Time Series modeling of the groundwater fluctuations in Deurne, The Netherlands

Time Series modeling meeting

Dutch Hydrological Society

3 November 2017

Raoul Collenteur, Frans Shaars, Artesia Water; Mark Bakker, TU Delft

Mijn doel: Wat is de bijdrage van de verdamping?

`https://github.com/pastas/pastas`

In [1]:

```
# Import the necessary packages
import pastas as ps
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use("seaborn")
%matplotlib notebook

import seaborn as sns
sns.set_context(context='talk')
```

Import Time Series

We start by importing the following time series:

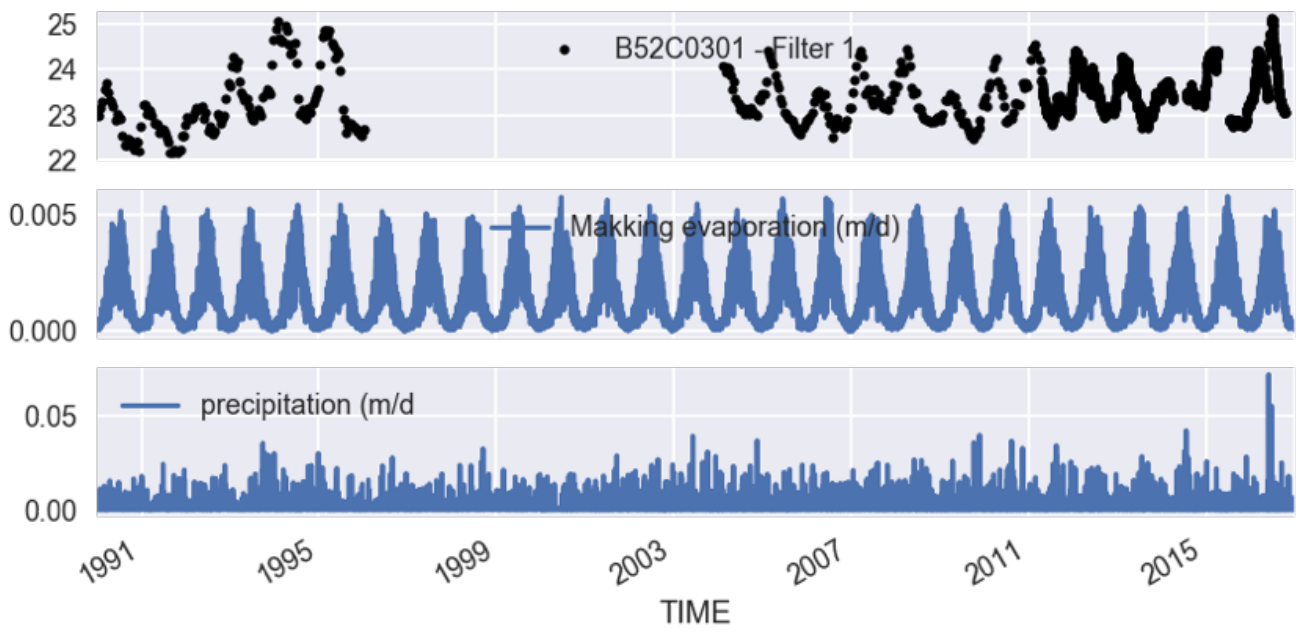
- The groundwater levels from B52C0301, filter 1 (http://dinoloket.nl (http://dinoloket.nl))
- Precipitation data from meteobase (http://meteobase.nl/ (http://meteobase.nl/))
- Makking evaporation data from meteobase (http://meteobase.nl/ (http://meteobase.nl/))

For comparison, here is a link to the TNO Grondwatertools model:
https://www.grondwatertools.nl/metran/html/showloc.html?wellid=B52C0301&screenid=001
(https://www.grondwatertools.nl/metran/html/showloc.html?wellid=B52C0301&screenid=001)

In [2]:

```
oseries = ps.read_dino("B52C0301001_1.csv")
evap= pd.read_csv("MAKdeurne.csv", squeeze=True, usecols=["TIME", "MAK_M_DAY"],
                  skipinitialspace=True, parse_dates=True, index_col="TIME")
evap = ps.TimeSeries(evap, kind="evap")
prec = pd.read_csv("meteobasedeurne.csv", squeeze=True, usecols=["TIME", "PREC_M
_DAY"],
                  skipinitialspace=True, parse_dates=True, index_col="TIME")
prec = ps.TimeSeries(prec/24, kind="prec", freq_original="H", freq="D")

fig, ax = plt.subplots(3, 1, figsize=[10, 4.8], sharex=True)
oseries.plot(ax=ax[0], x_compat=True, style="k.", label="B52C0301 - Filter 1", l
egend=True)
evap.plot(ax=ax[1], x_compat=True, legend=True, label="Makking evaporation (m/d)
")
prec.plot(ax=ax[2], x_compat=True, legend=True, label="precipitation (m/d")
```



Out[2]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x105259be0>
```

Create model and solve

Calibrated on data starting in 2004.

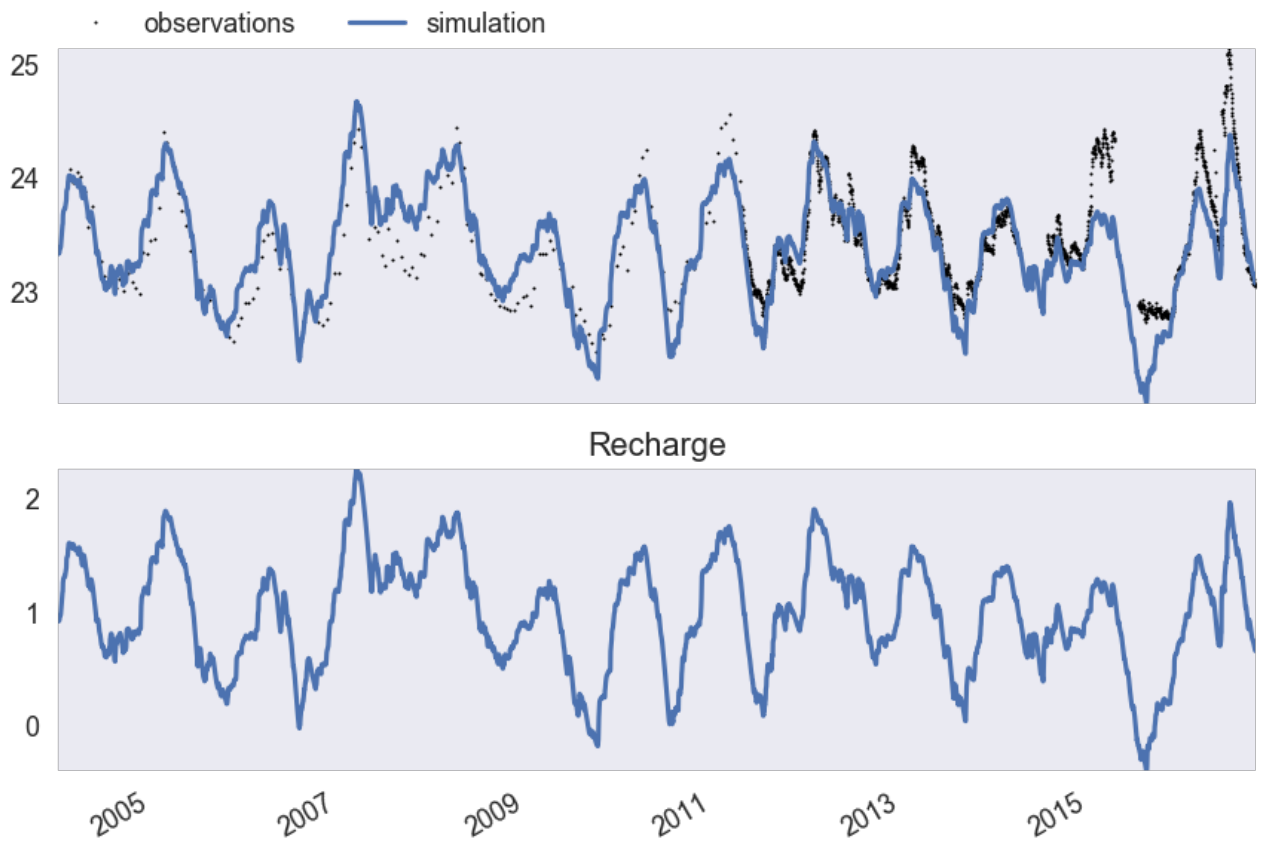
In [3]:

```
ml = ps.Model(oseries, name="Deurne")

sm = ps.StressModel2([prec, evap], ps.Gamma, "Recharge")
ml.add_stressmodel(sm)

ml.solve(freq="D", tmin="2004")
ml.plots.decomposition(tmin="2004")
```

```
[[Fit Statistics]]
    # function evals   = 45
    # data points      = 1975
    # variables        = 6
    chi-square         = 3.827
    reduced chi-square = 0.002
    Akaike info crit   = -12324.200
    Bayesian info crit = -12290.670
[[Variables]]
    Recharge_A:    1319.11457 +/- 141.2042 (10.70%) (init= 2156.159)
    Recharge_n:    1.15924580 +/- 0.014686 (1.27%) (init= 1)
    Recharge_a:    143.619839 +/- 14.91788 (10.39%) (init= 100)
    Recharge_f:   -0.81324649 +/- 0.073599 (9.05%) (init=-1)
    constant_d:    22.4122132 +/- 0.235428 (1.05%) (init= 23.45595)
    noise_alpha:   98.7908892 +/- 21.94905 (22.22%) (init= 14)
[[Correlations]] (unreported correlations are <  0.100)
    C(Recharge_f, constant_d)    = -0.858
    C(Recharge_A, Recharge_a)    =  0.795
    C(Recharge_A, constant_d)    = -0.712
    C(Recharge_n, Recharge_a)    = -0.427
    C(Recharge_a, constant_d)    = -0.372
    C(Recharge_n, Recharge_f)    =  0.351
    C(Recharge_A, Recharge_f)    =  0.323
    C(Recharge_n, constant_d)    = -0.290
    C(Recharge_f, noise_alpha)   = -0.190
    C(Recharge_a, noise_alpha)   =  0.175
    C(Recharge_A, noise_alpha)   =  0.169
```



Out[3]:

