OPTIMAL GROUNDWATER INJECTION DESIGN FOR SEAWATER INTRUSION PREVENTION

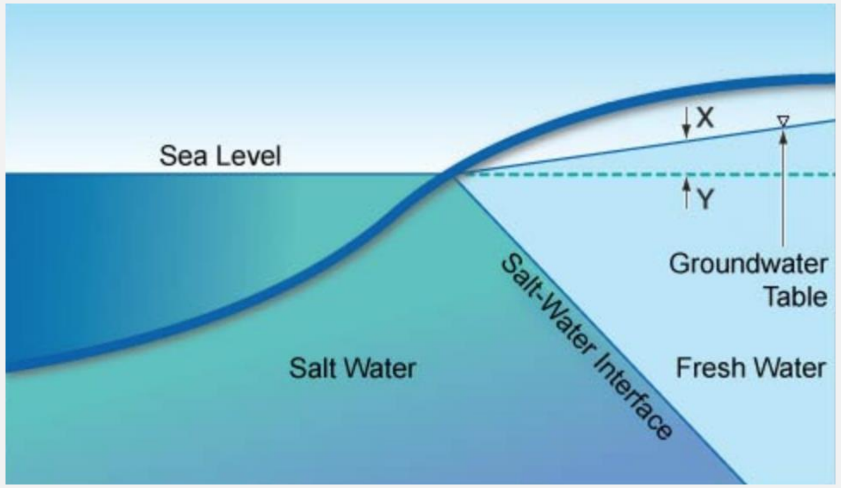
CEE696

Bing Hu

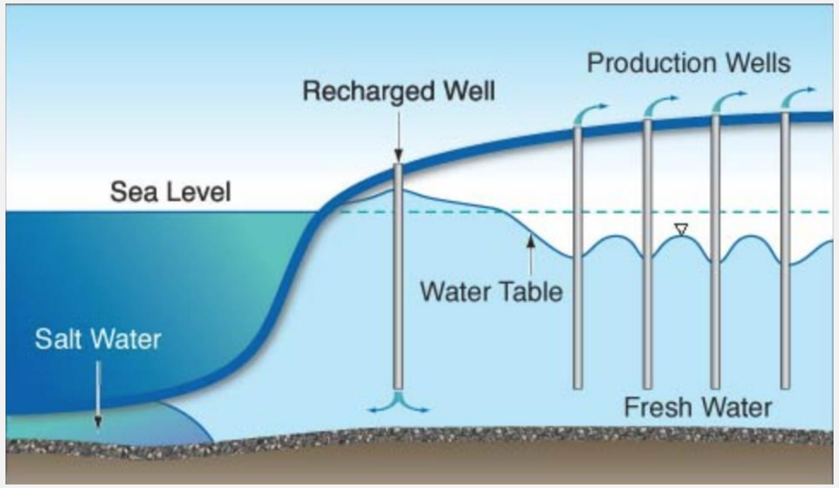
hub at hawaii dot edu

OUTLINE

INTRODUCTION



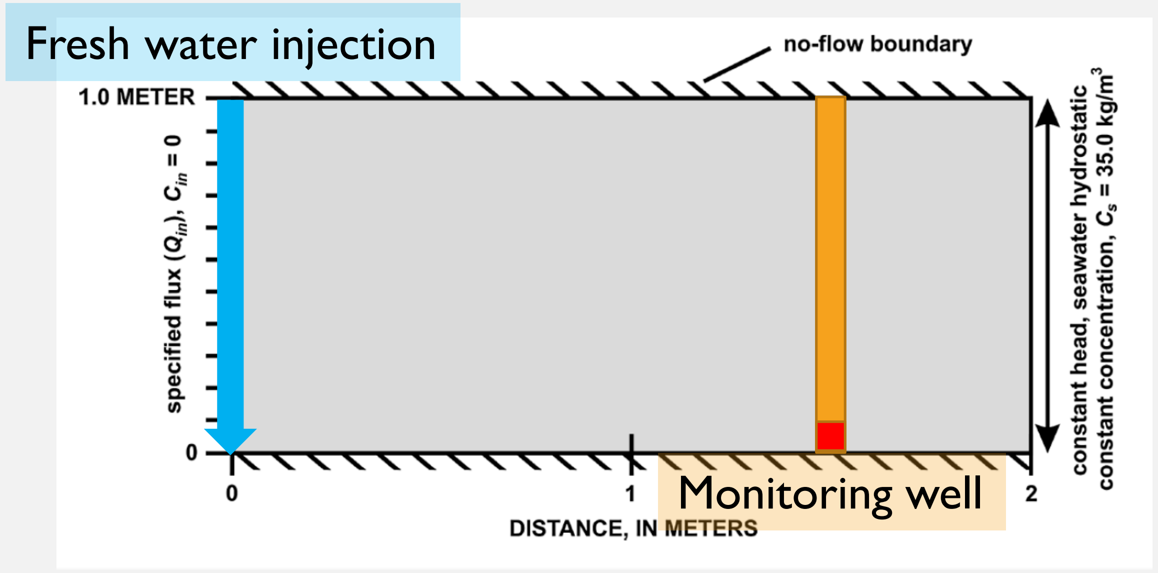
Picture from google

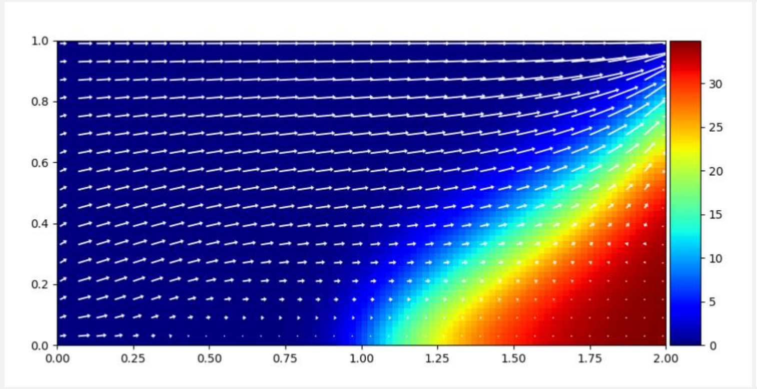


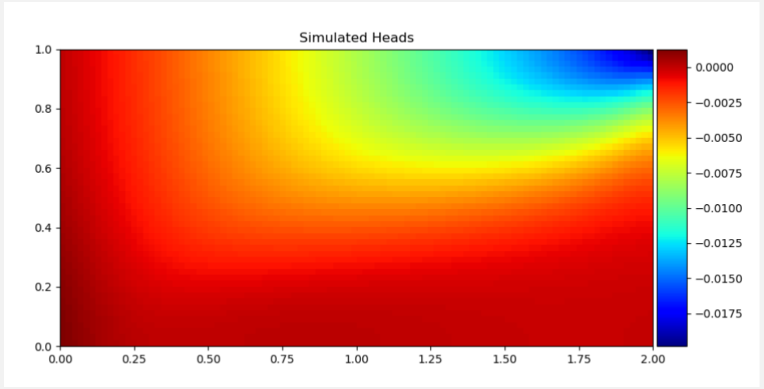
Picture from google

- Saltwater intrusion is the movement of saline water into freshwater aquifers, which can lead to contamination of drinking water sources and other consequences.
- It occurs naturally to some degree in most coastal aquifers, owing to the hydraulic connection between groundwater and seawater.
- By injection freshwater into aquifer near the coastal, seawater intrusion can be relieved or even prevented.
- In this project, Henry problem is used as the study domain.

