MODFLOW-MT3DMS with Flopy

project topics & contaminant transport simulation

Harry Lee

3/12/2018

CEE 696

Potential project topics

Ana	Contaminant Transport?
Bing	Fresh GW-Seawater interaction?
Brytne	Hydraulic conductivity estimation
Chris	Regional aquifer GW flow modeling?
Olkeba	Python-based estimation
Sabrina	Resident time optimization/contaminant degradation
Shelby	Unsaturated flow?
Harrison	Fresh GW - seawater interaction?

Potential applications covered in next two weeks

- Contaminant transport modeling
- 1D Unsaturated Flow
- GW-Seawater interaction
- Model Parameter Estimation

Potential optimization application ideas

- Pumping rate optimization
    - for pump & treatment/capture zone delineation - Ana, Sabrina
    - for sustainable yield determination - Chris
    - for seawater intrusion prevention - Bing/Harrison
- Hydraulic conductivity estimation - Brytne & Olkeba
- Unsaturated Flow? - Shelby

HW3

- Due on Friday 3/23
- Choose one application with your description
- Show initial simulation results if possible

Contaminant transport in MT3DMS

- MODPATH : Particle-tracking post-processing program based on MODFLOW
- MT3DMS : Modular 3-D Multi-Species Transport Model for Simulation of Advection, Dispersion, and Chemical Reactions of Contaminants in Groundwater Systems

We will learn how to use MT3MDS

MT3DMS installation

For Windows:

1. Download executable
   - from `https://hydro.geo.ua.edu/mt3d/mt3dms_530.exe`
   - or go to MT3DMS webpage: `https://hydro.geo.ua.edu/mt3d/index.htm` and click "MT3DMS 5.3"
2. Unzip the folders and copy bin/mt3dms5s.exe and bin/mt3dms5b.exe to your flopy working directory

For Mac and Linux:

1. Download or clone pyMake (`https://github.com/modflowpy/pymake`)
2. go to "examples" folder and run make_mt3d.py
3. copy "mt3dms" to your working directory

Example 2 (adaptive from Example 1)

A horizontal confined aquifer (1000 x 1000 x 50 m) with constant head on the western and eastern boundaries (h_{west} = 10 m, h_{east} = 0 m), no flow condition on northern and southern boundaries. Horizontal and vertical hydraulic conductivity are given by 10 m/d. A injection well at x = 100, y = 500 was installed and a conservative tracer C = 10 is injected with the flow rate of 1000 m^3/d into the aquifer for 1000 days. Longitudinal and transverse dispersity values are 10 m and 1 m respectively.

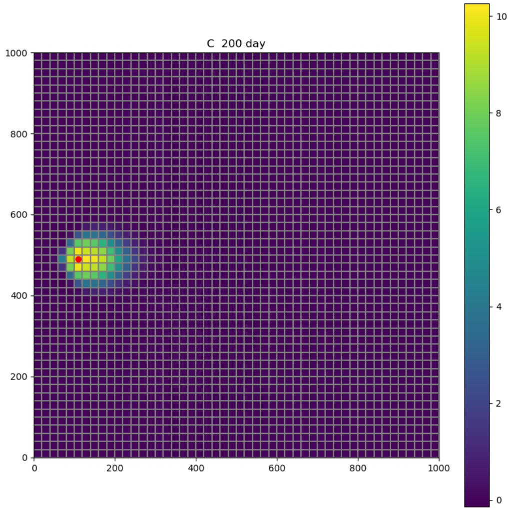
Run MODFLOW-MT3DMS simulation

Download a script from

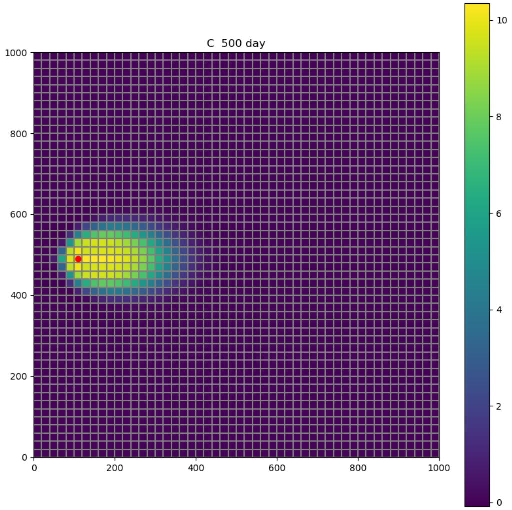
`https://www2.hawaii.edu/~jonghyun/classes/S18/CEE696/files/my_first_flopy_mt3dms_example.py`

and run the script.

