

Gevisualiseerde observatiepunt-resultaten D-HYDRO

Voor de regenwaterbuffer in Genbroek

Via dit notebook laten we de resultaten van de gevoeligheidsberekeningen zien. Daarnaast demonstreren we hoe modelresultaten kunnen worden ingelezen en gevisualiseerd.

De code voor de opbouw van het model is opgedeeld in logische stappen. Elke stap wordt ingeleid met een toelichting. De toelichting kan starten en eindigen met regels die beginnen met een *. Deze regels geven aan dat er een actie mogelijk of nodig is om instellingen te veranderen.

Onderstaand blok met code is bedoeld om de benodigde packages in te laden en hier hoeft niks aan veranderd te worden.

```
In [1]: import rasterio
import netCDF4
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon as PolygonPatch
from shapely.geometry import Polygon, LineString
from hkvpy import plotting
import os
import sys
sys.path.append('../Python')

%matplotlib inline
```

Laden functies voor het inlezen netcdf bestanden

Hier hoeft in principe niets aan te worden veranderd.

```

In [2]: class Results:
    def __init__(self, ncfile):
        self.file = ncfile
        self.open()
        self.variables = list(self.nc.variables.keys())
        self.close()

        self.network = Network(self)

    def open(self):
        self.nc = netCDF4.Dataset(self.file)

    def close(self):
        self.nc.close()

    def get_data(self, variable):
        if variable not in self.variables:
            raise KeyError('{} not in variables: {}'.format(variable, ', '.join(self.variables)))
        self.open()
        values = self.nc.variables[variable][:]
        self.close()
        return values

    def get_mesh2d_faces(self, as_polygon=False, as_patch=False):
        self.open()
        cells = np.dstack([self.nc.variables['mesh2d_face_x_bnd'], self.nc.variables['mesh2d_face_y_bnd']])
        nodata = (cells == self.nc.variables['mesh2d_face_x_bnd']._FillValue)
        cells = [cell[idx] for idx, cell in zip(nodata.sum(axis=-1) != 2, cells.data)]
        self.close()
        if as_polygon:
            cells = [Polygon(cell) for cell in cells]
        elif as_patch:
            cells = [PolygonPatch(cell, closed=True) for cell in cells]

        return cells

    def get_mesh2d_face_centers(self):
        self.open()
        coords = np.c_[self.nc.variables['mesh2d_face_x'][:].data, self.nc.variables['mesh2d_face_y'][:].data]
        self.close()
        return coords

    def get_max(self, variable, dimension=0):
        values = self.get_data(variable).data.max(axis=dimension)
        return values

    def get_network_count(self):
        self.open()
        counts = {
            'mesh1d': results.nc.dimensions['nmesh1d_node'].size,
            'mesh2d': results.nc.dimensions['nmesh2d_face'].size,
            'links1d2d': results.nc.dimensions['nlink1d2d_connections'].size

```

```

    }
    self.close()
    return counts

class Network:

    def __init__(self, parent):
        self.get_data = parent.get_data

        self.loaded_1d = False

    def load_1d_network(self):

        # Collect 1d network branches
        geomx = self.get_data('network_geom_x')
        geomy = self.get_data('network_geom_y')

        i = 0
        self.branches = []
        for nnodes in self.get_data('network_node_count'):
            self.branches.append(LineString([(geomx[j], geomy[j]) for j in range(i, i + nnodes)]))
            i += nnodes

        branchids = self.get_data('mesh1d_nodes_branch_id')
        offsets = self.get_data('mesh1d_nodes_branch_offset')

        crds = []
        for i in range(len(branchids)):
            crds.append(self.branches[branchids[i]].interpolate(offsets[i]).coordinates[0])

        # Lees 1D Links unit
        self.nodes1d = np.array(crds)
        self.edges1d = self.nodes1d[self.get_data('mesh1d_edge_nodes').data - 1]

        self.loaded_1d = True

    def get_nearest(self, xy, type='node', one_based=False):

        # Load the network if it is not loaded yet
        if not self.loaded_1d:
            self.load_1d_network()

        # Get nearest node
        if type == 'node':
            imin = np.argmin(np.hypot(*(self.nodes1d - xy).T)) + int(one_based)

        elif type == 'edge':
            dist = np.hypot(*(self.edges1d[:, 0, :] - xy).T) + np.hypot(*(self.edges1d[:, 1, :] - xy).T)
            imin = np.argmin(dist) + int(one_based)

        else:
            raise ValueError(f'Type "{type}" not recognized. Specify node or e

```

```
dge.')
```

```
return imin
```

D-HYDRO output bestanden verzamelen *

De code hieronder zoekt in een bestandslocatie naar D-HYDRO uitvoerbestanden. Onderaan wordt de lijst met gevonden bestanden geprint door de code.

Het runnen van D-HYDRO modellen geeft twee soorten resultaten. Beide resultaatbestanden zijn in NetCDF formaat (.nc). De gegevens van de observatiepunten worden weggeschreven in het bestand dat eindigt op '_his.nc', die hieronder ingelezen worden. Het tweede bestand eindigt op '_map.nc', dit is een rasterbestand met data per pixel.

*'outputdir' kan aangepast worden naar de eigen bestandslocatie van de D-hydro mode lresultaten.

*'skipmap' is bedoeld om bepaalde mappen over te slaan waarvan de resultaten niet m eegenomen moeten worden.


```

In [4]: outputdir = 'd:\PR\PR4139.10 regenwaterbuffers Limburg\output_netcdf\output_mo
dellen' #locatie his.nc bestanden
skipmap = 'oudemodellen' #verander wanneer in een bepaalde map niet gezocht mag
worden
def find_his(parent, skipmap=' ', ncfiles=[], ncfile_locations = []):
    #iteratie over alle bestanden in de opgegeven outputdir mao

    for file_name in os.listdir(parent):
        if file_name.endswith("his.nc"): #als een andere extensie moet worden
ingelesen worden kan dat hier aangepast worden.
            ncfiles.append(file_name)
            ncfile_locations.append(parent+'/'+file_name)

        else:
            if file_name != skipmap:
                current_path = "".join((parent, "/", file_name))
                if os.path.isdir(current_path):
                    find_his(current_path, skipmap) # wanneer een onderliggend
e map bekeken word, functie opnieuw

    return ncfiles, ncfile_locations

ncfiles, ncfile_locations = find_his(outputdir, skipmap) # verkrijg his besta
nden door de find_his functie, terugkoppeling is lijst met gevonden bestandsna
men.

for i, ncfile in enumerate(ncfiles):
    print('Loc:', i, ' Voor bestandsnaam: ', ncfile)

```

```

Loc: 0 Voor bestandsnaam: buffer1m_r0_b1bres_his.nc
Loc: 1 Voor bestandsnaam: buffer1m_r0_b1refinedv3_his.nc
Loc: 2 Voor bestandsnaam: buffer1m_r0_b1refinedv4_025_his.nc
Loc: 3 Voor bestandsnaam: buffer1m_r0_b1refinedv4_his.nc
Loc: 4 Voor bestandsnaam: buffer1m_r0_b1refined_his.nc
Loc: 5 Voor bestandsnaam: buffer1m_r0_b1ruw05_his.nc
Loc: 6 Voor bestandsnaam: buffer1m_r0_b1ruw2_his.nc
Loc: 7 Voor bestandsnaam: buffer1m_r0_b1_his.nc
Loc: 8 Voor bestandsnaam: buffer1m_r0_b3_his.nc
Loc: 9 Voor bestandsnaam: buffer1m_r60_b1_his.nc
Loc: 10 Voor bestandsnaam: buffer2m_r0_b1_his.nc
Loc: 11 Voor bestandsnaam: buffer5m_r0_b1_his.nc

```

Inzien van de bresgroei resultaten

Via de code in het onderstaande codeblok kan de bresgroei data worden ingelezen en bekeken. De variabelen die bekeken kunnen worden zijn geprint na de code. Als voorbeeld is hier de breedte van de doorbraak geplot over de tijd. In onderstaand figuur is te zien dat vanaf minuut 80 de breedte van de bres niet meer toeneemt.