HENRY PROBLEM

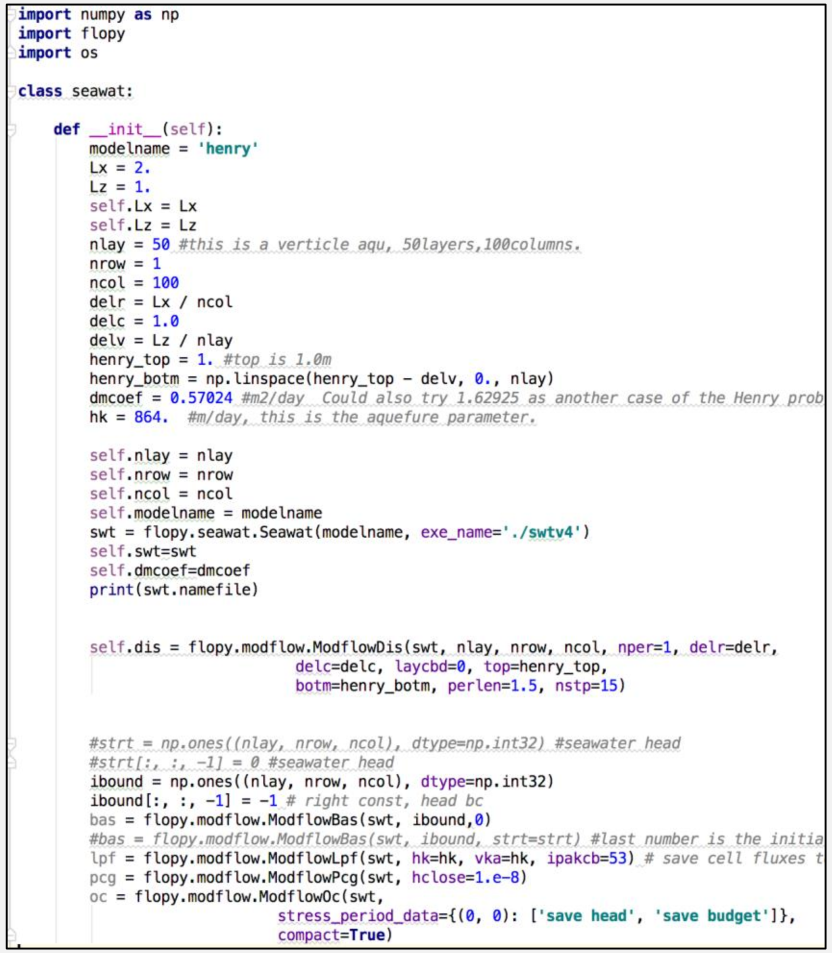

Fresh water injection
no-flow boundary
1.0 METER
specified flux (Q_{in}), C_{in} = 0
0
0
1
2
DISTANCE, IN METERS
Monitoring well
constant head, seawater hydrostatic
constant concentration, C_s = 35.0 kg/m^3



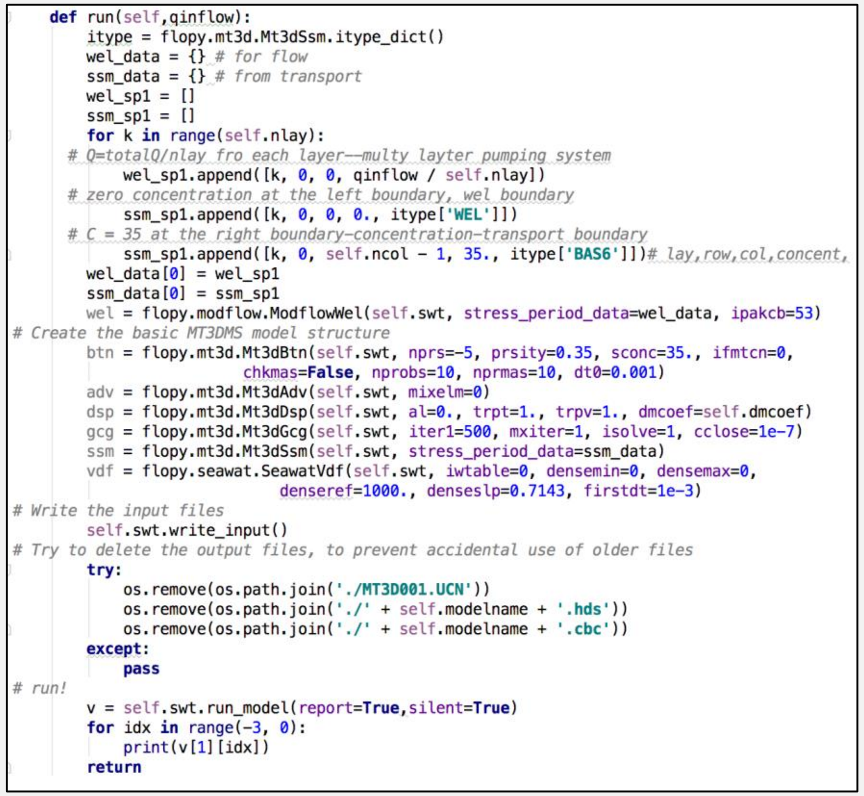


IMPLEMENT

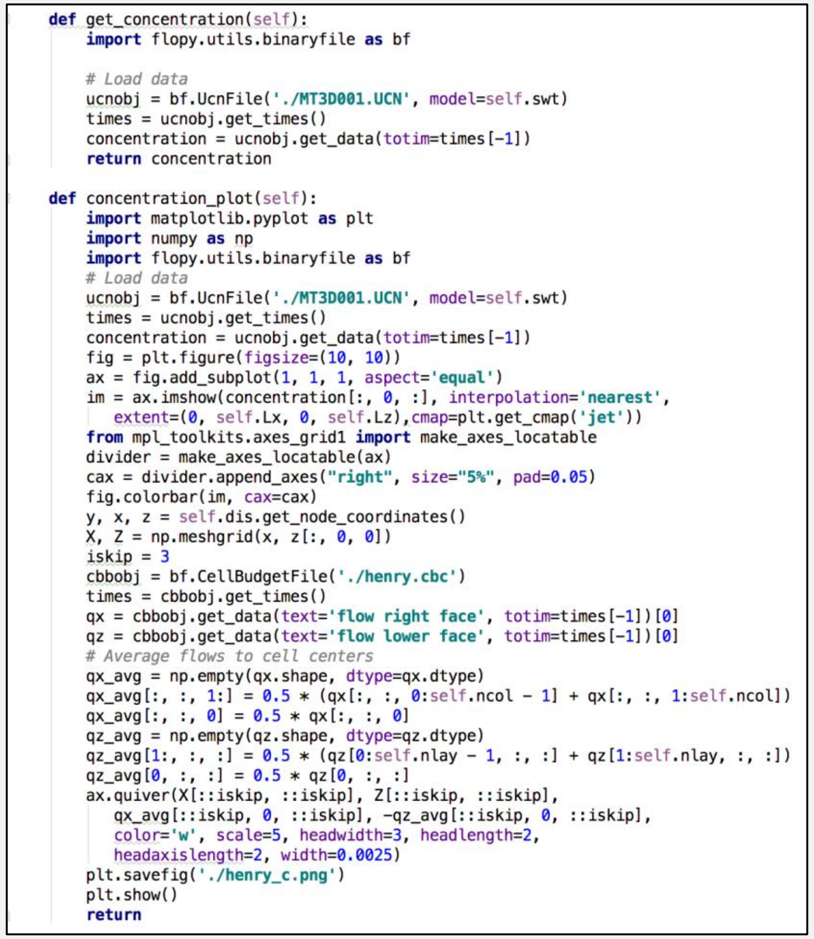
SEAWAT SIMULATION WRAPPER



```python
import numpy as np
import flopy
import os

class seawat:

    def __init__(self):
        modelname = 'henry'
        Lx = 2.
        Lz = 1.
        self.Lx = Lx
        self.Lz = Lz
        nlay = 50 #this is a verticle aqu, 50layers,100columns.
        nrow = 1
        ncol = 100
        delr = Lx / ncol
        delc = 1.0
        delv = Lz / nlay
        henry_top = 1. #top is 1.0m
        henry_botm = np.linspace(henry_top - delv, 0., nlay)
        dmcoef = 0.57024 #m2/day  Could also try 1.62925 as another case of the Henry prob
        hk = 864.  #m/day, this is the aquefure parameter.

        self.nlay = nlay
        self.nrow = nrow
        self.ncol = ncol
        self.modelname = modelname
        swt = flopy.seawat.Seawat(modelname, exe_name='./swtv4')
        self.swt=swt
        self.dmcoef=dmcoef
        print(swt.namefile)


        self.dis = flopy.modflow.ModflowDis(swt, nlay, nrow, ncol, nper=1, delr=delr,
                                       delc=delc, laycbd=0, top=henry_top,
                                       botm=henry_botm, perlen=1.5, nstp=15)


        #strt = np.ones((nlay, nrow, ncol), dtype=np.int32) #seawater head
        #strt[:, :, -1] = 0 #seawater head
        ibound = np.ones((nlay, nrow, ncol), dtype=np.int32)
        ibound[:, :, -1] = -1 # right const, head bc
        bas = flopy.modflow.ModflowBas(swt, ibound,0)
        #bas = flopy.modflow.ModflowBas(swt, ibound, strt=strt) #last number is the initia
        lpf = flopy.modflow.ModflowLpf(swt, hk=hk, vka=hk, ipakcb=53) # save cell fluxes t
        pcg = flopy.modflow.ModflowPcg(swt, hclose=1.e-8)
        oc = flopy.modflow.ModflowOc(swt,
                                     stress_period_data={(0, 0): ['save head', 'save budget']},
                                     compact=True)
```

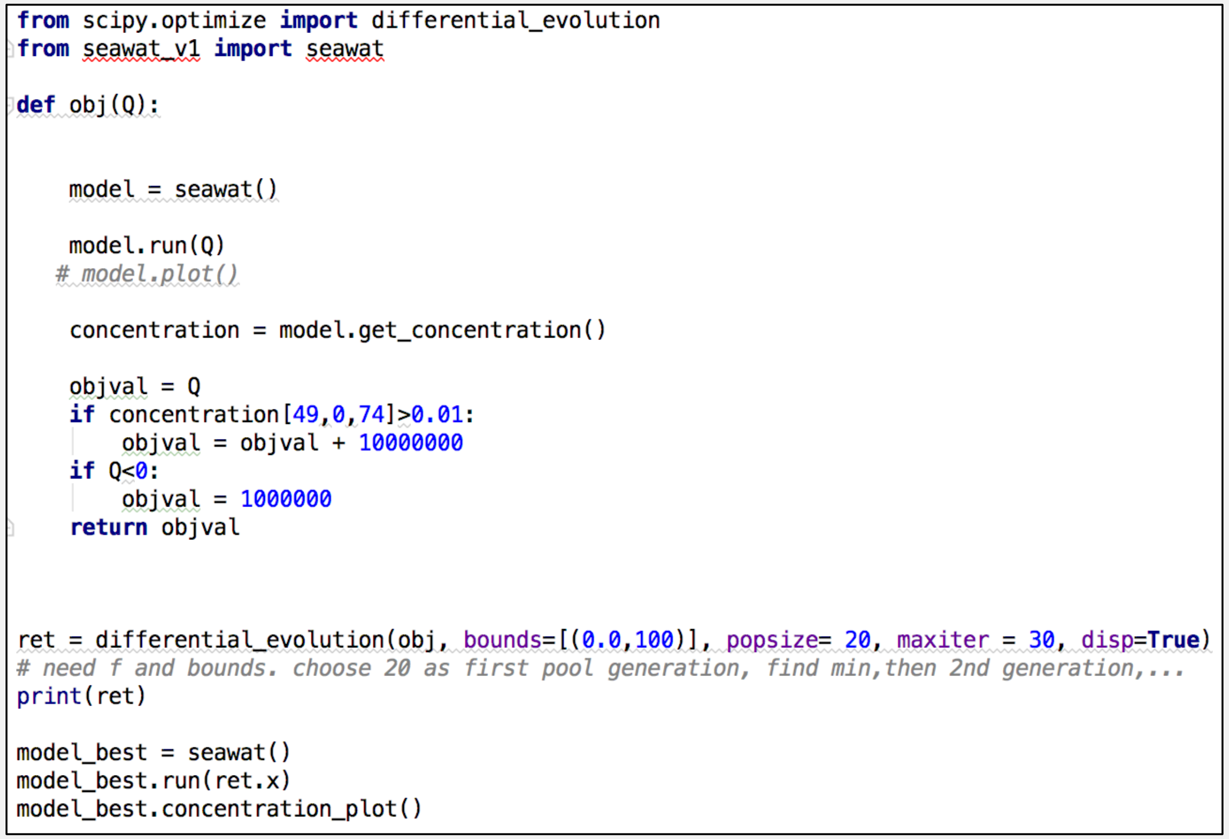


```python
    def run(self,qinflow):
        itype = flopy.mt3d.Mt3dSsm.itype_dict()
        wel_data = {} # for flow
        ssm_data = {} # from transport
        wel_sp1 = []
        ssm_sp1 = []
        for k in range(self.nlay):
      # Q=totalQ/nlay fro each layer--multy layter pumping system
            wel_sp1.append([k, 0, 0, qinflow / self.nlay])
      # zero concentration at the left boundary, wel boundary
            ssm_sp1.append([k, 0, 0, 0., itype['WEL']])
      # C = 35 at the right boundary-concentration-transport boundary
            ssm_sp1.append([k, 0, self.ncol - 1, 35., itype['BAS6']])# lay,row,col,concent,
        wel_data[0] = wel_sp1
        ssm_data[0] = ssm_sp1
        wel = flopy.modflow.ModflowWel(self.swt, stress_period_data=wel_data, ipakcb=53)
# Create the basic MT3DMS model structure
        btn = flopy.mt3d.Mt3dBtn(self.swt, nprs=-5, prsity=0.35, sconc=35., ifmtcn=0,
                                 chkmas=False, nprobs=10, nprmas=10, dt0=0.001)
        adv = flopy.mt3d.Mt3dAdv(self.swt, mixelm=0)
        dsp = flopy.mt3d.Mt3dDsp(self.swt, al=0., trpt=1., trpv=1., dmcoef=self.dmcoef)
        gcg = flopy.mt3d.Mt3dGcg(self.swt, iter1=500, mxiter=1, isolve=1, cclose=1e-7)
        ssm = flopy.mt3d.Mt3dSsm(self.swt, stress_period_data=ssm_data)
        vdf = flopy.seawat.SeawatVdf(self.swt, iwtable=0, densemin=0, densemax=0,
                                     denseref=1000., denseslp=0.7143, firstdt=1e-3)
# Write the input files
        self.swt.write_input()
# Try to delete the output files, to prevent accidental use of older files
        try:
            os.remove(os.path.join('./MT3D001.UCN'))
            os.remove(os.path.join('./' + self.modelname + '.hds'))
            os.remove(os.path.join('./' + self.modelname + '.cbc'))
        except:
            pass
# run!
        v = self.swt.run_model(report=True,silent=True)
        for idx in range(-3, 0):
            print(v[1][idx])
        return
```



