Groundwater supply optimization

simulation-optimization framework

Harry Lee

2/26/2018

CEE 696

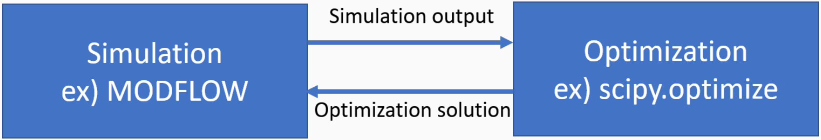
Table of contents

Simulation-Optimization Framework

Groundwater supply management

- One can manage and control the regional groundwater flow to make sure appropriate availability of water of adequate quantity
- “safe yield” vs. “sustainable yield”
- safe yield is intended to assure groundwater supply that meets need.
- Sustainable groundwater yield is a rate that can be pumped without causing unacceptable consequences.
- Sustainable yield computation can be posed as an optimization problem

Simulation-Optimization framework



- Simulation component is designed to predict how a physical system will respond to an input set including decision variables
- Optimization component solves a mathematical optimization problem using decision variables, objective function, constraints with system response from Simulation component
- The Simulation-Optimization (S-O) framework couples two components to provide the best strategy.
- Faster and more accurate than a trial and error approach using simulation model alone.

Back to my_first_opt_flopy_optimization.py

Our first application was groundwater supply maximization while keeping minimal head drops: constant head = 0 m at left and right boundaries, assume a pumping well at the center of domain and allow 1 m head drop.

Our optimization problem was formulated as

$$\begin{array}{ll} \max\limits_{Q} & Q \\ \text{subject to} & h_i(\mathrm{x}) \geq h_{min},\; i = 1, \ldots, m \end{array}$$

where Q is a pumping rate and h is hydraulic head.

Reformulation to unconstrained optimization

Since we have learned unconstrained optimization, we converted the constrained optimization to unconstrained optimization by adding a penalty term.

$$\begin{array}{ll} \max\limits_{Q} & Q \\ \text{subject to} & h_i(\mathbf{x}) \geq h_{min},\ i = 1, \ldots, m \end{array}$$

$$\min_{Q} \ -Q + \mathbb{1}_{\{h_i(x) < -1\}} \lambda (h_i(x) + 1)^2$$

where $\mathbb{1}_{condition}$ is an indicator function (1 for condition = true, otherwise 0) and λ is a big number for penalty.

Run this

We tried to perform an optimization with flopy-based modeling. Please copy an updated version of scripts below to your working directory and run opt_mymf.py:

`https://www2.hawaii.edu/~jonghyun/classes/S18/CEE696/files/mymf.py`

`https://www2.hawaii.edu/~jonghyun/classes/S18/CEE696/files/opt_mymf.py`

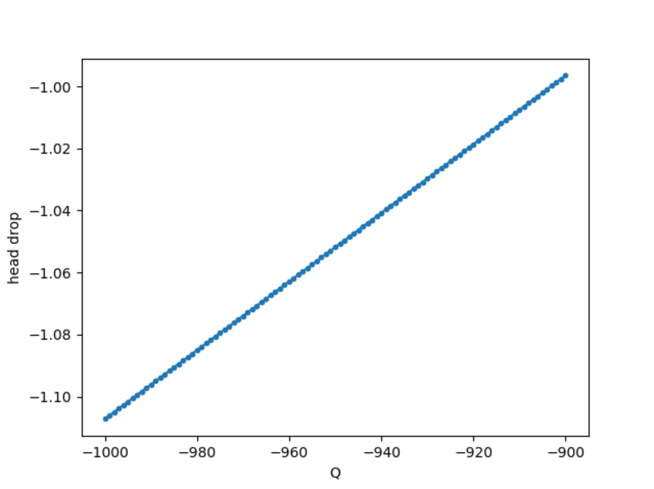
Flow in Confined Aquifer

GW Flow Equation

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) + \frac{\partial}{\partial x}\left(K_y \frac{\partial h}{\partial y}\right) + Q_s = S\frac{\partial h}{\partial t}$$

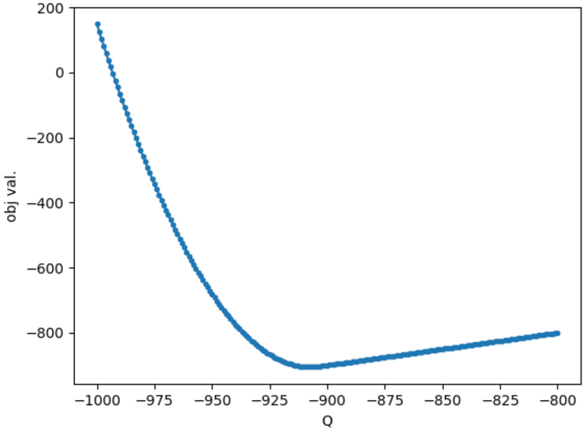
This is our simulation model. Also, the equation above is called "linear" - why?