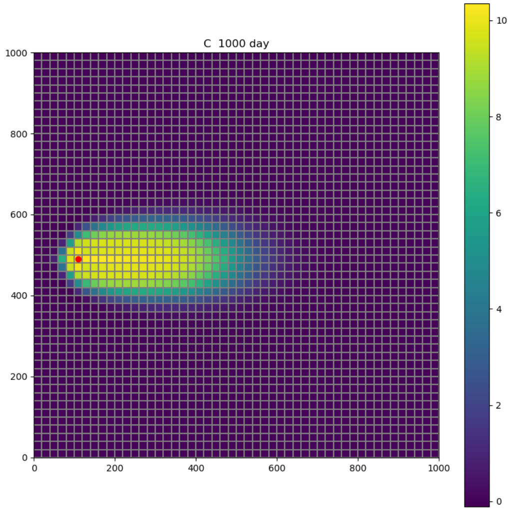
Results: C in day 200



Results: C in day 500



Results: C in day 1000



Governing equations (1)

Flow equation (as solved in MODFLOW)

- solve hydraulic h (then post-process q or v from h)

Transport equation (as solved in MT3DMS)

- solve concentration C

Governing equations (2)

Flow equation (in MODFLOW)

$$S_s \frac{\partial h}{\partial t} = \frac{\partial}{\partial t}\left(K \frac{\partial h}{\partial x}\right) + q_s \tag{1}$$

Transport equation (in MT3DMS) when porosity (θ) is constant

$$R\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right) - \frac{\partial}{\partial x}\left({\color{red}v}C\right) + q_s C_s + \sum_{k=1}^{N} R_k \tag{2}$$

my_first_flopy_mt3dms_scipt.py

1. Create a MODFLOW model object
2. Define packages (DIS, BAS, LPF, WELL, OC, PCG)
3. Add LMT (Link-MT3DMS) package
4. Write MODFLOW inputs
5. Run MODFLOW
6. Create a MT3DMS model object
7. Define MT3DMS packages (BTN, ADV)
8. Write MODFLOW inputs
9. Run MT3DMS
10. Post-process and plot results

MT3DMS Input-Output structure

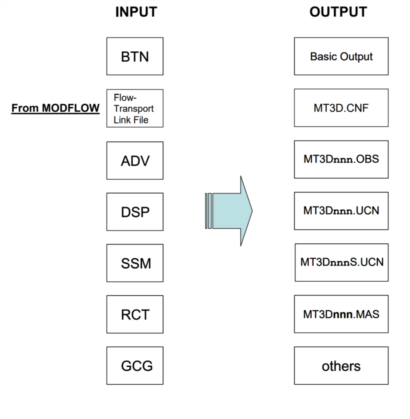


Figure 1: from
`http://inside.mines.edu/~epoeter/583CSM/12-1_MT3D.pdf`

my_first_flopy_mt3dms_scipt.py

1. MODFLOW simulation with LMT package
2. Create a MT3DMS model object
3. Define MT3DMS BTN packages (BTN)
4. Define MT3DMS Advection packages (ADV)
5. Define MT3DMS Dispersion packages (DSP)
6. Define MT3DMS Source/Sink Mixing packages (SSM)
7. Define MT3DMS Matrix Solver packages (GCG)
8. Write MT3DMS inputs
9. Run MT3DMS

Flopy MT3DMS modeling

```
# create mt3dms model object
mt = flopy.mt3d.Mt3dms(modflowmodel=mf,
                       modelname=modelname,
                       exe_name='./mt3dms5b.exe',
                       ftlfilename='mt3d_link.ftl')
#  for Mac and Linux: exe_name='./mt3dms'
```