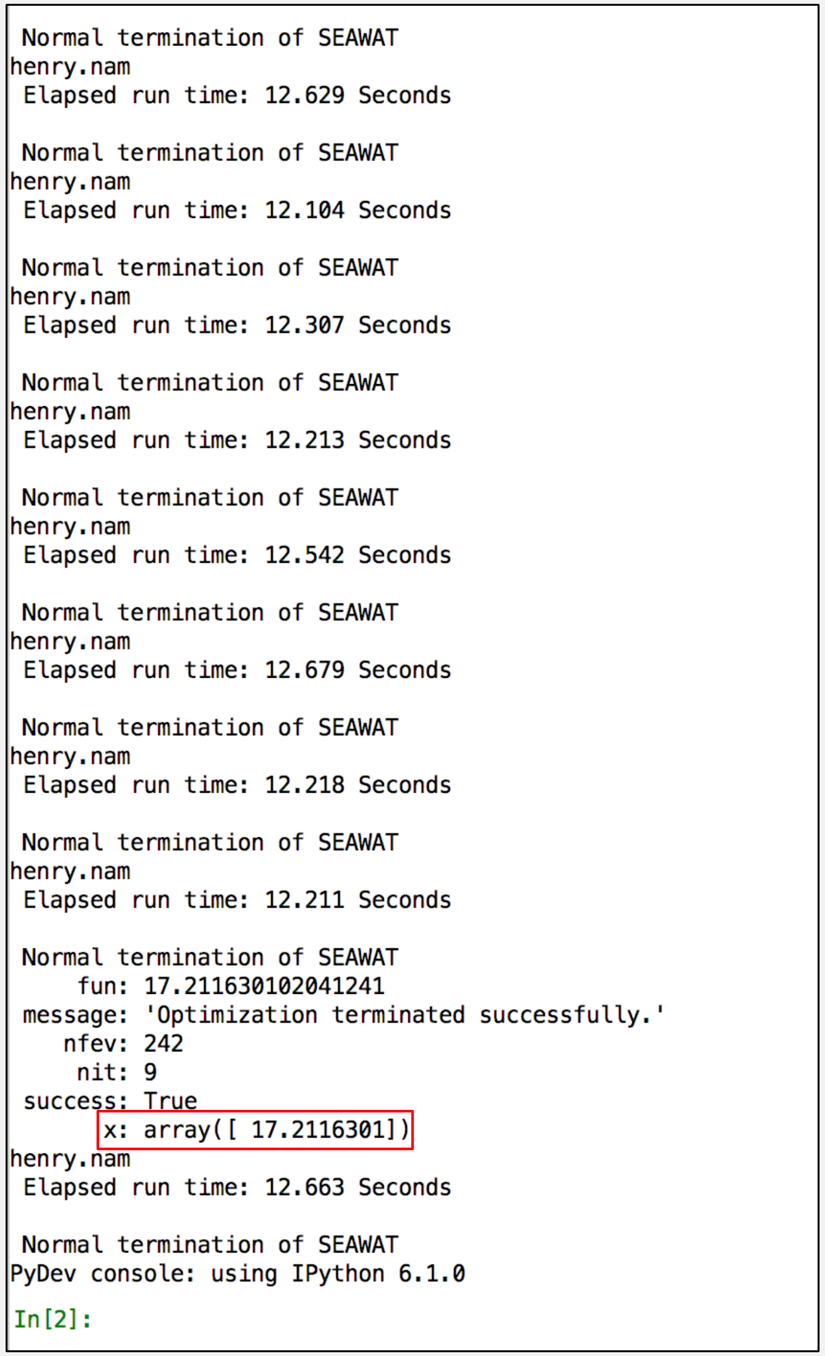
```python
    def get_concentration(self):
        import flopy.utils.binaryfile as bf

        # Load data
        ucnobj = bf.UcnFile('./MT3D001.UCN', model=self.swt)
        times = ucnobj.get_times()
        concentration = ucnobj.get_data(totim=times[-1])
        return concentration

    def concentration_plot(self):
        import matplotlib.pyplot as plt
        import numpy as np
        import flopy.utils.binaryfile as bf
        # Load data
        ucnobj = bf.UcnFile('./MT3D001.UCN', model=self.swt)
        times = ucnobj.get_times()
        concentration = ucnobj.get_data(totim=times[-1])
        fig = plt.figure(figsize=(10, 10))
        ax = fig.add_subplot(1, 1, 1, aspect='equal')
        im = ax.imshow(concentration[:, 0, :], interpolation='nearest',
          extent=(0, self.Lx, 0, self.Lz),cmap=plt.get_cmap('jet'))
        from mpl_toolkits.axes_grid1 import make_axes_locatable
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="5%", pad=0.05)
        fig.colorbar(im, cax=cax)
        y, x, z = self.dis.get_node_coordinates()
        X, Z = np.meshgrid(x, z[:, 0, 0])
        iskip = 3
        cbbobj = bf.CellBudgetFile('./henry.cbc')
        times = cbbobj.get_times()
        qx = cbbobj.get_data(text='flow right face', totim=times[-1])[0]
        qz = cbbobj.get_data(text='flow lower face', totim=times[-1])[0]
        # Average flows to cell centers
        qx_avg = np.empty(qx.shape, dtype=qx.dtype)
        qx_avg[:, :, 1:] = 0.5 * (qx[:, :, 0:self.ncol - 1] + qx[:, :, 1:self.ncol])
        qx_avg[:, :, 0] = 0.5 * qx[:, :, 0]
        qz_avg = np.empty(qz.shape, dtype=qz.dtype)
        qz_avg[1:, :, :] = 0.5 * (qz[0:self.nlay - 1, :, :] + qz[1:self.nlay, :, :])
        qz_avg[0, :, :] = 0.5 * qz[0, :, :]
        ax.quiver(X[::iskip, ::iskip], Z[::iskip, ::iskip],
          qx_avg[::iskip, 0, ::iskip], -qz_avg[::iskip, 0, ::iskip],
          color='w', scale=5, headwidth=3, headlength=2,
          headaxislength=2, width=0.0025)
        plt.savefig('./henry_c.png')
        plt.show()
        return
```

DIFFERENTIAL_EVOLUTION PROGRAMMING

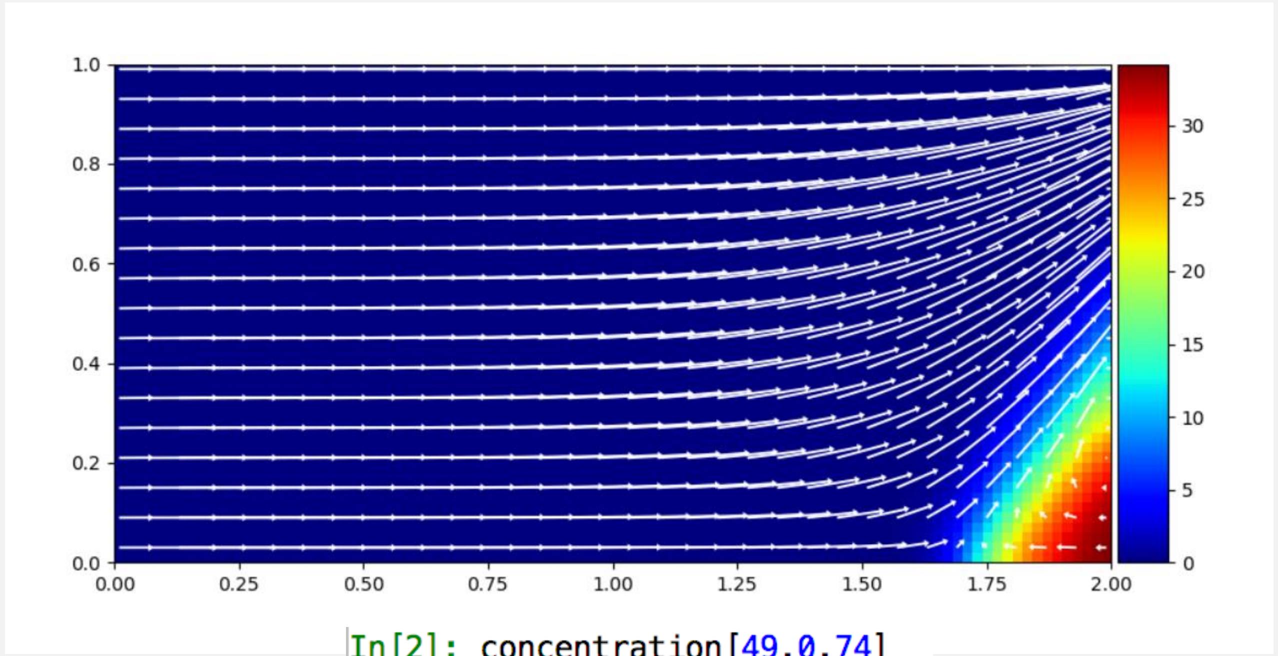


```
from scipy.optimize import differential_evolution
from seawat_v1 import seawat

def obj(Q):


    model = seawat()

    model.run(Q)
   # model.plot()

    concentration = model.get_concentration()

    objval = Q
    if concentration[49,0,74]>0.01:
        objval = objval + 10000000
    if Q<0:
        objval = 1000000
    return objval



ret = differential_evolution(obj, bounds=[(0.0,100)], popsize= 20, maxiter = 30, disp=True)
# need f and bounds. choose 20 as first pool generation, find min,then 2nd generation,...
print(ret)

model_best = seawat()
model_best.run(ret.x)
model_best.concentration_plot()
```

