In [2]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib notebook
import ipywidgets as widgets
import IPython.display as display
```

```
/Users/jklymak/anaconda2/lib/python2.7/site-packages/matplotlib/__init__.py:878:
UserWarning: axes.color_cycle is deprecated and replaced with axes.prop_cycle; p
lease use the latter.
  warnings.warn(self.msg_depr % (key, alt_key))
```

Airfoils and lift

Recall that around an object in an irrotational flow the lift is given by

$$L = \rho U \Gamma$$

where U is the free-stream velocity, and Γ is the circulation around the object.

In [5]:
```
# airfoils:
def plotAirfoil(ax,offset=0,a=1.2,b=1.):
    th = np.arange(0.,np.pi*2,0.001)
    R=a+1j*0.
    zeta = R*np.exp(1j*th)+offset
    zetaCylinder=zeta
    Zfoil = zeta+b**2/zeta
    ax.plot(np.real(Zfoil),np.imag(Zfoil),'b')
    ax.plot(np.real(zeta),np.imag(zeta),'g')
    ax.plot(np.real(offset),np.imag(offset),'+g')
    ax.axhline(y=0.,ls='--',color='k')
    ax.axvline(x=0.,ls='--',color='k')
    ax.set_aspect(1.)
    # get beta:
    bb = np.where(np.diff(np.signbit(np.imag(zeta))))[0]
    try:
        ax.plot(np.real(zeta[bb[1]]),np.imag(zeta[bb[1]]),'s')
        ax.plot(np.real(Zfoil[bb[1]]),np.imag(Zfoil[bb[1]]),'s')
        beta = -np.arctan2(np.imag(zeta[bb[1]]-offset),np.real(zeta[bb[1]]-offset
))
    except:
        beta=0.
    beta = np.arcsin(np.imag(offset)/a)
    ax.set_title(r'$\beta$'+ '= %1.2f degrees'%(beta*180./np.pi))
    return(beta,zetaCylinder,Zfoil)
```

Zhukowski Airfoil

We can create an airfoil by going from the $\chi = \eta + i\zeta$ plane to the $z = x + iy$ plane using the Zhukoskwi transform:

$$z = \zeta - \frac{b^2}{\zeta}$$

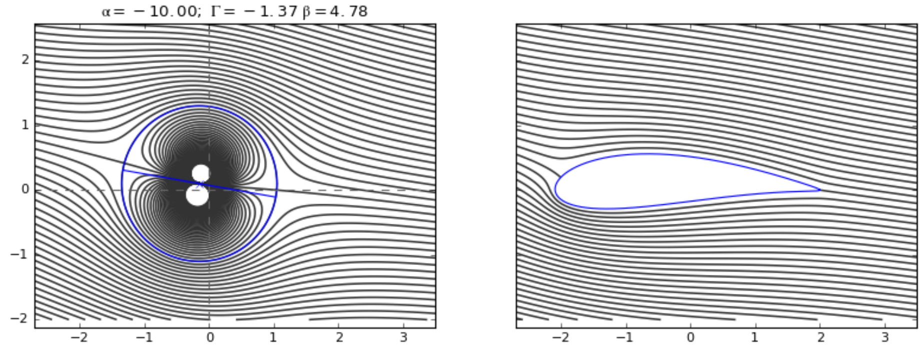
and back again using its inverse:

$$\zeta = \frac{z}{2} \pm \frac{1}{2}\left(z^2 - 4b^2\right)$$

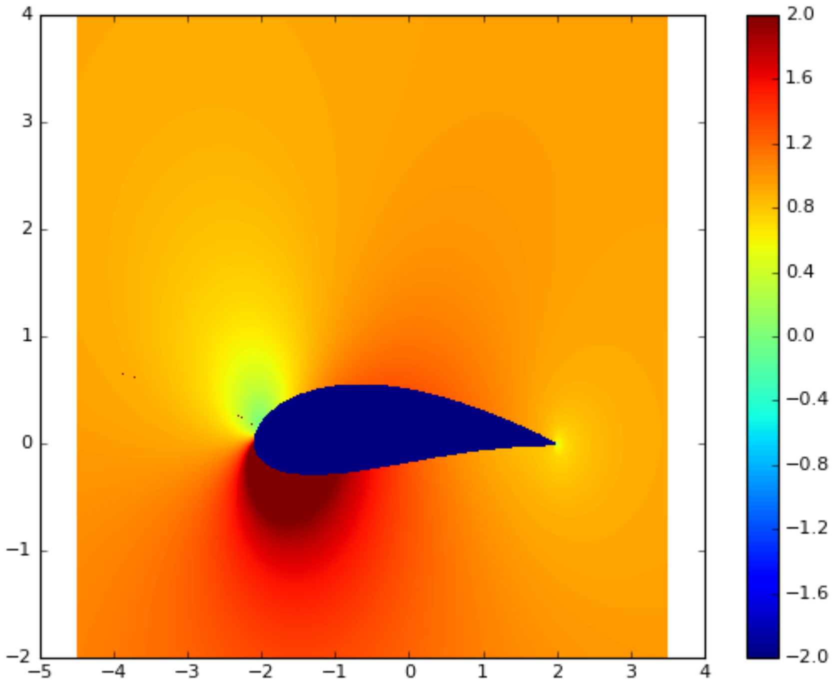
where b is an arbitrary parameter.

In [3]:
```
def plotFoil(axs,gammafac=0.,alpha=0.):

    # co-ordinates:
    b = 1.
    a = 1.2
    offset = -0.15+1j*0.1
    x = np.arange(-4.5,3.5,0.003)
    y = np.arange(-2.,4.,0.003)
    X,Y = np.meshgrid(x,y)
    R = np.sqrt(X**2+Y**2)
    Th = np.arctan2(Y,X)
    Th[Th<0]=Th[Th<0]+np.pi*2.
    Z = X+1j*Y
    U=1.
    ## zz is the cylander co-ordinates.
    zz = (Z-offset)*np.exp(-1j*alpha*np.pi/180.)

    cnts=np.arange(-4.,4.,0.1)
    ax=axs[0]

    ## cylinder co-ordinates:
    zetaCylinder = a*np.exp(1j*np.arange(0.,np.pi*2.,0.01))+offset
    ## where the cylander crosses the x-axis:
    beta = np.arcsin(np.imag(offset)/a)
    # get the gamma necesary for the stagnation point to be at the tip of the air
foil,
    # ie. where the cylader intersects the x axis.  gammafac is 1 if we want this
to satisfy the Jukowski
    # condition
    Gamma = 4*np.pi*U*a*np.sin(alpha*np.pi/180.+beta)*gammafac
    print(Gamma)

    # calculate W (psi = imag(W)) in the cylinader co-ordinate system:
    W = U*(zz+a**2/zz)+(0.+1j)/2./np.pi*np.log(zz/a)*Gamma
    Wcyl=W.copy()
    ############################
    # plot:
    ax.contour(x,y,np.imag(W),cnts,linewidths=1.4,colors='0.2')
    ax.set_aspect(1.)
    try:
        ax.plot(np.real(zetaCylinder[bb[1]]),np.imag(zetaCylinder[bb[1]]),'s')
    except:
        pass
    ax.plot(np.real(zetaCylinder),np.imag(zetaCylinder),'b')
    xx=50.*np.exp(1j*0.*np.pi/180.)
    ax.plot([-np.real(xx),np.real(xx)],[-np.imag(xx),np.imag(xx)],'--',color='0.4
')
    ax.plot([np.imag(xx),-np.imag(xx)],[-np.real(xx),np.real(xx)],'--',color='0.4
')
    ax.plot(np.real(offset),np.imag(offset),'bx')
    # plot the equator of cylander:
    aa = a*(np.array([-1.+0.*1j,0.+0.*1j,1.+0.*1j]))*np.exp(1j*alpha*np.pi/180.)+
offset
    print(aa)
    ax.plot(np.real(aa),np.imag(aa),'b')

    #######################################################
    ## get the result in the air-foil co-ordinates...
    ax=axs[1]
    # zeta is the inverse transform, so for each point in Z we can get a unique v
aule in the zeta
    # plane:
    zeta = Z/2.+0.5*np.sqrt(Z**2-4*b**2)
    zeta[X<np.real(0.)] = Z[X<np.real(0.)]/2.-0.5*np.sqrt(Z[X<np.real(0.)]**2-4*b
**2)
```

In [15]:
```
for alpha in [-10.,0.,15.]:
    fig,axs = plt.subplots(1,2,sharex=True,sharey=True,figsize=(12,4))

    axs=axs.flatten()

    x,y,Wcyl,Wfoil,Gamma=plotFoil(axs,gammafac=1.,alpha=alpha)

    # get the pressure:
    dwdz = np.diff(Wfoil,axis=1)/np.median(np.diff(x))
    fig,ax = plt.subplots()
    pc=ax.pcolormesh(x,y,np.abs(dwdz)**2,rasterized=True,vmin=-2.,vmax=2.)
    fig.colorbar(pc,ax=ax)
    # integrate along the top edge
    top = 0.
    bot = 0.
    dz = np.median(np.diff(x))
    for i in range(len(x)-1):
        bad = np.where((np.isnan(dwdz[:,i]))|(np.abs(dwdz[:,i])>20.) )[0]
        if len(bad)>2:
            bot +=dz*np.abs(dwdz[bad[0]-1,i])**2
            top +=dz*np.abs(dwdz[bad[-1]+1,i])**2
    print(top)
    print(bot)
    print('$\\rho \\int (dw/dz)*2 dz$=%1.4f'% ((top-bot)/2.))
    print('$\\rho U \Gamma$ = %1.4f'%Gamma)
```

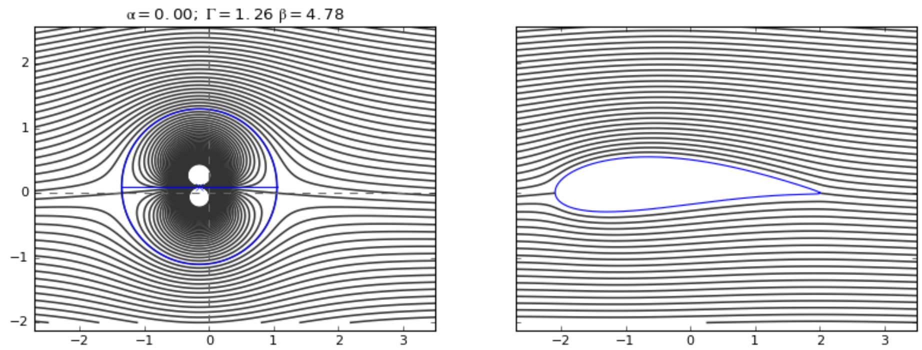



```
-1.37189887023
[-1.3317693+0.30837781j -0.1500000+0.1j         1.0317693-0.10837781j]
```

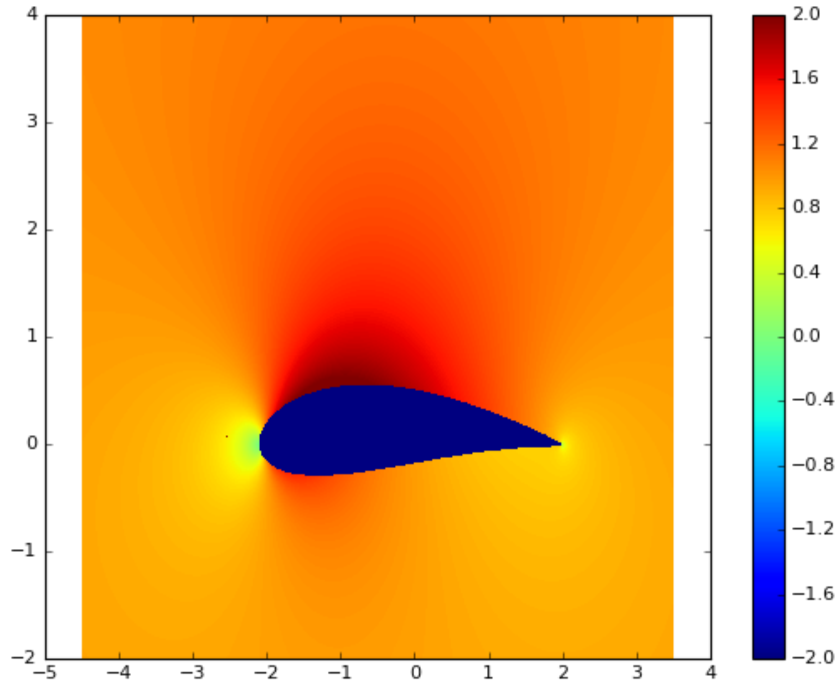


```
4.18417111895
6.89237443331
$\rho \int (dw/dz)*2 dz$=-1.3541
$\rho U \Gamma$ = -1.3719
```

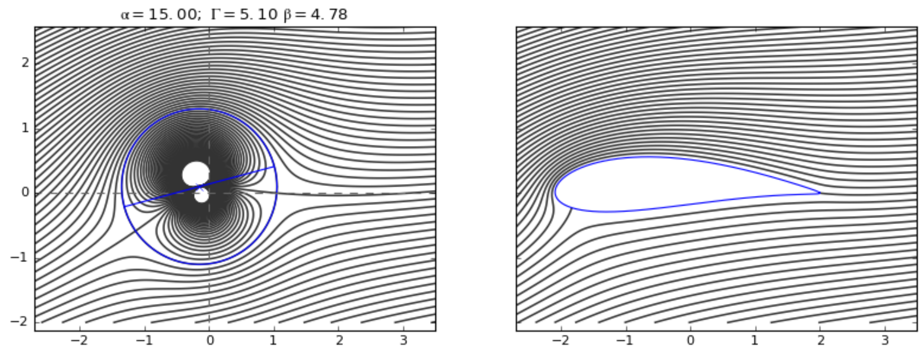
```
/Users/jklymak/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:18: R
untimeWarning: invalid value encountered in greater
```



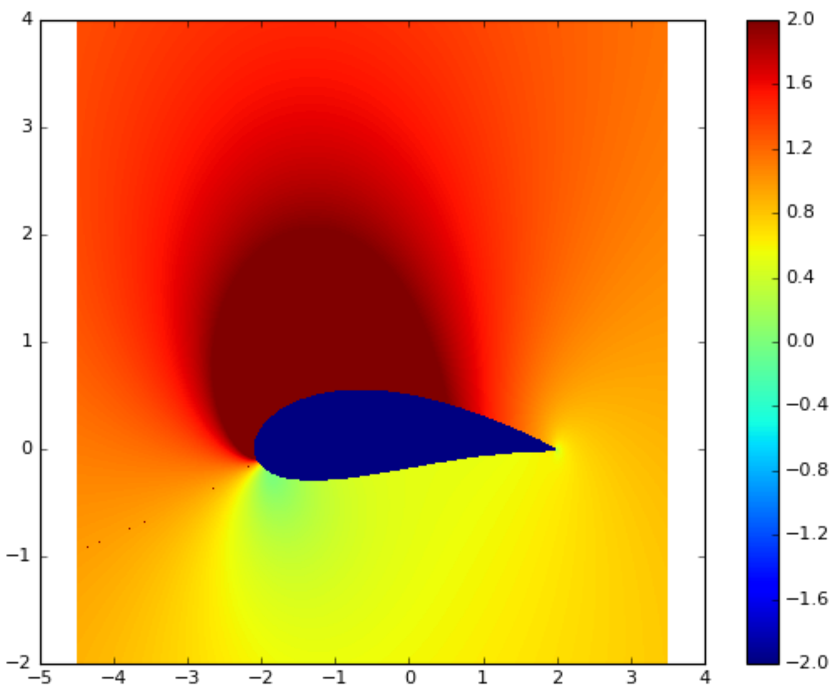

```
1.25663706144
[-1.35+0.1j -0.15+0.1j  1.05+0.1j]
```



```
6.69724741996
4.18570769077
$\rho \int (dw/dz)*2 dz$=1.2558
$\rho U \Gamma$ = 1.2566
```




```
5.10314210041
[-1.30911099-0.21058285j -0.15000000+0.1j          1.00911099+0.41058285j]
```