*Bij de variabele 'res' wordt nc_location[0] opgegeven. Verander het getal met de locatienummer uit de lijst hierboven die hoort bij het bestand waar een bresgroei is gesimuleerd.

*Het aantal tijdstappen dat gewenst is voor visualisatie en het aantal minuten per weggeschreven tijdstap kan worden aangepast.

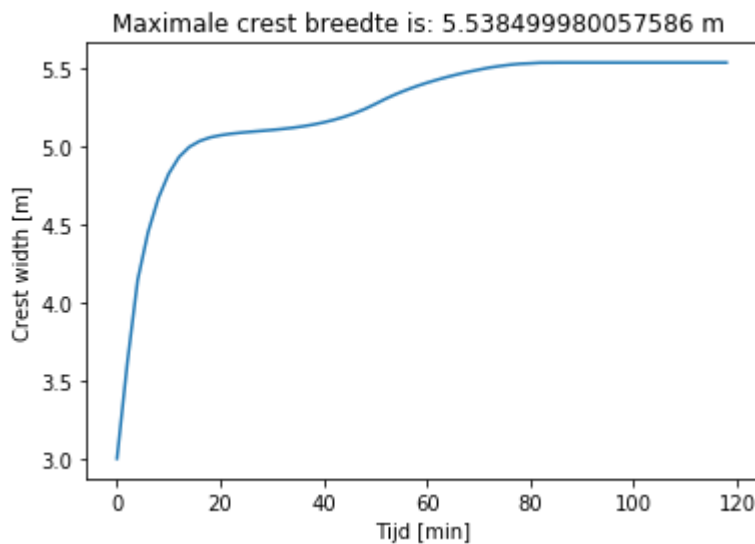
*Is er geen bresgroei gesimuleerd, dan hoeft het onderstaande codeblok niet te worden uitgevoerd.

```
In [5]: #uitlezen bres gegevens wanneer een bres is berekend
#print(ncfile_locations[0])
res = Results(ncfile_locations[0])
aantal_stappen = 60 #Geef hier het aantal tijdstappen op om te bekijken
min_per_stap = 2 # geef hier de wegschrijf interval op voor het his besta
nd om een kloppende x-as in minuten te hebben.
plt.plot(np.arange(0,aantal_stappen*min_per_stap,2), (res.get_data('dambreak_c
rest_width')[:aantal_stappen])) #tijdstap is elke 2 minuten bij deze modellen
plt.ylabel('Crest width [m]')
plt.xlabel('Tijd [min]')
plt.title('Maximale crest breedte is: {} m '.format(np.max(res.get_data('dambr
eak_crest_width')[::]),'m'))

#Opgeslagen parameters bij een bresgroei:
print('Opgeslagen bres parameters die bekeken kunnen worden zijn : \n', res.va
riables[26:])
```

Opgeslagen bres parameters die bekeken kunnen worden zijn :

```
['dambreak_id', 'dambreak_slup', 'dambreak_slidn', 'dambreak_discharge', 'dam
break_cumulative_discharge', 'dambreak_breach_width_time_derivative', 'dambre
ak_water_level_jump', 'dambreak_normal_velocity', 'dambreak_structure_head',
'dambreak_flow_area', 'dambreak_crest_level', 'dambreak_crest_width', 'time',
'timestep']
```



Aanmaken van plot functie

Functie voor het visualiseren van 2 of 3 verschillende modellen in 1 figuur. Bij gebruik voor 2 modelvergelijkingen geef voor loc3 het getal 999 op. De loc nummers voor de bijbehorende bestanden zijn hierboven geprint bij de bestanden zoeken.

```
Vb: plot_his_results(loc1=7, loc2=0, loc3=999, save = 'Yes', name_output='verg_1m_met_bres')
```

Hiermee wordt een plot gemaakt met in dit geval de 1m resolutie met de 1m resolutie bresgroei resultaten in 1 figuur en slaat het op met de naam 'verg_1m_met_bres'.

```

In [ ]: def plot_his_results(loc1, loc2, loc3, save='No', name_output='plot_his'):

    plotting.set_rcparams()

    files = [file for file in os.listdir(outputdir) if file.endswith('his.nc'
)]
    Numb_obs_points = Results(ncfile_locations[0]).get_data('bedlevel').count
    () #retrieve number of obs points, bedlevel is not variable over time therefore
    a good measure for the amount of points
    print('Number of observation points in his.nc file is: ',Numb_obs_points)
    lw=1.0

    if loc3 ==999:
        files = [ncfiles[loc1], ncfiles[loc2]]
        file_locations = [ncfile_locations[loc1], ncfile_locations[loc2]]

    if loc3 !=999:
        files = [ncfiles[loc1], ncfiles[loc2], ncfiles[loc3]]
        file_locations = [ncfile_locations[loc1], ncfile_locations[loc2], ncfile_locations[loc3]]

    for i in range(Numb_obs_points):
        fig = plt.figure(figsize=(15,9))
        axs = [
            plt.subplot2grid((2, 3), (0, 0), colspan=2),
            plt.subplot2grid((2, 3), (1, 0), colspan=2),
            plt.subplot2grid((2, 3), (0, 2), colspan=1),
            plt.subplot2grid((2, 3), (1, 2), colspan=1)
        ]
        for index, file in enumerate(files):
            res = Results(file_locations[index])
            time = res.get_data('time') / 3600.

            waterlevel = res.get_data('waterlevel')
            bedlevel = res.get_data('bedlevel')
            waterdepth = waterlevel - bedlevel

            x_vel= res.get_data('x_velocity')
            y_vel= res.get_data('y_velocity')
            v = (x_vel**2 + y_vel**2)**0.5

            label = file[0:-7]
            if waterdepth[:,i].any() == 0:
                label = label+'_NO_water_on_point'
            axs[0].plot(time, waterdepth[:,i], label=label, lw=lw)
            axs[2].plot(time, waterdepth[:,i], label=label, lw=lw)
            axs[1].plot(time, v[:,i], label=label, lw=lw)
            axs[3].plot(time, v[:,i], label=label, lw=lw)

            axs[0].legend()
            axs[1].legend()

        for ii, ax in enumerate(axs):
            if ii >1:
                ax.set_xlim(0, 4)

```

```

else:
    ax.set_xlim(0, 24)
    ax.grid()
    ylim = list(ax.get_ylim())
    ax.set_ylim(ylim)
    plotting.simplify_axes(ax=ax, rescale=False)
    ax.set_xlabel('Tijd [uren]')

    axs[0].set_title('Vergelijking {}'.format(name_output)+str(i+1)+ ' voo
r waterstanden ')
    axs[0].set_ylabel('Waterdiepte [m]')
    axs[1].set_title('Vergelijking {}'.format(name_output)+str(i+1)+ ' voo
r stroomsnelheden ')
    axs[1].set_ylabel('Stroomsnelheid [m/s]')

fig.tight_layout()
if save=='Yes':
    print('saved obs point ', str(i+1))
    fig.savefig('../plots/'+name_output+str(i+1)+'.png', dpi=220, bbox
_inches='tight') #toggle on for saving plots in folder

```

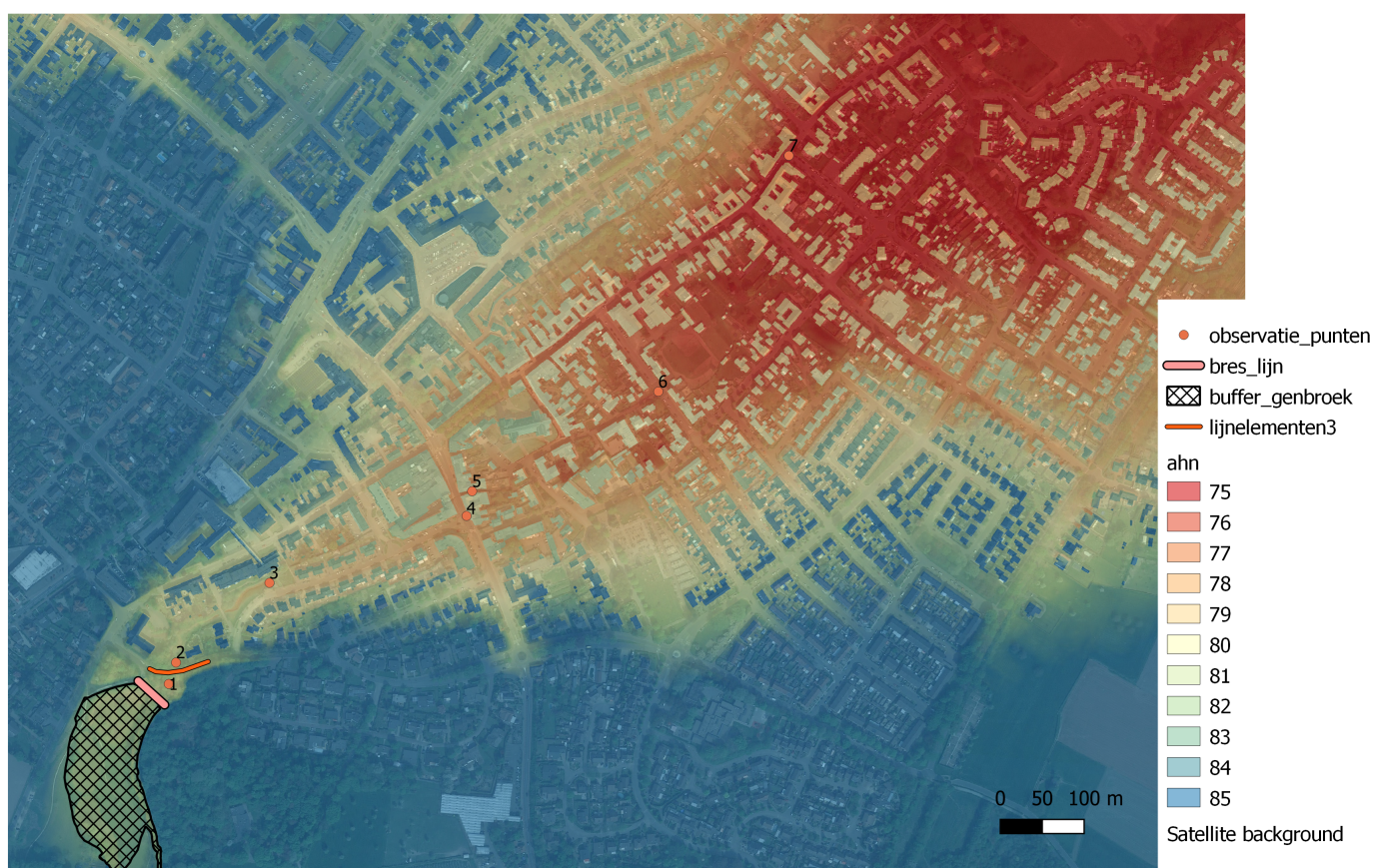
Resultaten gevoeligheidsanalyse

In het vervolg van het notebook worden de gevoeligheden van de modeluitkomsten behandeld voor:

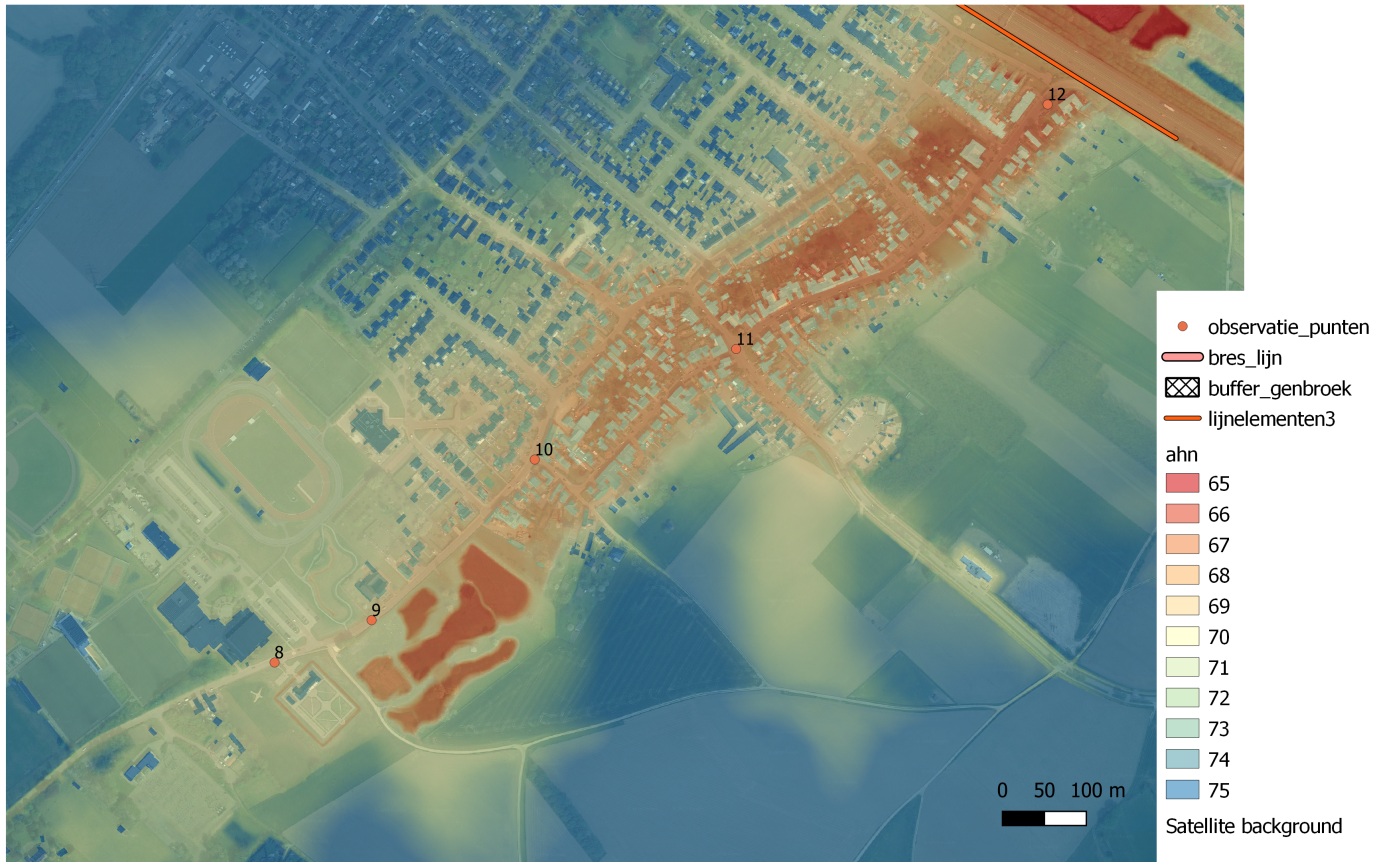
- 1) de resolutie
- 2) de ruwheid van het overstroomde maaiveld
- 3) de gridrotatie
- 4) de bres