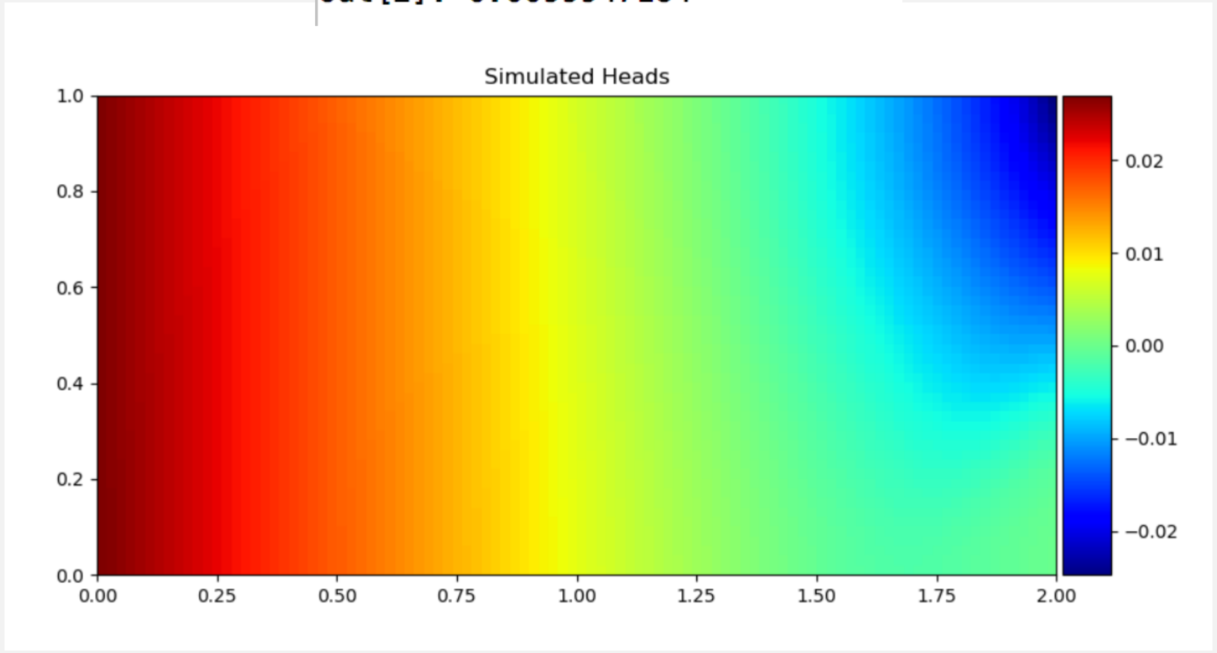
RESULTS



```
 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.629 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.104 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.307 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.213 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.542 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.679 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.218 Seconds

 Normal termination of SEAWAT
henry.nam
 Elapsed run time: 12.211 Seconds

 Normal termination of SEAWAT
     fun: 17.211630102041241
 message: 'Optimization terminated successfully.'
    nfev: 242
     nit: 9
 success: True
       x: array([ 17.2116301])
henry.nam
 Elapsed run time: 12.663 Seconds

 Normal termination of SEAWAT
PyDev console: using IPython 6.1.0
In[2]:
```



```
In[2]: concentration[49,0,74]
Out[2]: 0.0099947164
```



DISCUSSION

- Henry problem is not a good model to simulate the optimum injection when considering seawater level rise up.
- The optimum injection flow rate is always the same no matter how much the initial water head is in Henry problem.
- The water head distribution would be changed according to the initial water head.

QUESTIONS?