In [ ]:
```
%matplotlib inline
# zeros, cos, sin, pi
import numpy as np
from numpy import cos, sin, exp,pi
import matplotlib.pyplot as plt

dr= 0.05
#dz=.5

imax=int(((20-0.2)/dr))+1
jmax=int((10/dr))+1
miu= 0.01
H= 10
Dk= 8e-12
kave= 1e-11
N=3
Rfrac=1
rwell= 0.2
Rmax= 20
Pmax= 100e+6

count=0

#define the functions
def k0(z):
    return kave+ Dk*cos(2*pi*N*z/H)
def k(r,z):
    return k0(z) / (1-exp(-r/Rfrac))
def dkz(r,z):
    return -(1/(1-exp(-r/Rfrac)))* Dk*(2*pi*N/H)*sin(2*pi*N*z/H)
def dkr(r,z):
    return -(exp(-r)*k0(z))/((1-exp(-1/Rfrac))**2)

a=np.zeros(shape=(imax,jmax))
b=np.zeros(shape=(imax,jmax))
c=np.zeros(shape=(imax,jmax))
d=np.zeros(shape=(imax,jmax))
e=np.zeros(shape=(imax,jmax))
f=np.zeros(shape=(imax,jmax))
P=np.zeros(shape=(imax,jmax))

for i in range(imax):
    for j in range(jmax):
        P[i,j]= 100e+6* (i/397)

#set the inside
for i in range(1,imax-1):
    for j in range(1,jmax-1):
        r =0.2 + i*dr
        z = j*dr
        a[i,j] = (dr/2)*((-k(r,z)/r)+(-dkr(r,z)))-k(r,z)# instead o
```

```
f r-> i
        b[i,j] = (dr/2)*((k(r,z)/r)+(+dkr(r,z)))-k(r,z)
        c[i,j] = (dr/2)*((-dkz(r,z)))-k(r,z)
        d[i,j] = (dr/2)*((dkz(r,z)))-k(r,z)
        e[i,j] = 4*k(r,z)
        f[i,j] =0.

#boundaries

for j in range(jmax): #Vertical boundaries [for j in range(1,jmax-1
):]

    #right e=1 f=100
    a[imax-1,j]=0.
    b[imax-1,j]=0.
    c[imax-1,j]=0.
    d[imax-1,j]=0.
    e[imax-1,j]=1.
    f[imax-1,j]=100e+6

    z = j*dr
    if (z<1) : #e=1 a=-1
        a[0,j]=1.
        b[0,j]=0.
        c[0,j]=0.
        d[0,j]=0.
        e[0,j]=-1.
        f[0,j]=0.

    elif (z>(H-1)) : #WHEN BOTTOM HALF IS LOST CHANGE THE CONDITION
TO (z<(H/2))
        a[0,j]=1.
        b[0,j]=0.
        c[0,j]=0.
        d[0,j]=0.
        e[0,j]=-1.
        f[0,j]=0.

    else : #e=1
        a[0,j]=0.
        b[0,j]=0.
        c[0,j]=0.
        d[0,j]=0.
        e[0,j]=1.
        f[0,j]=0.

for i in range(imax): #horizontal boundaries
    #bottom c=-1 e=1
    a[i,0] = 0
    b[i,0] = 0
    c[i,0] = 1.
    d[i,0] = 0
    e[i,0] = -1.
```

```
    f[i,0] = 0

    #top d=-1 e=1
    a[i,jmax-1] =0.
    b[i,jmax-1] =0.
    c[i,jmax-1] =0.
    d[i,jmax-1] =1
    e[i,jmax-1] = -1.
    f[i,jmax-1] = 0


#plt.imshow(np.rot90(a), cmap='inferno')
#plt.colorbar()

residual=0
avg_res = 10


#SOR
while avg_res>1e-4:
    tot_res = 0
    num_pts = 0
    for i in range(imax):
        for j in range(jmax):
            omega= 1.6 #1.6 for purely convective but this has radi
al components
            if((i+j)%2==count%2):
                residual = (e[i,j]*P[i,j])-f[i,j]
                if (i<(imax-1)):
                    residual += a[i,j]*P[i+1,j]
                if (i>0):
                    residual += b[i,j]*P[i-1,j]
                if (j<(jmax-1)):
                    residual += c[i,j]*P[i,j+1]
                if (j>0):
                    residual += d[i,j]*P[i,j-1]
                P[i,j] = P[i,j] - omega * residual/e[i,j]
                tot_res += abs(residual)
                num_pts += 1
    avg_res = tot_res/num_pts
    count += 1
    #print(count)
    if(count%100==0):                    #Run this before plot
        print(avg_res)
        print("Hello")

plt.figure(1)
plt.imshow(np.rot90(P), cmap='plasma')
plt.colorbar()
plt.contour (np.rot90(P),cmap='plasma')
plt.xlabel('Radius')
plt.ylabel('Depth')
plt.title('Pressure in Pascals')
```

```
plt.show
plt.savefig('Pressure_res_.png', bbox_inches='tight')

fx=np.zeros((imax,jmax))
fy=np.zeros((imax,jmax))

#Flow
for i in range(imax):
    R= 0.2 + i*dr #times it with r
    for j in range(jmax):
        z=j*dr

        if ((i>0) and (i<imax-1)):
            fx[i,j] = (-k(r,z)/miu)*(P[i+1,j]-P[i-1,j])/(2.*dr)
        elif (i>0):
            fx[i,j] = (-k(r,z)/miu)*(P[i,j]-P[i-1,j])/(dr)
        else:
            fx[i,j] = (-k(r,z)/miu)*(P[i+1,j]-P[i,j])/(dr)
        if ((j>0) and (j<jmax-1)):
            fy[i,j] = (-k(r,z)/miu)*((P[i,j+1]-P[i,j-1])/(2.*dr))
        elif (j==jmax-1):
            fy[i,j] = (-k(r,z)/miu)*((P[i,j]-P[i,j-1])/(dr) )
        else:
            fy[i,j] = (-k(r,z)/miu)*((P[i,j+1]-P[i,j])/(dr))




#how flows with different sampling
Fx = np.flipud(np.rot90(fx))
Fy = np.flipud(np.rot90(fy))

plt.figure(2)
plt.quiver(Fx,Fy)

plt.xlabel('Radius')
plt.ylabel('Depth')
plt.title(' Flux ')
#plt.show()
plt.savefig('Flux.png', bbox_inches='tight')



Fxs = np.flipud(np.rot90(fx[::5,::5]))
Fys = np.flipud(np.rot90(fy[::5,::5]))

plt.figure(3)
plt.quiver(Fxs,Fys)
plt.xlabel('Radius')
plt.ylabel('Depth')
#plt.axis([0, 40, 0, 20])
plt.title('Flux (sampled every 20 points)')
plt.savefig('Fluxs.png', bbox_inches='tight')
```

```
#plt.show()



x_component_left = 0.2*(np.trapz(Fx[:,0]))
x_component_left_one = 0.2*(np.trapz(Fx[:,1]))

print ('The flow on the left boundary is ', x_component_left )
print ('The flow one column after the left boundary is ', x_compone
nt_left_one )

x_component_right = 0.2*(np.trapz(Fx[:,-1]))
x_component_right_one = 0.2*(np.trapz(Fx[:,-2]))

print ('The flow on the right boundary is :', x_component_right)
print ('The flow one column before the left  boundary is :', x_comp
onent_right_one)

print ("Difference of flow in the boundary:",x_component_left-x_com
ponent_right )
print ("Difference of flow one column before the boundary:",x_compo
nent_left_one-x_component_right_one )
```

In [ ]:
```
#SENSITIVITY ANALYSIS
#make the above code into a function (by going indenting it bellow
a: "def r(x)")
# put an x into the variable that will be changed; in this case Pre
ssure
# return the differences in the flux of the second and second to la
st columns as proxy of the algorith's accuracy
# (just change the print statements at the end of the above code wi
th:
# return x_component_left_one-x_component_right_one

w=[]
#this loop will change the pressures and create a list with the res
ults
for i in [1, 50e+6, 100e+6, 500e+6, 1000e+6, 100000e+6, 1000000e+6,
10000000e+6]:
    a= r(i)
    w.append(a)


#show the profile
plt.semilogx([1, 50e+6, 100e+6, 500e+6, 1000e+6, 100000e+6, 1000000
e+6, 10000000e+6],w,'rx')
plt.title('Effect of Difference in Pressure')
plt.ylabel("Difference in flow between second and second to last co
lumns")
plt.xlabel('Difference in Pressure between Reservoir and Well')
plt.savefig("Sensitivity2", bbox_inches='tight')

plt.plot([1, 50e+6, 100e+6, 500e+6, 1000e+6, 100000e+6, 1000000e+6,
10000000e+6],w, 'b')
plt.plot([1, 50e+6, 100e+6, 500e+6, 1000e+6, 100000e+6, 1000000e+6,
10000000e+6],w,'rx')
plt.title('Effect of Difference in Pressure')
plt.ylabel("Difference in flow between second and second to last co
lumns")
plt.xlabel('Difference in Pressure between Reservoir and Well')
plt.savefig("Sensitivity", bbox_inches='tight')
```