De figuur hieronder laat de locaties van de observatiepunten zien:

Observatiepunt overzicht Regenwaterbuffer Genbroek



Observatiepunt overzicht Regenwaterbuffer Genbroek



1) Invloed van resolutie:

Hieronder staan de resultaten voor de resolutie van het rekenrooster. Het doel is te komen tot een resolutie waarmee de inundaties voldoende nauwkeurig worden gesimuleerd binnen een acceptabele rekentijd. Gestart is met een resolutie van 5 m, gevolgd door een resolutie van 2 m en 1 m. De rekentijden voor deze resoluties zijn als volgt:

10 min. voor een resolutie van 5 m
50 min. voor een resolutie van 2 m
120 min. voor een resolutie van 1 m

Naast rekentijden zijn ook de resultaten van de berekeningen van belang. Als bij het verkleinen van de resolutie voldoende kleine verschillen worden gevonden in vergelijking met de vorige resolutie, kan worden gestopt met het verkleinen van de resolutie. Een aandachtspunt daarbij is de grote variatie van de hoogte in stedelijk gebied. Bij een lage resolutie kunnen er door het middelen van de hoogte, knelpunten ontstaan die er in werkelijkheid (en bij een hogere resolutie) niet zijn. Dit gebeurt vooral bij smalle doorgangen tussen huizen. Cellen die voor een deel op een huis liggen met een relatief hoge ligging, komen daardoor ook relatief hoog te liggen en zorgen voor knelpunten in de afwatering.

Hieronder is per observatiepunt en voor de beschouwde resoluties elke keer een figuur van de waterstanden en daaronder een figuur met de stroomsnelheden opgenomen. De verschillen in waterstanden bij de verschillende resoluties worden zichtbaar vanaf observatiepunt 2. De 5 meter resolutie heeft over het algemeen lagere waterdieptes. Dit lijkt voornamelijk te komen doordat hoogtevariëaties worden uitgemiddeld. Dit betekent dat er minder obstakels in de weg staan die bij een 2 m en de 1 m resolutie wel aanwezig zijn.

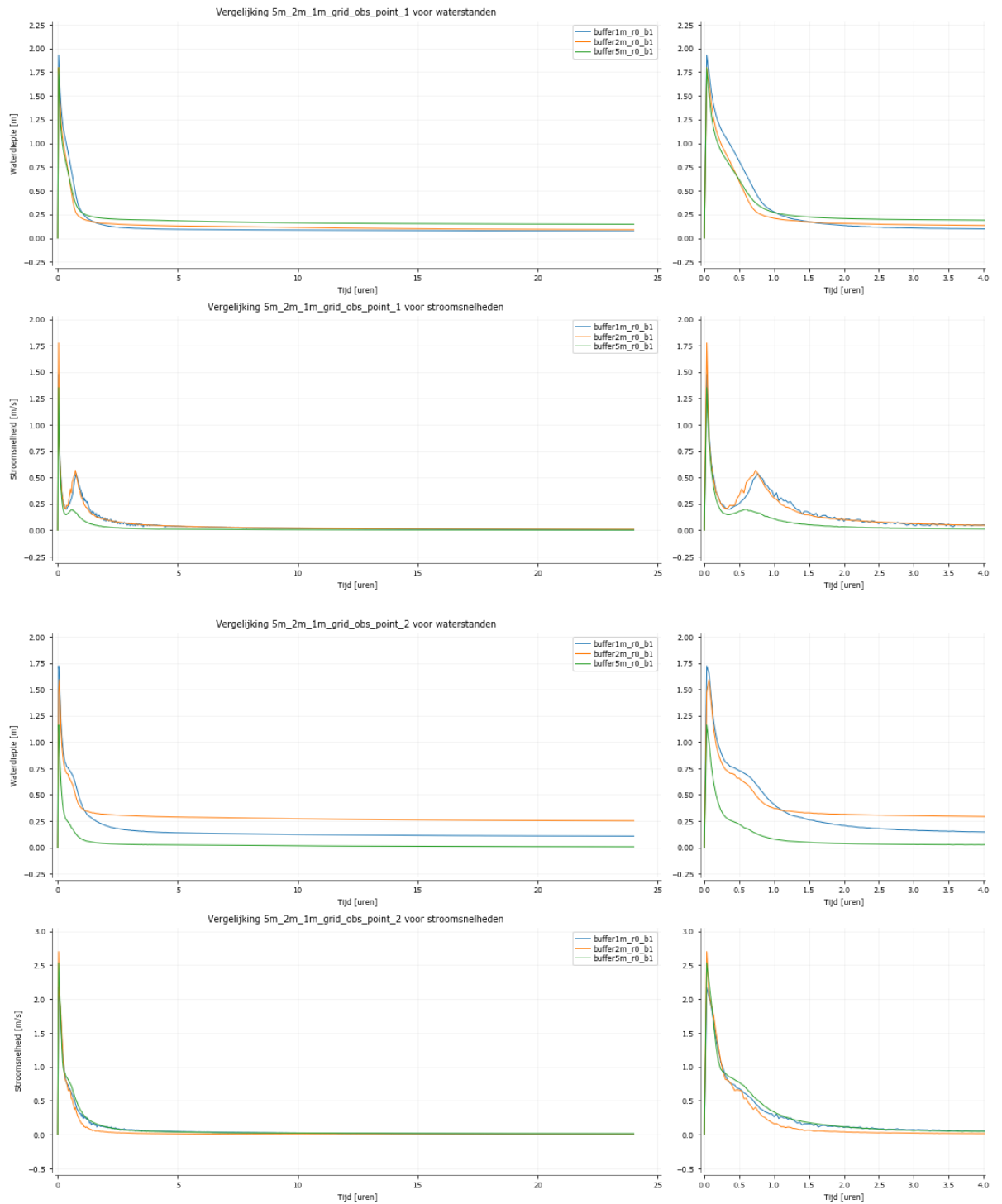
Observatiepunt 5 is gelegen in een smalle steeg van ongeveer 3 meter breed. Een groot deel van het water stroomt door deze steeg. In het 5 meter resolutiemodel is de maaiveldhoogte in de steeg te hoog, waardoor er geen stroming mogelijk is en het water een andere route neemt dan bij de modellen met een 1 m of een 2 m resolutie. Vlak voor punt 11 wordt al het water in het 5 m resolutiemodel tegengehouden waardoor de stroming hier stagneert.