```
11.7628628667
1.87405363033
$\rho \int (dw/dz)*2 dz$=4.9444
$\rho U \Gamma$ = 5.1031
```

Math for stagnation point

We want the point at $\theta = \alpha + \beta$ to be a stagnation point. We saw in class that the velocity around a cylander was given by:

$$u_\theta = -2U\sin\theta - \frac{\Gamma}{2\pi a}$$

So in this case, we want to move the stagnation point from $\theta = \alpha$ down to the angle $-\beta$, so this means that

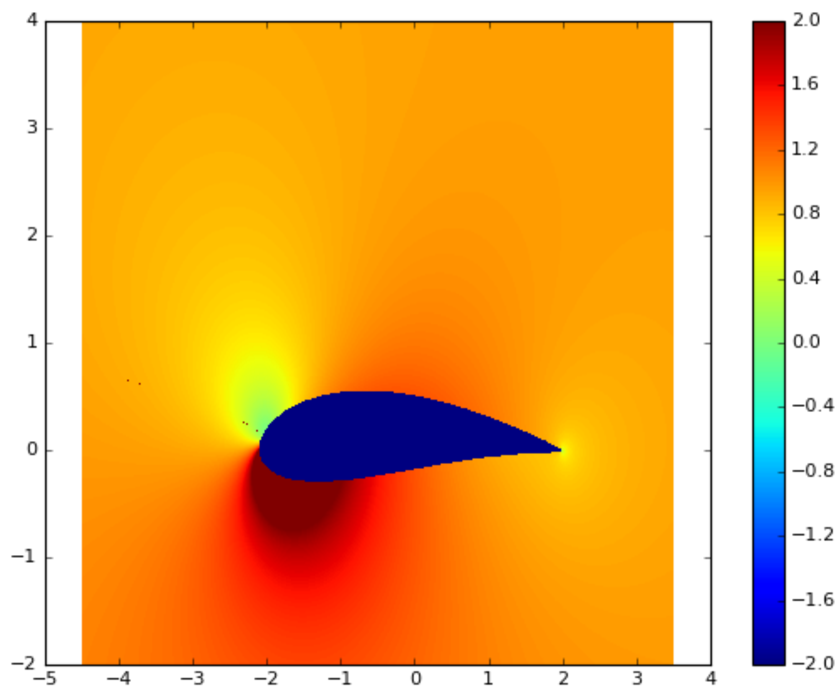
From 6.45:

$$D - iL = \frac{i}{2}\rho \oint_C \left(\frac{dw}{dz}\right)^2 dl$$

We need to calculate $dw/dz = \delta w/\delta x$

In [5]:

In [6]:



Out[6]: <matplotlib.colorbar.Colorbar at 0x11a135610>

In [7]:

```
4.19651927543
6.89237443331
$\rho \int (dw/dz)*2 dz$=-1.3479
$\rho U \Gamma$ = -1.3719
```

In [ ]:

In [1]:
```
import numpy as np
import matplotlib.pylab as plt
%matplotlib notebook
```

```
/Users/jklymak/anaconda2/lib/python2.7/site-packages/matplotlib/__init__.py:878:
UserWarning: axes.color_cycle is deprecated and replaced with axes.prop_cycle; p
lease use the latter.
  warnings.warn(self.msg_depr % (key, alt_key))
```

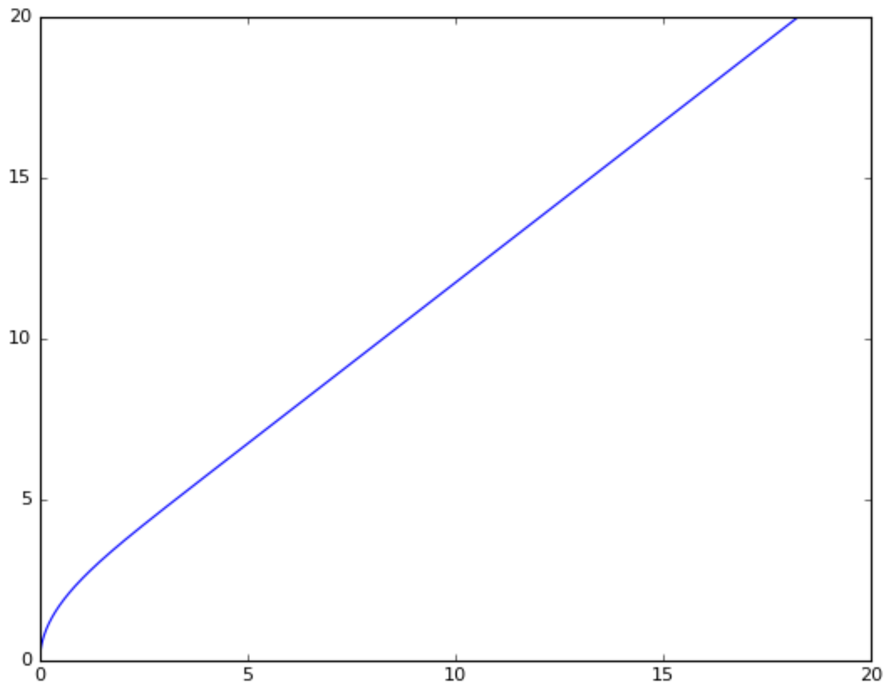
In [2]:
```
eta = np.arange(0,20,0.02)
deta = np.median(np.diff(eta))
```

In [3]:
```
g = 0.*eta
f = 0.*eta
h = 0.*eta
h_1=[]
g_N=[]