Basic Transport Package (BTN)

```
btn = flopy.mt3d.Mt3dBtn(mt, prsity=0.3, icbund = 1,
                         sconc=0.0, ncomp=1,
                         perlen = 1000, nper=1,
                         nstp = 51, tsmult = 1.0,
                         nprs = -1, nprobs = 10,
                         cinact = -1, chkmas=True)
```

prsity porosity array(nlay,nrow,ncol)

icbund = 0 inactive, <0 constant C, >0 active, array(nlay,nr,nc)

sconc start concentration array(nlay,nrow,ncol)

cinact value for inactive concentration cell

nprs > 0 simulation saved as specified in “timprs” parameter
= 0, only saved at the end of simulation,
< 0 saved whenever the number of transport steps is an even multiple of nprs

Basic Transport Package (BTN)

- Basic information
- Spatial discretization (same as MODFLOW)
- Boundary and initial conditions
- Output control
- Temporal discretization

Boundary Conditions

- No-flow boundary in MODFLOW - Zero mass flux boundary
- all other boundaries in MODFLOW, treated as specified mass flux boundary with mass flux Q*C

Basic Transport Package (BTN) - Temporal discretization

- PERLEN : An array of the stress period lengths (for steady-state, total simulation time)
- NSTP : Number of time steps in each stress period
- TSMULT : Time step multiplier
- DT0: initial transport stepsize
- TTSMULT: transport stepsize multiplier within a flow multiplier within a flow-model time model time step
- TTSMAX: maximum transport stepsize within a flow-model time step

Advection Package (ADV)

```
adv = flopy.mt3d.Mt3dAdv(mt, mixelm=-1,
                         percel=0.75)
```

Solution method (mixelm)

- Finite Difference Method (FDM)
- MOC : Method of Characteristics (MOC)
- MMOC : Modified Method of Characteristics (MMOC)
- HMOC : Hybrid Method of Characteristics (HMOC)
- TVD (MIXELM = -1 - try to use this one)

percel is the Courant number for numerical stability (≤ 1)

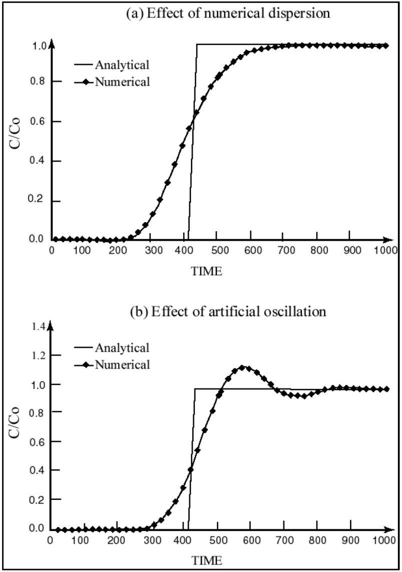


Figure 2: MT3DMS userguide (Report SERDP-99-1) Figure 1 - Illustration of common numerical errors in contaminant transport modeling

Dispersion Package (DSP)

```
dsp = flopy.mt3d.Mt3dDsp(mt, al=10.0, trpt=0.1,
                          trpv=0.1, dmcoef=1e-05)
```

Under some assumption:

$$
\begin{aligned}
D_L &= \alpha_L v_L + D^* \\
D_T &= \alpha_T v_T + D^*
\end{aligned}
$$

- al (α_L): longitudinal dispersivity [L]
- tprt : ratio of horizontal transverse dispersivity vs longitudinal dispersivity (0.01 - 0.1)
- trpv : ratio of vertical transverse (0.001 - 0.01) dispersivity vs longitudinal dispersivity
- dmcoef (D^*) : diffusion coefficient $[L^2/T]$

Dispersion modeling and its parameter assignment are very

Source/Sink Mixing Package (SSM)

```
ssm_data = {}
itype = flopy.mt3d.Mt3dSsm.itype_dict()
# print(itype)
ssm_data[0] = [(0, wrow, wcol, 10.0, itype['WEL'])]
# ssm_data.append((0, wrow1, wcol1, Q1, itype['WEL']))
ssm = flopy.mt3d.Mt3dSsm(mt,
                         stress_period_data=ssm_data)
```

Matrix Solver Package (GCG)

```
gcg = flopy.mt3d.Mt3dGcg(mt, cclose=1e-6)
```

cclose the convergence criterion in terms of relative concentration

Write input files for MT3DMS and Execution

```
# write mt3dms input
mt.write_input()
# run mt3dms
mt.run_model()
#mt.run_model(silent=True)
```

Simulation screen (1)

```
MT3DMS - Modular 3-D Multi-Species Transport Model [Version 5.30]
Developed at University of Alabama for U.S. Department of Defense

Using NAME File: mf-mt.nam

STRESS PERIOD NO.    1

TIME STEP NO.    1
FROM TIME =   0.0000     TO    20.000

Transport Step:    1   Step Size:    7.678     Total Elapsed Time:   7.6784
Outer Iter.  1  Inner Iter.   1:  Max. DC =   8.272      [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.   2:  Max. DC =  0.6572E-01  [K,I,J]    1   25    7
Outer Iter.  1  Inner Iter.   3:  Max. DC =  0.1469E-03  [K,I,J]    1   27    6
Outer Iter.  1  Inner Iter.   4:  Max. DC =  0.2146E-05  [K,I,J]    1   26    7
Outer Iter.  1  Inner Iter.   5:  Max. DC =  0.7451E-08  [K,I,J]    1   25    7
Transport Step:    2   Step Size:    7.678     Total Elapsed Time:   15.357
Outer Iter.  1  Inner Iter.   1:  Max. DC =   2.228      [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.   2:  Max. DC =  0.1770E-01  [K,I,J]    1   25    7
Outer Iter.  1  Inner Iter.   3:  Max. DC =  0.4137E-04  [K,I,J]    1   26    6
```

Simulation screen (2)

```
Outer Iter.  1  Inner Iter.  4:  Max. DC =  0.5684E-13  [K,I,J]    1   16   19

TIME STEP NO.   50
FROM TIME =   980.00     TO    1000.0

Transport Step:    1   Step Size:   7.678     Total Elapsed Time:   987.68
Outer Iter.  1  Inner Iter.  1:  Max. DC =   1.246      [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.  2:  Max. DC =  0.9935E-02  [K,I,J]    1   25    7
Outer Iter.  1  Inner Iter.  3:  Max. DC =  0.2354E-04  [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.  4:  Max. DC =  0.1788E-06  [K,I,J]    1   27    6
Transport Step:    2   Step Size:   7.678     Total Elapsed Time:   995.36
Outer Iter.  1  Inner Iter.  1:  Max. DC =   1.246      [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.  2:  Max. DC =  0.9936E-02  [K,I,J]    1   25    7
Outer Iter.  1  Inner Iter.  3:  Max. DC =  0.2354E-04  [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.  4:  Max. DC =  0.1788E-06  [K,I,J]    1   27    6
Transport Step:    3   Step Size:   4.643     Total Elapsed Time:   1000.0
Outer Iter.  1  Inner Iter.  1:  Max. DC =  0.7532      [K,I,J]    1   26    6
Outer Iter.  1  Inner Iter.  2:  Max. DC =  0.2588E-02  [K,I,J]    1   25    7
Outer Iter.  1  Inner Iter.  3:  Max. DC =  0.2265E-05  [K,I,J]    1   27    5
Outer Iter.  1  Inner Iter.  4:  Max. DC =  0.7105E-14  [K,I,J]    1   16   10
```

Plotting results

```
ucnobj = bf.UcnFile('MT3D001.UCN')

times = ucnobj.get_times() # simulation time
#times1 = times[round(len(times)/5.)-1] # 1/5 simulatio
#times2 = times[round(len(times)/2.)-1] # 1/2 simulatio
mytime = times[-1] # the last simulation time
conc = ucnobj.get_data(totim=mytime)
```