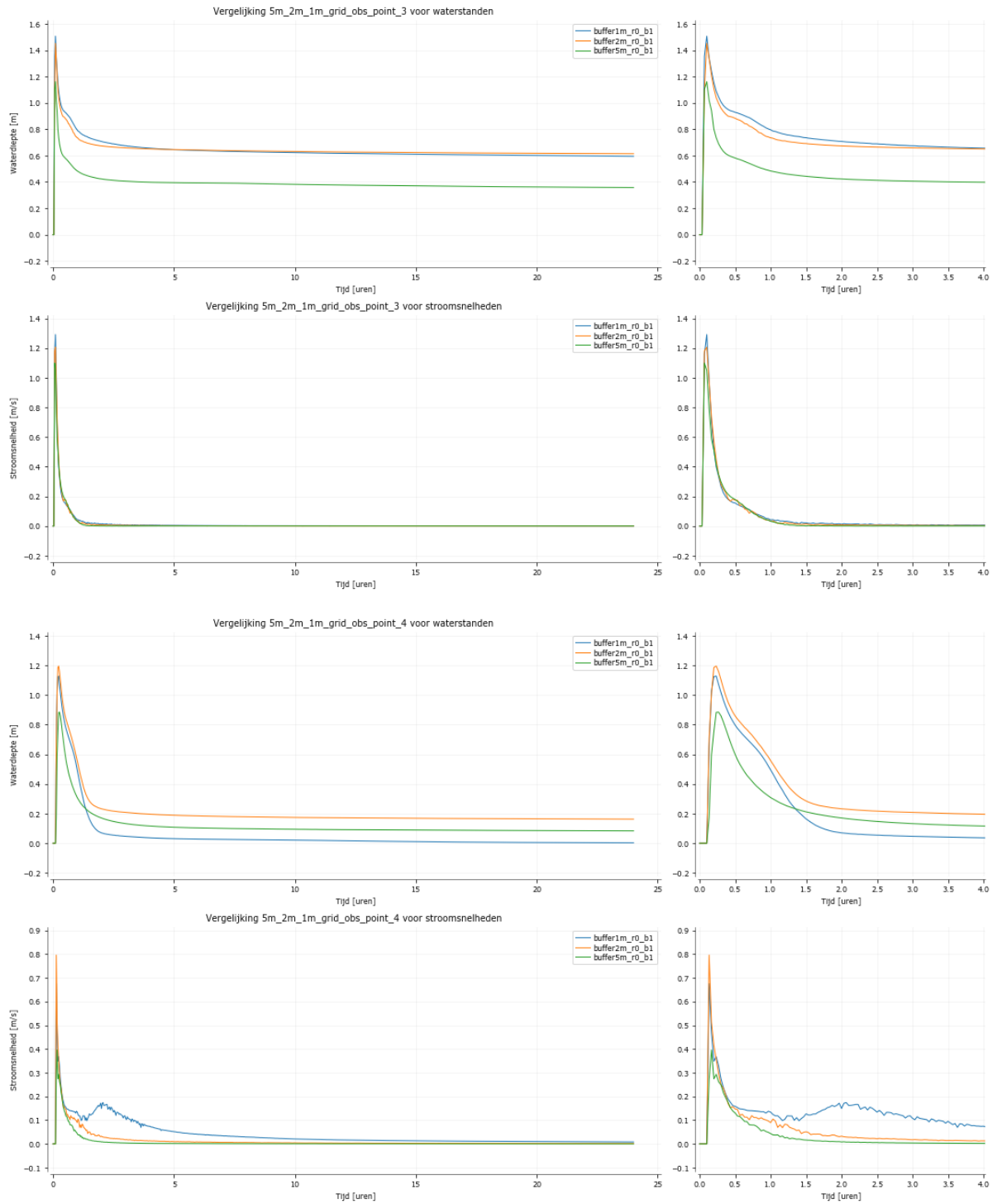
De resultaten voor de modellen met 1 m en 2 m resolutie zijn goed vergelijkbaar. In de smalle steeg (punt 5) is te zien dat het 1 m resolutiemodel met minder opstoppen en dus een lagere waterstand door de steeg stroomt.

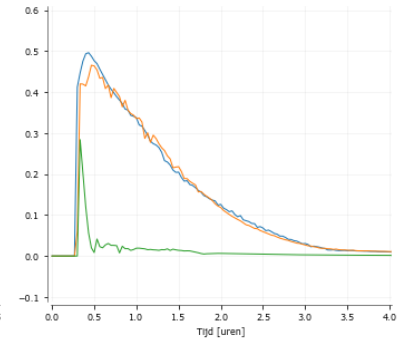
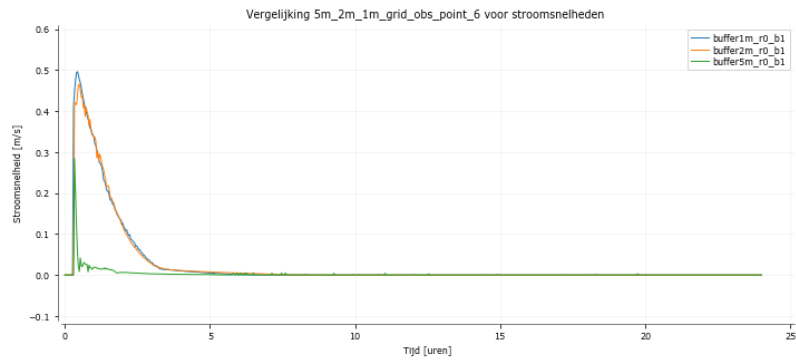
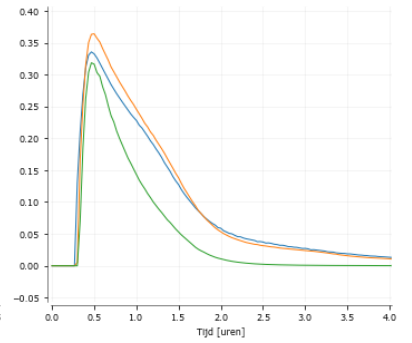
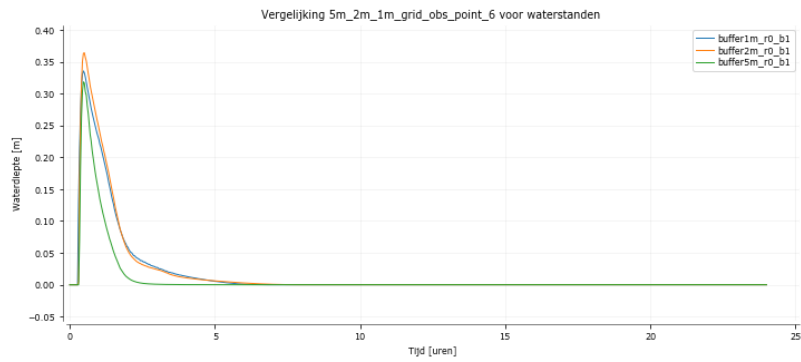
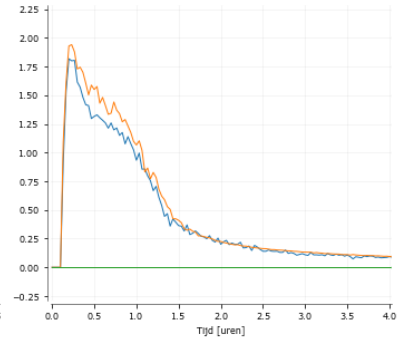
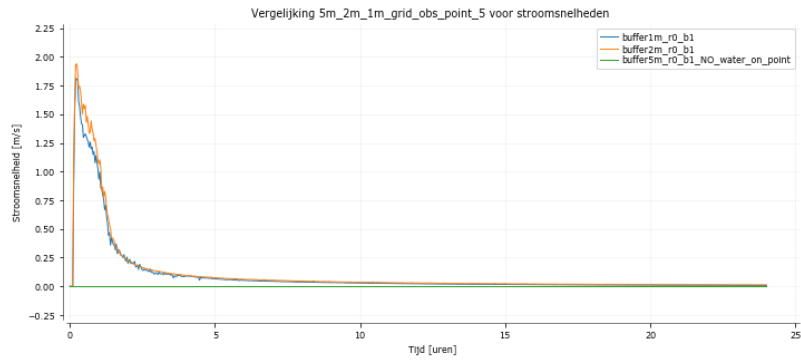
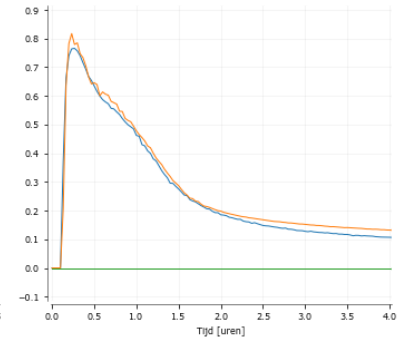
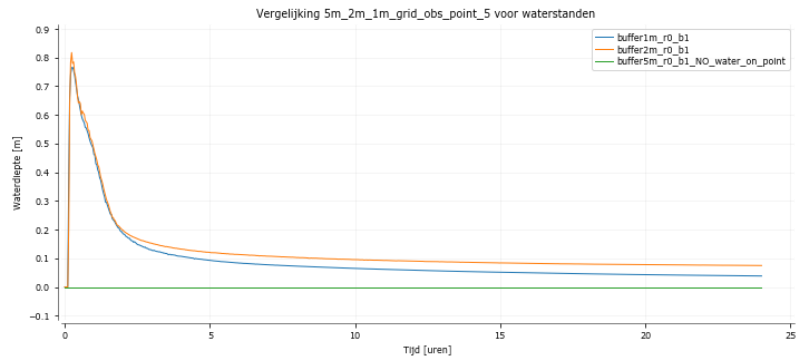
Voor een stedelijk gebied met smalle mee stromende stegen, zoals bij Genbroek het geval is, is een resolutie van 5 m te grof voor een nauwkeurige simulatie van de inundaties. De 1 m en 2 m rekenroosters geven een redelijk gelijk resultaat. Wanneer het gebiedsoppervlak met 1 m resolutiecellen onder de 1 miljoen cellen blijft heeft dit de voorkeur door de extra toevoeging van detail.