Please check hydraulic head changes with different pumping rates.



So, how does our objective function look like?

objective function



Response Matrix Approach

Response Matrix Approach

For steady-state flow in confined aquifer, we can efficiently compute the hydraulic head at n monitoring wells with pumping rate Q_j at m locations once we determine "unit response":

$$\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} = \begin{bmatrix} h_1^{init} \\ h_2^{init} \\ \vdots \\ h_n^{init} \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} & d_{13} & \dots & d_{1m} \\ d_{21} & d_{22} & d_{23} & \dots & d_{2m} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \dots & d_{nm} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix}$$

where h_i = hydraulic head at the monitoring well i

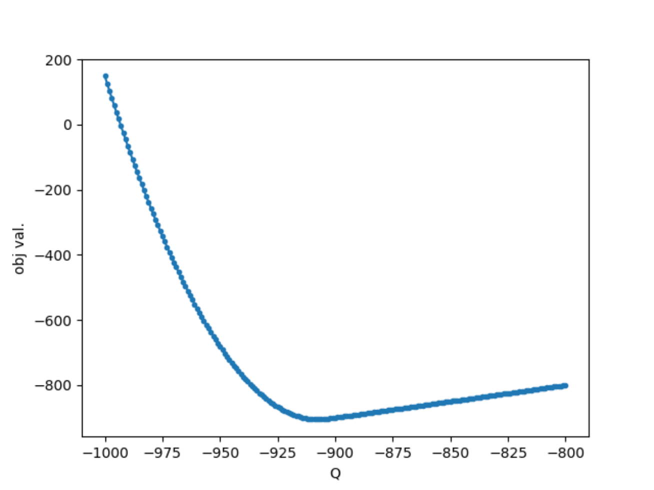
h_i^{init} = initial hydraulic head at the monitoring well i

d_{ij} = unit head drop/drawdown at the monitoring well i due to the pumping well j

q_j^{init} = pumping rate at the pumping well j

Linear Programming

Back to example



Which optimization routine would be the best for this problem?

Linear Programming in Python

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}_{ub}\mathbf{x} \leq \mathbf{b}_{ub}, \; i = 1, \ldots, m \\ & \mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}, \; i = 1, \ldots, n \end{aligned}$$

- Minimize a linear objective function subject to linear equality and inequality constraints.

```
scipy.optimize.linprog(c, A_ub=None, b_ub=None, A_eq=None,
b_eq=None, bounds=None, method='simplex', callback=None,
options=None)
```

Linear Programming - Example

- You run a business that produces 22 gallons of milk each week and sells dairy products - ice cream and butter
- 1 gallon of ice cream requires 3 gallons of milk
- 1 kilogram of butter requires 2 gallons of milk
- You have a refrigerator that can store unlimited amounts of butter, but hold at most 6 gallons of ice cream.
- You have only 6 hours per week to manufacture the dairy products.
- 1 hour of work is needed to produce either 4 gallons of ice cream or 1 kilogram of butter.
- Everything is sold out on a farm's market on Sunday at \$5 per gallon of milk and \$4 per 1 kg of butter

Q: How much ice cream and butter one should produce to maximize profit?

Linear Programming

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & -5x - 4y \\
\text{subject to} \quad & x \leq 6 \\
& 0.25x + y \leq 6 \\
& 3x + 2y \leq 22 \\
& x, y \geq 0
\end{aligned}
$$

