installing

April 16, 2023

1 MODFLOW6 on a JupyterHub server

This is pretty elaborate, but once done would enable one to run in a Jupyter Notebook.

1.1 Install FloPy

`sudo -H /opt/jupyterhub/bin/python3 -m pip install flopy`

1.2 Install MODFLOW

If you have intel/AMD chipset (processor) then you can get binaries directly by `! get-modflow /home/sensei/playground/modflow-python/mf6.4.1_linux/bin`

In the above command, run from Jupyter, the absolute path to the binary is specified - it will be where you choose (my path won't work on your machine!). The syntax is `get-modflow /path/to/bin`

If you are on aarch/arm chipset, then you will have to build from source - there are gfortran makefiles available on-line.

To test the install the following example should suffice:

[1]:

```
#RUN ONCE
# try to install current version modflow - here we send command to the os shell
#! get-modflow /home/sensei/playground/modflow-python/mf6.4.1_linux/bin
# note: sensei (local) is bin owner, may want to rerun as root in future
```

[2]:

```
# FloPy Examples
import warnings
warnings.filterwarnings('ignore') # suppress warnings (there are several!)
# Now attempt an example
import os
import numpy as np
import matplotlib.pyplot as plt
import flopy
```

[3]:

```
name = "example01_mf6"
h1 = 100
h2 = 90
Nlay = 10 #number layers
```

```
N = 101 # number rows and columns Nr=Nc  (a square)
L = 400.0 # length of sides
H = 50.0 # aquifer thickness
k = 1.0 # hydraulic condictivity
workspace = "./modflow-python/" + name # this appears in last few blocks below
```

[4]:

```
sim = flopy.mf6.MFSimulation(
    sim_name=name, exe_name="/home/sensei/playground/modflow-python/mf6.4.
↪1_linux/bin/mf6", version="mf6", sim_ws="./modflow-python/" + name
)
```

[5]:

```
tdis = flopy.mf6.ModflowTdis(
    sim, pname="tdis", time_units="DAYS", nper=1, perioddata=[(1.0, 1, 1.0)]
)
```

[6]:

```
ims = flopy.mf6.ModflowIms(sim, pname="ims", complexity="SIMPLE")
```

[7]:

```
model_nam_file = "{}.nam".format(name)
gwf = flopy.mf6.ModflowGwf(sim, modelname=name, model_nam_file=model_nam_file)
```

[8]:

```
bot = np.linspace(-H / Nlay, -H, Nlay)
delrow = delcol = L / (N - 1)
dis = flopy.mf6.ModflowGwfdis(
    gwf,
    nlay=Nlay,
    nrow=N,
    ncol=N,
    delr=delrow,
    delc=delcol,
    top=0.0,
    botm=bot,
)
```

[9]:

```
start = h1 * np.ones((Nlay, N, N))
ic = flopy.mf6.ModflowGwfic(gwf, pname="ic", strt=start)
```

[10]:

```
npf = flopy.mf6.ModflowGwfnpf(gwf, icelltype=1, k=k, save_flows=True)
```

[11]:

```
chd_rec = []
chd_rec.append(((0, int(N / 4), int(N / 4)), h2)) # set head in corner of top␣
↪layer to h2
for layer in range(0, Nlay):
    for row_col in range(0, N):
        chd_rec.append(((layer, row_col, 0), h1)) #set head left column all␣
↪layers to h1
        chd_rec.append(((layer, row_col, N - 1), h1)) #set right column all␣
↪layers to h1
```

```
        if row_col != 0 and row_col != N - 1:
            chd_rec.append(((layer, 0, row_col), h1)) #set top row all layers␣
 ↪to h1
            chd_rec.append(((layer, N - 1, row_col), h1)) #set bottom row all␣
 ↪layers to h1
chd = flopy.mf6.ModflowGwfchd(
    gwf,
    maxbound=len(chd_rec),
    stress_period_data=chd_rec,
    save_flows=True,
)
```