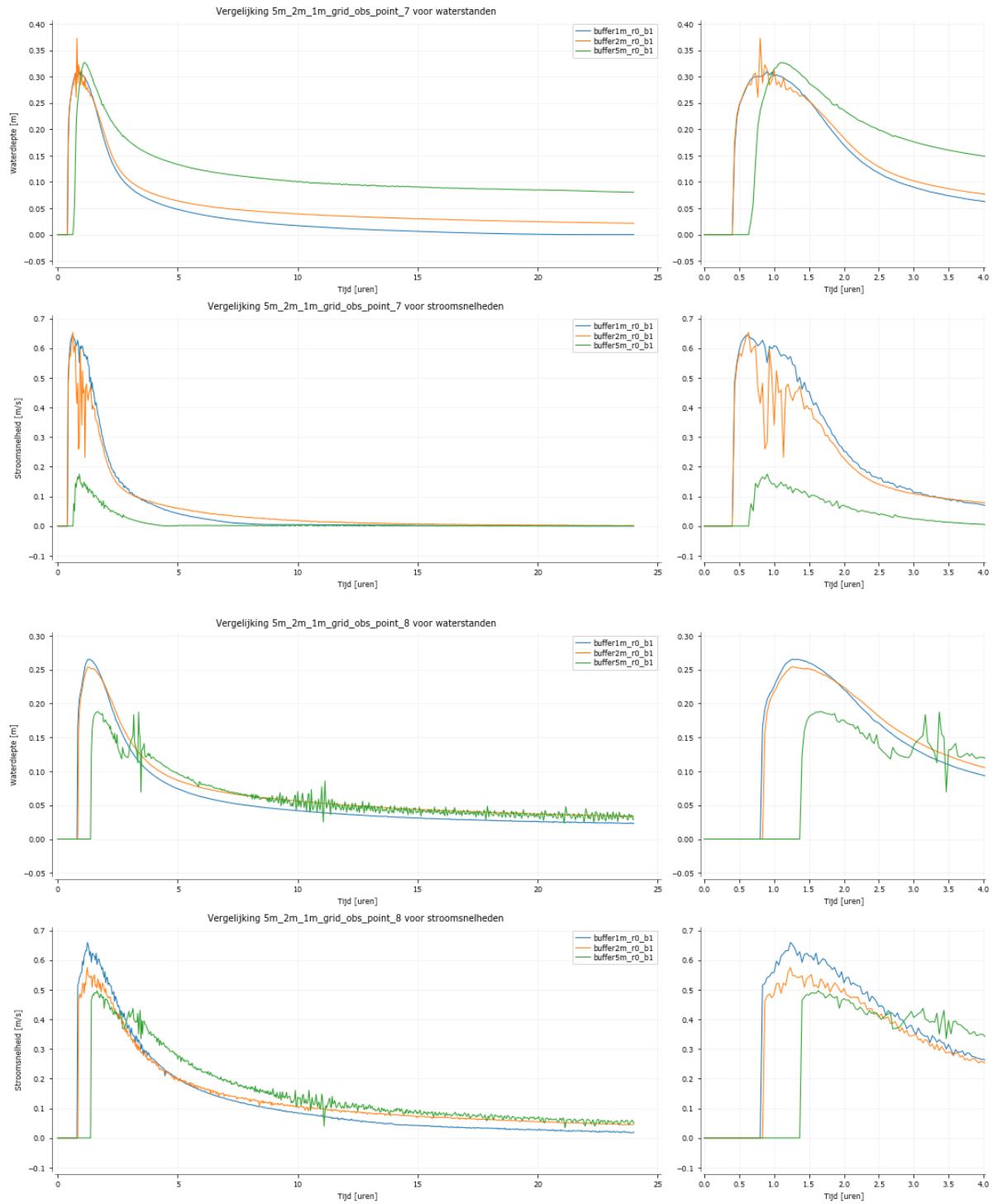
```
In [42]: plot_his_results(loc1=7, loc2=10, loc3=11, save = 'No', name_output= '5m_2m_1m_g  
rid_obs_point_')
```

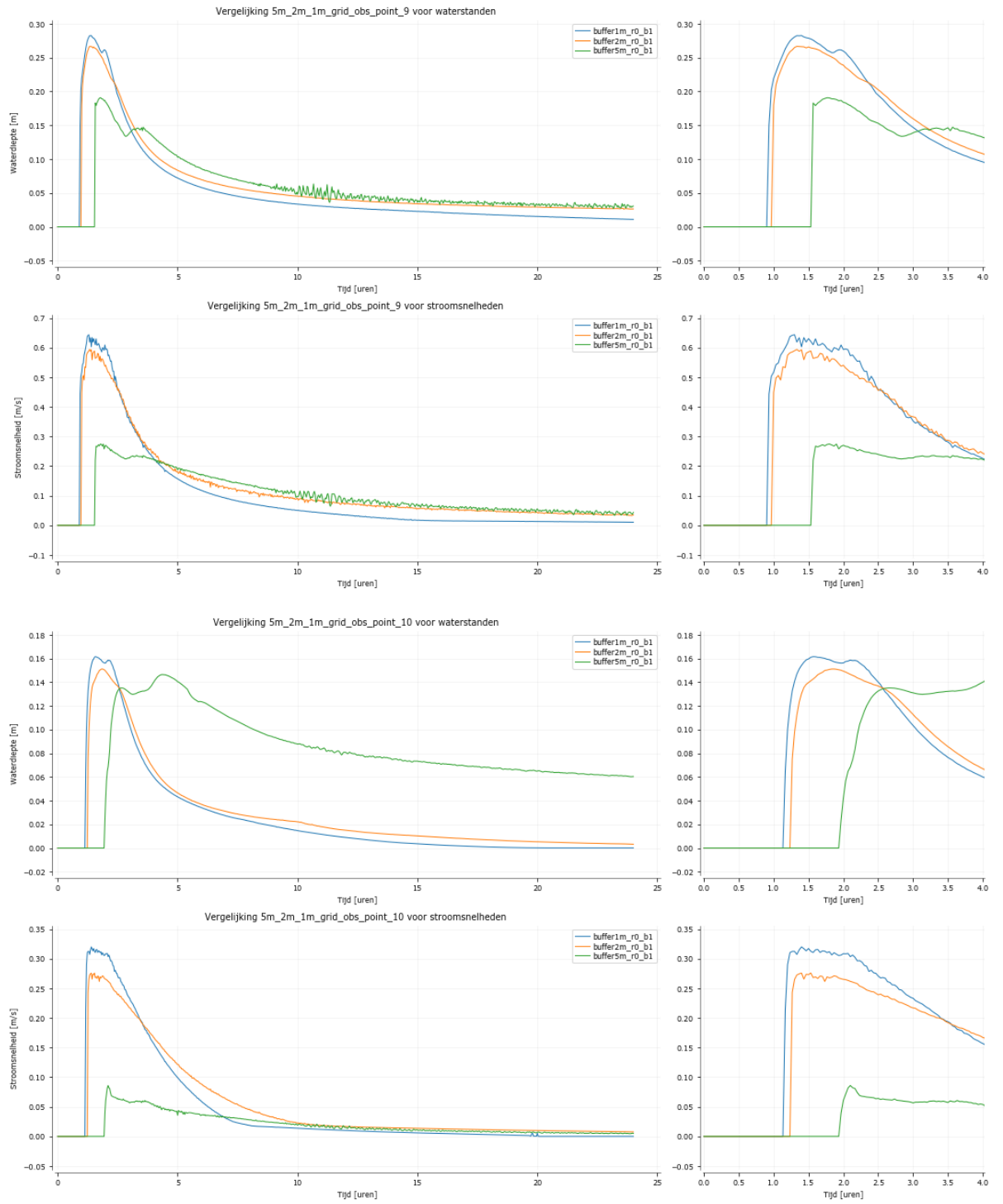
Number of observation points in his.nc file is: 12