```
[<matplotlib.axes._subplots.AxesSubplot at 0x110965198>,
 <matplotlib.axes._subplots.AxesSubplot at 0x110a63cf8>]
```

Summary statistics

In [4]:

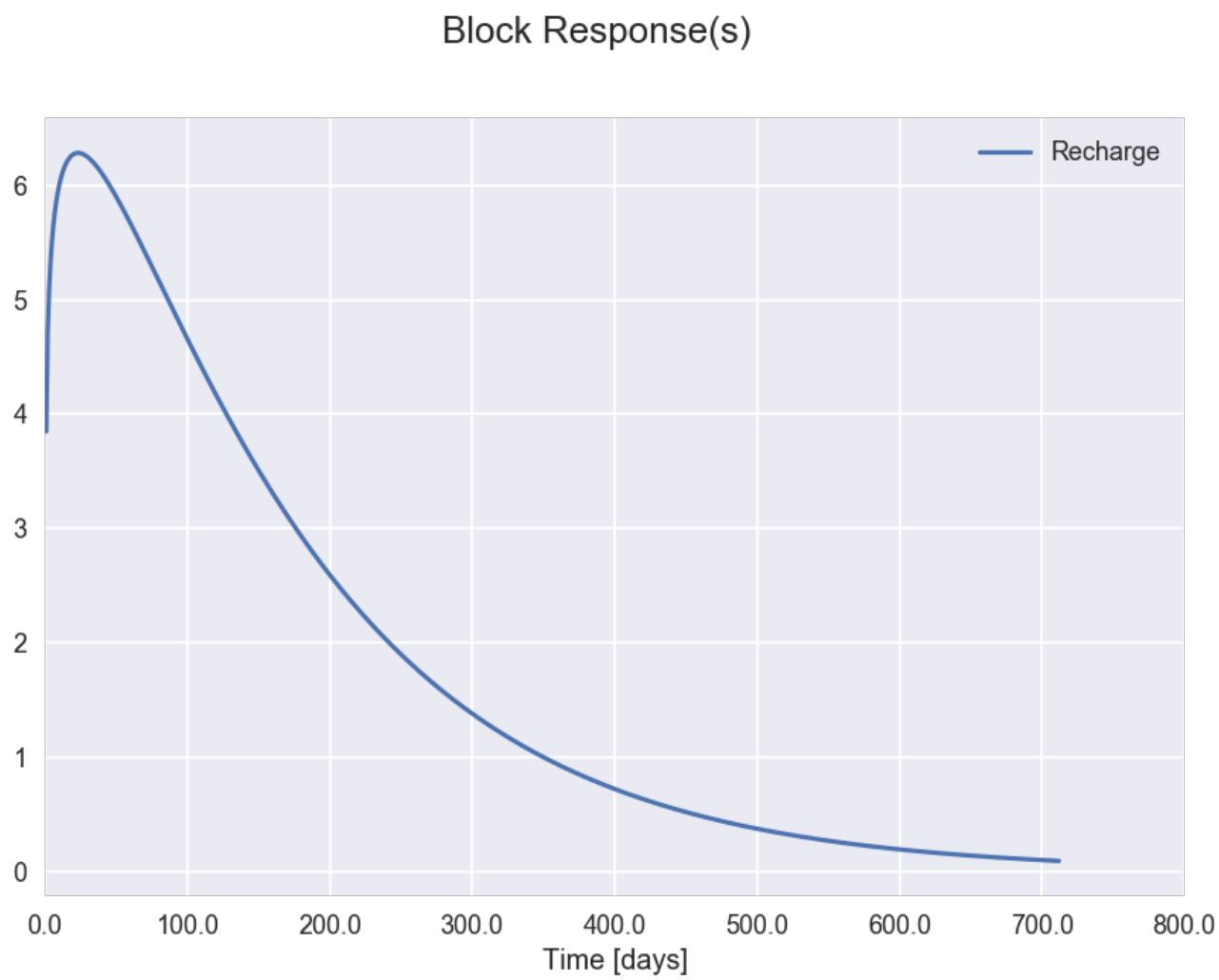
```
print(ml.stats.summary())
```

```
                                   Value
Statistic
Akaike Information Criterion    9.315707
Average Deviation               0.101059
Bayesian Information Criterion 42.845649
Explained variance percentage  71.644492
Pearson R^2                     0.853817
Root mean squared error         0.276105
```

Response function

In [5]:

```
ml.plots.block_response()
```



Out[5]:

```
[<matplotlib.axes._subplots.AxesSubplot at 0x1120472e8>]
```

Model with separate response function for precipitation and evaporation

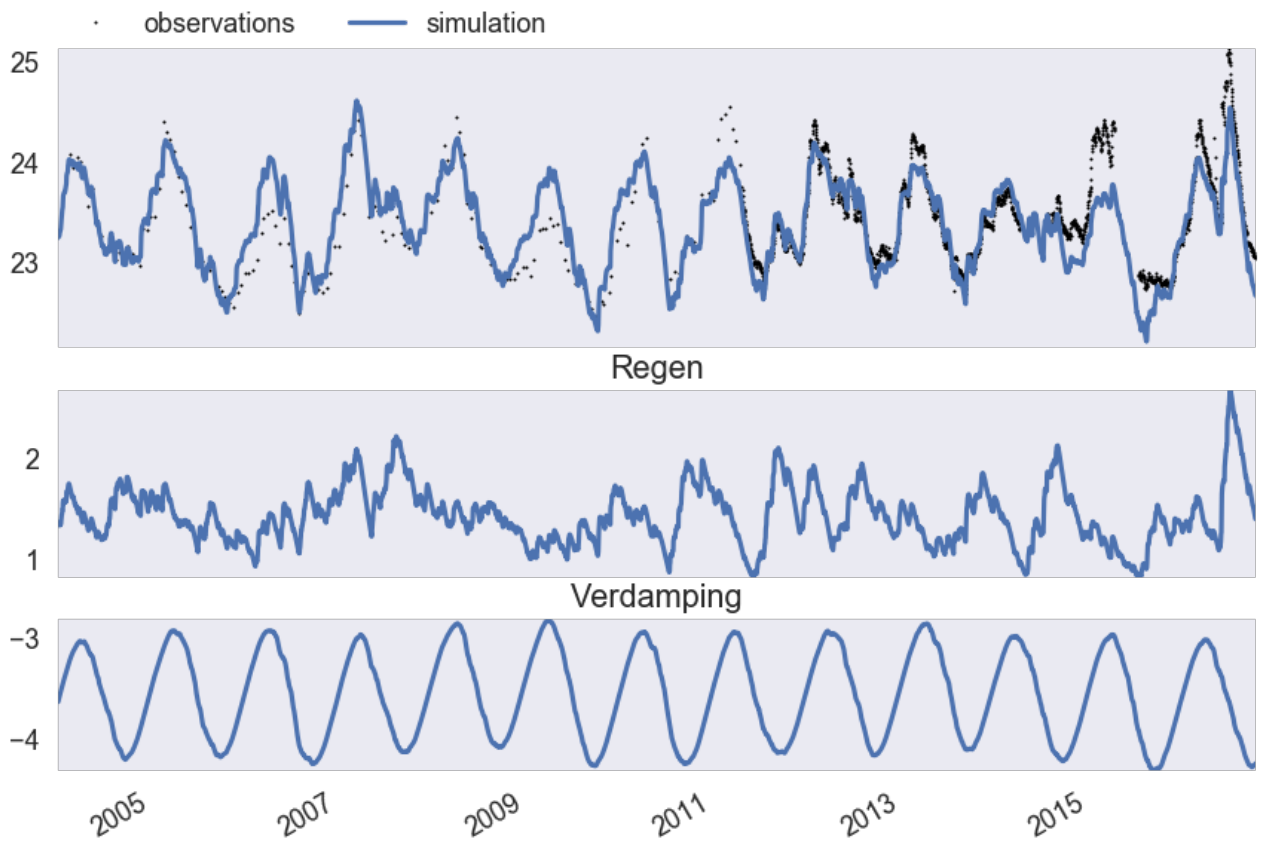
In [6]:

```
ml2 = ps.Model(oseries, name="Deurne")

smr = ps.StressModel(prec, ps.Gamma, "Regen")
ml2.add_stressmodel(smr)
smv = ps.StressModel(evap, ps.Gamma, "Verdamping", up=False)
ml2.add_stressmodel(smv)

ml2.solve(freq="D", tmin="2004")
ml2.plots.decomposition(tmin="2004")
```

```
[[Fit Statistics]]
    # function evals   = 48
    # data points      = 1975
    # variables        = 8
    chi-square         = 3.784
    reduced chi-square = 0.002
    Akaike info crit   = -12342.562
    Bayesian info crit = -12297.855
[[Variables]]
    Regen_A:         690.436903 +/- 88.84033 (12.87%) (init= 481.878)
    Regen_n:         1.19563935 +/- 0.015791 (1.32%) (init= 1)
    Regen_a:         70.3662236 +/- 9.708148 (13.80%) (init= 100)
    Verdamping_A:    2182.32764 +/- 524.9086 (24.05%) (init= 627.2627
)
    Verdamping_n:    1.13412044 +/- 0.074777 (6.59%) (init= 1)
    Verdamping_a:    265.039624 +/- 121.9755 (46.02%) (init= 100)
    constant_d:      25.5206367 +/- 0.863905 (3.39%) (init= 23.45595)
    noise_alpha:     78.0237346 +/- 15.82981 (20.29%) (init= 14)
[[Correlations]] (unreported correlations are <  0.100)
    C(Verdamping_A, constant_d)  =  0.974
    C(Verdamping_A, Verdamping_a)  =  0.906
    C(Regen_A, Regen_a)          =  0.892
    C(Verdamping_a, constant_d)  =  0.889
    C(Verdamping_n, Verdamping_a)  = -0.816
    C(Verdamping_n, constant_d)  = -0.559
    C(Verdamping_A, Verdamping_n)  = -0.554
    C(Regen_n, Regen_a)          = -0.481
    C(Regen_A, Verdamping_A)     =  0.224
    C(Regen_A, Verdamping_a)     =  0.175
    C(Regen_a, Verdamping_A)     =  0.170
    C(Regen_a, Verdamping_a)     =  0.148
    C(Regen_A, Regen_n)          = -0.137
```



Out[6]:

```
[<matplotlib.axes._subplots.AxesSubplot at 0x111100b70>,
 <matplotlib.axes._subplots.AxesSubplot at 0x11416b860>,
 <matplotlib.axes._subplots.AxesSubplot at 0x1141becf8>]
```

Summary statistics

Explained variance percentage has gone up from 71 to 79.

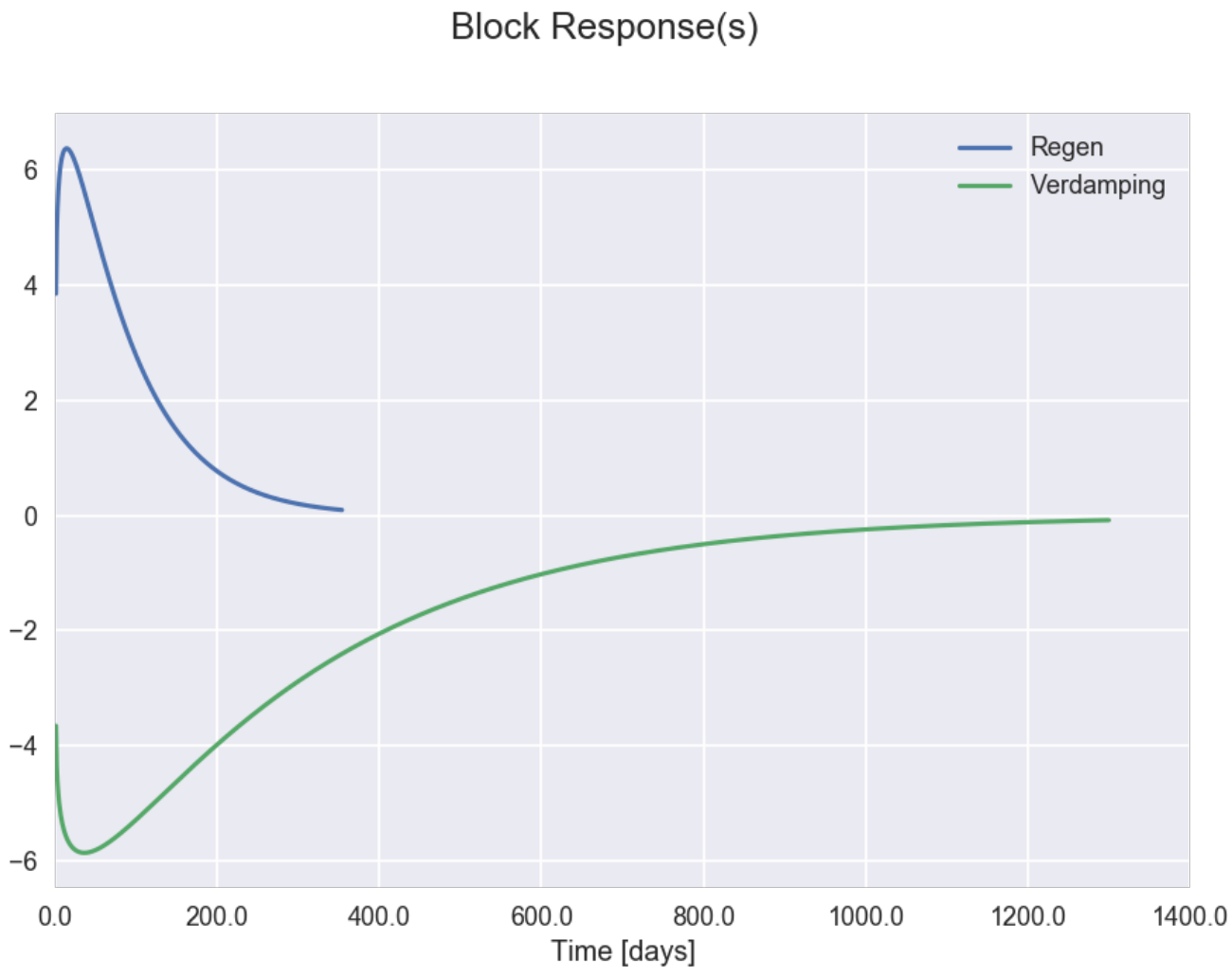
In [7]:

```
print('one respose function')
print(ml.stats.summary())
print('two response functions')
print(ml2.stats.summary())
```

```
one respose function
                                  Value
Statistic
Akaike Information Criterion     9.315707
Average Deviation                0.101059
Bayesian Information Criterion  42.845649
Explained variance percentage   71.644492
Pearson R^2                      0.853817
Root mean squared error          0.276105
two response functions
                                  Value
Statistic
Akaike Information Criterion    13.338352
Average Deviation                0.097509
Bayesian Information Criterion  58.044942
Explained variance percentage   79.422182
Pearson R^2                      0.891814
Root mean squared error          0.239625
```

In [8]:

```
ml2.plots.block_response()
```



Out[8]:

```
[<matplotlib.axes._subplots.AxesSubplot at 0x114416710>]
```

Evaporation factor for two response functions

Very high apparent evaporation factor. Probably not reasonable.

In [9]:

```
f = ml2.parameters.loc["Verdamping_A", "optimal"] / ml2.parameters.loc["Regen_A"
, "optimal"]
print('Apparent evaporation factor with two response functions:', f)
```

```
Apparent evaporation factor with two response functions: 3.160792291
1
```

Use all observation data

Use of all data does not give significantly better fit

In [10]:

```
ml.solve(report=False)
```

In [11]:

```
print(ml.stats.summary())
```

```
                                  Value
Statistic
Akaike Information Criterion     8.845344
Average Deviation                0.130872
Bayesian Information Criterion  42.803202
Explained variance percentage   71.393654
Pearson R^2                      0.849981
Root mean squared error          0.301394
```

In [12]:

```
ml2.solve(report=False)
```

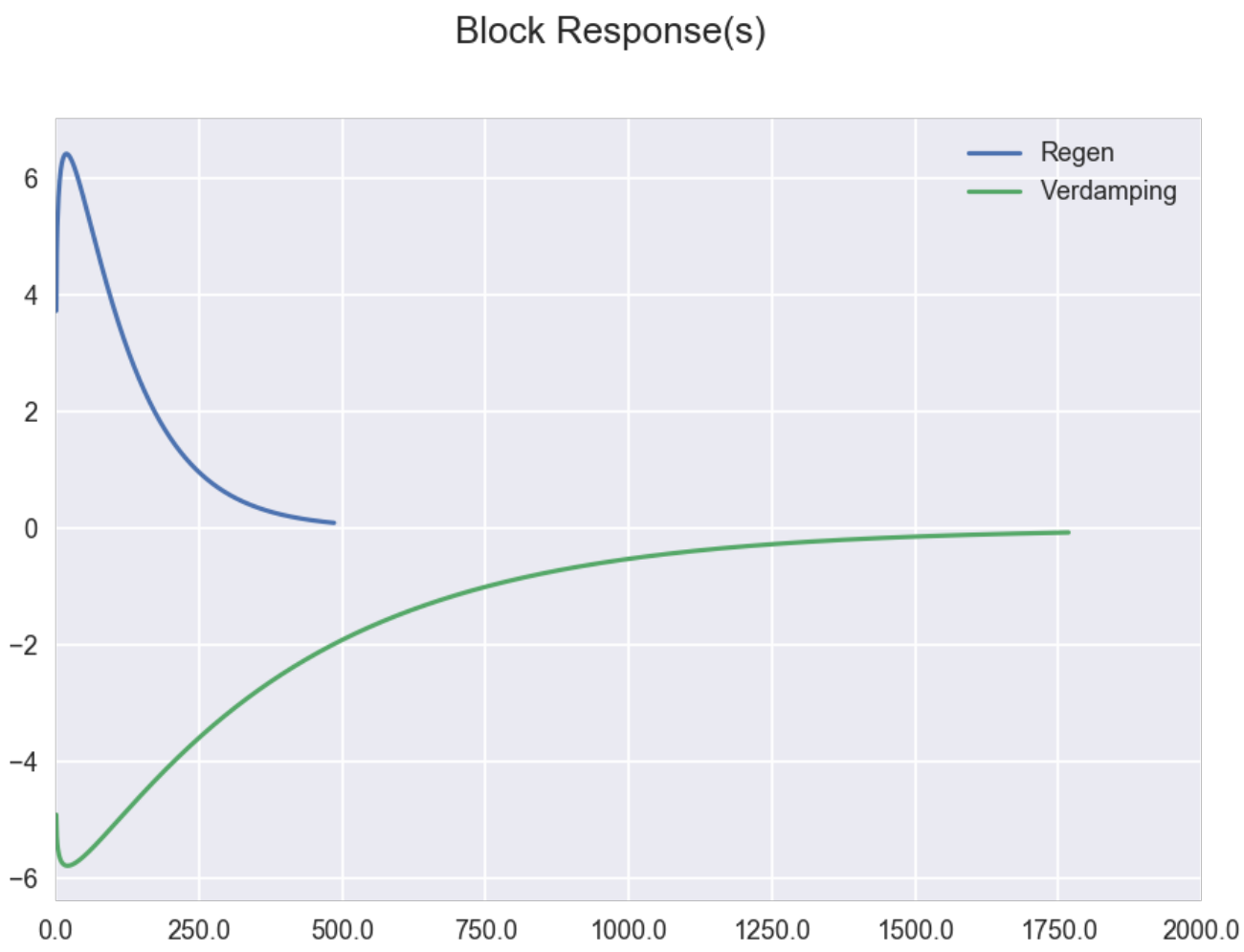
In [13]:

```
print(ml2.stats.summary())
```

```
                                  Value
Statistic
Akaike Information Criterion    12.147058
Average Deviation                0.189157
Bayesian Information Criterion  57.864478
Explained variance percentage   71.118329
Pearson R^2                      0.846001
Root mean squared error          0.335392
```

In [14]:

```
ml2.plots.block_response()
```



Out[14]:

```
[<matplotlib.axes._subplots.AxesSubplot at 0x111fe4c18>]
```

In [15]:

```
f = ml2.parameters.loc["Verdamping_A", "optimal"] / ml2.parameters.loc["Regen_A"
, "optimal"]
print('Apparent evaporation factor with two response functions using all data:',
f)
```

```
Apparent evaporation factor with two response functions using all da
ta: 2.73919762522
```

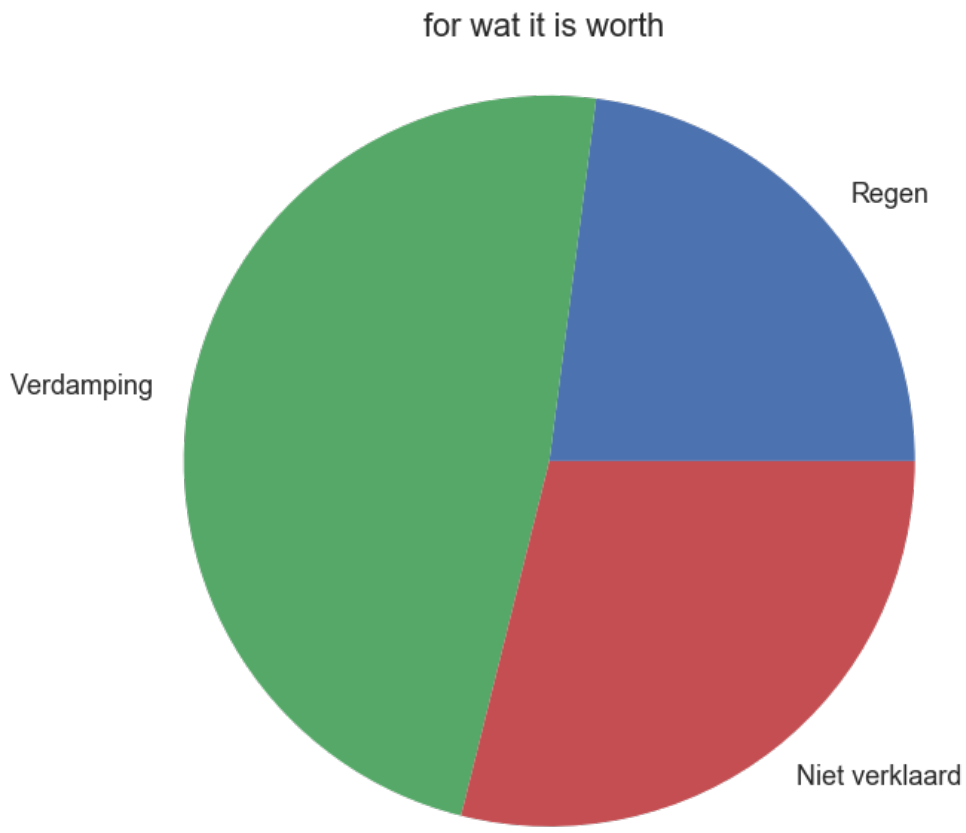
In [16]:

```
def make_pie(ml, tmin=None, tmax=None, **kwargs):
    import numpy as np
    import matplotlib.pyplot as plt
    plt.figure()
    sum = 0.0
    frac = []
    if tmin is None:
        tmin = ml.settings["tmin"]
    if tmax is None:
        tmax = ml.settings["tmax"]
    for name in ml.stressmodels.keys():
        a = np.abs(ml.get_contribution(name)[tmin:tmax]).sum()
        sum += a
        frac.append(a)

    evp = ml.stats.evp(tmin=tmin)/100
    frac = np.array(frac) / sum * evp
    frac = frac.tolist()
    frac.append(1-evp)
    frac = np.array(frac)
    import matplotlib.pyplot as plt
    labels = list(ml.stressmodels.keys())
    labels.append("Niet verklaard")
    plt.pie(frac, labels=labels, **kwargs)
    plt.axis('scaled')



make_pie(ml2)
plt.title('for wat it is worth')
```



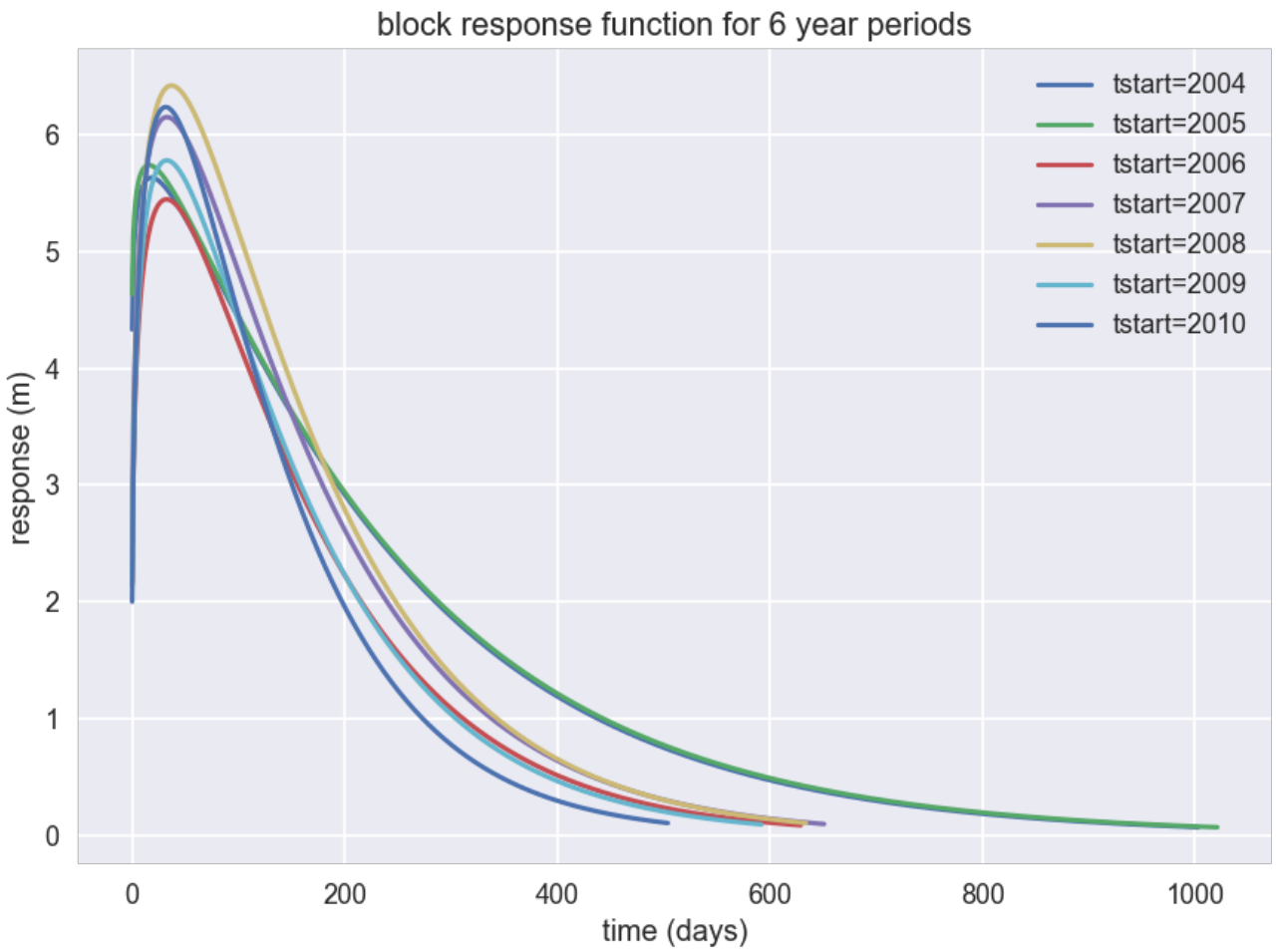
Out[16]:

```
<matplotlib.text.Text at 0x11529fda0>
```

6-year moving window

In [17]:

```
plt.figure()
fevap = []
for t in range(2004, 2011):
    ml.solve(tmin=str(t), tmax=str(t+6), report=False)
    f = ml.stressmodels['Recharge'].rfunc
    plt.plot(f.block(ml.parameters.iloc[:3, 2].values), label='tstart=' + str(t)
)
    fevap.append(ml.parameters.loc['Recharge_f', 'optimal'])
plt.legend()
plt.title('block response function for 6 year periods')
plt.xlabel('time (days)')
plt.ylabel('response (m)')
```

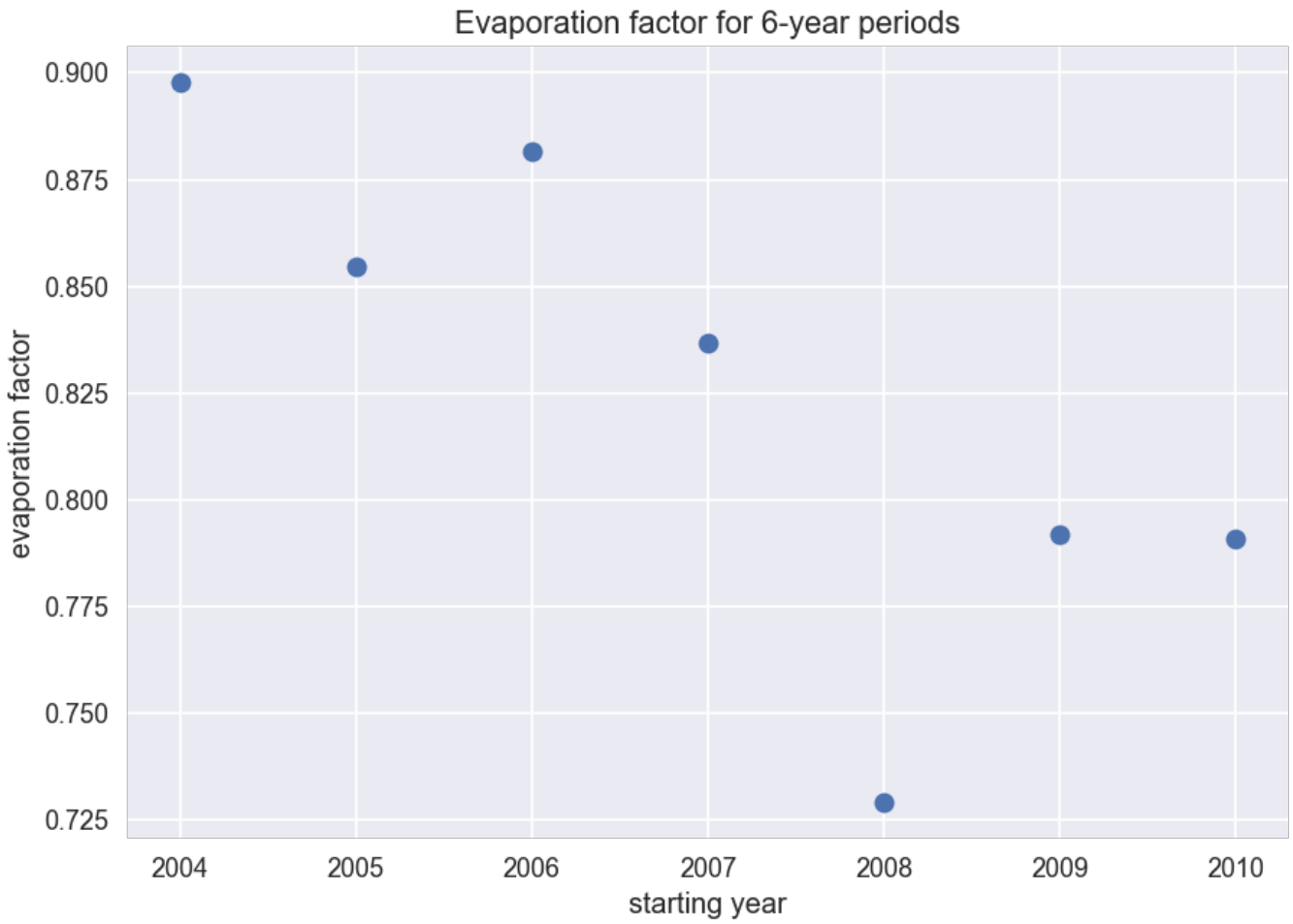


Out[17]:

<matplotlib.text.Text at 0x11239d7f0>

In [18]:

```
plt.figure()
plt.plot(range(2004, 2011), -np.array(fevap), 'o', markersize=10)
plt.title('Evaporation factor for 6-year periods')
plt.xlabel('starting year')
plt.ylabel('evaporation factor')
```



Out[18]:

```
<matplotlib.text.Text at 0x11521db70>
```