[12]:

```
iper = 0
ra = chd.stress_period_data.get_data(key=iper)
ra
```

[12]:

```
rec.array([((0, 25, 25),  90.), ((0, 0, 0), 100.), ((0, 0, 100), 100.),
           …, ((9, 100, 99), 100.), ((9, 100, 0), 100.),
           ((9, 100, 100), 100.)],
          dtype=[('cellid', 'O'), ('head', '<f8')])
```

[13]:

```
# Create the output control (`OC`) Package
headfile = "{}.hds".format(name)
head_filerecord = [headfile]
budgetfile = "{}.cbb".format(name)
budget_filerecord = [budgetfile]
saverecord = [("HEAD", "ALL"), ("BUDGET", "ALL")]
printrecord = [("HEAD", "LAST")]
oc = flopy.mf6.ModflowGwfoc(
    gwf,
    saverecord=saverecord,
    head_filerecord=head_filerecord,
    budget_filerecord=budget_filerecord,
    printrecord=printrecord,
)
```

[14]:

```
sim.write_simulation()
```

```
writing simulation…
  writing simulation name file…
  writing simulation tdis package…
  writing ims package ims…
  writing model example01_mf6…
    writing model name file…
    writing package dis…
    writing package ic…
    writing package npf…
    writing package chd_0…
    writing package oc…
```

[15]:

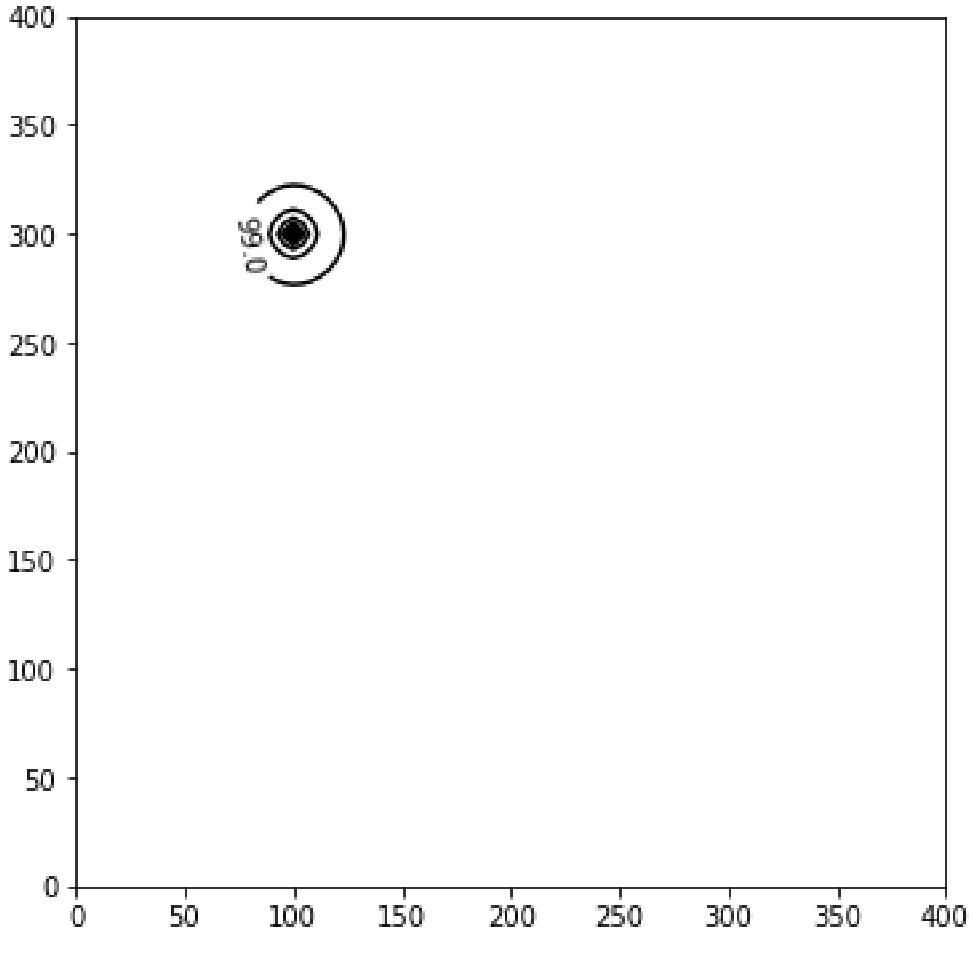
```
#import tracemalloc
#tracemalloc.start() # this is to suppress an asyncronous warning

# attempt to run the model, will see if binary loaded OK
success, buff = sim.run_simulation()
if not success:
    raise Exception("MODFLOW 6 did not terminate normally.")

#current, peak = tracemalloc.get_traced_memory()
#print("Current memory usage is %d bytes; peak was %d bytes" % (current, peak))
```

```
FloPy is using the following executable to run the model:
/home/sensei/playground/modflow-python/mf6.4.1_linux/bin/mf6
                                   MODFLOW 6
                U.S. GEOLOGICAL SURVEY MODULAR HYDROLOGIC MODEL
                        VERSION 6.4.1 Release 12/09/2022

   MODFLOW 6 compiled Apr 12 2023 19:02:29 with Intel(R) Fortran Intel(R) 64
   Compiler Classic for applications running on Intel(R) 64, Version 2021.7.0
                             Build 20220726_000000

This software has been approved for release by the U.S. Geological
Survey (USGS). Although the software has been subjected to rigorous
review, the USGS reserves the right to update the software as needed
pursuant to further analysis and review. No warranty, expressed or
implied, is made by the USGS or the U.S. Government as to the
functionality of the software and related material nor shall the
fact of release constitute any such warranty. Furthermore, the
software is released on condition that neither the USGS nor the U.S.
Government shall be held liable for any damages resulting from its
authorized or unauthorized use. Also refer to the USGS Water
Resources Software User Rights Notice for complete use, copyright,
and distribution information.


 Run start date and time (yyyy/mm/dd hh:mm:ss): 2023/04/16 11:50:11

 Writing simulation list file: mfsim.lst
 Using Simulation name file: mfsim.nam

    Solving:  Stress period:     1    Time step:     1

 Run end date and time (yyyy/mm/dd hh:mm:ss): 2023/04/16 11:50:11
 Elapsed run time:  0.555 Seconds
```

```
Normal termination of simulation.
```

[16]:
```
# now attempt to postprocess
h = gwf.output.head().get_data(kstpkper=(0, 0))
x = y = np.linspace(0, L, N)
y = y[::-1]
vmin, vmax = 90.0, 100.0
contour_intervals = np.arange(90, 100.1, 1.0)

# ### Plot a Map of Layer 1

fig = plt.figure(figsize=(6, 6))
ax = fig.add_subplot(1, 1, 1, aspect="equal")
c = ax.contour(x, y, h[0], contour_intervals, colors="black")
plt.clabel(c, fmt="%2.1f")
```

[16]:
```
<a list of 1 text.Text objects>
```



[ ]:

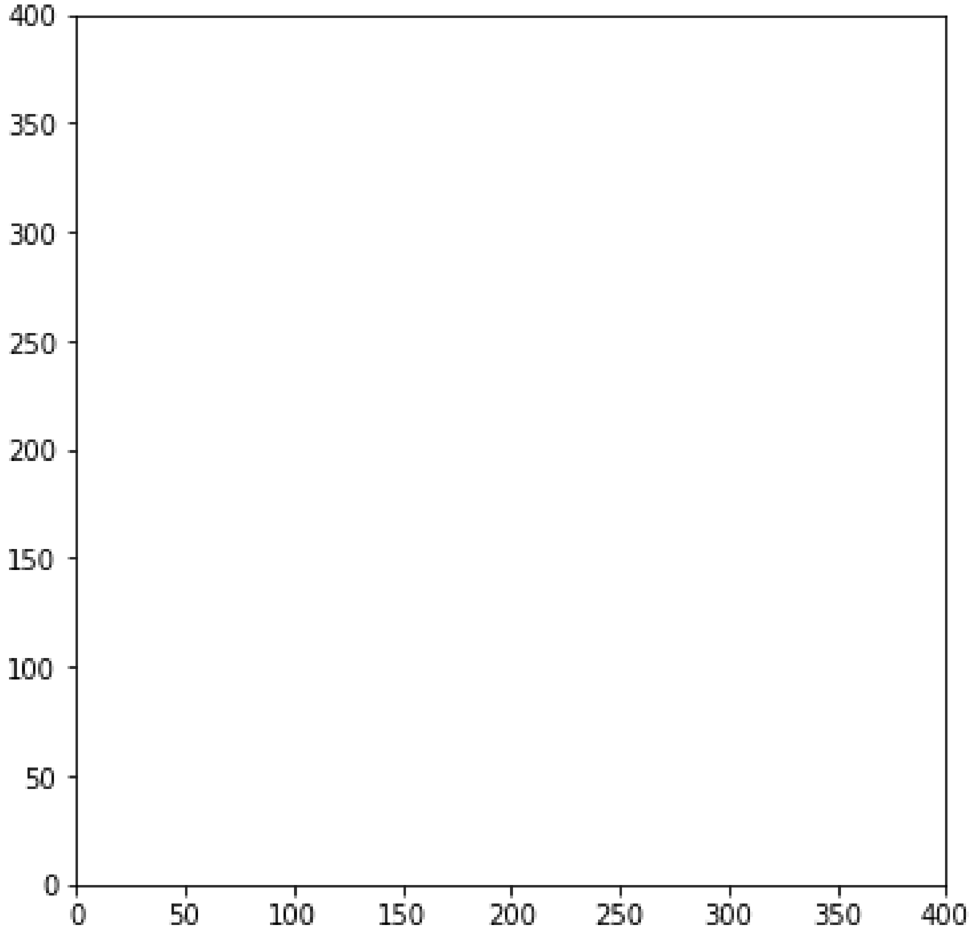
```
```

[17]:

```
# ### Plot a Map of Layer 10

x = y = np.linspace(0, L, N)
y = y[::-1]
fig = plt.figure(figsize=(6, 6))
ax = fig.add_subplot(1, 1, 1, aspect="equal")
c = ax.contour(x, y, h[-1], contour_intervals, colors="black")
plt.clabel(c, fmt="%1.1f")
```

[17]:

```
<a list of 0 text.Text objects>
```



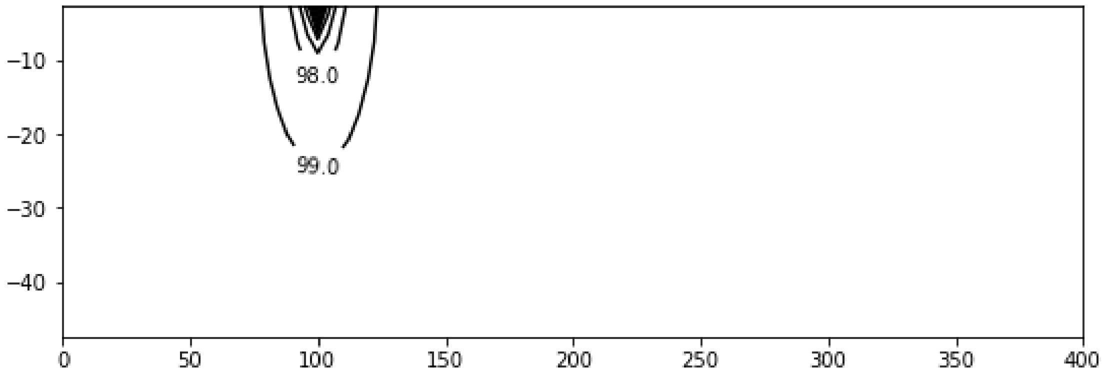
[18]:

```
# ### Plot a Cross-section along row 25

z = np.linspace(-H / Nlay / 2, -H + H / Nlay / 2, Nlay)
fig = plt.figure(figsize=(9, 3))
```

```python
ax = fig.add_subplot(1, 1, 1, aspect="auto")
c = ax.contour(x, z, h[:, int(N / 4), :], contour_intervals, colors="black")
plt.clabel(c, fmt="%1.1f")
```

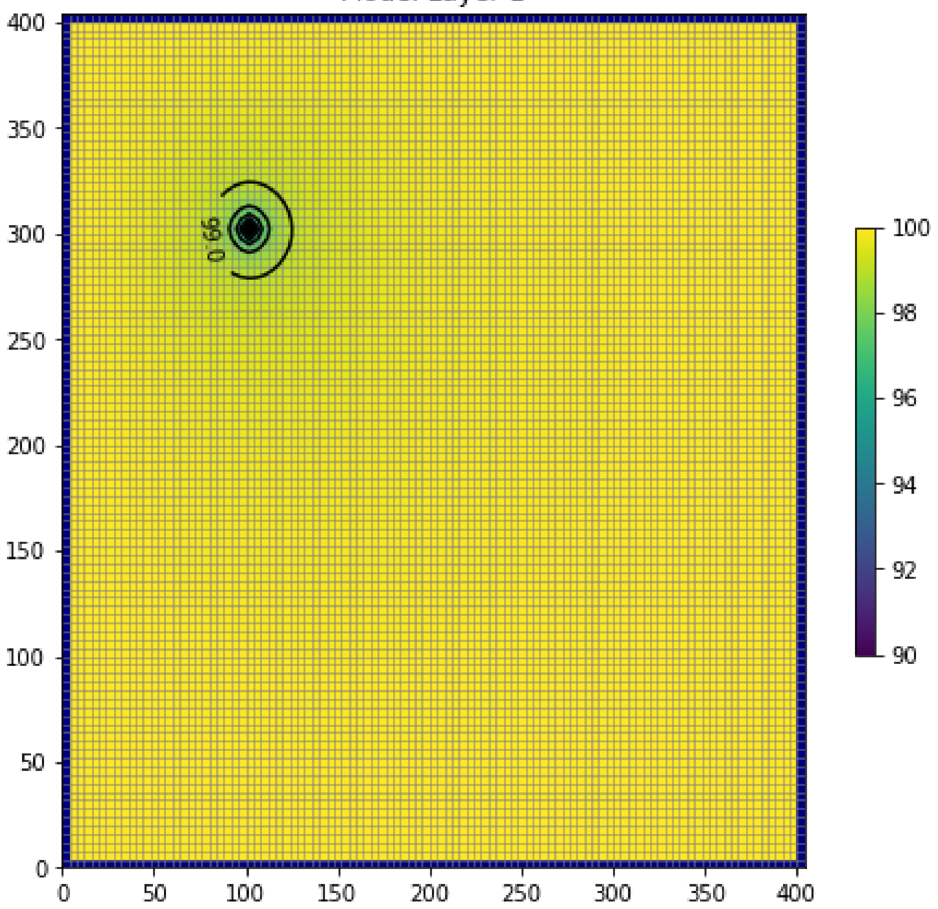
[18]:

```
<a list of 2 text.Text objects>
```

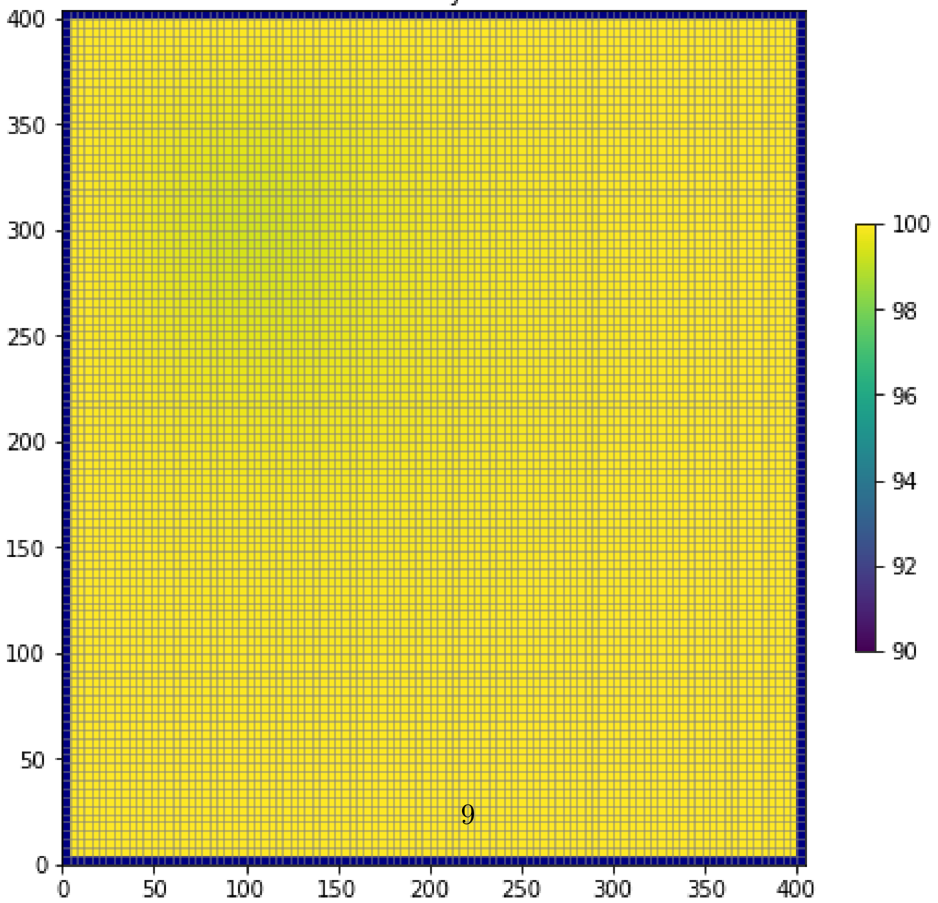


[19]:

```python
# ### Use the FloPy `PlotMapView()` capabilities for MODFLOW 6
#
# ### Plot a Map of heads in Layers 1 and 10

fig, axes = plt.subplots(2, 1, figsize=(6, 12), constrained_layout=True)
# first subplot
ax = axes[0]
ax.set_title("Model Layer 1")
modelmap = flopy.plot.PlotMapView(model=gwf, ax=ax)
pa = modelmap.plot_array(h, vmin=vmin, vmax=vmax)
quadmesh = modelmap.plot_bc("CHD")
linecollection = modelmap.plot_grid(lw=0.5, color="0.5")
contours = modelmap.contour_array(
    h,
    levels=contour_intervals,
    colors="black",
)
ax.clabel(contours, fmt="%2.1f")
cb = plt.colorbar(pa, shrink=0.5, ax=ax)
# second subplot
ax = axes[1]
ax.set_title(f"Model Layer {Nlay}")
modelmap = flopy.plot.PlotMapView(model=gwf, ax=ax, layer=Nlay - 1)
linecollection = modelmap.plot_grid(lw=0.5, color="0.5")
pa = modelmap.plot_array(h, vmin=vmin, vmax=vmax)
quadmesh = modelmap.plot_bc("CHD")
contours = modelmap.contour_array(
    h,
    levels=contour_intervals,
    colors="black",
)
ax.clabel(contours, fmt="%2.1f")
cb = plt.colorbar(pa, shrink=0.5, ax=ax)
```



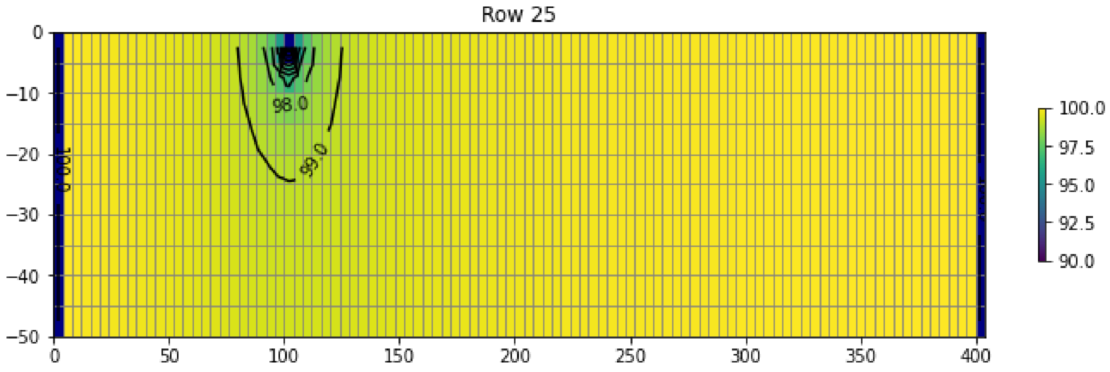
Model Layer 1



Model Layer 10

[20]:

```
# ### Use the FloPy `PlotCrossSection()` capabilities for MODFLOW 6
#
# ### Plot a cross-section of heads along row 25

fig, ax = plt.subplots(1, 1, figsize=(9, 3), constrained_layout=True)
# first subplot
ax.set_title("Row 25")
modelmap = flopy.plot.PlotCrossSection(
    model=gwf,
    ax=ax,
    line={"row": int(N / 4)},
)
pa = modelmap.plot_array(h, vmin=vmin, vmax=vmax)
quadmesh = modelmap.plot_bc("CHD")
linecollection = modelmap.plot_grid(lw=0.5, color="0.5")
contours = modelmap.contour_array(
    h,
    levels=contour_intervals,
    colors="black",
)
ax.clabel(contours, fmt="%2.1f")
cb = plt.colorbar(pa, shrink=0.5, ax=ax)
```



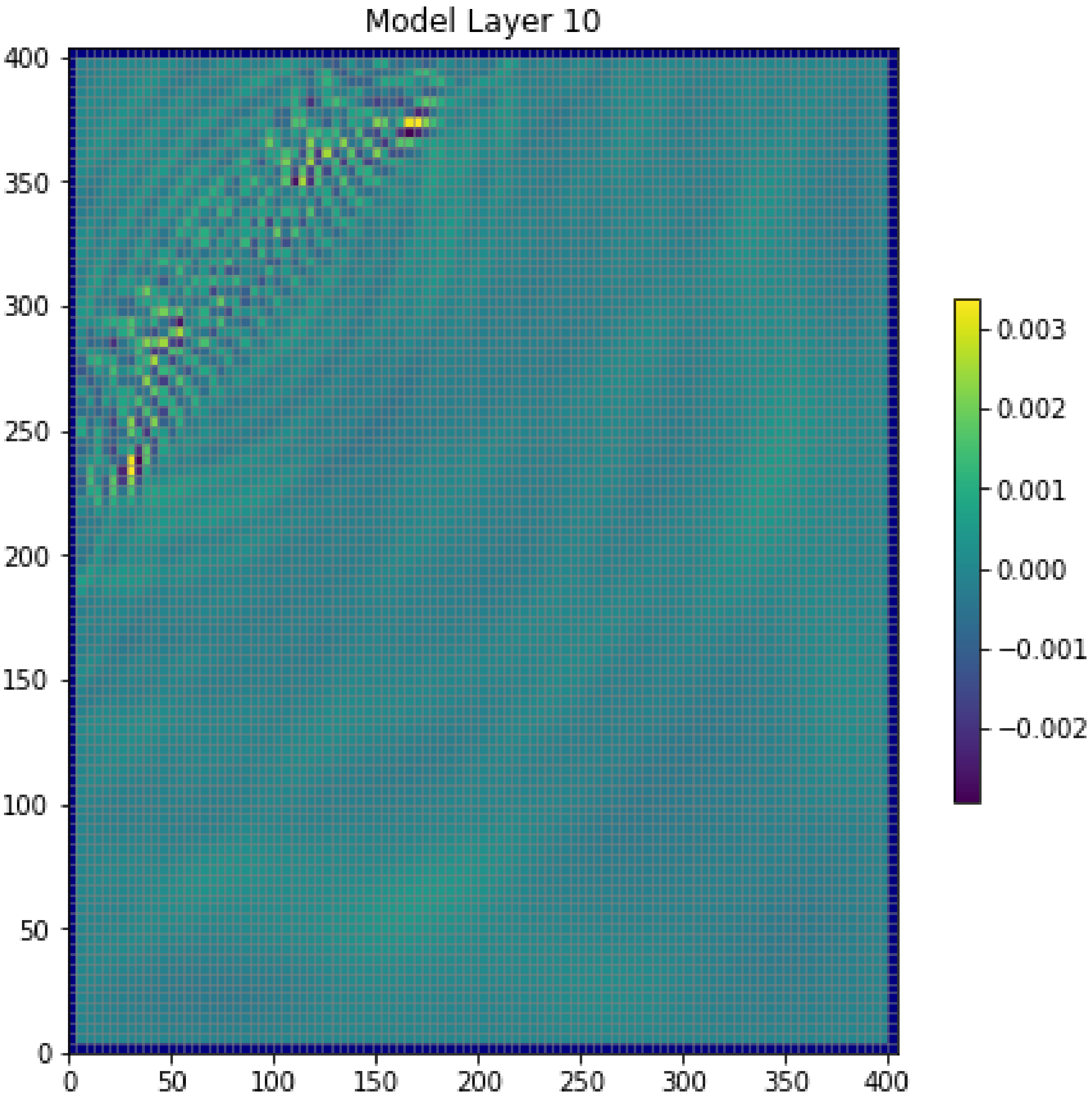
[21]:

```
# ## Determine the Flow Residual
#
# The `FLOW-JA-FACE` cell-by-cell budget data can be processed to
# determine the flow residual for each cell in a MODFLOW 6 model. The
# diagonal position for each row in the `FLOW-JA-FACE` cell-by-cell
# budget data contains the flow residual for each cell and can be
# extracted using the `flopy.mf6.utils.get_residuals()` function.
#
```

```python
# First extract the `FLOW-JA-FACE` array from the cell-by-cell budget file

flowja = gwf.oc.output.budget().get_data(text="FLOW-JA-FACE", kstpkper=(0, 0))[
    0
]

# Next extract the flow residual. The MODFLOW 6 binary grid file is passed
# into the function because it contains the ia array that defines
# the location of the diagonal position in the `FLOW-JA-FACE` array.
print(workspace)
grb_file = workspace + "/" + f"{name}.dis.grb"
#grb_file = workspace + ".dis.grb"
residual = flopy.mf6.utils.get_residuals(flowja, grb_file=grb_file)

# ### Plot a Map of the flow error in Layer 10

fig, ax = plt.subplots(1, 1, figsize=(6, 6), constrained_layout=True)
ax.set_title("Model Layer 10")
modelmap = flopy.plot.PlotMapView(model=gwf, ax=ax, layer=Nlay - 1)
pa = modelmap.plot_array(residual)
quadmesh = modelmap.plot_bc("CHD")
linecollection = modelmap.plot_grid(lw=0.5, color="0.5")
contours = modelmap.contour_array(
    h,
    levels=contour_intervals,
    colors="black",
)
ax.clabel(contours, fmt="%2.1f")
plt.colorbar(pa, shrink=0.5)
```

```
./modflow-python/example01_mf6
```

```
[21]: <matplotlib.colorbar.Colorbar at 0x7f5fc1466160>
```


Model Layer 10
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300
350
400
0.003
0.002
0.001
0.000
−0.001
−0.002