=(0, 1, 2)
analyze point: [-8.    4.62  1.01] ([-319.81  184.64   40.34] µm)
 minimum is at offset: [ 0.   0.   0.]
 rf, dc potentials: 1.3e-07, 0.0087 (4.9e-07 eV, 0.033 eV)
 saddle offset, height: [ 1.54 -0.89  0.24], 0.0082 (0.032 eV)
 pp+dc normal curvatures: [ 0.02  0.18  0.18]
 motion is bounded: True
 pseudopotential modes:
  2.209 MHz, [ 0.87 -0.5   0.  ]
  6.629 MHz, [  4.53e-04  -1.90e-03  -1.00e+00]
  6.71 MHz, [ 0.5   0.87 -0.  ]
  euler angles: [ -9.01e+01   3.00e+01  -3.00e-02]
 mathieu modes:
  2.209 MHz, [ 0.86 -0.5   0.  ]
  6.798 MHz, [ -1.04e-03   7.08e-04   9.35e-01]
  6.882 MHz, [  4.67e-01   8.09e-01  -9.20e-05]
  euler angles: [ -9.00e+01   3.00e+01   6.92e-02]
 heating for 1 nV²/Hz white on each electrode:
  field-noise psd: [  5.72e-12   1.91e-12   2.11e-11] V²/(m² Hz)
  pot-noise psd: 2.67785238592e-19 V²/Hz
  2.209 MHz: ndot=842.9/s, S_E*f=1.693e-05 (V² Hz)/(m² Hz)
  6.629 MHz: ndot=774.9/s, S_E*f=0.0001401 (V² Hz)/(m² Hz)
  6.71 MHz: ndot=208.4/s, S_E*f=3.861e-05 (V² Hz)/(m² Hz)
```



In [7]:
```
x0 = channel(0.) # center of b-c channel
els = "a1 a2 a3 b1 b2 b3 b4 b5 c1 c2 c3 c4 c5".split()
for line in s.analyze_shims([x0], electrodes=els, use_modes=True,
    forces=["x y z".split()], curvatures=["xx yy xy xz yz".split()]):
    if type(line) == type(""):
        print line
forces, curvatures, us = line

# add some xx curvature
for eli, ui in zip(els, us[:, 3]):
    s.electrode(eli).voltage_dc = .02*ui

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, aspect="equal")
ax.set_xlim((-6,6))
ax.set_ylim((-3,5))
s.plot_voltages(ax)
ax.plot(x0[0], x0[1], "kx")

l = 40e-6
u = 27.
m = 24*constants.atomic_mass
q = 1*constants.elementary_charge
o = 2*pi*55e6
scale = (u*q/l/o)**2/(4*m) # rf pseudopotential energy scale
dc_scale = scale/q # dc energy scale

for line in s.analyze_static(x0, l=l, u=u, o=o, m=m, q=q):
    print line

# undo yy curvature at x0
for eli, ui in zip(els, us[:, 2]):
```

```
shim analysis for points: [[ 0.    0.64  0.91]]
 forces: [[&apos;x&apos;, &apos;y&apos;, &apos;z&apos;]]
 curvatures: [[&apos;xx&apos;, &apos;yy&apos;, &apos;xy&apos;, &apos;xz&apos;,
&apos;yz&apos;]]
 matrix shape: (13, 8), rank: 8
 electrodes: [&apos;a1&apos; &apos;a2&apos; &apos;a3&apos; &apos;b1&apos;
&apos;b2&apos; &apos;b3&apos; &apos;b4&apos; &apos;b5&apos; &apos;c1&apos;
&apos;c2&apos; &apos;c3&apos; &apos;c4&apos; &apos;c5&apos;]
 sh_0x : [ -3.20e-14   1.64e-13  -9.54e-14   1.38e+01  -4.69e+00  -3.79e+01
```

```
  -1.87e+01  -9.54e+00  -1.38e+01   4.69e+00   3.79e+01   1.87e+01
   9.54e+00]
 sh_0y : [ 12.06 -52.59  -1.89  -9.26   1.9    4.53 -10.05  -9.99  -9.26   1.9
   4.53 -10.05  -9.99]
 sh_0z : [ -11.69   19.88  -34.09  -13.16  -22.23   38.87  118.32  102.79  -13.16
  -22.23   38.87  118.32  102.79]
 sh_0xx: [ 14.58 -43.62  26.43  38.39  20.39  -9.32 -76.01 -68.6   38.39  20.39
  -9.32 -76.01 -68.6 ]
 sh_0yy: [ 20.17 -23.43  33.89  39.65  14.28 -22.96 -97.63 -86.3   39.65  14.28
 -22.96 -97.63 -86.3 ]
 sh_0xy: [ -1.69e-14   1.19e-13  -5.24e-14  -4.19e+01   2.68e-01  -4.75e+01
  -2.14e+01  -1.01e+01   4.19e+01  -2.68e-01   4.75e+01   2.14e+01
   1.01e+01]
 sh_0xz: [ -7.90e-15  -1.88e-13  -4.18e-15   1.11e+01   1.12e+01  -2.56e+01
  -1.31e+01  -6.74e+00  -1.11e+01  -1.12e+01   2.56e+01   1.31e+01
   6.74e+00]
 sh_0yz: [  -8.47  125.13  -41.14  -59.83   -1.67   12.44  127.01  114.91  -59.83
   -1.67   12.44  127.01  114.91]
u = 27 V, f = 55 MHz, m = 24 amu, q = 1 qe, l = 40 µm, axis=(0, 1, 2)
analyze point: [ 0.    0.64  0.91] ([  0.    25.79  36.24] µm)
 minimum is at offset: [  5.03e-05  -1.80e-05  -3.75e-05]
 rf, dc potentials: 1.7e-10, 0.022 (6.5e-10 eV, 0.083 eV)
 saddle offset, height: [ 0.52  0.14  0.06], 0.0013 (0.005 eV)
 pp+dc normal curvatures: [ 0.02  0.18  0.19]
 motion is bounded: True
 pseudopotential modes:
  2.215 MHz, [  1.00e+00   6.71e-16   6.78e-16]
  6.691 MHz, [ -7.87e-16   1.79e-01   9.84e-01]
  6.888 MHz, [  5.39e-16  -9.84e-01   1.79e-01]
  euler angles: [  7.97e+01   3.09e-14   4.51e-14]
 mathieu modes:
  2.215 MHz, [  9.97e-01   4.24e-16   7.29e-16]
  6.869 MHz, [ -3.03e-15   1.68e-01   9.17e-01]
  7.069 MHz, [ -1.83e-15   9.17e-01  -1.68e-01]
  euler angles: [ -1.00e+02  -1.12e-13   1.74e-13]
 heating for 1 nV²/Hz white on each electrode:
  field-noise psd: [  2.30e-12   1.16e-11   3.15e-11] V²/(m² Hz)
  pot-noise psd: 1.42709049349e-19 V²/Hz
  2.215 MHz: ndot=251.8/s, S_E*f=5.084e-06 (V² Hz)/(m² Hz)
  6.691 MHz: ndot=1367/s, S_E*f=0.0002518 (V² Hz)/(m² Hz)
  6.888 MHz: ndot=670.3/s, S_E*f=0.0001309 (V² Hz)/(m² Hz)
```



5
4
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4
6