dataworth

July 16, 2019

1 data worth and related assessments

In this notebook, we will use outputs from previous notebooks (in particular `pestpp-glm_part1.ipynb`) to undertake data worth assessments based on first-order second-moment (FOSM) techniques. "Worth" is framed here in the context of the extent to which the uncertainty surrounding a model prediction of management interest is reduced through data collection. Given that these anayses can help target and optimize data acquisition strategies, this is a concept that really resonates with decision makers.

In [1]:
```
%matplotlib inline
import os
import shutil
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.rcParams['font.size']=12
import flopy
import pyemu
```

```
flopy is installed in C:\Users\knowling\Dev\GW1876\activities_csiro\notebooks\flopy
```

In [2]:
```
m_d = "master_glm"
```

In [3]:
```
pst = pyemu.Pst(os.path.join(m_d,"freyberg_pp.pst"))
print(pst.npar_adj)
pst.write_par_summary_table(filename="none")
```

```
527
```

Out[3]:
```
                         type transform  count initial value upper bound  \
cn_hk6              cn_hk6       log      1             0           1
cn_hk7              cn_hk7       log      1             0           1
cn_hk8              cn_hk8       log      1             0           1
cn_prsity6      cn_prsity6       log      1             0           1
cn_prsity7      cn_prsity7       log      1             0           1
```

cn_prsity8	cn_prsity8	log	1	0	1
cn_rech4	cn_rech4	log	1	0	0.0413927
cn_rech5	cn_rech5	log	1	0	0.0413927
cn_ss6	cn_ss6	log	1	0	1
cn_ss7	cn_ss7	log	1	0	1
cn_ss8	cn_ss8	log	1	0	1
cn_strt6	cn_strt6	log	1	0	0.0211893
cn_strt7	cn_strt7	log	1	0	0.0211893
cn_strt8	cn_strt8	log	1	0	0.0211893
cn_sy6	cn_sy6	log	1	0	0.243038
cn_sy7	cn_sy7	log	1	0	0.243038
cn_sy8	cn_sy8	log	1	0	0.243038
cn_vka6	cn_vka6	log	1	0	1
cn_vka7	cn_vka7	log	1	0	1
cn_vka8	cn_vka8	log	1	0	1
drncond_k00	drncond_k00	log	10	0	1
flow	flow	log	1	0	0.09691
gr_hk3	gr_hk3	fixed	705	1	10
gr_hk4	gr_hk4	fixed	705	1	10
gr_hk5	gr_hk5	fixed	705	1	10
gr_prsity3	gr_prsity3	fixed	705	1	10
gr_prsity4	gr_prsity4	fixed	705	1	10
gr_prsity5	gr_prsity5	fixed	705	1	10
gr_rech2	gr_rech2	fixed	705	1	1.1
gr_rech3	gr_rech3	fixed	705	1	1.1
...	...	...	...	...	...
gr_strt5	gr_strt5	fixed	705	1	1.05
gr_sy3	gr_sy3	fixed	705	1	1.75
gr_sy4	gr_sy4	fixed	705	1	1.75
gr_sy5	gr_sy5	fixed	705	1	1.75
gr_vka3	gr_vka3	fixed	705	1	10
gr_vka4	gr_vka4	fixed	705	1	10
gr_vka5	gr_vka5	fixed	705	1	10
pp_hk0	pp_hk0	log	32	0	1
pp_hk1	pp_hk1	log	32	0	1
pp_hk2	pp_hk2	log	32	0	1
pp_prsity0	pp_prsity0	log	32	0	10.0414
pp_prsity1	pp_prsity1	log	32	0	10.0414
pp_prsity2	pp_prsity2	log	32	0	10.0414
pp_rech0	pp_rech0	log	32	0	0.0413927
pp_rech1	pp_rech1	fixed	32	1	1.1
pp_ss0	pp_ss0	log	32	0	1
pp_ss1	pp_ss1	log	32	0	1
pp_ss2	pp_ss2	log	32	0	1
pp_strt0	pp_strt0	fixed	32	1	1.05
pp_strt1	pp_strt1	fixed	32	1	1.05
pp_strt2	pp_strt2	fixed	32	1	1.05
pp_sy0	pp_sy0	log	32	0	0.243038

pp_sy1	pp_sy1	log	32	0	0.243038
pp_sy2	pp_sy2	log	32	0	0.243038
pp_vka0	pp_vka0	fixed	32	1	10
pp_vka1	pp_vka1	log	32	0	1
pp_vka2	pp_vka2	fixed	32	1	10
strk	strk	log	40	0	2
welflux	welflux	log	2	0	1
welflux_k02	welflux_k02	log	6	0	1

	lower bound	standard deviation
cn_hk6	-1	0.5
cn_hk7	-1	0.5
cn_hk8	-1	0.5
cn_prsity6	-1	0.5
cn_prsity7	-1	0.5
cn_prsity8	-1	0.5
cn_rech4	-0.0457575	0.0217875
cn_rech5	-0.0457575	0.0217875
cn_ss6	-1	0.5
cn_ss7	-1	0.5
cn_ss8	-1	0.5
cn_strt6	-0.0222764	0.0108664
cn_strt7	-0.0222764	0.0108664
cn_strt8	-0.0222764	0.0108664
cn_sy6	-0.60206	0.211275
cn_sy7	-0.60206	0.211275
cn_sy8	-0.60206	0.211275
cn_vka6	-1	0.5
cn_vka7	-1	0.5
cn_vka8	-1	0.5
drncond_k00	-1	0.5
flow	-0.124939	0.0554622
gr_hk3	0.1	2.475
gr_hk4	0.1	2.475
gr_hk5	0.1	2.475
gr_prsity3	0.1	2.475
gr_prsity4	0.1	2.475
gr_prsity5	0.1	2.475
gr_rech2	0.9	0.05
gr_rech3	0.9	0.05
...	...	...
gr_strt5	0.95	0.025
gr_sy3	0.25	0.375
gr_sy4	0.25	0.375
gr_sy5	0.25	0.375
gr_vka3	0.1	2.475
gr_vka4	0.1	2.475
gr_vka5	0.1	2.475

```
pp_hk0              -1            0.5
pp_hk1              -1            0.5
pp_hk2              -1            0.5
pp_prsity0    -9.95861              5
pp_prsity1    -9.95861              5
pp_prsity2    -9.95861              5
pp_rech0    -0.0457575      0.0217875
pp_rech1           0.9           0.05
pp_ss0              -1            0.5
pp_ss1              -1            0.5
pp_ss2              -1            0.5
pp_strt0          0.95          0.025
pp_strt1          0.95          0.025
pp_strt2          0.95          0.025
pp_sy0        -0.60206       0.211275
pp_sy1        -0.60206       0.211275
pp_sy2        -0.60206       0.211275
pp_vka0            0.1          2.475
pp_vka1             -1            0.5
pp_vka2            0.1          2.475
strk                -2              1
welflux             -1            0.5
welflux_k02         -1            0.5

[65 rows x 7 columns]
```

first ingredient: parameter covariance matrix (representing prior uncertainty in this instance)

```
In [4]: cov = pyemu.Cov.from_binary(os.path.join(m_d,"prior_cov.jcb")).to_dataframe()
        cov = cov.loc[pst.adj_par_names,pst.adj_par_names]
        cov = pyemu.Cov.from_dataframe(cov)
```

```
new binary format detected...
```

second ingredient: jacobian matrix

```
In [5]: jco = os.path.join(m_d,"freyberg_pp.jcb")
```

the third ingredient--the (diagonal) noise covariance matrix--populated on-the-fly using weights when constructing the Schur object below...

```
In [6]: sc = pyemu.Schur(jco=jco,parcov=cov)
```

1.0.1 there we have it--all computations done and contained within `sc`. We will only be required to access different parts of `sc` below...

1.0.2 Parameter uncertainty

First let's inspect the (approx) posterior parameter covariance matrix and the reduction in parameter uncertainty through "data assimilation", before mapping to forecasts... (note that this matrix is *not* forecast-specific)

```
In [7]: sc.posterior_parameter.to_dataframe().sort_index(axis=1).iloc[100:105:,100:105]
```

Out[7]:

	hk225	hk226	hk227	hk228	hk229
hk225	0.093333	0.019926	0.003091	0.001128	0.020917
hk226	0.019926	0.081598	0.019668	0.006277	0.029243
hk227	0.003091	0.019668	0.090720	0.030622	0.011145
hk228	0.001128	0.006277	0.030622	0.104660	0.001329
hk229	0.020917	0.029243	0.011145	0.001329	0.080951

We can see the posterior variance for each parameter along the diagonal. The off-diags are symmetric.

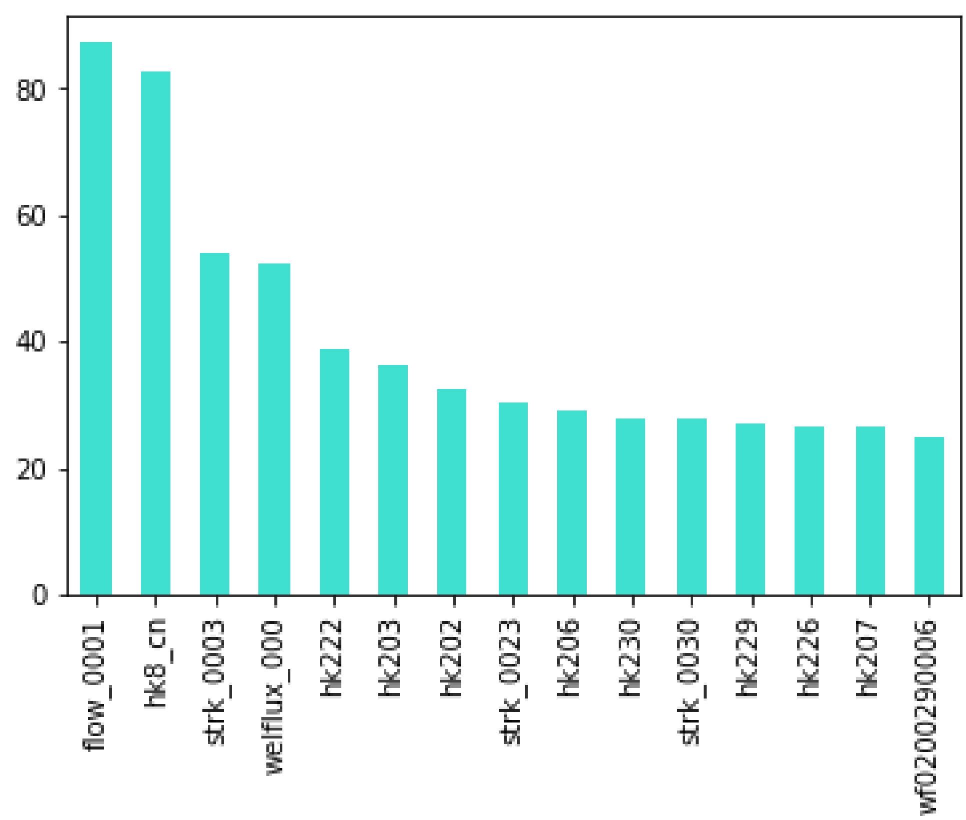
```
In [8]: par_sum = sc.get_parameter_summary().sort_values("percent_reduction",ascending=False)
        par_sum.head()
```

Out[8]:

	percent_reduction	post_var	prior_var
flow_0001	87.379227	0.000173	0.001367
hk8_cn	82.586485	0.019348	0.111111
strk_0003	54.023848	0.204338	0.444444
welflux_000	52.465889	0.052816	0.111111
hk222	38.842096	0.067953	0.111111

```
In [9]: par_sum.loc[par_sum.index[:15],"percent_reduction"].plot(kind="bar",color="turquoise")
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x28f0263ad30>



What have we achieved by "notionally calibrating" our model to 13 head and 1 stream flow observations? Which parameters are informed? Will they matter for the forecast of interest? Which ones are un-informed?

1.1 Forecast uncertainty

In [10]:
```
forecasts = sc.pst.pestpp_options['forecasts'].split(",")
forecasts
```

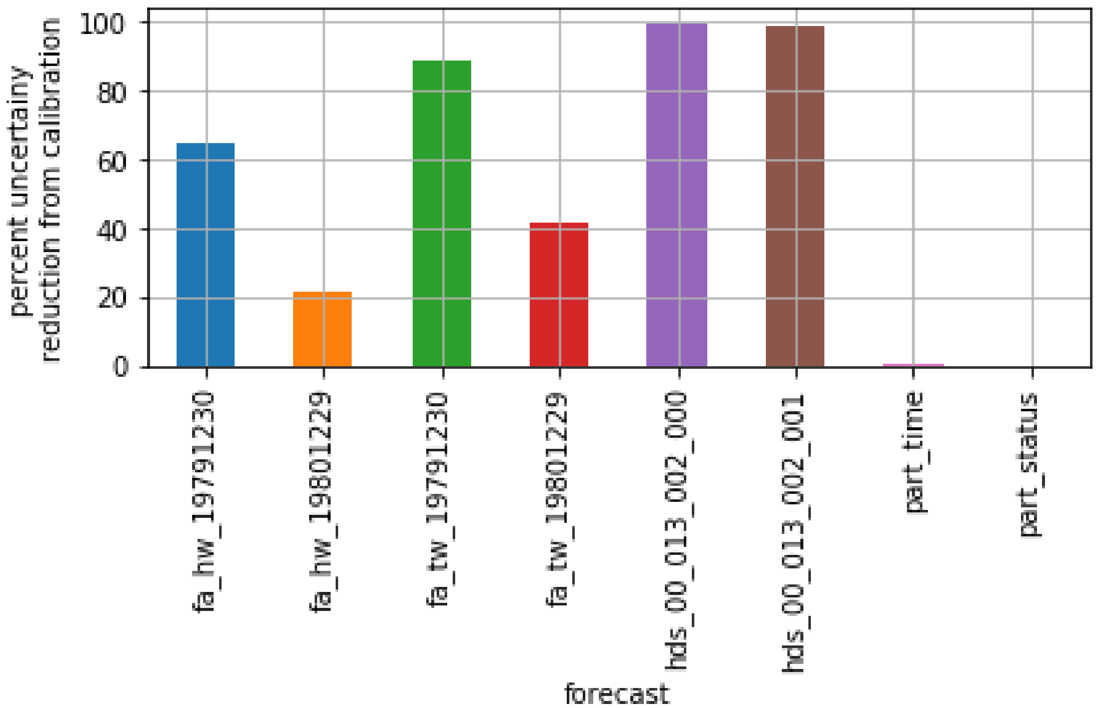
Out[10]:
```
['fa_hw_19791230',
 'fa_hw_19801229',
 'fa_tw_19791230',
 'fa_tw_19801229',
 'hds_00_013_002_000',
 'hds_00_013_002_001',
 'part_time',
 'part_status']
```

In [11]:
```
df = sc.get_forecast_summary()
df
```

Out[11]:

	percent_reduction	post_var	prior_var
fa_hw_19791230	64.350080	4.599921e+04	1.290303e+05
fa_hw_19801229	21.344325	2.731591e+05	3.472847e+05
fa_tw_19791230	88.127961	2.043748e+04	1.721480e+05
fa_tw_19801229	41.327422	2.184906e+05	3.723897e+05
hds_00_013_002_000	98.824410	8.972545e-02	7.632373e+00
hds_00_013_002_001	98.009129	1.689069e-01	8.484073e+00
part_time	0.779595	1.258207e+07	1.268093e+07
part_status	NaN	0.000000e+00	0.000000e+00

In [12]:
```
# make a pretty plot
fig = plt.figure()
ax = plt.subplot(111)
ax = df["percent_reduction"].plot(kind='bar',ax=ax,grid=True)
ax.set_ylabel("percent uncertainy\nreduction from calibration")
ax.set_xlabel("forecast")
plt.tight_layout()
```



Surprise, surprise... Some forecasts benefit from calibration, some do not!

1.1.1 Before moving onto data worth, let's look at the contribution of different parameters to forecast uncertainty

Parameter contributions to uncertainty are quantified by "fixing" parameters (or parameter groups) and observing the uncertainty reduction as a result. This approach is of course subject to some sizable assumptions--related to parameter representativeness. But it can be very informative. Let's do by group.

```
In [13]: par_contrib = sc.get_par_group_contribution()
```

```
In [14]: par_contrib.head()
```

```
Out[14]:              fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  fa_tw_19801229  \
         base           45999.208695   273159.126193    20437.475638   218490.640452
         cn_hk6         45670.367149   273002.628427    20337.569326   218435.182951
         cn_hk7         45999.203632   273159.121981    20437.473127   218490.580039
         cn_hk8         42643.883560   271632.419963    19337.356972   218274.514865
         cn_prsity6     45999.208695   273159.126193    20437.475638   218490.640452

                      hds_00_013_002_000  hds_00_013_002_001  part_status     part_time
         base                   0.089725            0.168907          0.0  1.258207e+07
         cn_hk6                 0.089376            0.168861          0.0  1.257663e+07
         cn_hk7                 0.089725            0.168907          0.0  1.258207e+07
```

```
cn_hk8                0.087930            0.165124           0.0   1.257893e+07
cn_prsity6            0.089725            0.168907           0.0   1.257428e+07
```

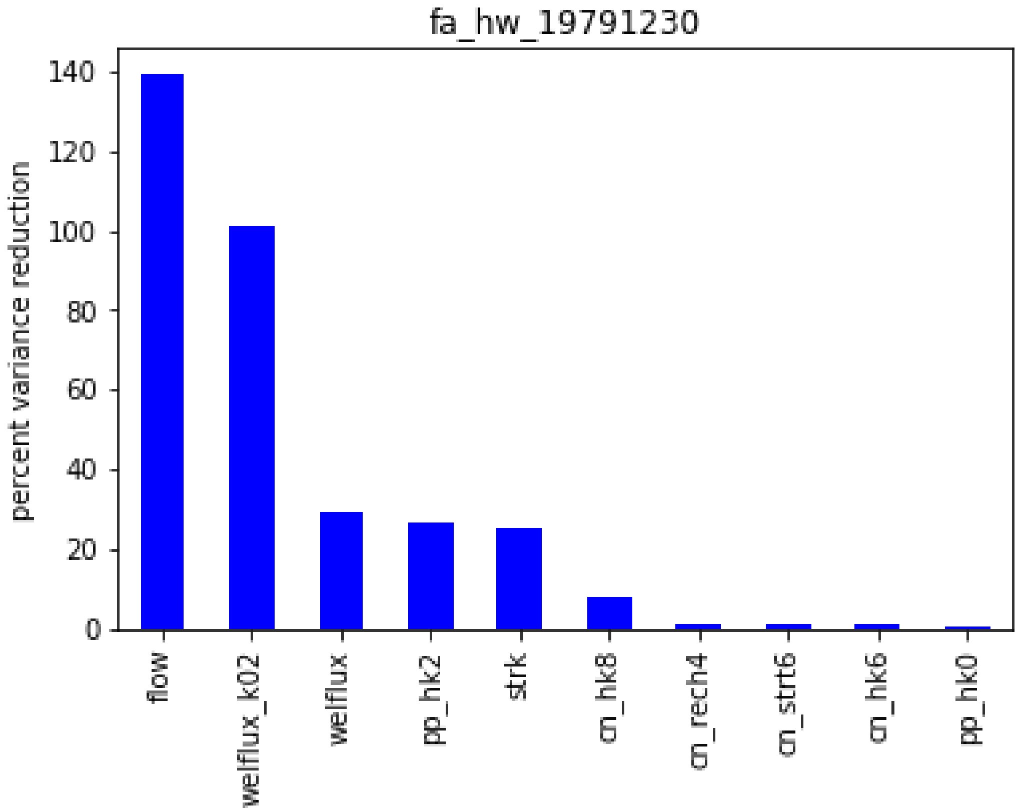
In [15]:
```python
base = par_contrib.loc["base",:]
par_contrib = 100.0 * (base - par_contrib) / par_contrib
par_contrib.sort_index()
```

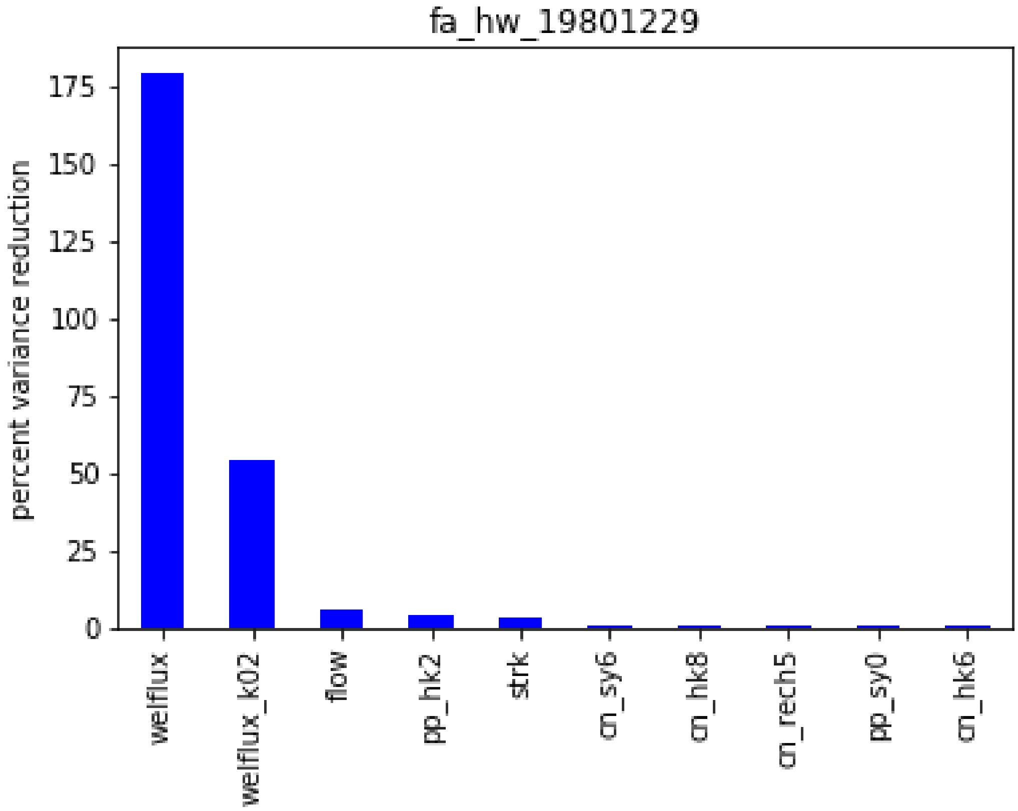
Out[15]:
```
             fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  fa_tw_19801229  \
base           0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00
cn_hk6         7.200326e-01    5.732464e-02    4.912402e-01    2.538854e-02
cn_hk7         1.100704e-05    1.542071e-06    1.228760e-05    2.765020e-05
cn_hk8         7.868245e+00    5.620486e-01    5.689085e+00    9.901549e-02
cn_prsity6     1.581757e-14   -2.130907e-14    1.780053e-14   -3.996120e-14
cn_prsity7     1.581757e-14   -2.130907e-14    1.780053e-14   -3.996120e-14
cn_prsity8     1.581757e-14   -2.130907e-14    1.780053e-14   -3.996120e-14
cn_rech4       1.348055e+00    3.609799e-03    5.298375e-01    5.789648e-02
cn_rech5      -3.163514e-14    1.545966e-01   -7.120212e-14    1.499301e-01
cn_ss6         3.236505e-06    2.198815e-05    3.774045e-05    4.564123e-05
cn_ss7         1.581757e-14    1.361376e-05   -5.340159e-14    4.690049e-05
cn_ss8         6.526454e-06    1.910420e-03    1.264430e-05    3.151138e-03
cn_strt6       7.450759e-01    8.104834e-04    2.548655e-01    2.772938e-02
cn_strt7       5.477110e-06    7.637383e-10    2.596380e-06    1.425282e-07
cn_strt8       2.446861e-04    5.344060e-06    1.721387e-04    2.029535e-05
cn_sy6         6.323762e-03    6.543298e-01    6.387675e-04    7.553426e-01
cn_sy7         0.000000e+00   -4.261813e-14   -1.780053e-14   -1.332040e-14
cn_sy8         0.000000e+00   -4.261813e-14   -1.780053e-14   -1.332040e-14
cn_vka6        1.147905e-03    1.444725e-04    1.136181e-03    1.036603e-05
cn_vka7        6.078694e-02    5.246166e-03    1.724905e-02    5.221210e-04
cn_vka8        8.327031e-02    7.668260e-03    2.127669e-02    8.034385e-04
drncond_k00    8.261107e-04    5.019712e-06    3.444939e-03    1.284969e-05
flow           1.390969e+02    5.570652e+00    9.875874e+01    4.322948e-01
pp_hk0         4.640144e-01    5.717760e-02    4.388836e-01    1.990442e-02
pp_hk1         9.714172e-04    2.977460e-05    5.180516e-04    1.756521e-05
pp_hk2         2.632325e+01    3.859421e+00    7.703933e+01    3.477940e+00
pp_prsity0    -3.163514e-14   -2.130907e-14   -1.780053e-14   -1.332040e-14
pp_prsity1    -3.163514e-14   -2.130907e-14   -1.780053e-14   -1.332040e-14
pp_prsity2    -3.163514e-14   -2.130907e-14   -1.780053e-14   -1.332040e-14
pp_rech0       3.886904e-01    1.537794e-02    1.629436e-01    1.611215e-02
pp_ss0         0.000000e+00    5.959034e-08   -1.780053e-14    3.439249e-06
pp_ss1         0.000000e+00    2.178200e-08   -1.780053e-14    2.432966e-05
pp_ss2         6.809523e-07    4.443653e-04    6.993913e-07    9.689916e-04
pp_sy0         1.652762e-03    1.042446e-01    5.194962e-04    1.428086e-01
pp_sy1         3.163514e-14    0.000000e+00    3.560106e-14   -1.332040e-14
pp_sy2         3.163514e-14    0.000000e+00    3.560106e-14   -1.332040e-14
pp_vka1        2.022176e-02    2.237218e-03    1.643048e-02    5.337044e-04
strk           2.552502e+01    2.678685e+00    4.276074e+01    1.259642e+00
welflux        2.931378e+01    1.786585e+02    1.737590e+01    3.969062e+02
welflux_k02    1.009574e+02    5.415224e+01    1.819651e+01    2.986519e+01
```

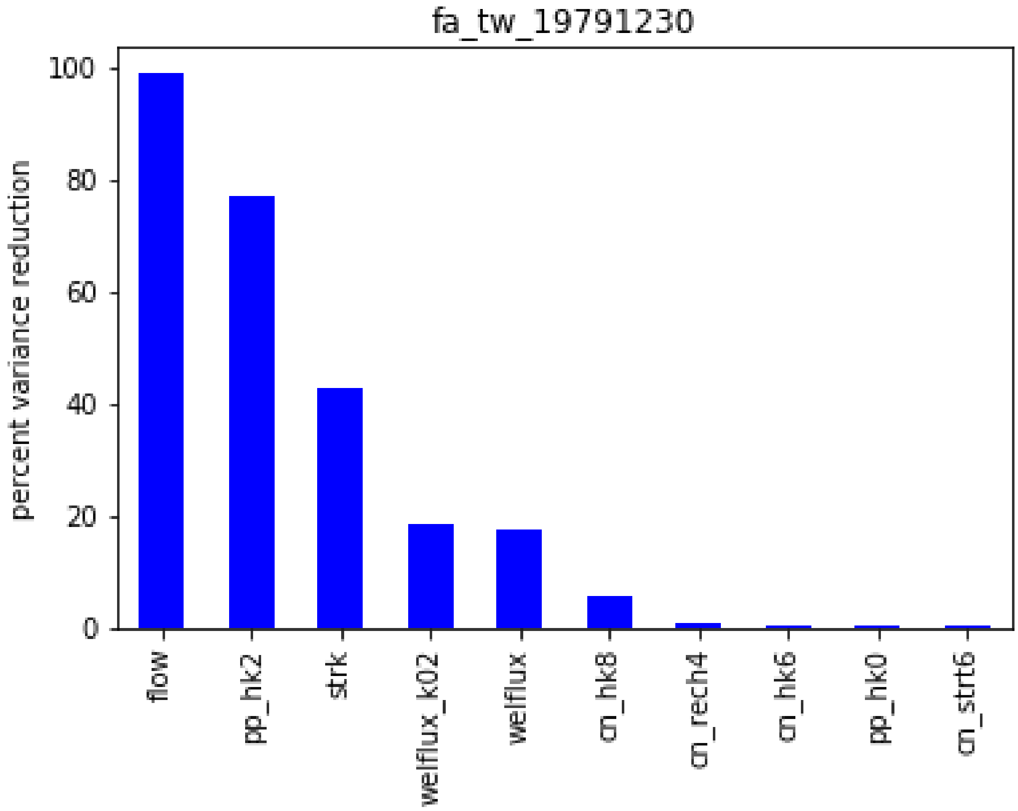
	hds_00_013_002_000	hds_00_013_002_001	part_status	part_time
base	0.000000e+00	0.000000e+00	NaN	0.000000e+00
cn_hk6	3.907923e-01	2.711848e-02	NaN	4.330495e-02
cn_hk7	1.409848e-05	7.712379e-06	NaN	4.995530e-06
cn_hk8	2.041916e+00	2.290750e+00	NaN	2.497670e-02
cn_prsity6	-1.701364e-13	0.000000e+00	NaN	6.199667e-02
cn_prsity7	-1.701364e-13	0.000000e+00	NaN	1.031702e-02
cn_prsity8	-1.701364e-13	0.000000e+00	NaN	2.706847e+00
cn_rech4	1.766098e-03	2.046518e+00	NaN	7.965004e-03
cn_rech5	-6.186779e-14	1.402045e+00	NaN	0.000000e+00
cn_ss6	3.380205e-06	1.936788e-03	NaN	9.680755e-10
cn_ss7	1.392025e-13	4.613505e-04	NaN	2.146264e-09
cn_ss8	5.598416e-10	2.236178e-02	NaN	1.095581e-07
cn_strt6	1.777425e-03	1.201548e+00	NaN	4.687886e-03
cn_strt7	1.246586e-08	8.328904e-06	NaN	5.719802e-06
cn_strt8	4.218370e-06	3.595262e-04	NaN	5.124408e-05
cn_sy6	4.721875e-04	1.312509e+01	NaN	1.128800e-04
cn_sy7	-1.082686e-13	-6.572986e-14	NaN	0.000000e+00
cn_sy8	-1.082686e-13	-6.572986e-14	NaN	0.000000e+00
cn_vka6	8.942020e-05	1.268586e-11	NaN	5.819374e-08
cn_vka7	6.980633e-04	5.332341e-04	NaN	4.446734e-06
cn_vka8	3.221093e-04	5.353557e-04	NaN	3.498805e-06
drncond_k00	3.868335e-04	8.478465e-04	NaN	1.574046e-06
flow	6.069304e-01	1.737717e+00	NaN	3.928388e-03
pp_hk0	5.170918e+00	2.949620e+00	NaN	3.270703e-02
pp_hk1	9.605049e-04	2.500788e-03	NaN	4.337329e-06
pp_hk2	1.439076e+02	6.819699e+01	NaN	6.091880e-01
pp_prsity0	3.093389e-13	4.929740e-13	NaN	5.448398e+00
pp_prsity1	3.093389e-13	4.929740e-13	NaN	6.926106e-01
pp_prsity2	3.093389e-13	4.929740e-13	NaN	9.807321e+02
pp_rech0	9.796288e-02	4.051834e-01	NaN	1.797803e-03
pp_ss0	2.629381e-13	2.430896e-04	NaN	6.194792e-09
pp_ss1	2.629381e-13	4.929740e-13	NaN	9.931534e-10
pp_ss2	1.312719e-06	2.724502e-03	NaN	3.697979e-08
pp_sy0	2.209137e-03	2.636021e+00	NaN	3.377781e-05
pp_sy1	1.546695e-14	-1.643247e-14	NaN	0.000000e+00
pp_sy2	1.546695e-14	-1.643247e-14	NaN	0.000000e+00
pp_vka1	1.019290e-03	1.262804e-03	NaN	3.056187e-06
strk	3.861402e-01	2.085921e-01	NaN	2.937591e-03
welflux	4.119541e-01	2.116091e+01	NaN	1.317691e-03
welflux_k02	1.829228e-01	2.805127e+00	NaN	9.376455e-03

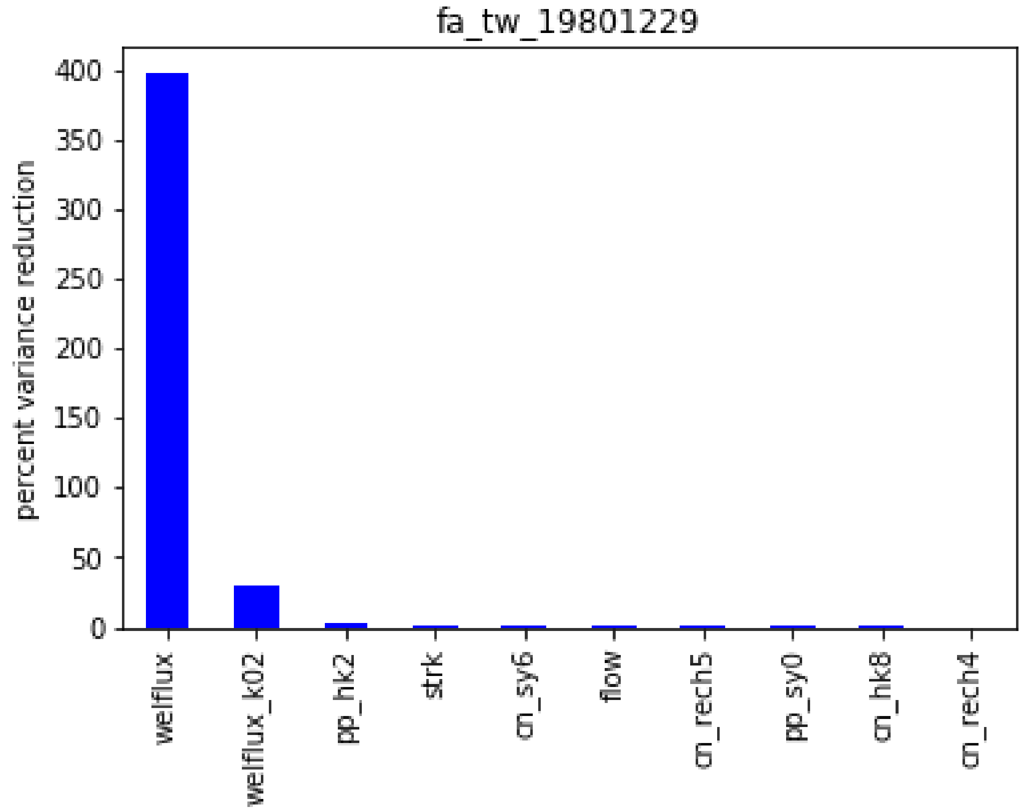
In [16]:
```
for forecast in par_contrib.columns:
    fore_df = par_contrib.loc[:,forecast].copy()
    fore_df.sort_values(inplace=True, ascending=False)
    ax = fore_df.iloc[:10].plot(kind="bar",color="b")
    ax.set_title(forecast)
```

```
ax.set_ylabel("percent variance reduction")
plt.show()
```

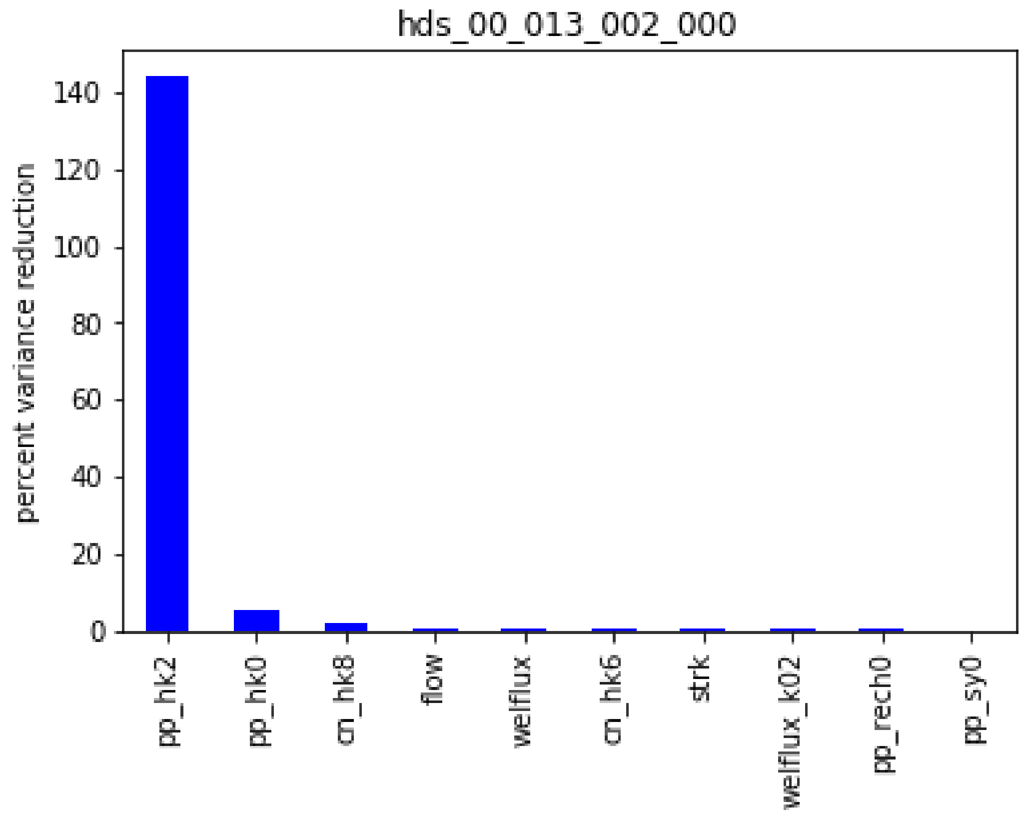


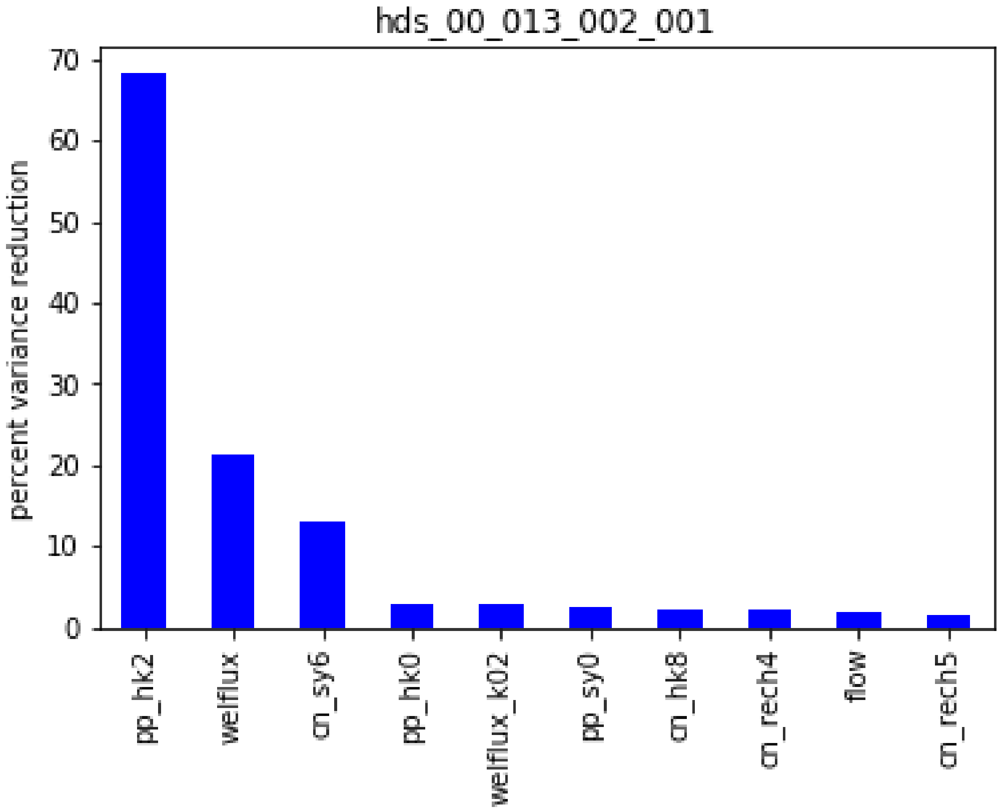

fa_hw_19801229
percent variance reduction
0
25
50
75
100
125
150
175
welflux
welflux_k02_
flow
pp_hk2
strk
cn_sy6
cn_hk8
cn_rech5
pp_sy0
cn_hk6

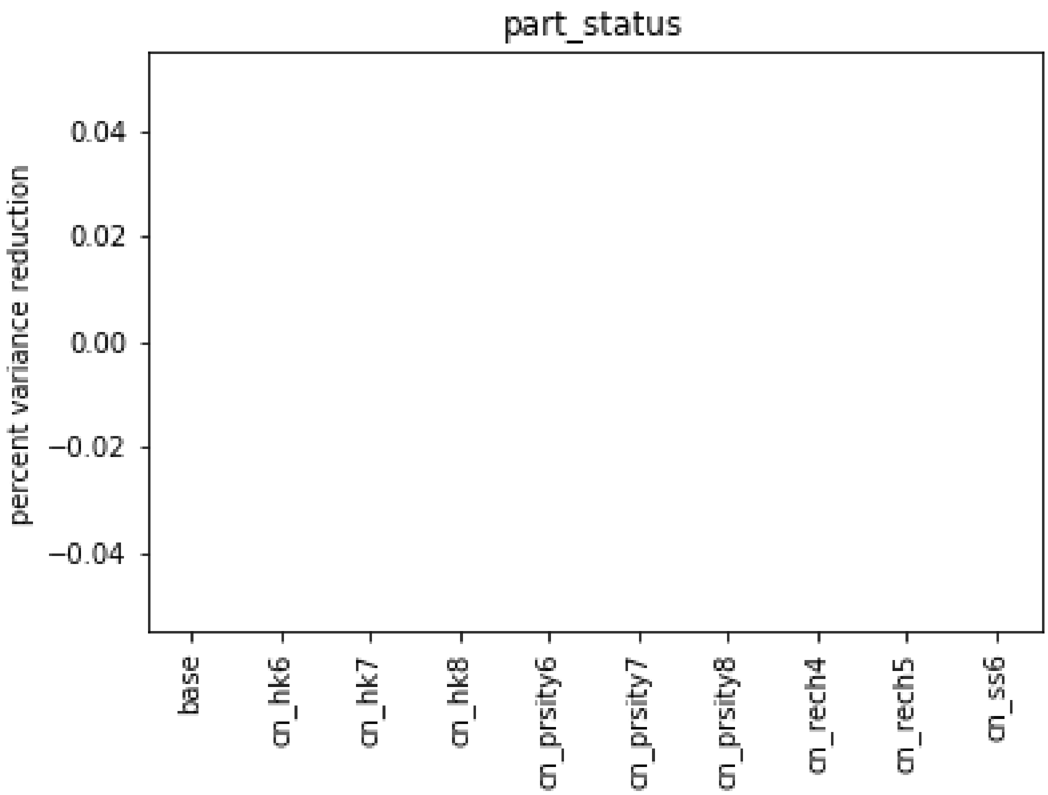

fa_tw_19791230
percent variance reduction
100
80
60
40
20
0
flow
pp_hk2
strk
welflux_k02
welflux
cn_hk8
cn_rech4
cn_hk6
pp_hk0
cn_strt6

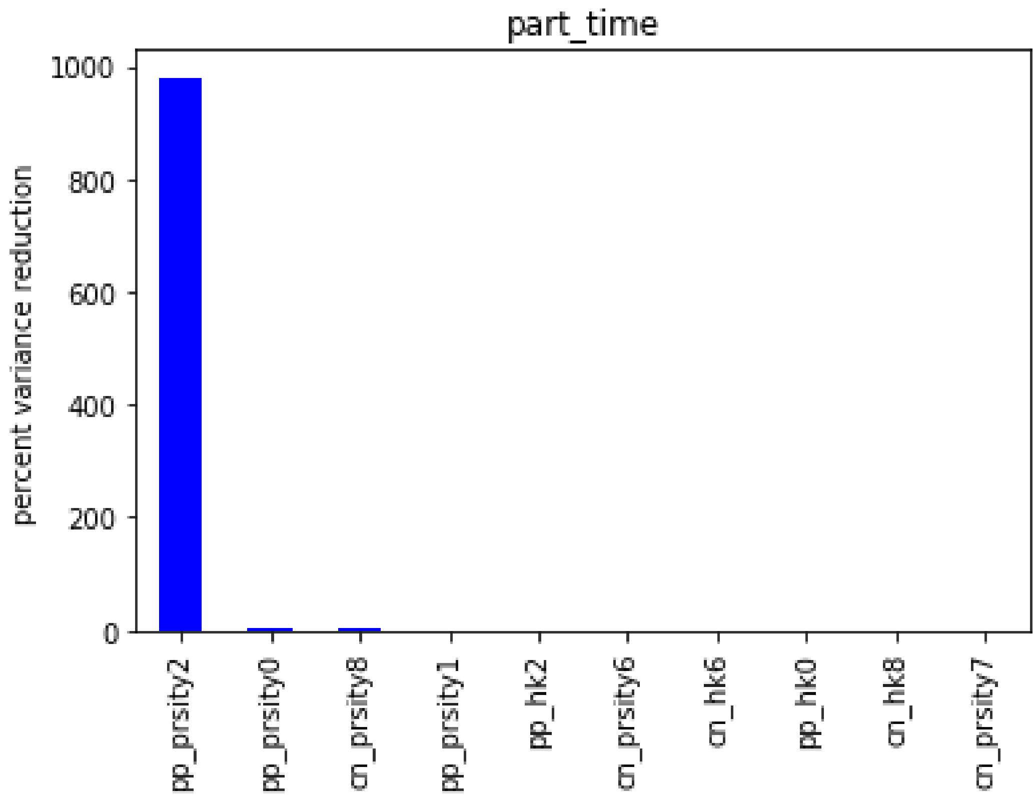


fa_tw_19801229
percent variance reduction
400
350
300
250
200
150
100
50
0
welflux
welflux_k02
pp_hk2
strk
cn_sy6
flow
cn_rech5
pp_sy0
cn_hk8
cn_rech4


hds_00_013_002_000
percent variance reduction
0
20
40
60
80
100
120
140
pp_hk2
pp_hk0
cn_hk8
flow
welflux
cn_hk6
strk
welflux_k02
pp_rech0
pp_sy0


hds_00_013_002_001
percent variance reduction
70
60
50
40
30
20
10
0
pp_hk2
welflux
cn_sy6
pp_hk0
welflux_k02
pp_sy0
cn_hk8
cn_rech4
flow
cn_rech5


part_status
percent variance reduction
0.04
0.02
0.00
−0.02
−0.04
base
cn_hk6
cn_hk7
cn_hk8
cn_prsity6
cn_prsity7
cn_prsity8
cn_rech4
cn_rech5
cn_ss6



1.1.2 Data worth

1.1.3 what is the worth of *existing* observations?

What is happening under the hood is that we are recalculating the Schur complement without some of the observations to see how the posterior forecast uncertainty increases (wrt a "base" condition in which we have all observation data available).

In [17]:
```
dw_rm = sc.get_removed_obs_importance()
dw_rm.head()
```

Out[17]:
```
                       fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
base                     45999.208695   273159.126193    20437.475638
fo_39_19791230           47845.132435   273818.307804    20783.056217
hds_00_002_009_000       46218.456517   273314.654780    20443.845591
hds_00_002_015_000       46021.367916   273241.159290    20442.761529
hds_00_003_008_000       46132.481010   273241.996021    20437.500048

                       fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
base                    218490.640452            0.089725            0.168907
fo_39_19791230          218519.965294            0.089792            0.169205
hds_00_002_009_000      218490.844125            0.089813            0.168970
```

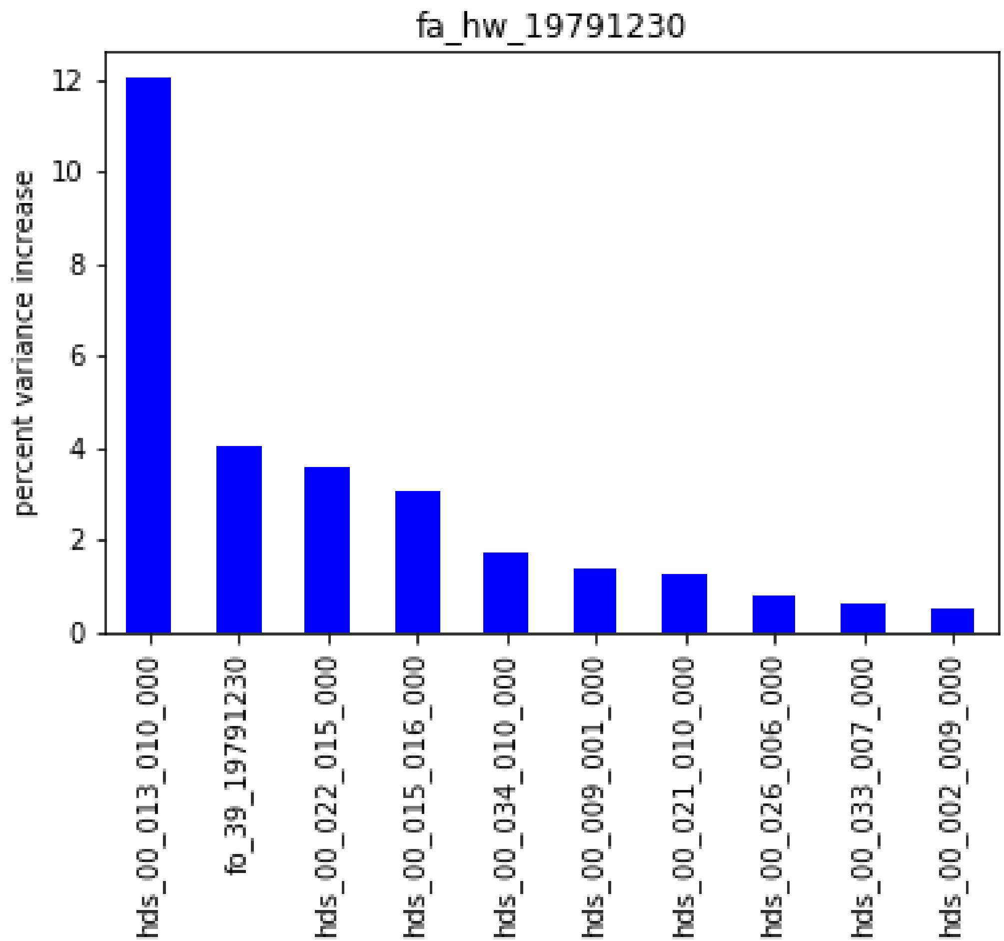
```
hds_00_002_015_000   218491.210009             0.089742             0.168909
hds_00_003_008_000   218490.692562             0.090262             0.169395

                    part_status     part_time
base                        0.0  1.258207e+07
fo_39_19791230              0.0  1.258212e+07
hds_00_002_009_000          0.0  1.258212e+07
hds_00_002_015_000          0.0  1.258210e+07
hds_00_003_008_000          0.0  1.258341e+07
```

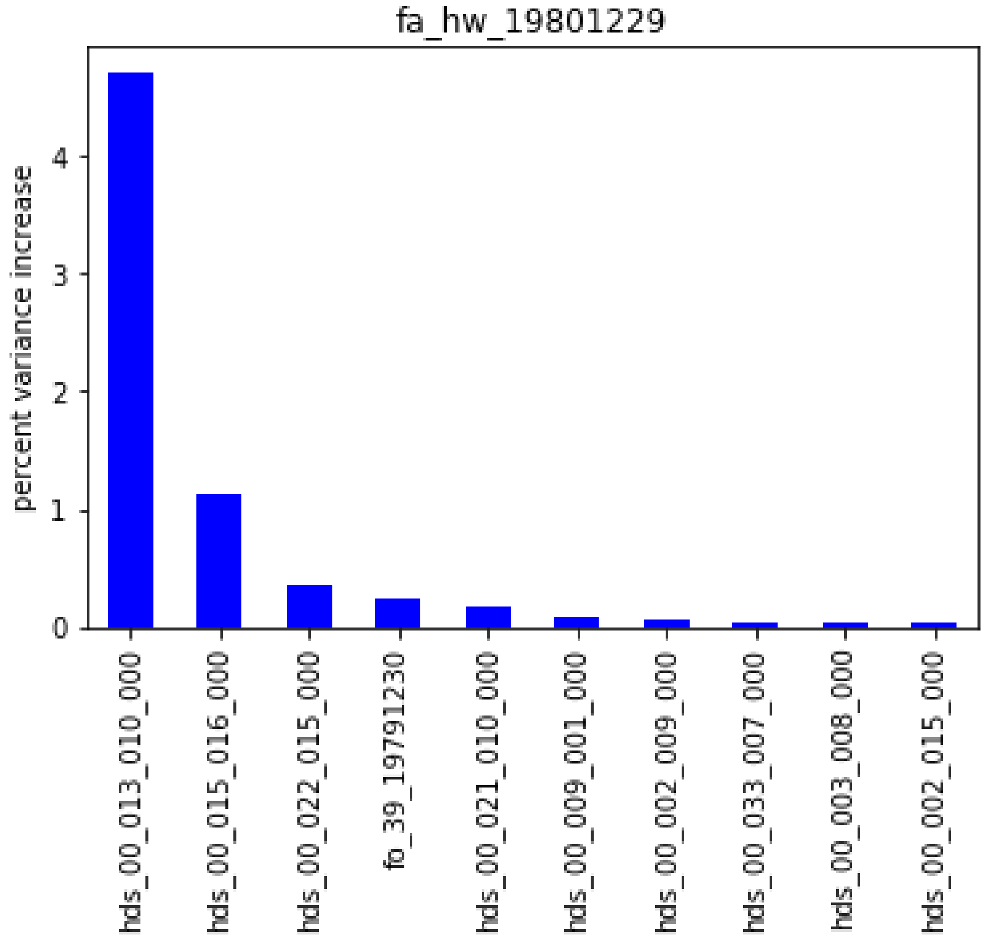
Here the base row contains the results of the Schur complement calculation (in terms of forecast uncertainty variance) using all observations.

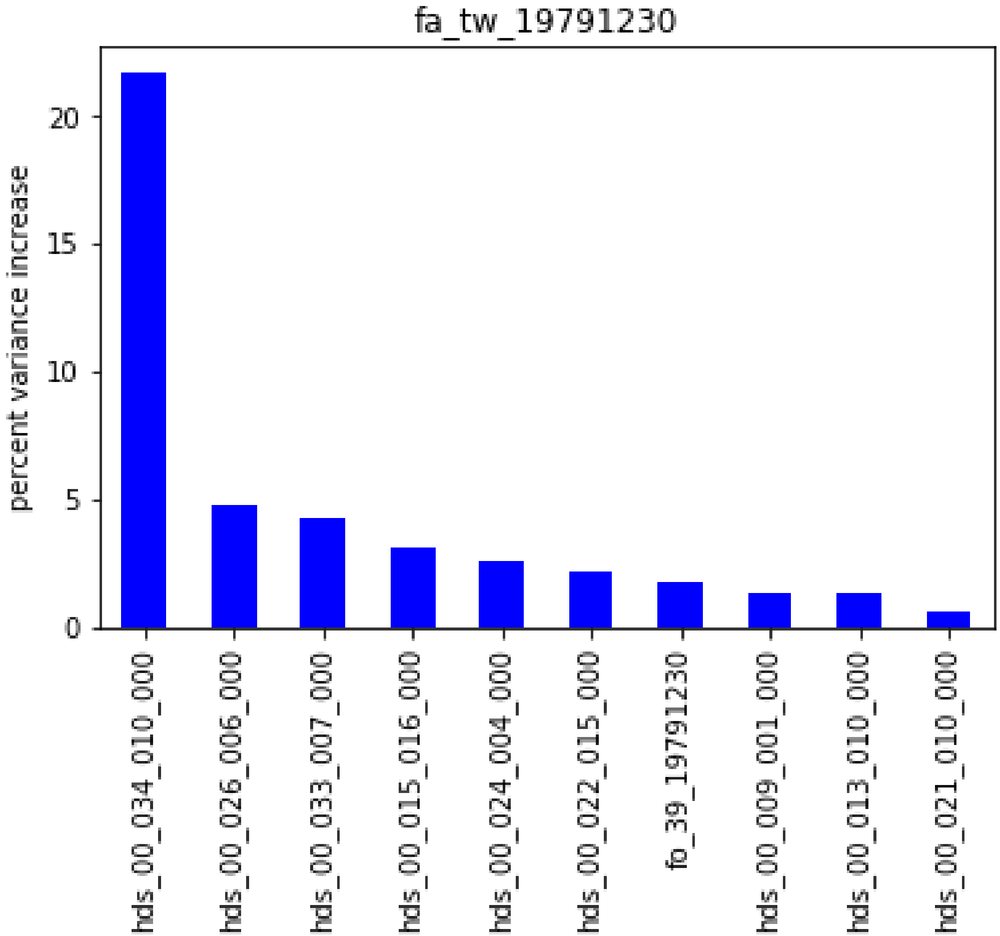
In [18]:
```
# let's normalize to make more meaningful comparisons of data worth (unctainty varian
base = dw_rm.loc["base",:]
dw_rm = 100 * (dw_rm  - base) / base
dw_rm.head()
```

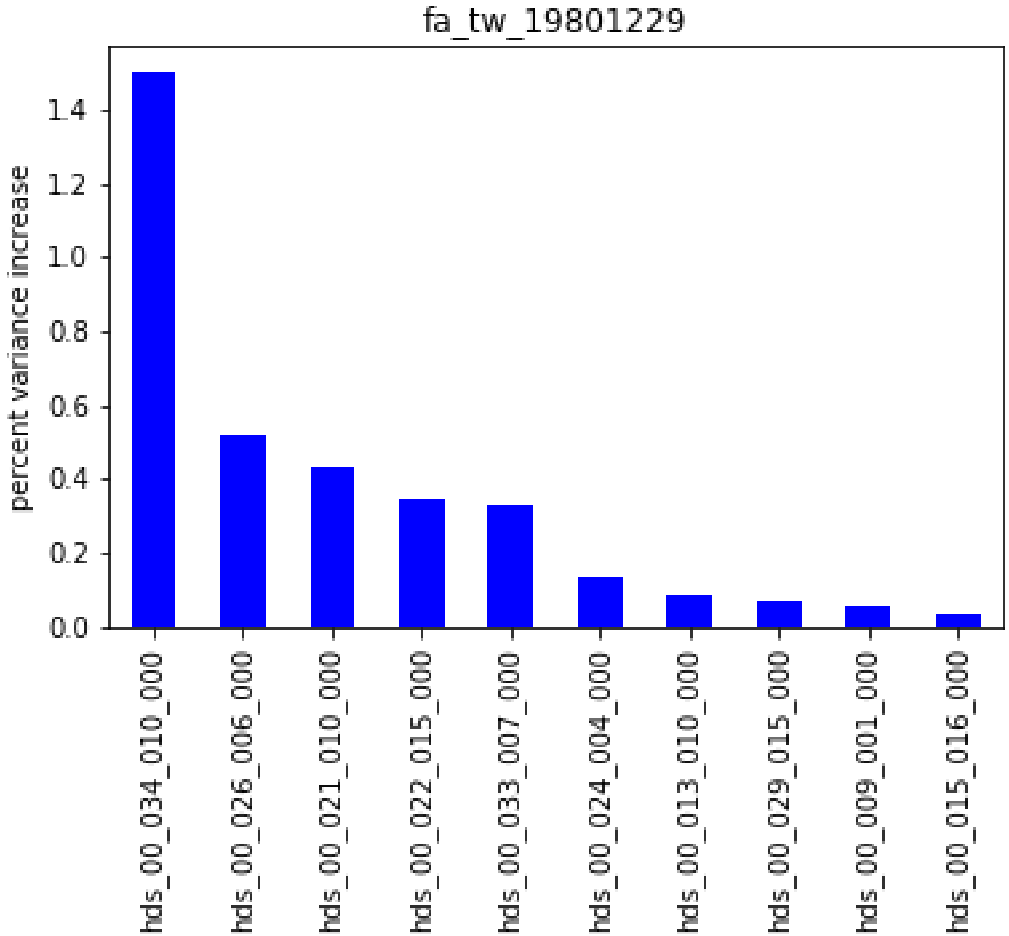
Out[18]:
```
                    fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
base                      0.000000        0.000000        0.000000
fo_39_19791230            4.012947        0.241318        1.690916
hds_00_002_009_000        0.476634        0.056937        0.031168
hds_00_002_015_000        0.048173        0.030031        0.025864
hds_00_003_008_000        0.289727        0.030338        0.000119

                    fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
base                      0.000000            0.000000            0.000000
fo_39_19791230            0.013422            0.074072            0.176625
hds_00_002_009_000        0.000093            0.097228            0.037509
hds_00_002_015_000        0.000261            0.018382            0.001039
hds_00_003_008_000        0.000024            0.598203            0.289228

                    part_status  part_time
base                        NaN   0.000000
fo_39_19791230              NaN   0.000377
hds_00_002_009_000          NaN   0.000375
hds_00_002_015_000          NaN   0.000207
hds_00_003_008_000          NaN   0.010619
```

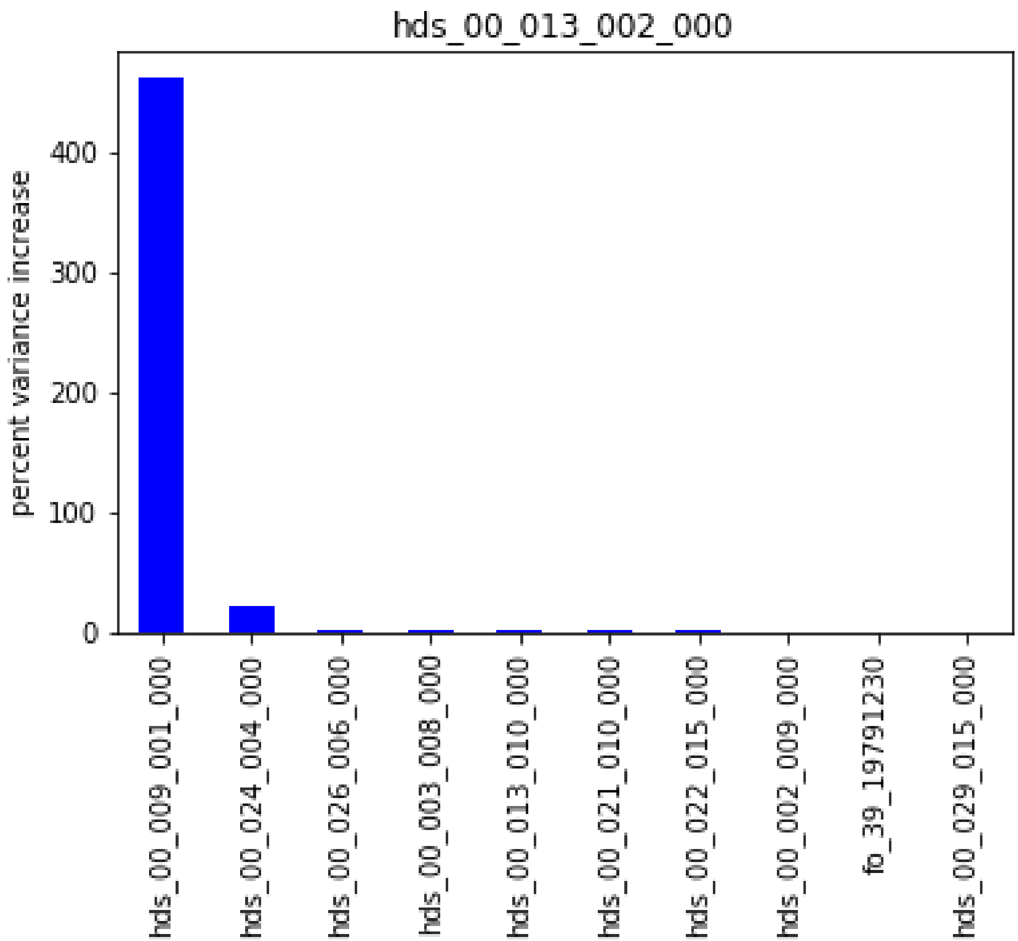
In [19]:
```
for forecast in dw_rm.columns:
    fore_df = dw_rm.loc[:,forecast].copy()
    fore_df.sort_values(inplace=True, ascending=False)
    ax = fore_df.iloc[:10].plot(kind="bar",color="b")
    ax.set_title(forecast)
    ax.set_ylabel("percent variance increase")
    plt.show()
```

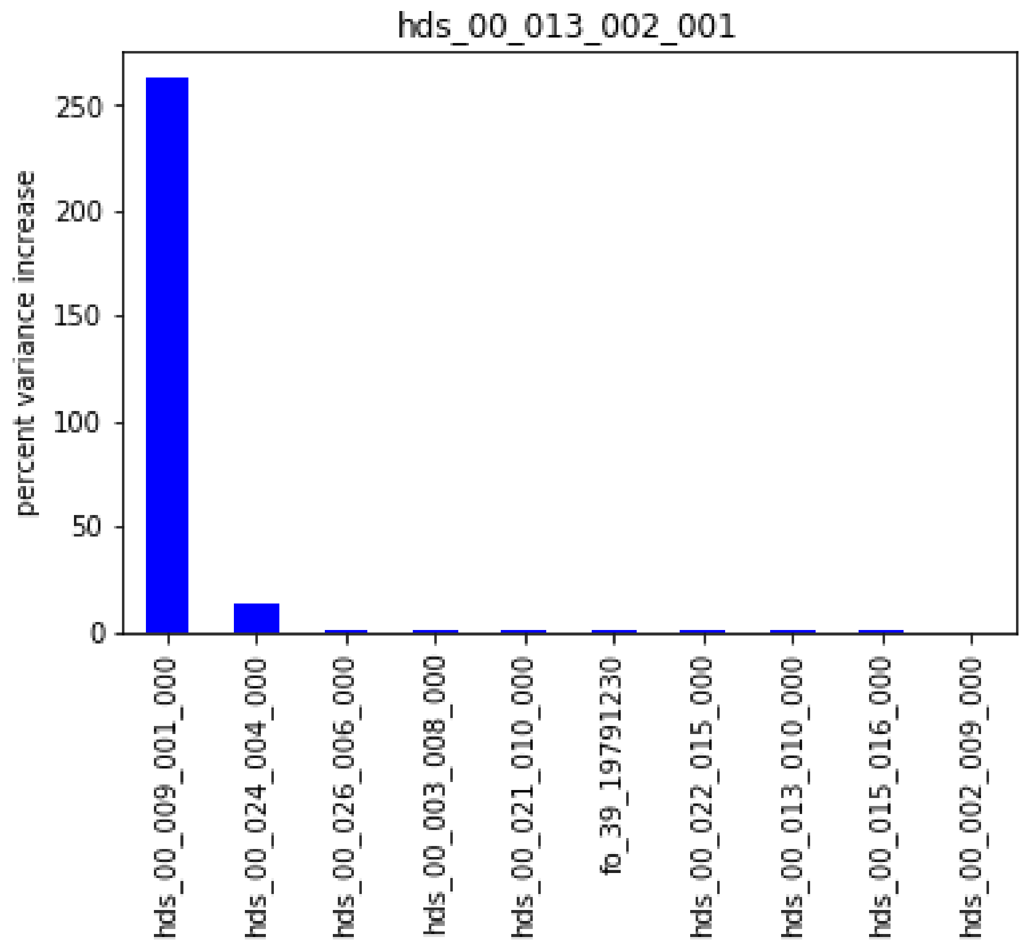

fa_hw_19791230
percent variance increase
12
10
8
6
4
2
0
hds_00_013_010_000
fo_39_19791230
hds_00_022_015_000
hds_00_015_016_000
hds_00_034_010_000
hds_00_009_001_000
hds_00_021_010_000
hds_00_026_006_000
hds_00_033_007_000
hds_00_002_009_000

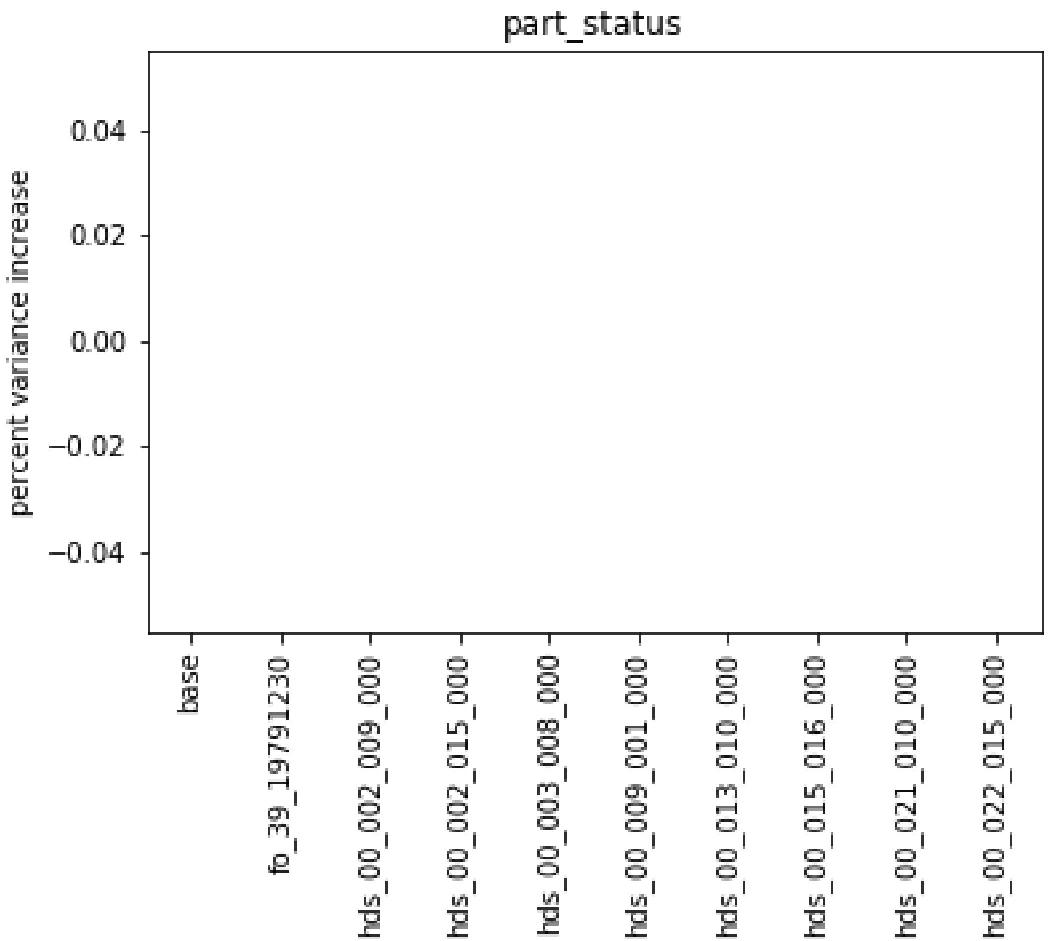

fa_hw_19801229
percent variance increase
0
1
2
3
4
hds_00_013_010_000
hds_00_015_016_000
hds_00_022_015_000
fo_39_19791230
hds_00_021_010_000
hds_00_009_001_000
hds_00_002_009_000
hds_00_033_007_000
hds_00_003_008_000
hds_00_002_015_000

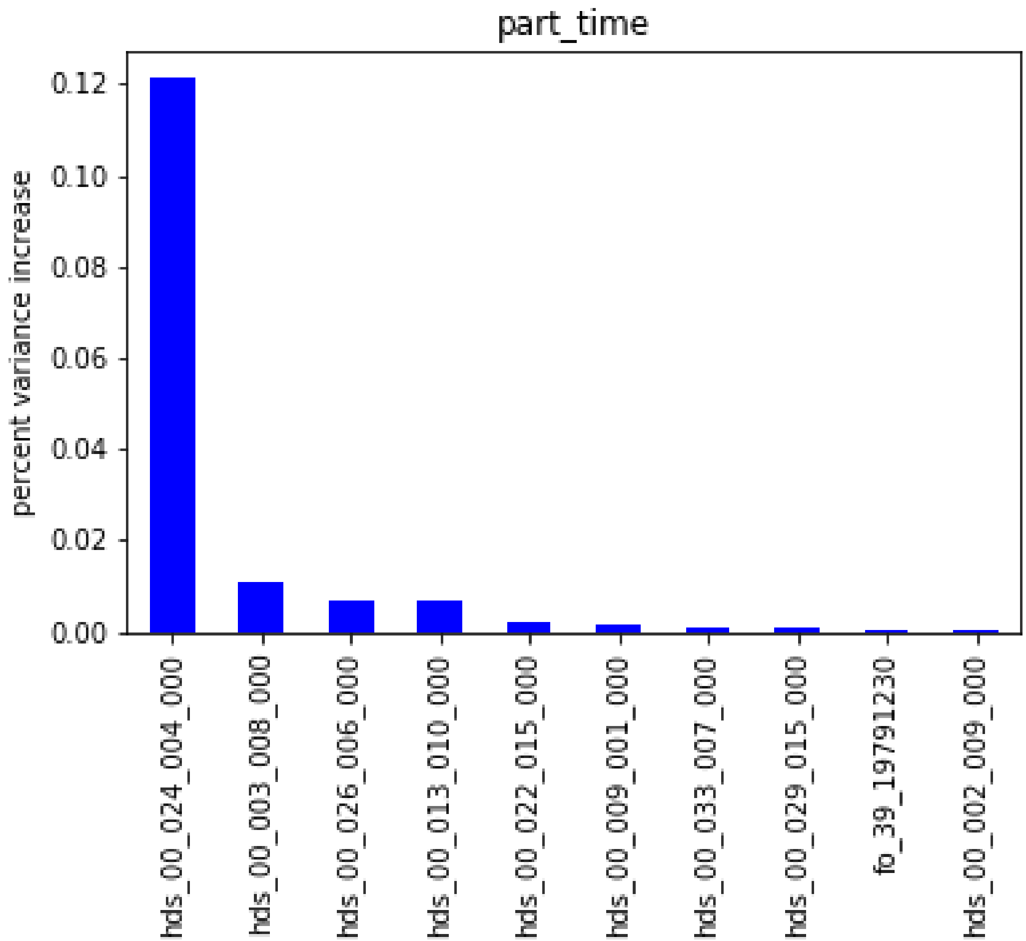

fa_tw_19791230
percent variance increase
0
5
10
15
20
hds_00_034_010_000
hds_00_026_006_000
hds_00_033_007_000
hds_00_015_016_000
hds_00_024_004_000
hds_00_022_015_000
fo_39_19791230
hds_00_009_001_000
hds_00_013_010_000
hds_00_021_010_000


fa_tw_19801229
percent variance increase
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
hds_00_034_010_000
hds_00_026_006_000
hds_00_021_010_000
hds_00_022_015_000
hds_00_033_007_000
hds_00_024_004_000
hds_00_013_010_000
hds_00_029_015_000
hds_00_009_001_000
hds_00_015_016_000


hds_00_013_002_000
percent variance increase
0
100
200
300
400
hds_00_009_001_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_003_008_000
hds_00_013_010_000
hds_00_021_010_000
hds_00_022_015_000
hds_00_002_009_000
fo_39_19791230
hds_00_029_015_000

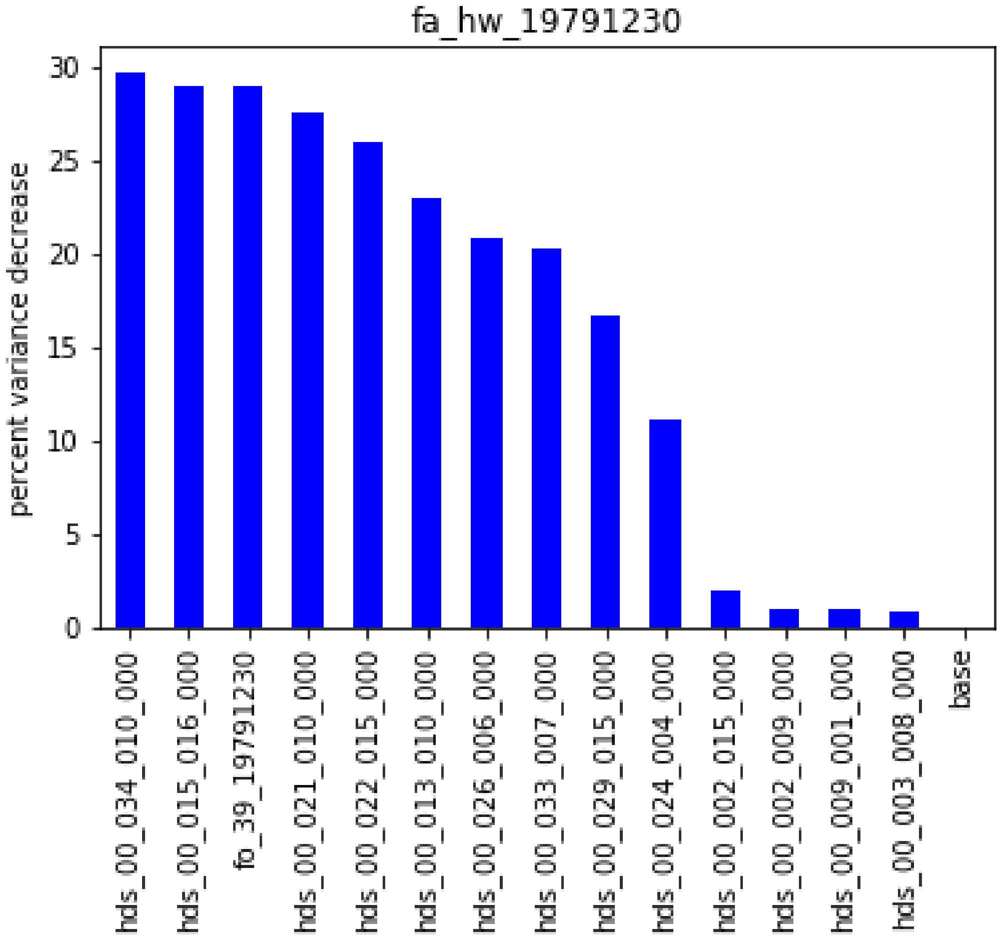

hds_00_013_002_001
percent variance increase
250
200
150
100
50
0
hds_00_009_001_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_003_008_000
hds_00_021_010_000
fo_39_19791230
hds_00_022_015_000
hds_00_013_010_000
hds_00_015_016_000
hds_00_002_009_000

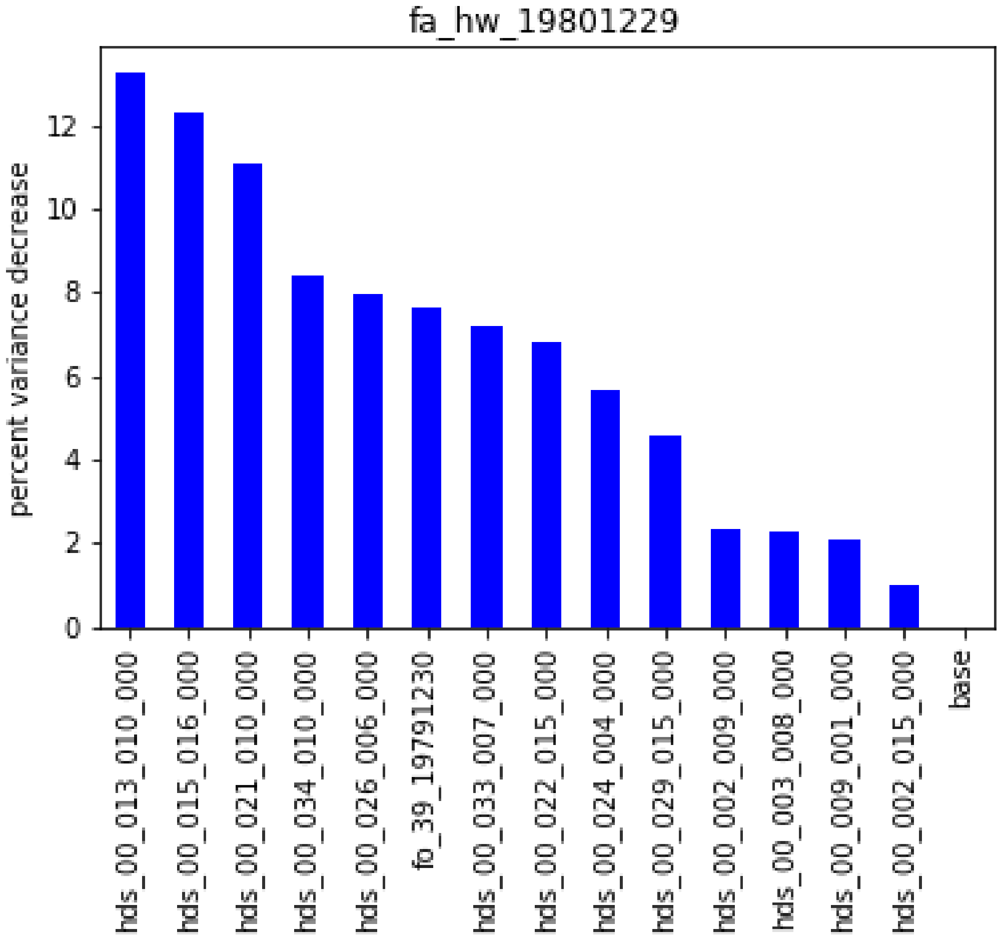

part_status
percent variance increase
0.04
0.02
0.00
−0.02
−0.04
base
fo_39_19791230
hds_00_002_009_000
hds_00_002_015_000
hds_00_003_008_000
hds_00_009_001_000
hds_00_013_010_000
hds_00_015_016_000
hds_00_021_010_000
hds_00_022_015_000

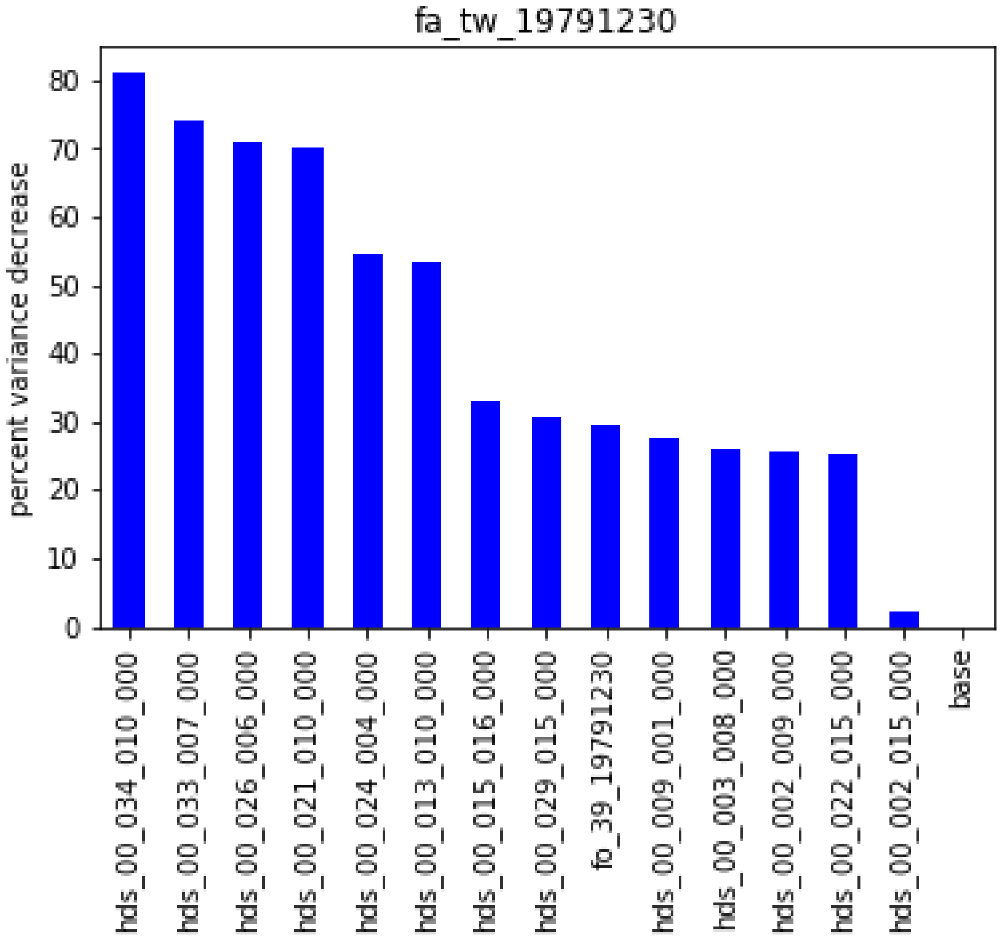


There is also an option to calculate the worth of observations by taking a "base" condition of zero observation (i.e., a priori) and calculating the reduction in uncertainty through adding observations to the dataset.

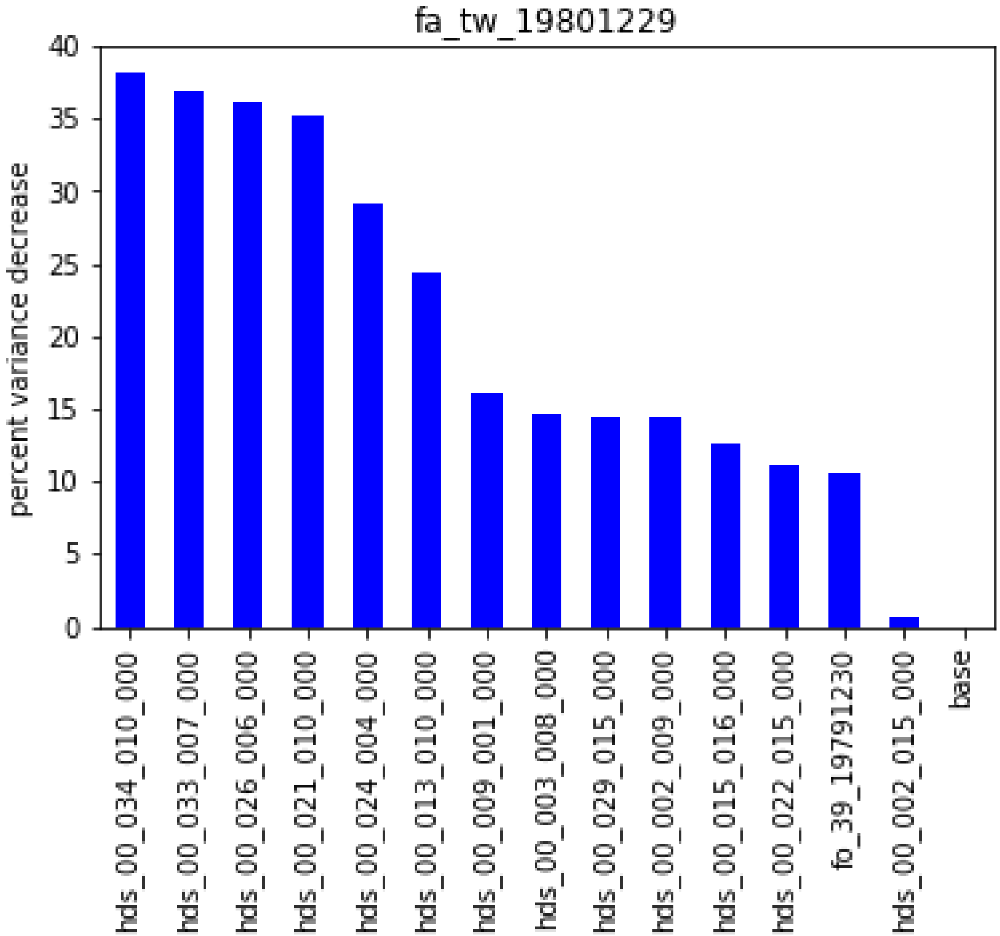
```
In [20]: dw_ad = sc.get_added_obs_importance()
         base = dw_ad.loc["base",:]
         dw_ad = 100 * (base - dw_ad) / base
         for forecast in dw_ad.columns:
             fore_df_ad = dw_ad.loc[:,forecast].copy()
             fore_df_ad.sort_values(inplace=True, ascending=False)
             ax = fore_df_ad.iloc[:20].plot(kind="bar",color="b")
             ax.set_title(forecast)
             ax.set_ylabel("percent variance decrease")
             plt.show()
```

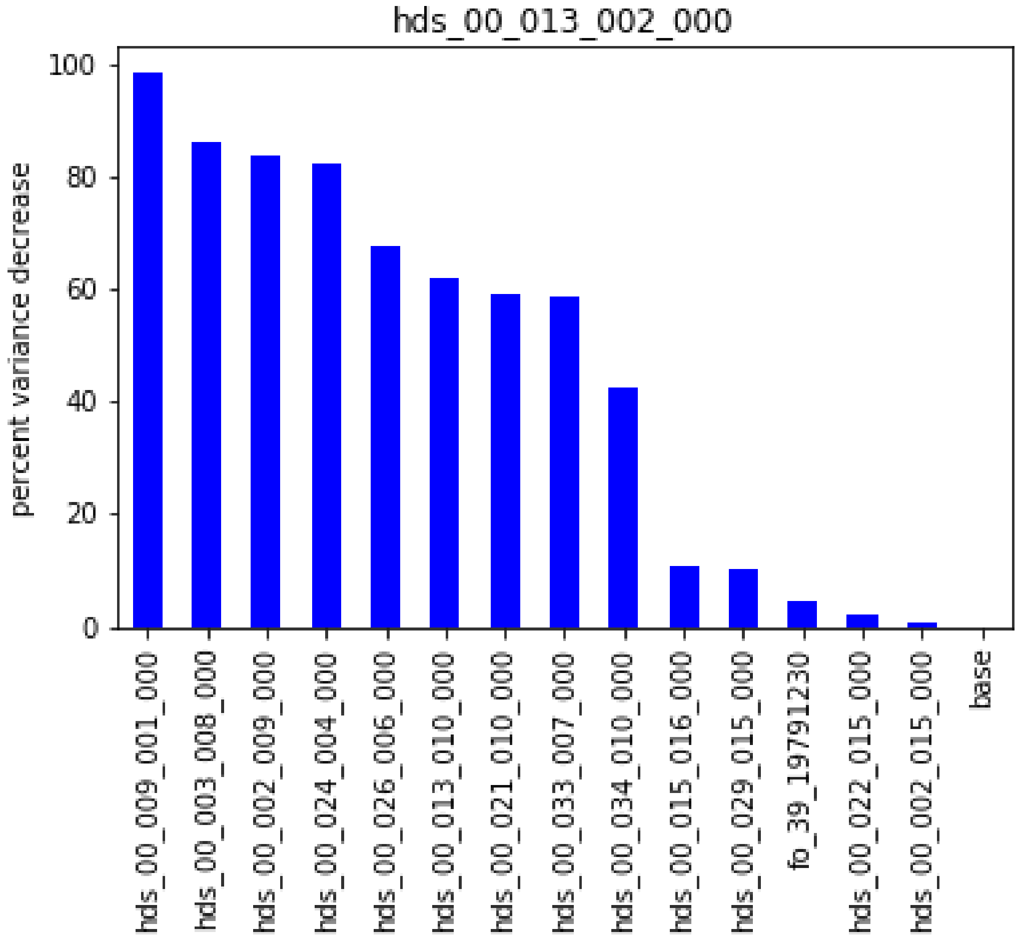

fa_hw_19791230
percent variance decrease
30
25
20
15
10
5
0
hds_00_034_010_000
hds_00_015_016_000
fo_39_19791230
hds_00_021_010_000
hds_00_022_015_000
hds_00_013_010_000
hds_00_026_006_000
hds_00_033_007_000
hds_00_029_015_000
hds_00_024_004_000
hds_00_002_015_000
hds_00_002_009_000
hds_00_009_001_000
hds_00_003_008_000
base

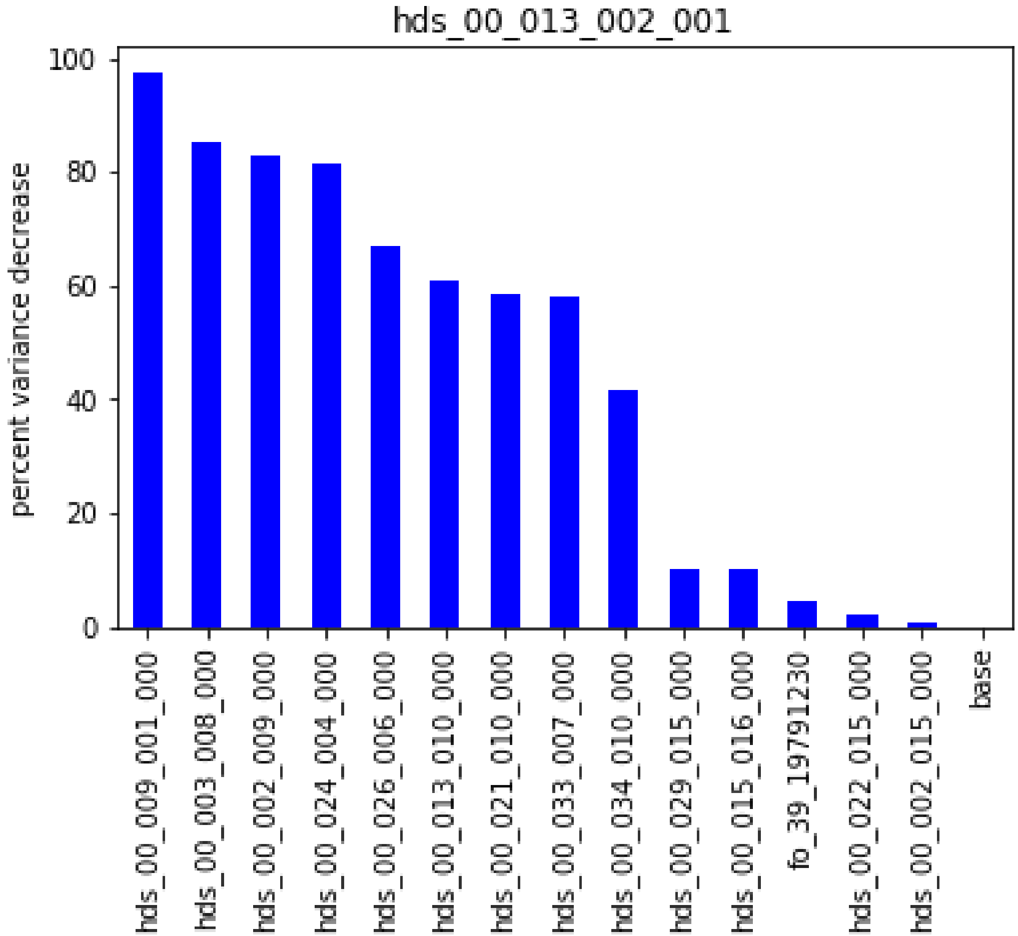

fa_hw_19801229
percent variance decrease
0
2
4
6
8
10
12
hds_00_013_010_000
hds_00_015_016_000
hds_00_021_010_000
hds_00_034_010_000
hds_00_026_006_000
fo_39_19791230
hds_00_033_007_000
hds_00_022_015_000
hds_00_024_004_000
hds_00_029_015_000
hds_00_002_009_000
hds_00_003_008_000
hds_00_009_001_000
hds_00_002_015_000
base

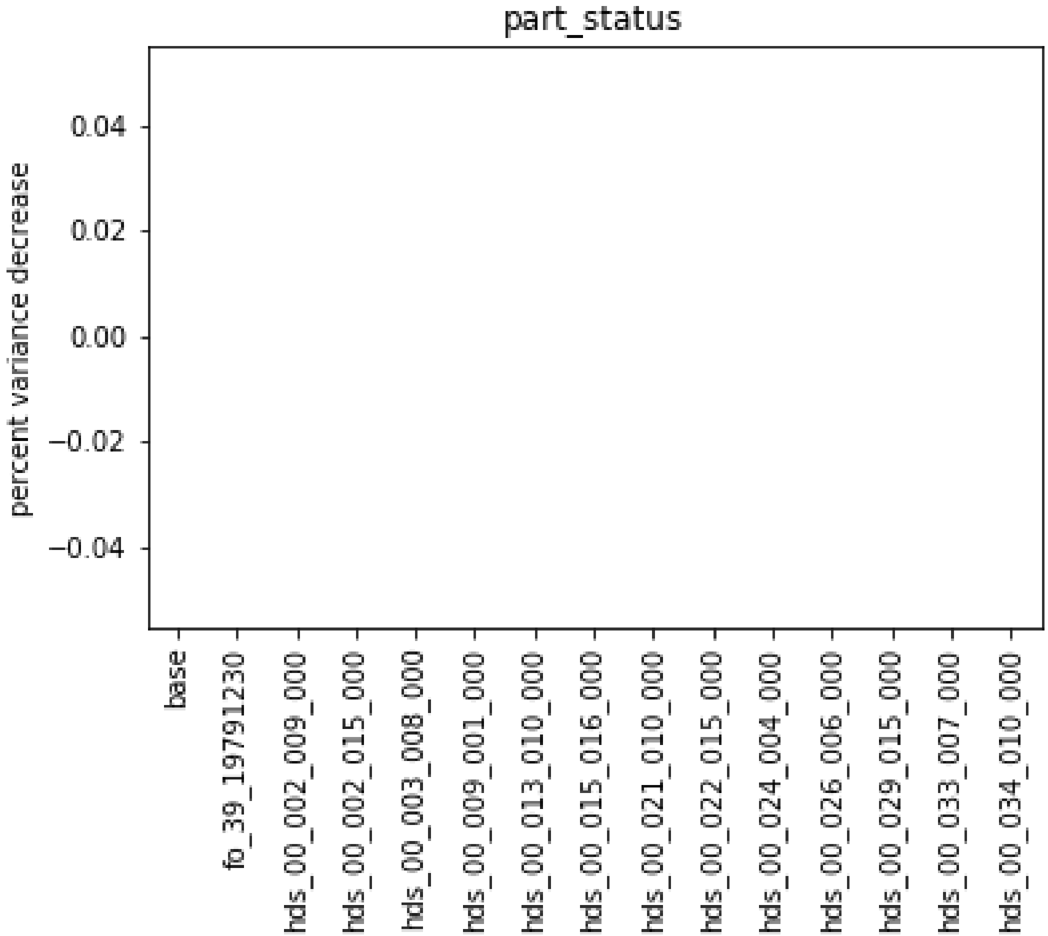


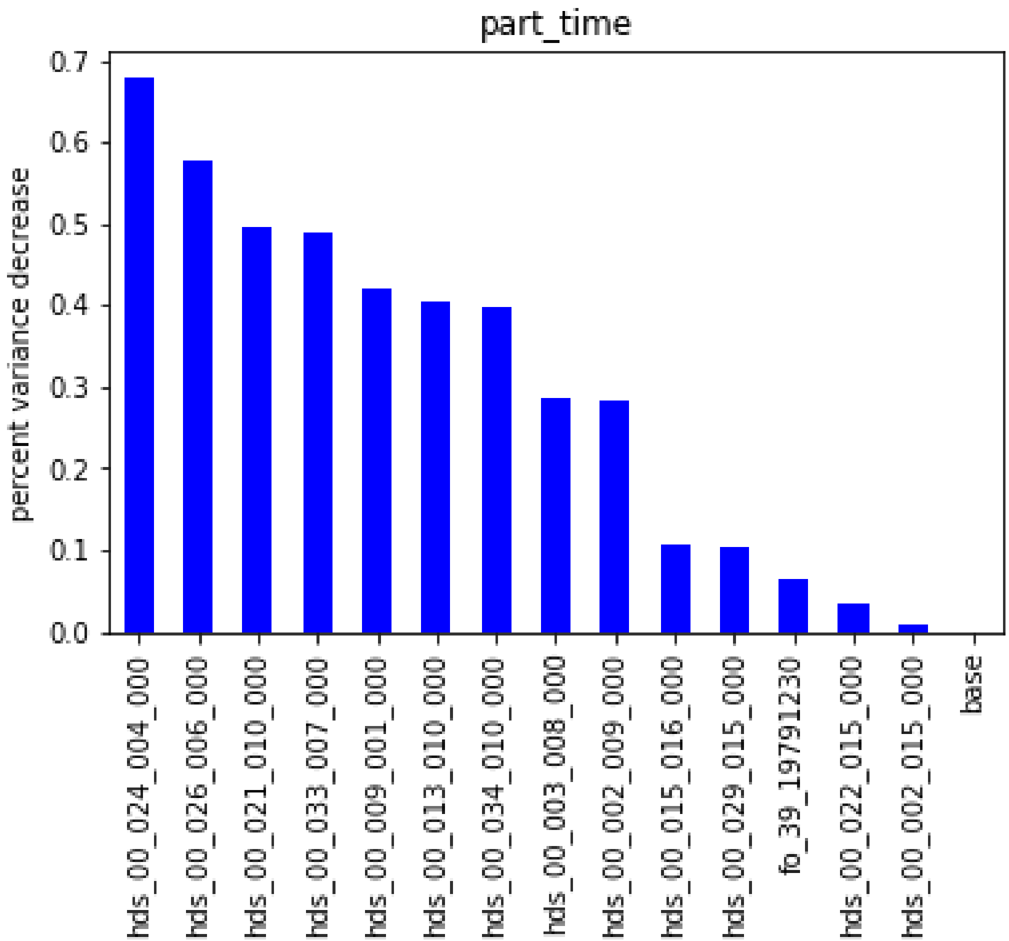
fa_tw_19791230
percent variance decrease
0
10
20
30
40
50
60
70
80
hds_00_034_010_000
hds_00_033_007_000
hds_00_026_006_000
hds_00_021_010_000
hds_00_024_004_000
hds_00_013_010_000
hds_00_015_016_000
hds_00_029_015_000
fo_39_19791230
hds_00_009_001_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_022_015_000
hds_00_002_015_000
base


fa_tw_19801229
percent variance decrease
40
35
30
25
20
15
10
5
0
hds_00_034_010_000
hds_00_033_007_000
hds_00_026_006_000
hds_00_021_010_000
hds_00_024_004_000
hds_00_013_010_000
hds_00_009_001_000
hds_00_003_008_000
hds_00_029_015_000
hds_00_002_009_000
hds_00_015_016_000
hds_00_022_015_000
fo_39_19791230
hds_00_002_015_000
base


hds_00_013_002_000
percent variance decrease
100
80
60
40
20
0
hds_00_009_001_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_013_010_000
hds_00_021_010_000
hds_00_033_007_000
hds_00_034_010_000
hds_00_015_016_000
hds_00_029_015_000
fo_39_19791230
hds_00_022_015_000
hds_00_002_015_000
base


hds_00_013_002_001
percent variance decrease
0
20
40
60
80
100
hds_00_009_001_000
hds_00_003_008_000
hds_00_002_009_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_013_010_000
hds_00_021_010_000
hds_00_033_007_000
hds_00_034_010_000
hds_00_029_015_000
hds_00_015_016_000
fo_39_19791230
hds_00_022_015_000
hds_00_002_015_000
base


part_status
percent variance decrease
0.04
0.02
0.00
−0.02
−0.04
base
fo_39_19791230
hds_00_002_009_000
hds_00_002_015_000
hds_00_003_008_000
hds_00_009_001_000
hds_00_013_010_000
hds_00_015_016_000
hds_00_021_010_000
hds_00_022_015_000
hds_00_024_004_000
hds_00_026_006_000
hds_00_029_015_000
hds_00_033_007_000
hds_00_034_010_000



Do these two approaches give the same answer? They shouldn't.. Why? Let's discuss..

1.1.4 what is the worth of *potential* observations? what data should we collect?

Recall we are "carrying" cell-by-cell heads, reach-based sfr flows, etc..

```
In [21]: z_obs = pst.observation_data.loc[(pst.observation_data.weight == 0),"obsnme"].tolist()
         z_obs = [x for x in z_obs if x not in forecasts]  # less our forecasts
         z_obs
```

```
Out[21]: ['fa_0_19791230',
          'fa_0_19801229',
          'fa_10_19791230',
          'fa_10_19801229',
          'fa_11_19791230',
          'fa_11_19801229',
          'fa_12_19791230',
          'fa_12_19801229',
          'fa_13_19791230',
```

```
'fa_13_19801229',
'fa_14_19791230',
'fa_14_19801229',
'fa_15_19791230',
'fa_15_19801229',
'fa_16_19791230',
'fa_16_19801229',
'fa_17_19791230',
'fa_17_19801229',
'fa_18_19791230',
'fa_18_19801229',
'fa_19_19791230',
'fa_19_19801229',
'fa_1_19791230',
'fa_1_19801229',
'fa_20_19791230',
'fa_20_19801229',
'fa_21_19791230',
'fa_21_19801229',
'fa_22_19791230',
'fa_22_19801229',
'fa_23_19791230',
'fa_23_19801229',
'fa_24_19791230',
'fa_24_19801229',
'fa_25_19791230',
'fa_25_19801229',
'fa_26_19791230',
'fa_26_19801229',
'fa_27_19791230',
'fa_27_19801229',
'fa_28_19791230',
'fa_28_19801229',
'fa_29_19791230',
'fa_29_19801229',
'fa_2_19791230',
'fa_2_19801229',
'fa_30_19791230',
'fa_30_19801229',
'fa_31_19791230',
'fa_31_19801229',
'fa_32_19791230',
'fa_32_19801229',
'fa_33_19791230',
'fa_33_19801229',
'fa_34_19791230',
'fa_34_19801229',
'fa_35_19791230',
```

```
'fa_35_19801229',
'fa_36_19791230',
'fa_36_19801229',
'fa_37_19791230',
'fa_37_19801229',
'fa_38_19791230',
'fa_38_19801229',
'fa_39_19791230',
'fa_39_19801229',
'fa_3_19791230',
'fa_3_19801229',
'fa_4_19791230',
'fa_4_19801229',
'fa_5_19791230',
'fa_5_19801229',
'fa_6_19791230',
'fa_6_19801229',
'fa_7_19791230',
'fa_7_19801229',
'fa_8_19791230',
'fa_8_19801229',
'fa_9_19791230',
'fa_9_19801229',
'flx_constan_19791230',
'flx_constan_19801229',
'flx_drains_19791230',
'flx_drains_19801229',
'flx_in-out_19791230',
'flx_in-out_19801229',
'flx_percent_19791230',
'flx_percent_19801229',
'flx_recharg_19791230',
'flx_recharg_19801229',
'flx_storage_19791230',
'flx_storage_19801229',
'flx_stream__19791230',
'flx_stream__19801229',
'flx_total_19791230',
'flx_total_19801229',
'flx_wells_19791230',
'flx_wells_19801229',
'fo_0_19791230',
'fo_0_19801229',
'fo_10_19791230',
'fo_10_19801229',
'fo_11_19791230',
'fo_11_19801229',
'fo_12_19791230',
```

```
'fo_12_19801229',
'fo_13_19791230',
'fo_13_19801229',
'fo_14_19791230',
'fo_14_19801229',
'fo_15_19791230',
'fo_15_19801229',
'fo_16_19791230',
'fo_16_19801229',
'fo_17_19791230',
'fo_17_19801229',
'fo_18_19791230',
'fo_18_19801229',
'fo_19_19791230',
'fo_19_19801229',
'fo_1_19791230',
'fo_1_19801229',
'fo_20_19791230',
'fo_20_19801229',
'fo_21_19791230',
'fo_21_19801229',
'fo_22_19791230',
'fo_22_19801229',
'fo_23_19791230',
'fo_23_19801229',
'fo_24_19791230',
'fo_24_19801229',
'fo_25_19791230',
'fo_25_19801229',
'fo_26_19791230',
'fo_26_19801229',
'fo_27_19791230',
'fo_27_19801229',
'fo_28_19791230',
'fo_28_19801229',
'fo_29_19791230',
'fo_29_19801229',
'fo_2_19791230',
'fo_2_19801229',
'fo_30_19791230',
'fo_30_19801229',
'fo_31_19791230',
'fo_31_19801229',
'fo_32_19791230',
'fo_32_19801229',
'fo_33_19791230',
'fo_33_19801229',
'fo_34_19791230',
```

```
'fo_34_19801229',
'fo_35_19791230',
'fo_35_19801229',
'fo_36_19791230',
'fo_36_19801229',
'fo_37_19791230',
'fo_37_19801229',
'fo_38_19791230',
'fo_38_19801229',
'fo_39_19801229',
'fo_3_19791230',
'fo_3_19801229',
'fo_4_19791230',
'fo_4_19801229',
'fo_5_19791230',
'fo_5_19801229',
'fo_6_19791230',
'fo_6_19801229',
'fo_7_19791230',
'fo_7_19801229',
'fo_8_19791230',
'fo_8_19801229',
'fo_9_19791230',
'fo_9_19801229',
'fo_hw_19791230',
'fo_hw_19801229',
'fo_tw_19791230',
'fo_tw_19801229',
'hds_00_000_000_000',
'hds_00_000_000_001',
'hds_00_000_001_000',
'hds_00_000_001_001',
'hds_00_000_002_000',
'hds_00_000_002_001',
'hds_00_000_003_000',
'hds_00_000_003_001',
'hds_00_000_004_000',
'hds_00_000_004_001',
'hds_00_000_005_000',
'hds_00_000_005_001',
'hds_00_000_006_000',
'hds_00_000_006_001',
'hds_00_000_007_000',
'hds_00_000_007_001',
'hds_00_000_008_000',
'hds_00_000_008_001',
'hds_00_000_009_000',
'hds_00_000_009_001',
```

```
'hds_00_000_010_000',
'hds_00_000_010_001',
'hds_00_000_011_000',
'hds_00_000_011_001',
'hds_00_000_012_000',
'hds_00_000_012_001',
'hds_00_000_013_000',
'hds_00_000_013_001',
'hds_00_000_014_000',
'hds_00_000_014_001',
'hds_00_000_015_000',
'hds_00_000_015_001',
'hds_00_000_016_000',
'hds_00_000_016_001',
'hds_00_000_017_000',
'hds_00_000_017_001',
'hds_00_000_018_000',
'hds_00_000_018_001',
'hds_00_000_019_000',
'hds_00_000_019_001',
'hds_00_001_000_000',
'hds_00_001_000_001',
'hds_00_001_001_000',
'hds_00_001_001_001',
'hds_00_001_002_000',
'hds_00_001_002_001',
'hds_00_001_003_000',
'hds_00_001_003_001',
'hds_00_001_004_000',
'hds_00_001_004_001',
'hds_00_001_005_000',
'hds_00_001_005_001',
'hds_00_001_006_000',
'hds_00_001_006_001',
'hds_00_001_007_000',
'hds_00_001_007_001',
'hds_00_001_008_000',
'hds_00_001_008_001',
'hds_00_001_009_000',
'hds_00_001_009_001',
'hds_00_001_010_000',
'hds_00_001_010_001',
'hds_00_001_011_000',
'hds_00_001_011_001',
'hds_00_001_012_000',
'hds_00_001_012_001',
'hds_00_001_013_000',
'hds_00_001_013_001',
```

```
'hds_00_001_014_000',
'hds_00_001_014_001',
'hds_00_001_015_000',
'hds_00_001_015_001',
'hds_00_001_016_000',
'hds_00_001_016_001',
'hds_00_001_017_000',
'hds_00_001_017_001',
'hds_00_001_018_000',
'hds_00_001_018_001',
'hds_00_001_019_000',
'hds_00_001_019_001',
'hds_00_002_000_000',
'hds_00_002_000_001',
'hds_00_002_001_000',
'hds_00_002_001_001',
'hds_00_002_002_000',
'hds_00_002_002_001',
'hds_00_002_003_000',
'hds_00_002_003_001',
'hds_00_002_004_000',
'hds_00_002_004_001',
'hds_00_002_005_000',
'hds_00_002_005_001',
'hds_00_002_006_000',
'hds_00_002_006_001',
'hds_00_002_007_000',
'hds_00_002_007_001',
'hds_00_002_008_000',
'hds_00_002_008_001',
'hds_00_002_009_001',
'hds_00_002_010_000',
'hds_00_002_010_001',
'hds_00_002_011_000',
'hds_00_002_011_001',
'hds_00_002_012_000',
'hds_00_002_012_001',
'hds_00_002_013_000',
'hds_00_002_013_001',
'hds_00_002_014_000',
'hds_00_002_014_001',
'hds_00_002_015_001',
'hds_00_002_016_000',
'hds_00_002_016_001',
'hds_00_002_017_000',
'hds_00_002_017_001',
'hds_00_002_018_000',
'hds_00_002_018_001',
```

```
'hds_00_002_019_000',
'hds_00_002_019_001',
'hds_00_003_000_000',
'hds_00_003_000_001',
'hds_00_003_001_000',
'hds_00_003_001_001',
'hds_00_003_002_000',
'hds_00_003_002_001',
'hds_00_003_003_000',
'hds_00_003_003_001',
'hds_00_003_004_000',
'hds_00_003_004_001',
'hds_00_003_005_000',
'hds_00_003_005_001',
'hds_00_003_006_000',
'hds_00_003_006_001',
'hds_00_003_007_000',
'hds_00_003_007_001',
'hds_00_003_008_001',
'hds_00_003_009_000',
'hds_00_003_009_001',
'hds_00_003_010_000',
'hds_00_003_010_001',
'hds_00_003_011_000',
'hds_00_003_011_001',
'hds_00_003_012_000',
'hds_00_003_012_001',
'hds_00_003_013_000',
'hds_00_003_013_001',
'hds_00_003_014_000',
'hds_00_003_014_001',
'hds_00_003_015_000',
'hds_00_003_015_001',
'hds_00_003_016_000',
'hds_00_003_016_001',
'hds_00_003_017_000',
'hds_00_003_017_001',
'hds_00_003_018_000',
'hds_00_003_018_001',
'hds_00_003_019_000',
'hds_00_003_019_001',
'hds_00_004_000_000',
'hds_00_004_000_001',
'hds_00_004_001_000',
'hds_00_004_001_001',
'hds_00_004_002_000',
'hds_00_004_002_001',
'hds_00_004_003_000',
```

```
'hds_00_004_003_001',
'hds_00_004_004_000',
'hds_00_004_004_001',
'hds_00_004_005_000',
'hds_00_004_005_001',
'hds_00_004_006_000',
'hds_00_004_006_001',
'hds_00_004_007_000',
'hds_00_004_007_001',
'hds_00_004_008_000',
'hds_00_004_008_001',
'hds_00_004_009_000',
'hds_00_004_009_001',
'hds_00_004_010_000',
'hds_00_004_010_001',
'hds_00_004_011_000',
'hds_00_004_011_001',
'hds_00_004_012_000',
'hds_00_004_012_001',
'hds_00_004_013_000',
'hds_00_004_013_001',
'hds_00_004_014_000',
'hds_00_004_014_001',
'hds_00_004_015_000',
'hds_00_004_015_001',
'hds_00_004_016_000',
'hds_00_004_016_001',
'hds_00_004_017_000',
'hds_00_004_017_001',
'hds_00_004_018_000',
'hds_00_004_018_001',
'hds_00_004_019_000',
'hds_00_004_019_001',
'hds_00_005_000_000',
'hds_00_005_000_001',
'hds_00_005_001_000',
'hds_00_005_001_001',
'hds_00_005_002_000',
'hds_00_005_002_001',
'hds_00_005_003_000',
'hds_00_005_003_001',
'hds_00_005_004_000',
'hds_00_005_004_001',
'hds_00_005_005_000',
'hds_00_005_005_001',
'hds_00_005_006_000',
'hds_00_005_006_001',
'hds_00_005_007_000',
```

```
'hds_00_005_007_001',
'hds_00_005_008_000',
'hds_00_005_008_001',
'hds_00_005_009_000',
'hds_00_005_009_001',
'hds_00_005_010_000',
'hds_00_005_010_001',
'hds_00_005_011_000',
'hds_00_005_011_001',
'hds_00_005_012_000',
'hds_00_005_012_001',
'hds_00_005_013_000',
'hds_00_005_013_001',
'hds_00_005_014_000',
'hds_00_005_014_001',
'hds_00_005_015_000',
'hds_00_005_015_001',
'hds_00_005_016_000',
'hds_00_005_016_001',
'hds_00_005_017_000',
'hds_00_005_017_001',
'hds_00_005_018_000',
'hds_00_005_018_001',
'hds_00_005_019_000',
'hds_00_005_019_001',
'hds_00_006_000_000',
'hds_00_006_000_001',
'hds_00_006_001_000',
'hds_00_006_001_001',
'hds_00_006_002_000',
'hds_00_006_002_001',
'hds_00_006_003_000',
'hds_00_006_003_001',
'hds_00_006_004_000',
'hds_00_006_004_001',
'hds_00_006_005_000',
'hds_00_006_005_001',
'hds_00_006_006_000',
'hds_00_006_006_001',
'hds_00_006_007_000',
'hds_00_006_007_001',
'hds_00_006_008_000',
'hds_00_006_008_001',
'hds_00_006_009_000',
'hds_00_006_009_001',
'hds_00_006_010_000',
'hds_00_006_010_001',
'hds_00_006_011_000',
```

```
'hds_00_006_011_001',
'hds_00_006_012_000',
'hds_00_006_012_001',
'hds_00_006_013_000',
'hds_00_006_013_001',
'hds_00_006_014_000',
'hds_00_006_014_001',
'hds_00_006_015_000',
'hds_00_006_015_001',
'hds_00_006_016_000',
'hds_00_006_016_001',
'hds_00_006_017_000',
'hds_00_006_017_001',
'hds_00_006_018_000',
'hds_00_006_018_001',
'hds_00_006_019_000',
'hds_00_006_019_001',
'hds_00_007_000_000',
'hds_00_007_000_001',
'hds_00_007_001_000',
'hds_00_007_001_001',
'hds_00_007_002_000',
'hds_00_007_002_001',
'hds_00_007_003_000',
'hds_00_007_003_001',
'hds_00_007_004_000',
'hds_00_007_004_001',
'hds_00_007_005_000',
'hds_00_007_005_001',
'hds_00_007_006_000',
'hds_00_007_006_001',
'hds_00_007_007_000',
'hds_00_007_007_001',
'hds_00_007_008_000',
'hds_00_007_008_001',
'hds_00_007_009_000',
'hds_00_007_009_001',
'hds_00_007_010_000',
'hds_00_007_010_001',
'hds_00_007_011_000',
'hds_00_007_011_001',
'hds_00_007_012_000',
'hds_00_007_012_001',
'hds_00_007_013_000',
'hds_00_007_013_001',
'hds_00_007_014_000',
'hds_00_007_014_001',
'hds_00_007_015_000',
```

```
'hds_00_007_015_001',
'hds_00_007_016_000',
'hds_00_007_016_001',
'hds_00_007_017_000',
'hds_00_007_017_001',
'hds_00_007_018_000',
'hds_00_007_018_001',
'hds_00_007_019_000',
'hds_00_007_019_001',
'hds_00_008_000_000',
'hds_00_008_000_001',
'hds_00_008_001_000',
'hds_00_008_001_001',
'hds_00_008_002_000',
'hds_00_008_002_001',
'hds_00_008_003_000',
'hds_00_008_003_001',
'hds_00_008_007_000',
'hds_00_008_007_001',
'hds_00_008_008_000',
'hds_00_008_008_001',
'hds_00_008_009_000',
'hds_00_008_009_001',
'hds_00_008_010_000',
'hds_00_008_010_001',
'hds_00_008_011_000',
'hds_00_008_011_001',
'hds_00_008_012_000',
'hds_00_008_012_001',
'hds_00_008_013_000',
'hds_00_008_013_001',
'hds_00_008_014_000',
'hds_00_008_014_001',
'hds_00_008_015_000',
'hds_00_008_015_001',
'hds_00_008_016_000',
'hds_00_008_016_001',
'hds_00_008_017_000',
'hds_00_008_017_001',
'hds_00_008_018_000',
'hds_00_008_018_001',
'hds_00_008_019_000',
'hds_00_008_019_001',
'hds_00_009_000_000',
'hds_00_009_000_001',
'hds_00_009_001_001',
'hds_00_009_002_000',
'hds_00_009_002_001',
```

```
'hds_00_009_003_000',
'hds_00_009_003_001',
'hds_00_009_008_000',
'hds_00_009_008_001',
'hds_00_009_009_000',
'hds_00_009_009_001',
'hds_00_009_010_000',
'hds_00_009_010_001',
'hds_00_009_011_000',
'hds_00_009_011_001',
'hds_00_009_012_000',
'hds_00_009_012_001',
'hds_00_009_013_000',
'hds_00_009_013_001',
'hds_00_009_014_000',
'hds_00_009_014_001',
'hds_00_009_015_000',
'hds_00_009_015_001',
'hds_00_009_016_000',
'hds_00_009_016_001',
'hds_00_009_017_000',
'hds_00_009_017_001',
'hds_00_009_018_000',
'hds_00_009_018_001',
'hds_00_009_019_000',
'hds_00_009_019_001',
'hds_00_010_000_000',
'hds_00_010_000_001',
'hds_00_010_001_000',
'hds_00_010_001_001',
'hds_00_010_002_000',
'hds_00_010_002_001',
'hds_00_010_003_000',
'hds_00_010_003_001',
'hds_00_010_008_000',
'hds_00_010_008_001',
'hds_00_010_009_000',
'hds_00_010_009_001',
'hds_00_010_010_000',
'hds_00_010_010_001',
'hds_00_010_011_000',
'hds_00_010_011_001',
'hds_00_010_012_000',
'hds_00_010_012_001',
'hds_00_010_013_000',
'hds_00_010_013_001',
'hds_00_010_014_000',
'hds_00_010_014_001',
```

```
'hds_00_010_015_000',
'hds_00_010_015_001',
'hds_00_010_016_000',
'hds_00_010_016_001',
'hds_00_010_017_000',
'hds_00_010_017_001',
'hds_00_010_018_000',
'hds_00_010_018_001',
'hds_00_010_019_000',
'hds_00_010_019_001',
'hds_00_011_000_000',
'hds_00_011_000_001',
'hds_00_011_001_000',
'hds_00_011_001_001',
'hds_00_011_002_000',
'hds_00_011_002_001',
'hds_00_011_003_000',
'hds_00_011_003_001',
'hds_00_011_008_000',
'hds_00_011_008_001',
'hds_00_011_009_000',
'hds_00_011_009_001',
'hds_00_011_010_000',
'hds_00_011_010_001',
'hds_00_011_011_000',
'hds_00_011_011_001',
'hds_00_011_012_000',
'hds_00_011_012_001',
'hds_00_011_013_000',
'hds_00_011_013_001',
'hds_00_011_014_000',
'hds_00_011_014_001',
'hds_00_011_015_000',
'hds_00_011_015_001',
'hds_00_011_016_000',
'hds_00_011_016_001',
'hds_00_011_017_000',
'hds_00_011_017_001',
'hds_00_011_018_000',
'hds_00_011_018_001',
'hds_00_011_019_000',
'hds_00_011_019_001',
'hds_00_012_000_000',
'hds_00_012_000_001',
'hds_00_012_001_000',
'hds_00_012_001_001',
'hds_00_012_002_000',
'hds_00_012_002_001',
```

```
'hds_00_012_003_000',
'hds_00_012_003_001',
'hds_00_012_008_000',
'hds_00_012_008_001',
'hds_00_012_009_000',
'hds_00_012_009_001',
'hds_00_012_010_000',
'hds_00_012_010_001',
'hds_00_012_011_000',
'hds_00_012_011_001',
'hds_00_012_012_000',
'hds_00_012_012_001',
'hds_00_012_013_000',
'hds_00_012_013_001',
'hds_00_012_014_000',
'hds_00_012_014_001',
'hds_00_012_015_000',
'hds_00_012_015_001',
'hds_00_012_016_000',
'hds_00_012_016_001',
'hds_00_012_017_000',
'hds_00_012_017_001',
'hds_00_012_018_000',
'hds_00_012_018_001',
'hds_00_012_019_000',
'hds_00_012_019_001',
'hds_00_013_000_000',
'hds_00_013_000_001',
'hds_00_013_001_000',
'hds_00_013_001_001',
'hds_00_013_003_000',
'hds_00_013_003_001',
'hds_00_013_008_000',
'hds_00_013_008_001',
'hds_00_013_009_000',
'hds_00_013_009_001',
'hds_00_013_010_001',
'hds_00_013_011_000',
'hds_00_013_011_001',
'hds_00_013_012_000',
'hds_00_013_012_001',
'hds_00_013_013_000',
'hds_00_013_013_001',
'hds_00_013_014_000',
'hds_00_013_014_001',
'hds_00_013_015_000',
'hds_00_013_015_001',
'hds_00_013_016_000',
```

```
'hds_00_013_016_001',
'hds_00_013_017_000',
'hds_00_013_017_001',
'hds_00_013_018_000',
'hds_00_013_018_001',
'hds_00_013_019_000',
'hds_00_013_019_001',
'hds_00_014_000_000',
'hds_00_014_000_001',
'hds_00_014_001_000',
'hds_00_014_001_001',
'hds_00_014_002_000',
'hds_00_014_002_001',
'hds_00_014_003_000',
'hds_00_014_003_001',
'hds_00_014_008_000',
'hds_00_014_008_001',
'hds_00_014_009_000',
'hds_00_014_009_001',
'hds_00_014_010_000',
'hds_00_014_010_001',
'hds_00_014_011_000',
'hds_00_014_011_001',
'hds_00_014_012_000',
'hds_00_014_012_001',
'hds_00_014_013_000',
'hds_00_014_013_001',
'hds_00_014_014_000',
'hds_00_014_014_001',
'hds_00_014_015_000',
'hds_00_014_015_001',
'hds_00_014_016_000',
'hds_00_014_016_001',
'hds_00_014_017_000',
'hds_00_014_017_001',
'hds_00_014_018_000',
'hds_00_014_018_001',
'hds_00_014_019_000',
'hds_00_014_019_001',
'hds_00_015_000_000',
'hds_00_015_000_001',
'hds_00_015_001_000',
'hds_00_015_001_001',
'hds_00_015_002_000',
'hds_00_015_002_001',
'hds_00_015_003_000',
'hds_00_015_003_001',
'hds_00_015_008_000',
```

```
'hds_00_015_008_001',
'hds_00_015_009_000',
'hds_00_015_009_001',
'hds_00_015_010_000',
'hds_00_015_010_001',
'hds_00_015_011_000',
'hds_00_015_011_001',
'hds_00_015_012_000',
'hds_00_015_012_001',
'hds_00_015_013_000',
'hds_00_015_013_001',
'hds_00_015_014_000',
'hds_00_015_014_001',
'hds_00_015_015_000',
'hds_00_015_015_001',
'hds_00_015_016_001',
'hds_00_015_017_000',
'hds_00_015_017_001',
'hds_00_015_018_000',
'hds_00_015_018_001',
'hds_00_015_019_000',
'hds_00_015_019_001',
'hds_00_016_000_000',
'hds_00_016_000_001',
'hds_00_016_001_000',
'hds_00_016_001_001',
'hds_00_016_002_000',
'hds_00_016_002_001',
'hds_00_016_003_000',
'hds_00_016_003_001',
'hds_00_016_008_000',
'hds_00_016_008_001',
'hds_00_016_009_000',
'hds_00_016_009_001',
'hds_00_016_010_000',
'hds_00_016_010_001',
'hds_00_016_011_000',
'hds_00_016_011_001',
'hds_00_016_012_000',
'hds_00_016_012_001',
'hds_00_016_013_000',
'hds_00_016_013_001',
'hds_00_016_014_000',
'hds_00_016_014_001',
'hds_00_016_015_000',
'hds_00_016_015_001',
'hds_00_016_016_000',
'hds_00_016_016_001',
```

```
'hds_00_016_017_000',
'hds_00_016_017_001',
'hds_00_016_018_000',
'hds_00_016_018_001',
'hds_00_016_019_000',
'hds_00_016_019_001',
'hds_00_017_000_000',
'hds_00_017_000_001',
'hds_00_017_001_000',
'hds_00_017_001_001',
'hds_00_017_002_000',
'hds_00_017_002_001',
'hds_00_017_003_000',
'hds_00_017_003_001',
'hds_00_017_008_000',
'hds_00_017_008_001',
'hds_00_017_009_000',
'hds_00_017_009_001',
'hds_00_017_010_000',
'hds_00_017_010_001',
'hds_00_017_011_000',
'hds_00_017_011_001',
'hds_00_017_012_000',
'hds_00_017_012_001',
'hds_00_017_013_000',
'hds_00_017_013_001',
'hds_00_017_014_000',
'hds_00_017_014_001',
'hds_00_017_015_000',
'hds_00_017_015_001',
'hds_00_017_016_000',
'hds_00_017_016_001',
'hds_00_017_017_000',
'hds_00_017_017_001',
'hds_00_017_018_000',
'hds_00_017_018_001',
'hds_00_017_019_000',
'hds_00_017_019_001',
'hds_00_018_000_000',
'hds_00_018_000_001',
'hds_00_018_001_000',
'hds_00_018_001_001',
'hds_00_018_002_000',
'hds_00_018_002_001',
'hds_00_018_003_000',
'hds_00_018_003_001',
'hds_00_018_008_000',
'hds_00_018_008_001',
```

```
'hds_00_018_009_000',
'hds_00_018_009_001',
'hds_00_018_010_000',
'hds_00_018_010_001',
'hds_00_018_011_000',
'hds_00_018_011_001',
'hds_00_018_012_000',
'hds_00_018_012_001',
'hds_00_018_013_000',
'hds_00_018_013_001',
'hds_00_018_014_000',
'hds_00_018_014_001',
'hds_00_018_015_000',
'hds_00_018_015_001',
'hds_00_018_016_000',
'hds_00_018_016_001',
'hds_00_018_017_000',
'hds_00_018_017_001',
'hds_00_018_018_000',
'hds_00_018_018_001',
'hds_00_018_019_000',
'hds_00_018_019_001',
'hds_00_019_000_000',
'hds_00_019_000_001',
'hds_00_019_001_000',
'hds_00_019_001_001',
'hds_00_019_002_000',
'hds_00_019_002_001',
'hds_00_019_003_000',
'hds_00_019_003_001',
'hds_00_019_008_000',
'hds_00_019_008_001',
'hds_00_019_009_000',
'hds_00_019_009_001',
'hds_00_019_010_000',
'hds_00_019_010_001',
'hds_00_019_011_000',
'hds_00_019_011_001',
'hds_00_019_012_000',
'hds_00_019_012_001',
'hds_00_019_013_000',
'hds_00_019_013_001',
'hds_00_019_014_000',
'hds_00_019_014_001',
'hds_00_019_015_000',
'hds_00_019_015_001',
'hds_00_019_016_000',
'hds_00_019_016_001',
```

```
'hds_00_019_017_000',
'hds_00_019_017_001',
'hds_00_019_018_000',
'hds_00_019_018_001',
'hds_00_019_019_000',
'hds_00_019_019_001',
'hds_00_020_000_000',
'hds_00_020_000_001',
'hds_00_020_001_000',
'hds_00_020_001_001',
'hds_00_020_002_000',
'hds_00_020_002_001',
'hds_00_020_003_000',
'hds_00_020_003_001',
'hds_00_020_008_000',
'hds_00_020_008_001',
'hds_00_020_009_000',
'hds_00_020_009_001',
'hds_00_020_010_000',
'hds_00_020_010_001',
'hds_00_020_011_000',
'hds_00_020_011_001',
'hds_00_020_012_000',
'hds_00_020_012_001',
'hds_00_020_013_000',
'hds_00_020_013_001',
'hds_00_020_014_000',
'hds_00_020_014_001',
'hds_00_020_015_000',
'hds_00_020_015_001',
'hds_00_020_016_000',
'hds_00_020_016_001',
'hds_00_020_017_000',
'hds_00_020_017_001',
'hds_00_020_018_000',
'hds_00_020_018_001',
'hds_00_020_019_000',
'hds_00_020_019_001',
'hds_00_021_000_000',
'hds_00_021_000_001',
'hds_00_021_001_000',
'hds_00_021_001_001',
'hds_00_021_002_000',
'hds_00_021_002_001',
'hds_00_021_003_000',
'hds_00_021_003_001',
'hds_00_021_006_000',
'hds_00_021_006_001',
```

```
'hds_00_021_007_000',
'hds_00_021_007_001',
'hds_00_021_008_000',
'hds_00_021_008_001',
'hds_00_021_009_000',
'hds_00_021_009_001',
'hds_00_021_010_001',
'hds_00_021_011_000',
'hds_00_021_011_001',
'hds_00_021_012_000',
'hds_00_021_012_001',
'hds_00_021_013_000',
'hds_00_021_013_001',
'hds_00_021_014_000',
'hds_00_021_014_001',
'hds_00_021_015_000',
'hds_00_021_015_001',
'hds_00_021_016_000',
'hds_00_021_016_001',
'hds_00_021_017_000',
'hds_00_021_017_001',
'hds_00_021_018_000',
'hds_00_021_018_001',
'hds_00_021_019_000',
'hds_00_021_019_001',
'hds_00_022_000_000',
'hds_00_022_000_001',
'hds_00_022_001_000',
'hds_00_022_001_001',
'hds_00_022_002_000',
'hds_00_022_002_001',
'hds_00_022_003_000',
'hds_00_022_003_001',
'hds_00_022_004_000',
'hds_00_022_004_001',
'hds_00_022_006_000',
'hds_00_022_006_001',
'hds_00_022_007_000',
'hds_00_022_007_001',
'hds_00_022_008_000',
'hds_00_022_008_001',
'hds_00_022_009_000',
'hds_00_022_009_001',
'hds_00_022_010_000',
'hds_00_022_010_001',
'hds_00_022_011_000',
'hds_00_022_011_001',
'hds_00_022_012_000',
```

```
 'hds_00_022_012_001',
 'hds_00_022_013_000',
 'hds_00_022_013_001',
 'hds_00_022_014_000',
 'hds_00_022_014_001',
 'hds_00_022_015_001',
 'hds_00_022_016_000',
 'hds_00_022_016_001',
 'hds_00_022_017_000',
 'hds_00_022_017_001',
 'hds_00_022_018_000',
 'hds_00_022_018_001',
 'hds_00_022_019_000',
 'hds_00_022_019_001',
 'hds_00_023_000_000',
 'hds_00_023_000_001',
 'hds_00_023_001_000',
 'hds_00_023_001_001',
 'hds_00_023_002_000',
 'hds_00_023_002_001',
 'hds_00_023_003_000',
 'hds_00_023_003_001',
 'hds_00_023_004_000',
 'hds_00_023_004_001',
 'hds_00_023_005_000',
 'hds_00_023_005_001',
 'hds_00_023_006_000',
 'hds_00_023_006_001',
 'hds_00_023_007_000',
 'hds_00_023_007_001',
 'hds_00_023_008_000',
 ...]
```

We can therefore repeat above analysis for the observations that currently have zero weight by turning those observations "on".

Beware: calculating the Schur complement for all potential observation types and locations could take some time!! So we will sample to speed things up. You may need to further reduce the number of potential obs - you can do this by adding [0::2] to take every second element for example.

```
In [22]: new_obs = [x for x in z_obs if "hds_00" in x]#and x.endswith("_000")  # all heads in
         print("number of new potential head observation locations considered: {}".format(len(r
```

```
number of new potential head observation locations considered: 1395
```

```
In [23]: from datetime import datetime
         start = datetime.now()
```

```
df_worth_new = sc.get_added_obs_importance(obslist_dict=new_obs, base_obslist=sc.pst.r
print("took:",datetime.now() - start)
```

```
took: 0:02:12.042580
```

In [24]:
```
df_worth_new.head()
```

Out[24]:
```
                     fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
base                   45999.208695   273159.126193    20437.475638
hds_00_000_000_000     45981.155345   273145.362839    20437.088584
hds_00_000_000_001     45978.901180   271004.867935    20437.165963
hds_00_000_001_000     45982.314955   273145.553598    20436.955632
hds_00_000_001_001     45980.505927   270945.569451    20437.264005

                     fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
base                  218490.640452            0.089725            0.168907
hds_00_000_000_000    218490.404369            0.089529            0.168618
hds_00_000_000_001    216417.089852            0.089573            0.168305
hds_00_000_001_000    218490.428861            0.089518            0.168610
hds_00_000_001_001    216373.244627            0.089562            0.168319

                     part_status     part_time
base                         0.0  1.258207e+07
hds_00_000_000_000           0.0  1.258051e+07
hds_00_000_000_001           0.0  1.258088e+07
hds_00_000_001_000           0.0  1.258057e+07
hds_00_000_001_001           0.0  1.258093e+07
```

1.1.5 nice! now let's process a little bit and make some plots of (potential) data worth

In [25]:
```
def worth_plot_prep(df):
    # some processing
    df_new_base = df.loc["base",:].copy()  # "base" row
    df_new_imax = df.apply(lambda x: df_new_base - x, axis=1).idxmax()  # obs with la
    df_new_worth = 100.0 * (df.apply(lambda x: df_new_base - x, axis=1) / df_new_base)

    # plot prep
    df_new_worth_plot = df_new_worth[df_new_worth.index != 'base'].copy()
    df_new_worth_plot.loc[:,'names'] = df_new_worth_plot.index
    names = df_new_worth_plot.names
    df_new_worth_plot.loc[:,"i"] = names.apply(lambda x: int(x[8:10]))
    df_new_worth_plot.loc[:,"j"] = names.apply(lambda x: int(x[11:14]))
    df_new_worth_plot.loc[:,'kper'] = names.apply(lambda x: int(x[-3:]))
    #df_new_worth_plot.head()

    return df_new_worth_plot, df_new_imax
```

In [26]:
```
df_worth_new_plot, df_worth_new_imax = worth_plot_prep(df_worth_new)
```

In [27]:
```
df_worth_new_plot.head()
```

Out[27]:
```
                    fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
hds_00_000_000_000        0.039247        0.005039        0.001894
hds_00_000_000_001        0.044148        0.788646        0.001515
hds_00_000_001_000        0.036726        0.004969        0.002544
hds_00_000_001_001        0.040659        0.810354        0.001036
hds_00_000_002_000        0.035804        0.005004        0.002372

                    fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
hds_00_000_000_000        0.000108            0.219015            0.170793
hds_00_000_000_001        0.949034            0.170139            0.356402
hds_00_000_001_000        0.000097            0.231211            0.175668
hds_00_000_001_001        0.969101            0.182607            0.347883
hds_00_000_002_000        0.000086            0.246462            0.177513

                    part_status  part_time               names  i  j  kper
hds_00_000_000_000          NaN   0.012406  hds_00_000_000_000  0  0     0
hds_00_000_000_001          NaN   0.009440  hds_00_000_000_001  0  0     1
hds_00_000_001_000          NaN   0.011919  hds_00_000_001_000  0  1     0
hds_00_000_001_001          NaN   0.009101  hds_00_000_001_001  0  1     1
hds_00_000_002_000          NaN   0.010760  hds_00_000_002_000  0  2     0
```

In [28]:
```
df_worth_new_imax  # which obs causes largest unc var reduction?
```

Out[28]:
```
fa_hw_19791230        hds_00_009_016_001
fa_hw_19801229        hds_00_011_013_001
fa_tw_19791230        hds_00_020_014_001
fa_tw_19801229        hds_00_026_010_001
hds_00_013_002_000    hds_00_016_001_001
hds_00_013_002_001    hds_00_017_002_001
part_status                         base
part_time             hds_00_017_002_001
dtype: object
```

In [29]:
```
df_worth_new_plot.drop(axis=1,labels=["part_status"],inplace=True) # drop "part_statu
df_worth_new_plot.head()
```

Out[29]:
```
                    fa_hw_19791230  fa_hw_19801229  fa_tw_19791230  \
hds_00_000_000_000        0.039247        0.005039        0.001894
hds_00_000_000_001        0.044148        0.788646        0.001515
hds_00_000_001_000        0.036726        0.004969        0.002544
hds_00_000_001_001        0.040659        0.810354        0.001036
hds_00_000_002_000        0.035804        0.005004        0.002372

                    fa_tw_19801229  hds_00_013_002_000  hds_00_013_002_001  \
hds_00_000_000_000        0.000108            0.219015            0.170793
hds_00_000_000_001        0.949034            0.170139            0.356402
hds_00_000_001_000        0.000097            0.231211            0.175668
```

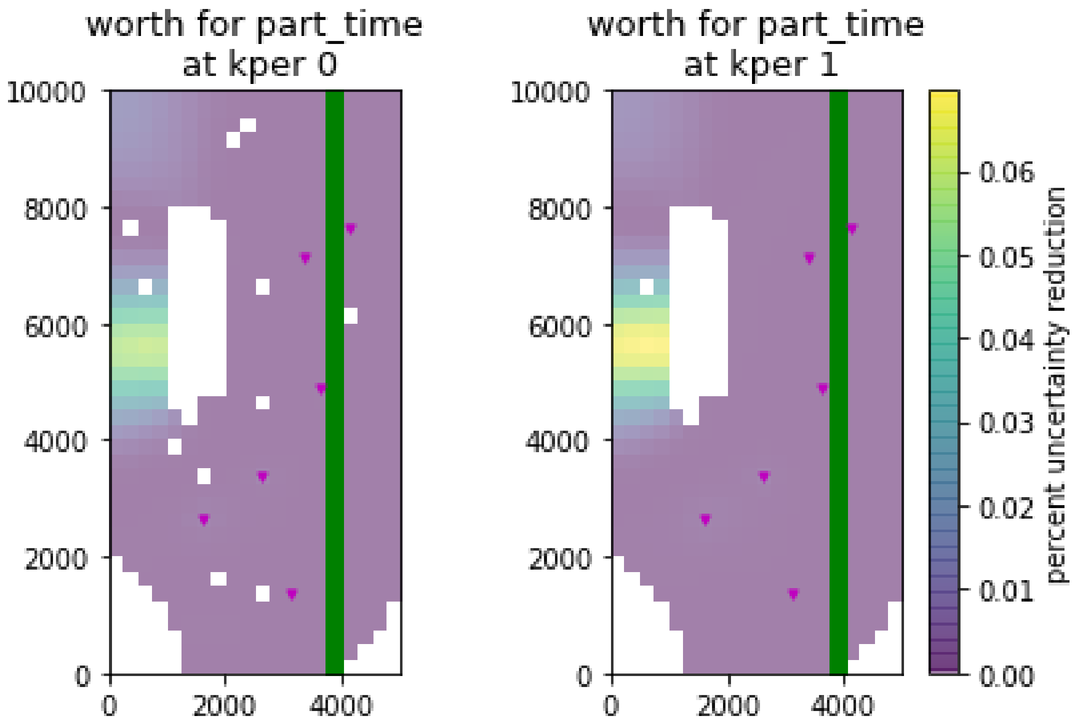
```
hds_00_000_001_001        0.969101           0.182607           0.347883
hds_00_000_002_000        0.000086           0.246462           0.177513

                     part_time              names  i  j  kper
hds_00_000_000_000    0.012406  hds_00_000_000_000  0  0     0
hds_00_000_000_001    0.009440  hds_00_000_000_001  0  0     1
hds_00_000_001_000    0.011919  hds_00_000_001_000  0  1     0
hds_00_000_001_001    0.009101  hds_00_000_001_001  0  1     1
hds_00_000_002_000    0.010760  hds_00_000_002_000  0  2     0
```

1.1.6 plotting

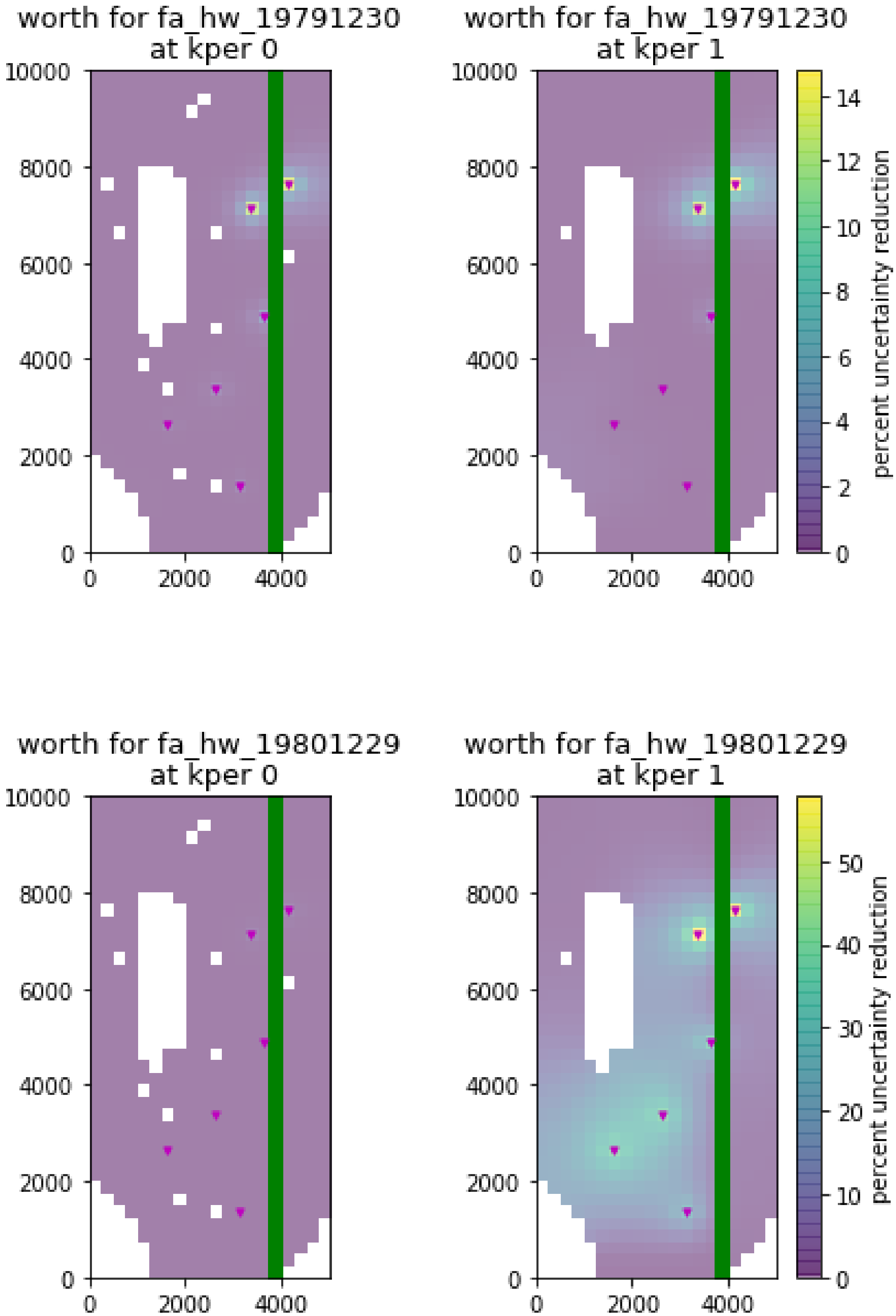
```
In [30]: m = flopy.modflow.Modflow.load("freyberg.nam", model_ws=os.path.join(m_d))

In [31]: def plot_added_importance(df_worth_plot, ml, forecast_name=None,
                                   newlox=None,):

             vmax = df_worth_plot[forecast_name].max()

             fig, axs = plt.subplots(1,2)
             if newlox:
                 currx = []
                 curry = []
                 for i,clox in enumerate(newlox):
                     crow = int(clox[8:10])
                     ccol = int(clox[11:14])
                     currx.append(ml.sr.xcentergrid[crow,ccol])
                     curry.append(ml.sr.ycentergrid[crow,ccol])

             for sp,ax in enumerate(axs): # by kpers
                 unc_array = np.zeros_like(ml.upw.hk[0].array) - 1
                 df_worth_csp = df_worth_plot.groupby('kper').get_group(sp)
                 for i,j,unc in zip(df_worth_csp.i,df_worth_csp.j,
                                    df_worth_csp[forecast_name]):
                     unc_array[i,j] = unc
                 unc_array[unc_array == -1] = np.NaN
                 cb = ax.imshow(unc_array,interpolation="nearest",
                                alpha=0.5,extent=ml.sr.get_extent(),
                                vmin=0, vmax=vmax)
                 if sp==1:
                     plt.colorbar(cb,label="percent uncertainty reduction")

                 # plot sfr
                 sfr_data = ml.sfr.stress_period_data[0]
                 sfr_x = ml.sr.xcentergrid[sfr_data["i"],sfr_data["j"]]
                 sfr_y = ml.sr.ycentergrid[sfr_data["i"],sfr_data["j"]]
                 for (x,y) in zip(sfr_x,sfr_y):
                     ax.scatter([x],[y],marker="s",color="g",s=26)
```

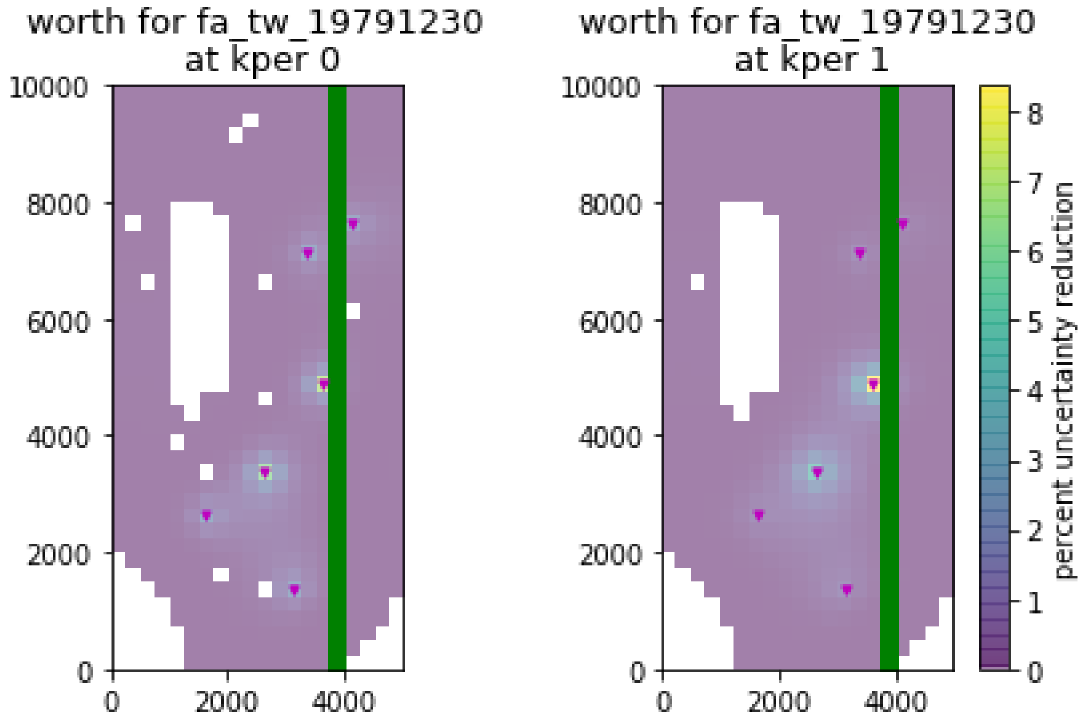
```python
        # plot the pumping wells
        wel_data = ml.wel.stress_period_data[0]
        wel_x = ml.sr.xcentergrid[wel_data["i"],wel_data["j"]]
        wel_y = ml.sr.ycentergrid[wel_data["i"],wel_data["j"]]
        for w,(x,y) in enumerate(zip(wel_x,wel_y)):
            ax.scatter([x],[y],marker="v",color="m",s=10)

        if newlox:
            for nl,(cx,cy,cobs) in enumerate(zip(currx, curry, newlox)):
                csp = int(cobs[-1])
                if csp == sp:
                    ax.plot(cx, cy, 'rd', mfc=None, ms=10, alpha=0.8)
                    ax.text(cx-50,cy-50, nl, size=10)

        # plot the location of the forecast if possible
        if forecast_name.startswith('hds'):
            i = int(forecast_name[8:10])
            j = int(forecast_name[11:14])
            forecast_x = ml.sr.xcentergrid[i,j]
            forecast_y = ml.sr.ycentergrid[i,j]
            ax.scatter(forecast_x, forecast_y, marker='o', s=600, alpha=0.5)

        ax.set_title("worth for {0}\n at kper {1}".format(forecast_name,sp), fontsize=
        plt.tight_layout()
    return fig
```

In [32]:
```python
fig = plot_added_importance(df_worth_plot=df_worth_new_plot, ml=m,forecast_name="part_
```

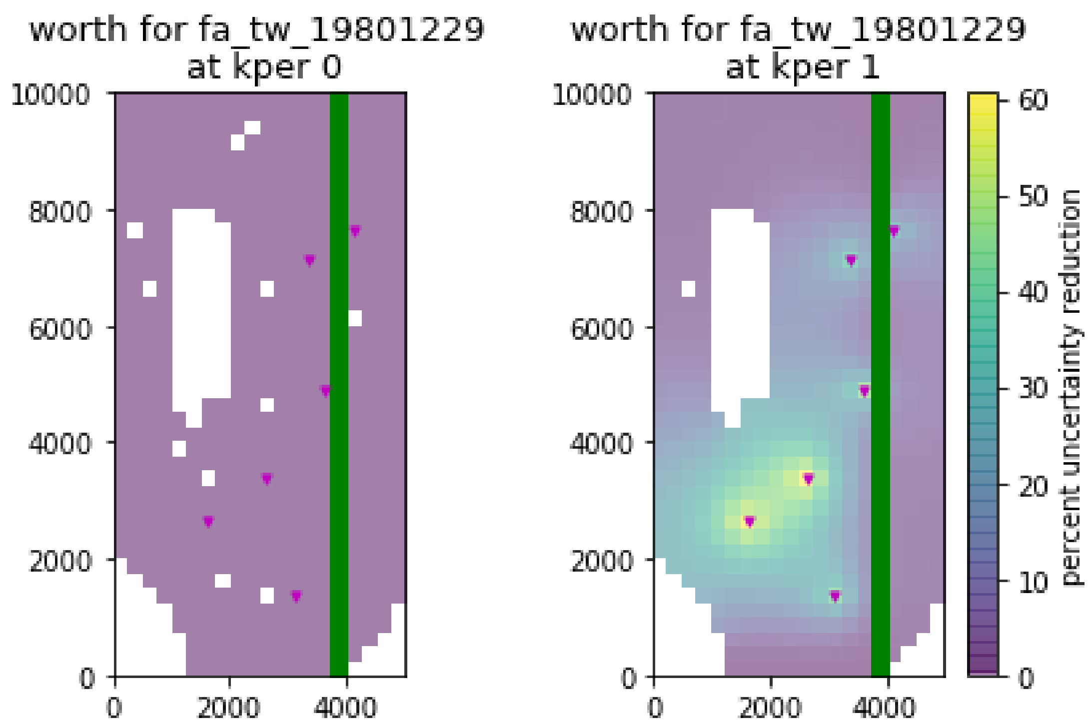



```
In [33]: for i in [x for x in forecasts if "part_status" not in x]:
             fig = plot_added_importance(df_worth_plot=df_worth_new_plot, ml=m,
                                         forecast_name=i)
             #fig.savefig('add_worth_{}.pdf'.format(i))
```

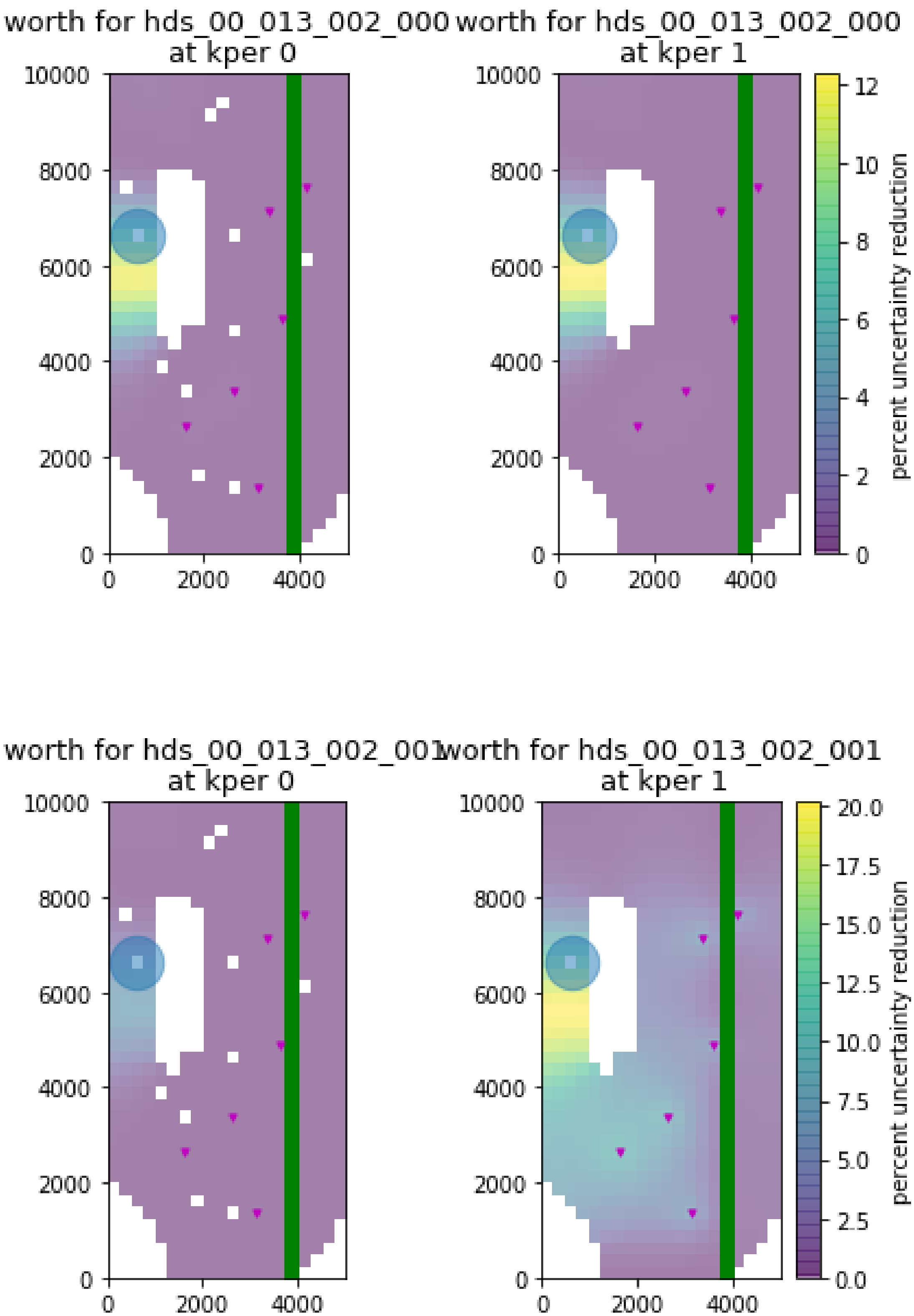

worth for fa_hw_19791230 at kper 0
worth for fa_hw_19791230 at kper 1
percent uncertainty reduction
worth for fa_hw_19801229 at kper 0
worth for fa_hw_19801229 at kper 1
percent uncertainty reduction
10000
8000
6000
4000
2000
0
2000
4000
0
2
4
6
8
10
12
14
10
20
30
40
50

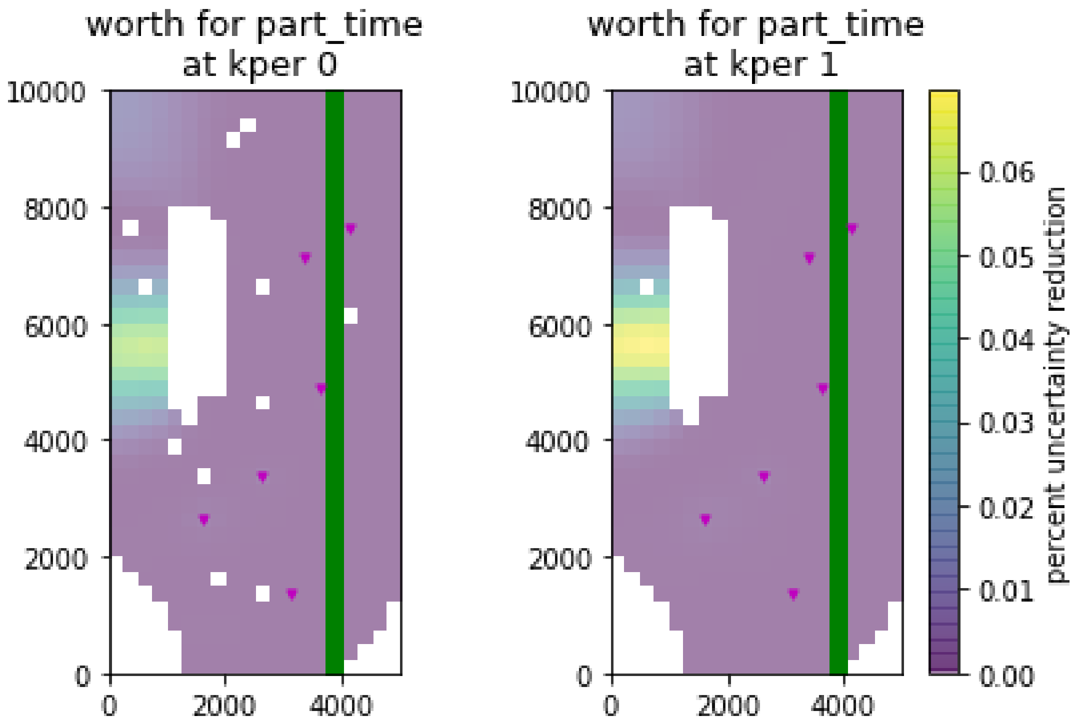


worth for fa_tw_19791230
at kper 0
worth for fa_tw_19791230
at kper 1
percent uncertainty reduction



worth for fa_tw_19801229
at kper 0
worth for fa_tw_19801229
at kper 1
percent uncertainty reduction


worth for hds_00_013_002_000
at kper 0
worth for hds_00_013_002_000
at kper 1
worth for hds_00_013_002_001
at kper 0
worth for hds_00_013_002_001
at kper 1
percent uncertainty reduction
percent uncertainty reduction
0
2000
4000
6000
8000
10000



1.2 the "next best" observation

This is what we would ultimately like to know... Takes into account what we already know through incrementally making additional observations. For example, consider making an observation in the middle of the zone of highest worth. Where should we subsequently collect data?

Let's just use the same potential observation list for now (the head in every top-layer cell) and evaluate which ones to collect, if we only had the budget for 5, in the context of the particle travel time prediction.

In [34]:
```
start = datetime.now()
next_most_df = sc.next_most_important_added_obs(forecast='part_time',niter=5,obslist_
                                                base_obslist=sc.pst.nnz_obs_names,rese
print("took:",datetime.now() - start)
```

```
took: 0:10:43.885949
```

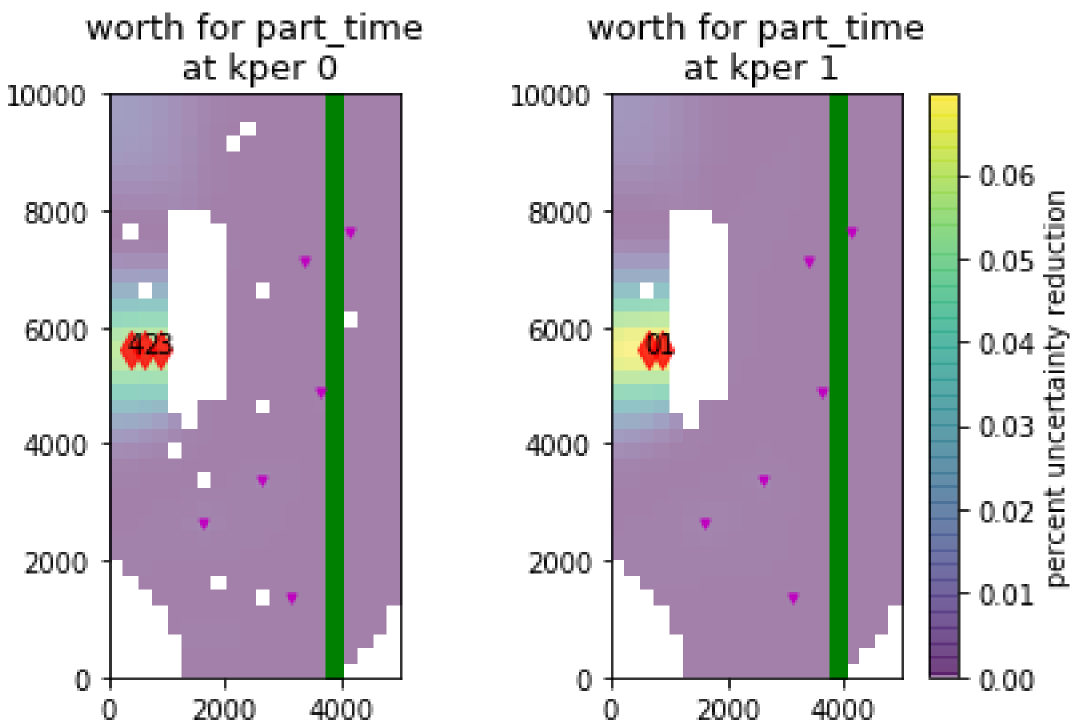
In [35]:
```
next_most_df
```

Out[35]:

	best_obs	part_time_variance	\
hds_00_017_002_001	hds_00_017_002_001	1.257332e+07	
hds_00_017_003_001	hds_00_017_003_001	1.256794e+07	
hds_00_017_002_000	hds_00_017_002_000	1.256376e+07	
hds_00_017_003_000	hds_00_017_003_000	1.256048e+07	

```
hds_00_017_001_000  hds_00_017_001_000        1.255786e+07

                    unc_reduce_iter_base  unc_reduce_initial_base
hds_00_017_002_001              0.069594                 0.069594
hds_00_017_003_001              0.042718                 0.112282
hds_00_017_002_000              0.033300                 0.145545
hds_00_017_003_000              0.026110                 0.171617
hds_00_017_001_000              0.020883                 0.192464
```

In [36]:
```
fig = plot_added_importance(df_worth_new_plot, m, 'part_time',
                            newlox=next_most_df.best_obs.tolist())
```



In [37]:
```
# for fun after class
'''
for i in [x for x in forecasts if "part_status" not in x]:
    next_most_df = sc.next_most_important_added_obs(forecast=i,niter=5,obslist_dict=d
                                                    base_obslist=sc.pst.nnz_obs_names
    fig = plot_added_importance(df_worth_new_plot, m, forcast_name=i,
                                newlox=next_most_df.best_obs.tolist())
    fig.savefig('next_best_5_worth_{}.pdf'.format(i))
'''
```

Out[37]:
```
'\nfor i in [x for x in forecasts if "part_status" not in x]:\n    next_most_df = sc.n
```

1.2.1 Note: an important assumption underpinning the above is that the model is able to fit observations to a level that is commensurate with measurement noise... Are we comfortable with this assumption? We will discuss this more in `pestpp-glm_part2.ipynb`

In [38]:
```
# recall...
pst.observation_data.loc[pst.nnz_obs_names,:]
```

Out[38]:

obsnme	obsnme	obsval	weight	obgnme	extra
fo_39_19791230	fo_39_19791230	13056.405235	0.01	calflux	NaN
hds_00_002_009_000	hds_00_002_009_000	37.561214	5.00	calhead	NaN
hds_00_002_015_000	hds_00_002_015_000	35.287045	5.00	calhead	NaN
hds_00_003_008_000	hds_00_003_008_000	38.506551	5.00	calhead	NaN
hds_00_009_001_000	hds_00_009_001_000	41.829578	5.00	calhead	NaN
hds_00_013_010_000	hds_00_013_010_000	35.071779	5.00	calhead	NaN
hds_00_015_016_000	hds_00_015_016_000	34.762081	5.00	calhead	NaN
hds_00_021_010_000	hds_00_021_010_000	35.130549	5.00	calhead	NaN
hds_00_022_015_000	hds_00_022_015_000	34.659017	5.00	calhead	NaN
hds_00_024_004_000	hds_00_024_004_000	36.851231	5.00	calhead	NaN
hds_00_026_006_000	hds_00_026_006_000	35.725369	5.00	calhead	NaN
hds_00_029_015_000	hds_00_029_015_000	34.562965	5.00	calhead	NaN
hds_00_033_007_000	hds_00_033_007_000	35.093152	5.00	calhead	NaN
hds_00_034_010_000	hds_00_034_010_000	34.407247	5.00	calhead	NaN

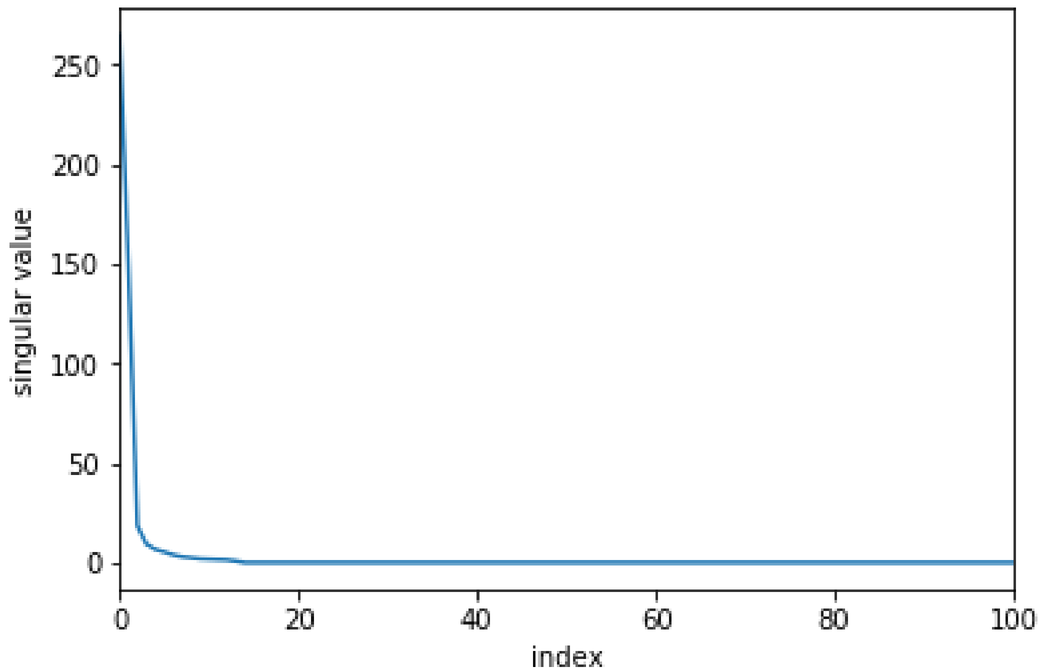
1.2.2 an "extra" if we have time: parameter identifiability

In [39]:
```
la = pyemu.ErrVar(jco=jco)
```

In [40]:
```
s = la.qhalfx.s  # singular spectrum
s.x[:10]
```

Out[40]:
```
array([[264.97056776],
       [147.97387188],
       [ 18.51044468],
       [  9.39137973],
       [  6.67548911],
       [  5.39991184],
       [  3.9177492 ],
       [  2.94841828],
       [  2.3809987 ],
       [  1.95277321]])
```

In [41]:
```
figure = plt.figure()
ax = plt.subplot(111)
ax.plot(s.x)
ax.set_ylabel("singular value")
ax.set_xlabel("index")
ax.set_xlim(0,100)
plt.show()
```



As expected, singluar spectrum decays rapidly.

```
In [42]: truncation_thresh = 1e-6
         n_signif_singvals = ((s.x / s[0].x) > 1e-6).sum()
```

```
In [43]: print("This means that, on the basis of the {0} (non-zero) weighted observations, \
         there are {1} unique pieces of information in the calibration dataset.  \
         Recall the inverse problem we are trying to solve involves the estimation of {2} param
               format(pst.nnz_obs, n_signif_singvals, pst.npar_adj))
```

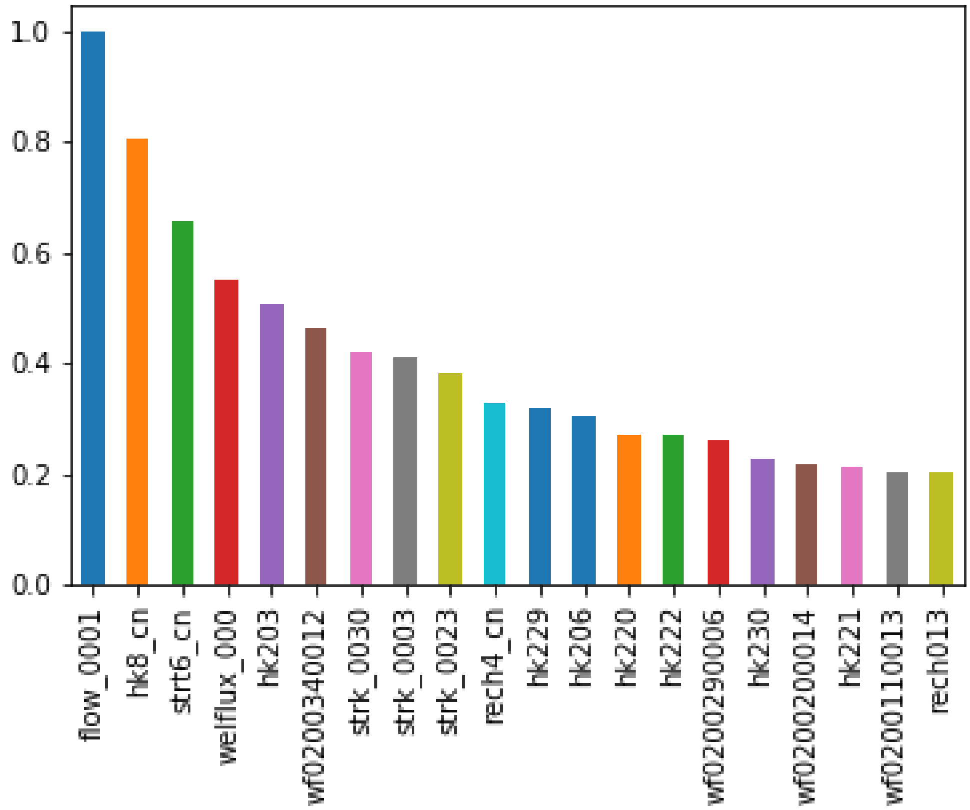
```
This means that, on the basis of the 14 (non-zero) weighted observations, there are 14 unique p
```

Now let's compute the identifiability of actual model parameters based on these singular vectors. Identifiability ranges from 0 (not identified by the data) to 1 (full identified by the data).

```
In [44]: ident_df = la.get_identifiability_dataframe()  # sing val trunc defaults to pst.nnz_ob
```

```
In [45]: ident_df.sort_values(by="ident",ascending=False).iloc[0:20].loc[:,"ident"].plot(kind="
```

```
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x28f02b63b38>
```



Note similarity with some of the earlier parameter contribution to forecast uncertainty results