- Show the solution graphically

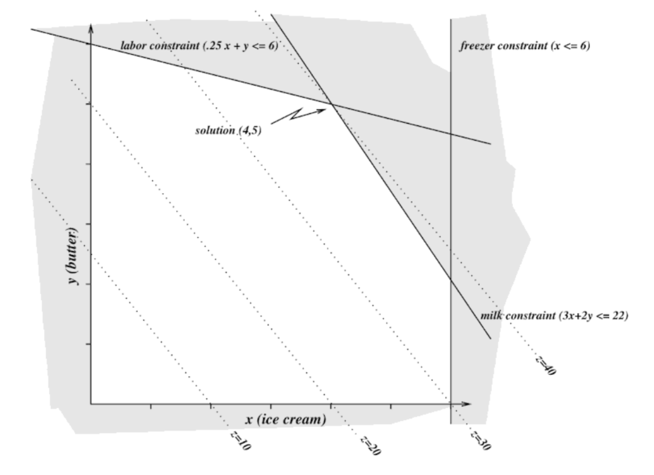


Figure 1: Figure 1.1 from Ferris et. al. [2007]

Example

```python
import scipy.optimize as opt
import numpy as np
c = np.array([-5, -4])
A = np.array([[0.25, 1], [3, 2]])
b = np.array([6, 22])
x0_bounds = (0, 6)
x1_bounds = (0, None)
res = opt.linprog(c, A_ub=A, b_ub=b,
                  bounds=(x0_bounds, x1_bounds),
                  options={"disp": True})
```

Application: Linear Programming with Response Matrix Approach

Problem description

- A horizontal confined aquifer (1000 x 1000 x 50 m) with constant hydraulic heads on the western and eastern boundaries (h_{west} = 1 m, h_{east} = 1 m), no flow condition on northern and southern boundaries.
- Horizontal and vertical hydraulic conductivity are given by 10 m/d.
- The domain is discretized in 10 blocks in x and 10 blocks in y and 1 block in z for MODFLOW simulation.
- Pumping well is located at the center of the domain (in our modeling grid row_idx, col_idx = (4,4)).
- We would like to maximize the pumping rate.
- Hydraulic head should be greater than 0 m

Download scripts

mymf_v3.py our modflow simulation wrapper

simul_mymf_v3.py modflow simulation example using mymf_v3.py

opt_RMA_LP_well1.py optimization (linear programming) using response matrix approach with single well configuration

Simulation Example

Now mymf takes initial head condition

```
from mymf_v3 import mymf
init_head = 1. # initial head
model = mymf(init_head=init_head)
well_rcs = [[4,4]]      # center
Qs = [-1.]              # unit pumping rate
model.run(well_rcs,Qs)
model.plot()
```

Optimization (1) - Response Matrix Approach

```
# we will start with initial head = 1 m
# construct response matrix
init_head = 1. # initial head
model = mymf(init_head=init_head)
well_rcs = [[4,4]]      # center
Qs = [-1.]              # unit pumping rate
model.run(well_rcs,Qs)
model.plot()

head = model.head() # note this array is 3D!
head_change = init_head - head
```

Optimization (2) - Linear Programming

Once we get response due to unit pumping rate

```
"""
min  Q
s.t.  unit_head_change * Q <= max_head_change
      Q <= 0
"""
c = np.array([1.]) # minimize Q (maximize extraction)
# A is head change because of unit pumping rate at (4,4)
A = np.array([[-head_change[0,4,4]]])
b = np.array([1.])    # maximum head change
x0_bounds = (None, 0) # negative value for pumping
res = opt.linprog(c, A_ub=A, b_ub=b,
                  bounds=(x0_bounds),
                  options={"disp": True})
print('### result with minimal head constraint ###')
print(res)
print('the maximum pumping rate is %f' % (res.x))
```

Head Constraints at monitoring wells

What if we measure hydraulic heads at monitoring wells (1,1) and (7,7) and want to keep hydraulic heads at those locations above 0 m?

Optimization (3) - Linear Programming with Monitoring Wells

head constraints at monitoring wells (1,1) and (7,7)

```
"""
min  Q
s.t.  unit_head_change * Q at (1,1) <= max_head_change
      unit_head_change * Q at (7,7) <= max_head_change
      Q <= 0
"""
c = np.array([1.]) # minimize Q (maximize extraction)
# now A is 2 by 1 array (A.shape = (2,1))
A = np.array([[-head_change[0,1,1]],
              [-head_change[0,7,7]]])
b = np.array([1.,1.]) # max head change at (1,1) & (7,7)
x0_bounds = (None, 0) # Q <= 0
res = opt.linprog(c, A_ub=A, b_ub=b,
                  bounds=(x0_bounds),
                  options={"disp": True})
print('### result with minimal head constraint ###')
```

Multiple Pumping Well Optimization

What if we have two pumping wells and want to keep hydraulic heads at a location above 0 m?