#Integrate once:
h[0]=0.02
for i in range(1,len(eta)):
    f[i]=f[i-1]+g[i-1]*deta
    g[i]=g[i-1]+h[i-1]*deta
    h[i]=h[i-1]-0.5*f[i-1]*h[i-1]*deta
g1 = g[-1]
h1 = h[0]
g_N+=[g1]
h_1+=[h1]

# integrate a second time
g = 0.*eta
f = 0.*eta
h = 0.*eta
h[0]=2.
print(g1)
for i in range(1,len(eta)):
    f[i]=f[i-1]+g[i-1]*deta
    g[i]=g[i-1]+h[i-1]*deta
    h[i]=h[i-1]-0.5*f[i-1]*h[i-1]*deta
g2 = g[-1]
h2 = h[0]
g_N+=[g2]
h_1+=[h2]
print g1, g2

# now iterate on two solutions using Newton-Raphson
while (np.abs(g2-1.))>0.001:
    if np.abs(g2-g1)>1e-10:
        h[0]=h[0] - (g1-1.)*(h2-h1)/(g2-g1)
    else: # this shouldn't happen, but if it does, then jump somewhere else and t
ry again...
        h[0]=h[0]+0.1
    if h[0]<0:
        h[0]=0.001
    g1 = g2
    h1 = h2
    g = 0.*eta
    f = 0.*eta
    h[1:]=0.
    for i in range(1,len(eta)):
        f[i]=f[i-1]+g[i-1]*deta
        g[i]=g[i-1]+h[i-1]*deta
        h[i]=h[i-1]-0.5*f[i-1]*h[i-1]*deta
    g2 = g[-1]
    g_N+=[g2]
    h2 = h[0]
    h_1+=[h2]
    #print(g2)

print(h2)
print(g2)
```

```
0.154078969943
0.154078969943 3.35317598398
0.328484420055
0.999820571757
```

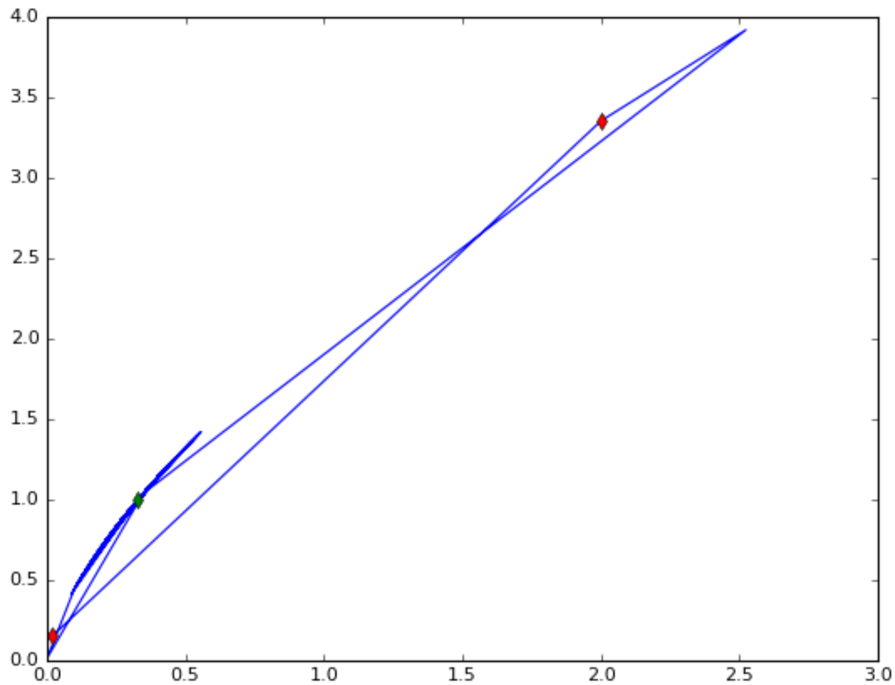
In [4]:
```
print h2
print g2
fig,ax=plt.subplots()
ax.plot(f,eta)
```

```
0.328484420055
0.999820571757
```



Out[4]:
```
[<matplotlib.lines.Line2D at 0x10aa7df10>]
```

In [5]:
```
fig,ax=plt.subplots()
ax.plot(h_1,g_N)
ax.plot(h_1[-1],g_N[-1],'dg')
ax.plot(h_1[:2],g_N[:2],'rd')
```

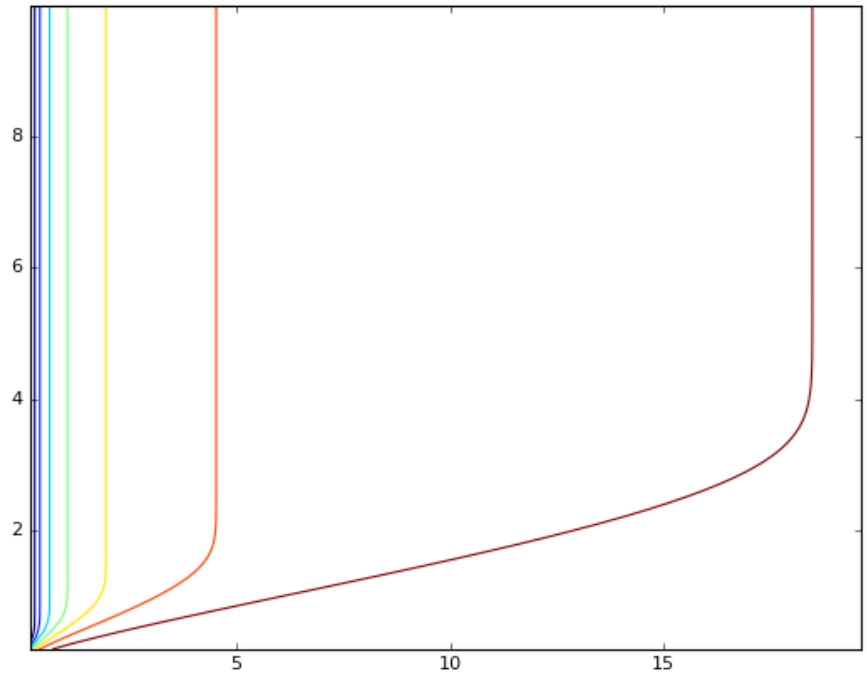


Out[5]: [<matplotlib.lines.Line2D at 0x10c499950>]

In [6]:
```
x = np.arange(0.2,20.,0.2)
z = np.arange(0.2,10.,0.03)
etaE = np.hstack((eta,1000.))
fE = np.hstack((f,1000.))
fE[-1] = fE[-2] + (etaE[-1]-etaE[-2])*(fE[-2]-fE[-3])/(etaE[-2]-etaE[-3])
print(etaE[-1])
X,Z = np.meshgrid(x,z)
nu = 0.022
U = 1.
Delta = np.sqrt(nu*X/U)
etag = Z/Delta
psi = np.interp(etag,etaE,fE)*U*Delta
print(Delta)
```

```
1000.0
[[ 0.0663325   0.09380832  0.11489125 ...,  0.65329932  0.65665821  0.66       ]
 [ 0.0663325   0.09380832  0.11489125 ...,  0.65329932  0.65665821  0.66       ]
 [ 0.0663325   0.09380832  0.11489125 ...,  0.65329932  0.65665821  0.66       ]
 ...,
 [ 0.0663325   0.09380832  0.11489125 ...,  0.65329932  0.65665821  0.66       ]
 [ 0.0663325   0.09380832  0.11489125 ...,  0.65329932  0.65665821  0.66       ]
 [ 0.0663325   0.09380832  0.11489125 ...,  0.65329932  0.65665821  0.66       ]]
```

In [7]:
```
fig,ax = plt.subplots()
ax.contour(x[:-1],z,np.diff(psi,axis=1))
```



Out[7]: <matplotlib.contour.QuadContourSet at 0x10c799150>

In [ ]: