In [1]:

```
### Zhe Zhang
### 2023-08-23
### python script to check output
### from wetland scheme

import numpy as np
import matplotlib.pyplot as plt
from netCDF4 import Dataset
import pandas as pd
```

In [2]:

```
### Run 3-year simulations at 30min interval, hence 52608 timesteps
### at Fen site, SK, Canada

def time_to_int(dateobj):
    total = int(dateobj.strftime('%S'))
    total += int(dateobj.strftime('%M')) * 60
    total += int(dateobj.strftime('%H')) * 60 * 60
    total += (int(dateobj.strftime('%j')) - 1) * 60 * 60 * 24
    total += (int(dateobj.strftime('%Y')) - 1970) * 60 * 60 * 24 * 365
    return total
times = pd.date_range('2003-01-01', periods=52608, freq='30min')
print (len(times))
```

```
52608
```

In [3]:

```
### Read two output files by their names,
### default means no wetland,
### wetland means with wetland model,
default = Dataset("200301010030_default","r")
wetland = Dataset("200301010030_wetland","r")
```

In [4]:

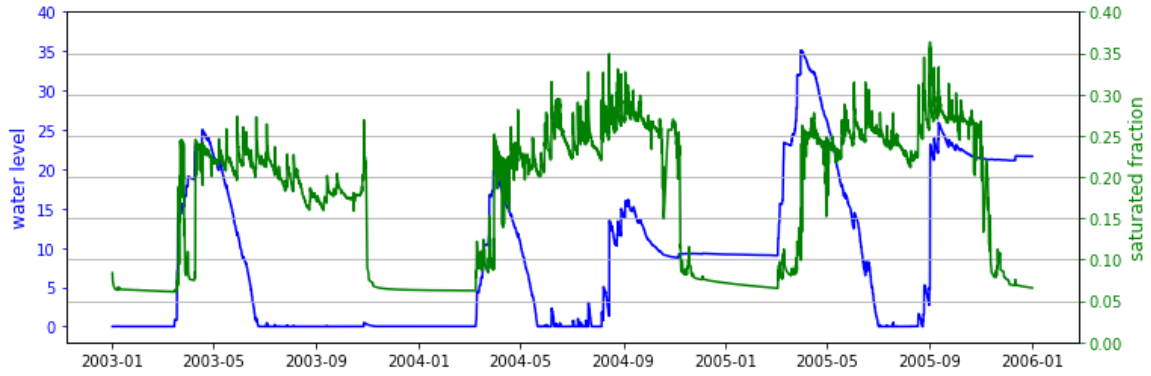
```
### First check two variables
### FSAT for saturated fraction of the grid cell (0~1)
### WSURF for water level in surface wetland, its capacity set in the parameter
table
### WSURF gets water from snowmelt and rainfall,
### filling wetlands in spring, sometimes drying out in summer
### get frozen and covered by snow in winter, untile next year

FSAT_wet  = wetland.variables["FSAT"][:]
WSURF_wet = wetland.variables["WSURF"][:]

fig, ax1 = plt.subplots(figsize=(12, 4))
ax2 = ax1.twinx()

ax1.plot(times,WSURF_wet[:,0,0],"b",label="water level")
ax1.set_ylabel("water level",fontsize=12,color="b")
ax1.set_yticks([0,5,10,15,20,25,30,35,40],color="b")
ax1.tick_params(axis='y', colors='b')
ax1.set_ylim(-2,40)

ax2.plot(times,FSAT_wet[:,0,0],"g",label="saturated fraction")
ax2.set_ylabel("saturated fraction",fontsize=12,color="g")
ax2.set_yticks([0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40])
ax2.tick_params(axis='y', colors='g')
ax2.set_ylim(0,0.4)
ax2.grid()
plt.show()
```



In [5]:

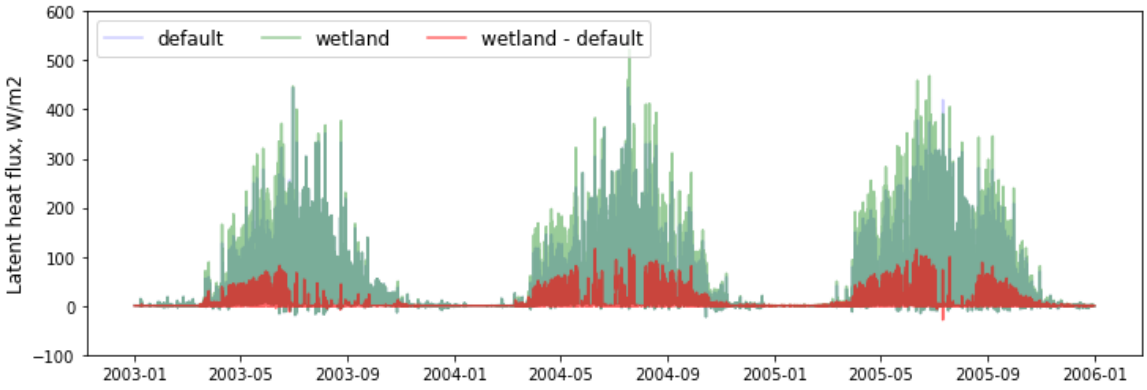
```
### Then check wetland contribution to latent heat flux
### the model gets higher LH when there is water in wetlands

LH_def = default.variables["LH"][:]
LH_wet = wetland.variables["LH"][:]
print (len(LH_wet))
plt.figure(figsize=(12,4))
plt.plot(times,LH_def[:,0,0],"b",label="default",alpha=0.2)
plt.plot(times,LH_wet[:,0,0],"g",label="wetland",alpha=0.4)
plt.plot(times,LH_wet[:,0,0]-LH_def[:,0,0],"r",label="wetland - default",alpha=
0.6)
plt.legend(ncol=3,loc="upper left",fontsize=12)
plt.ylabel("Latent heat flux, W/m2",fontsize=12)
plt.ylim(-100,600)
```

```
52608
```

Out[5]:

```
(-100.0, 600.0)
```



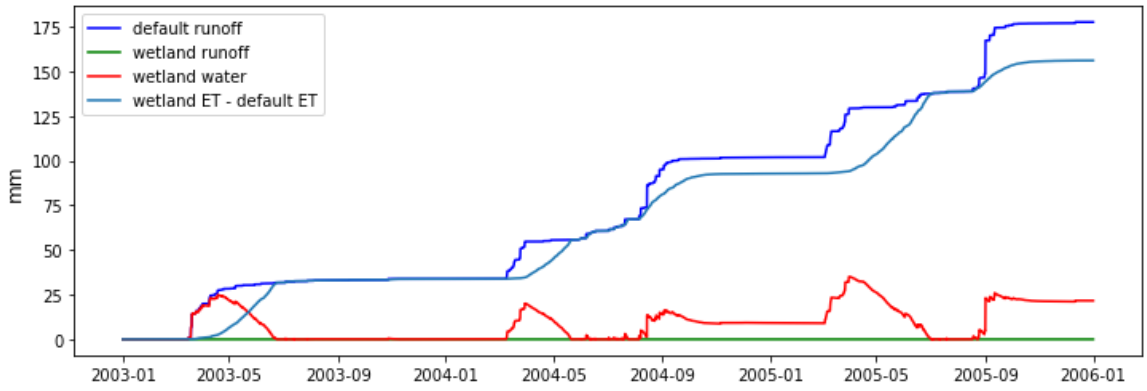
In [7]:

```
### Also check the water balance
### The default runoff will first go into wetland
### wetland water level rises
### water in wetlands contribute to higher ET
### RUNOFF_default = WSURF + (ET_wetland - ET_default) + RUNOFF_wetland
### in this case, water in wetlands does not exceed threshold, thus RUNOFF_wetla
nd = 0

WSURF_wet = wetland.variables["WSURF"][:]
ET_wet    = wetland.variables["EDIR"][:]+wetland.variables["ECAN"][:]+wetland.va
riables["ETRAN"][:]
RUN_wet   = wetland.variables["SFCRNOFF"][:]

RUN_def   = default.variables["SFCRNOFF"][:]
ET_def    = default.variables["EDIR"][:]+default.variables["ECAN"][:]+default.va
riables["ETRAN"][:]

plt.figure(figsize=(12,4))
plt.plot(times,RUN_def[:,0,0],"b",label="default runoff")
plt.plot(times,RUN_wet[:,0,0],"g",label="wetland runoff")
plt.plot(times,WSURF_wet[:,0,0],"r",label="wetland water")
plt.plot(times,np.cumsum(ET_wet[:,0,0]-ET_def[:,0,0])*1800.,label="wetland ET -
default ET")
plt.legend(loc="upper left")
plt.ylabel("mm",fontsize=12)
plt.show()
```



In [ ]: