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

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

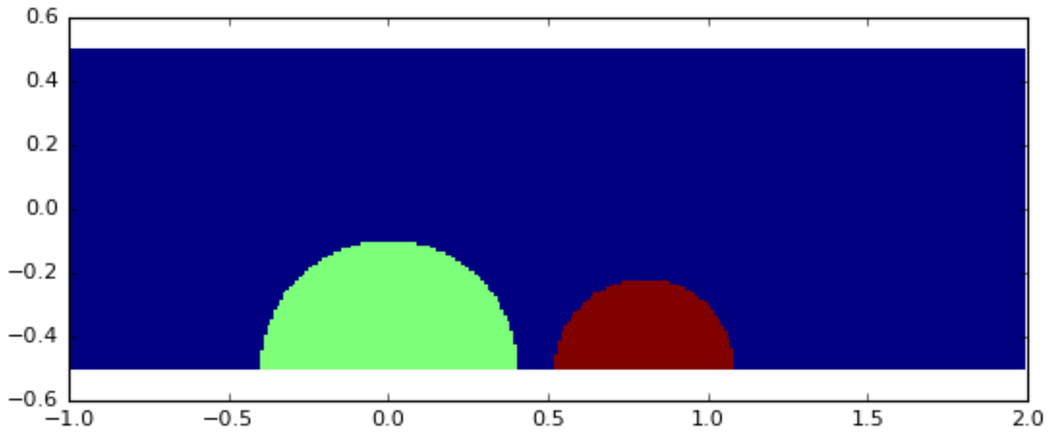
In [14]:
```
nx = 300
nz = 100
nx = 3*nz
x = np.arange(0.,3.,3./nx)
z = np.arange(0.,1.+1./nz,1./nz)
x = x-x[-1]/3.
z = z-z[-1]/2.

x1 = 0.
z1 = -0.5
r1 = 20./50.

x2 = 40./50.
z2 = -0.5
r2 = 14./50

X,Z = np.meshgrid(x,z)
# where is the circle?
in1= np.where(np.sqrt((X-x1)**2+(Z-z1)**2)<=r1);
minx1 = np.argmin(in1[1])
leadx1 = in1[1][minx1]
leadz1 = in1[0][minx1]
# where is the second circle?
in2= np.where(np.abs(X-x2+1j*(Z-z2))<=r2);
```

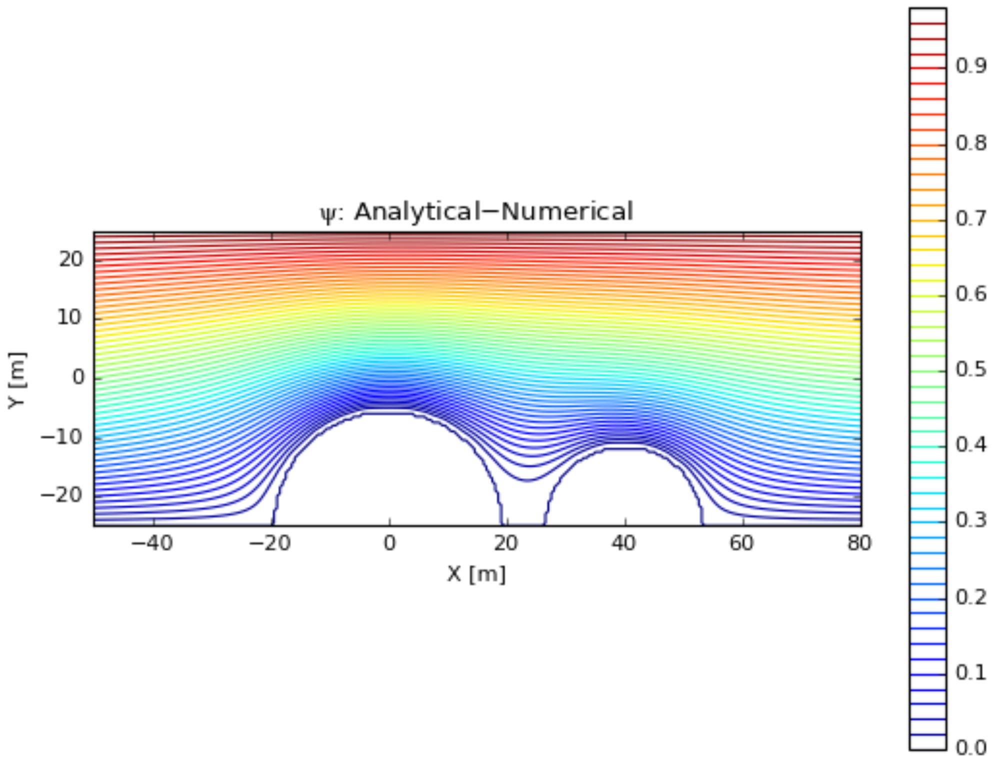
In [3]:
```
# plot and make sure OK
A=0.*Z
A[in1]=1.
A[in2]=2.
fig,ax=plt.subplots()
ax.pcolormesh(x,z,A,rasterized=True)
ax.set_aspect(1.)
```



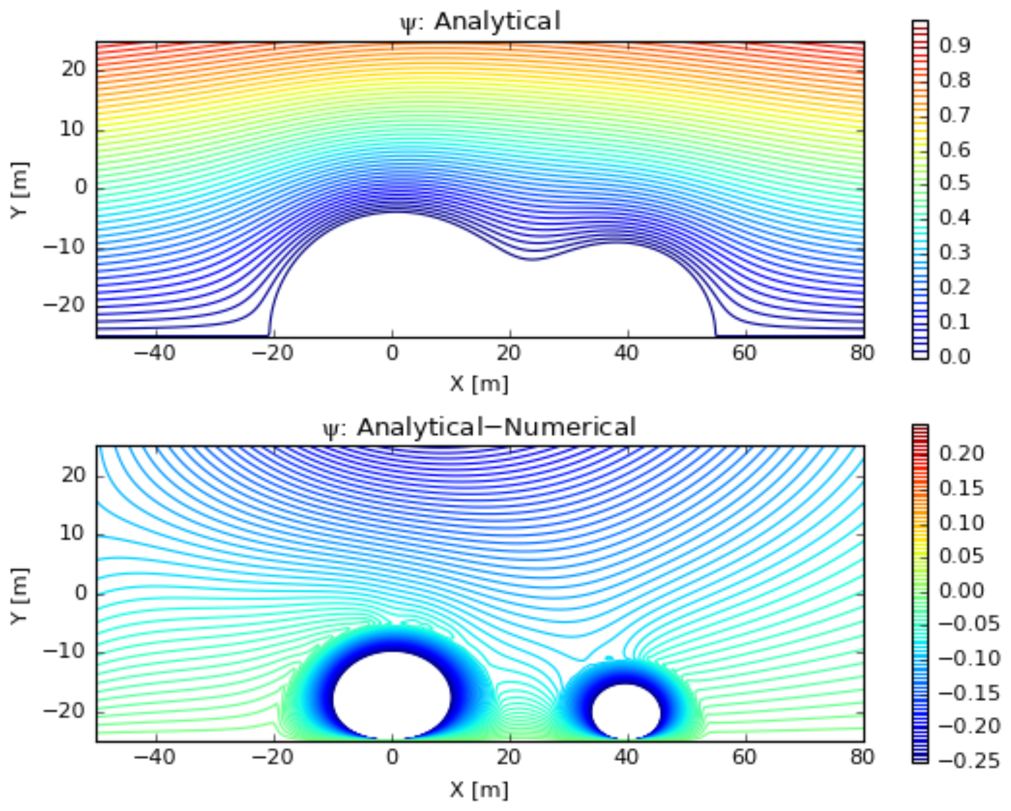
In [13]:
```python
psiold = X*0. # just to compare w/ psinew every iteration...
# initializes as if no obstacles:  i.e. psi[z]=z+0.5 =0 at bottom, 1 a
t top
psinew = Z.copy()+0.5

fig,axs=plt.subplots()
try:
    axs=axs.flatten()
except:
    axs = [axs]
psi = [[],[],[],[]]
numit=np.zeros(4)
for nn,tol in enumerate([1.e-6,]):
    ax=axs[nn]
    maxdiff=1.
    # initialize to be a free stream
    num=0
    while (maxdiff>tol):
        num+=1
        # set psi in the body to a value at the boundary of the body.
        psinew[in1]=0.
        psinew[in2]=0.
        # do the integration:
        psinew[1:-1,1:-1]=0.25*(psinew[0:-2,1:-1]+psinew[2:,1:-1]+
                                psinew[1:-1,0:-2,]+psinew[1:-1,2:])
        maxdiff = np.max(np.abs(psinew-psiold))
        psiold=psinew.copy()
    pc=ax.contour(x*50.,z*50.,psinew,np.arange(0.,1.,0.02))
    fig.colorbar(pc,ax=ax)
    ax.set_aspect(1.)
    ax.set_ylim([-25,25])
    ax.set_xlim([-50,80])
    ax.set_xlabel('X [m]')
    ax.set_ylabel('Y [m]')

    psi[nn]=psinew.copy()
    numit[nn]=num
    ax.set_title('$\psi$: Analytical-Numerical ')
```


ψ: Analytical−Numerical
Y [m]
X [m]
20
10
0
−10
−20
−40
−20
0
20
40
60
80
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

In [12]:
```
# get the analytic solution:
zz = X+1j*Z
comp = zz+r1**2/(zz-(x1+1j*z1))+r2**2/(zz-(x2+1j*z2))
fig,axs=plt.subplots(2,1)
ax = axs[0]
pc=ax.contour(x*50.,z*50.,np.imag(comp-comp[0,0]),np.arange(0.,1.,0.02
))
fig.colorbar(pc,ax=ax)
ax.set_aspect(1.)
ax.set_ylim([-25,25])
ax.set_xlim([-50,80])
ax.set_xlabel('X [m]')
ax.set_ylabel('Y [m]')
ax.set_title('$\psi$: Analytical')
ax=axs[1]
pc=ax.contour(x*50.,z*50.,np.imag(comp-comp[0,0])-psi[0],np.arange(-1.
,1.,0.02)/4.)
fig.colorbar(pc,ax=ax)
ax.set_aspect(1.)
ax.set_ylim([-25,25])
ax.set_xlim([-50,80])
ax.set_xlabel('X [m]')
ax.set_ylabel('Y [m]')
ax.set_title('$\psi$: Analytical-Numerical ')
```



```
Out[12]: <matplotlib.text.Text at 0x10b3e15d0>
```

Note the differences are mostly along the upper boundary. The analytical solution has no upper boundary, so the flow is still going around the obstacles at the top of the domain. Otherwise the responses are quite similar.

```
In [ ]:
```