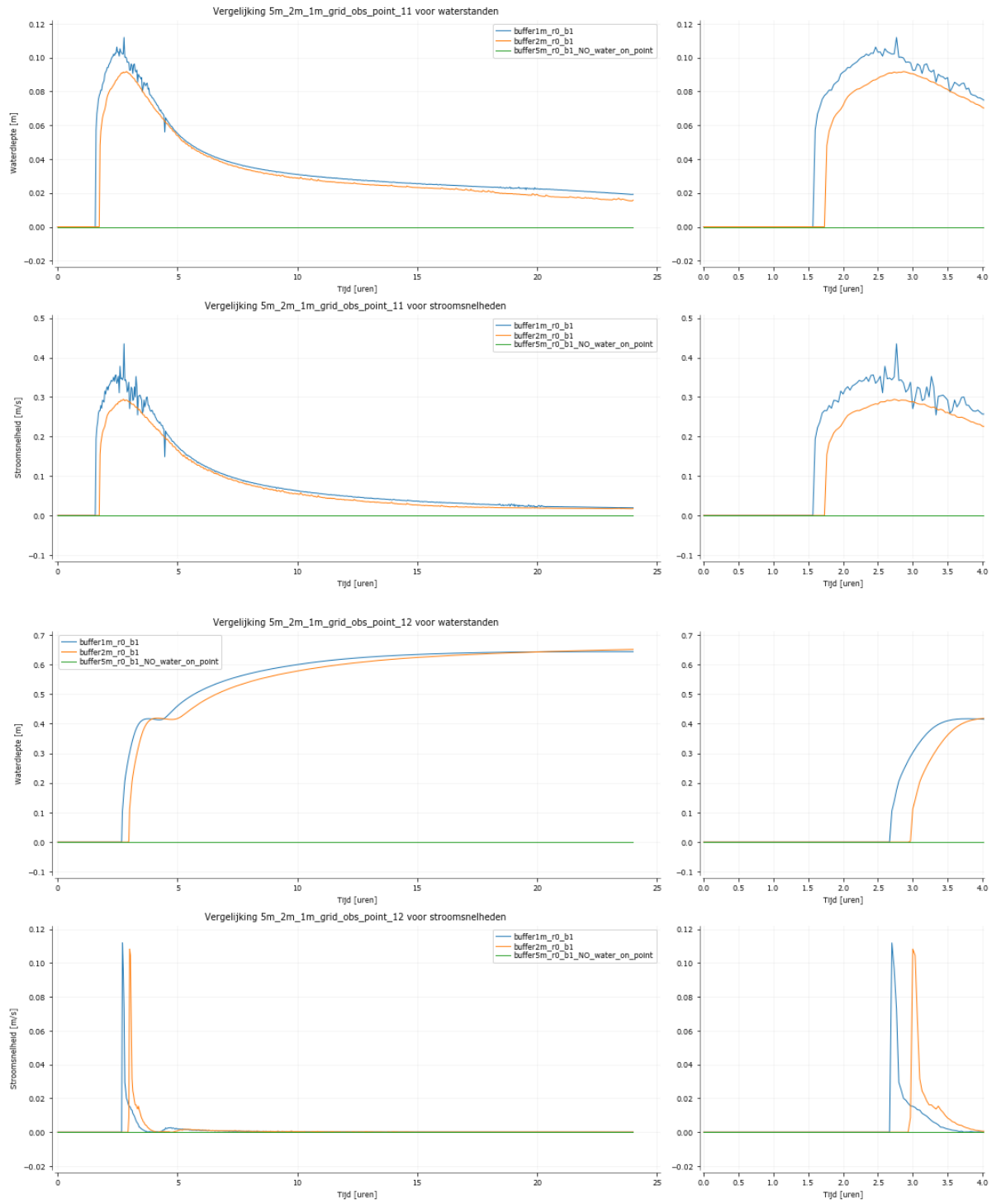










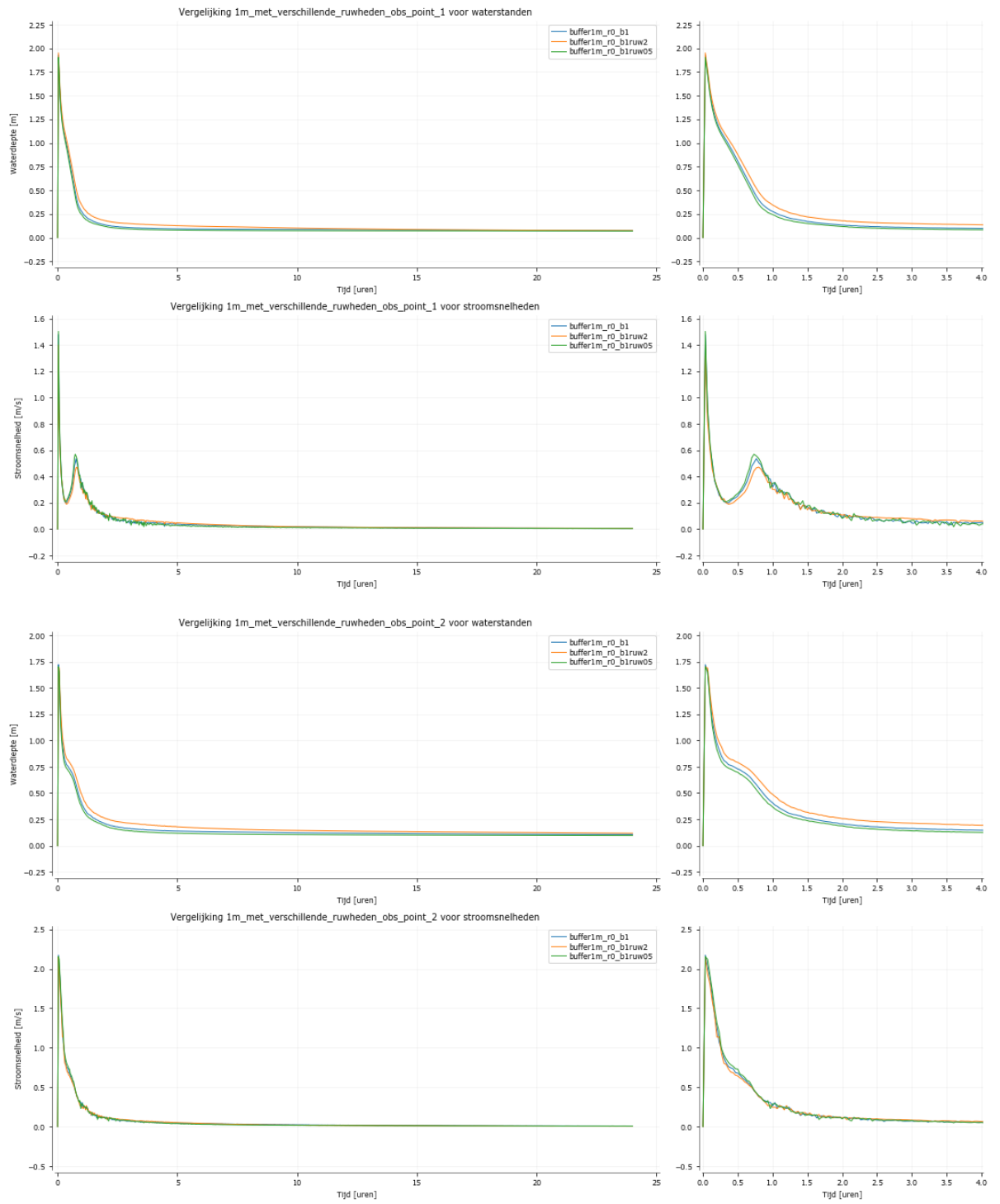


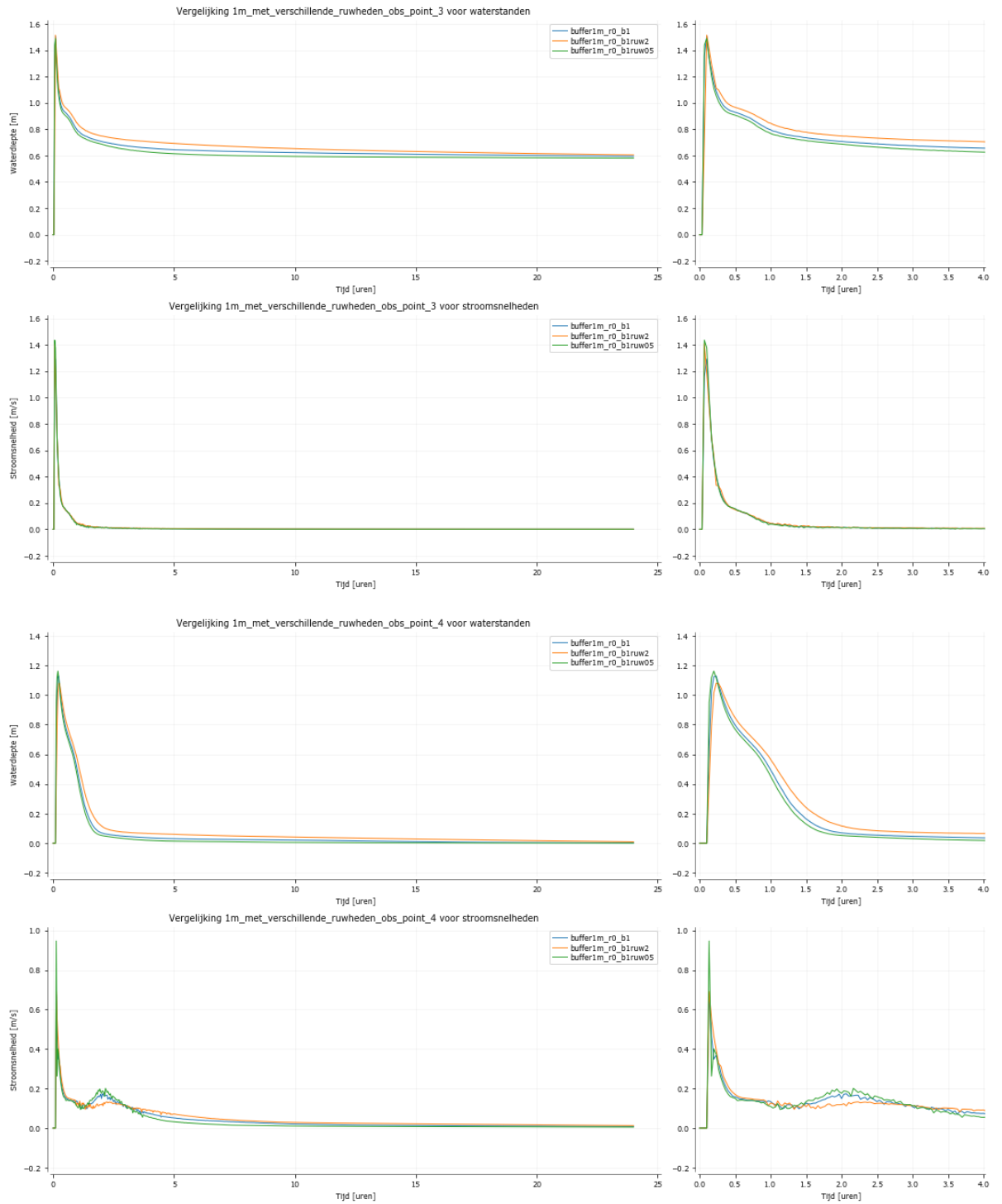
Invloed van de gevoeligheid op de gekozen ruwheid:

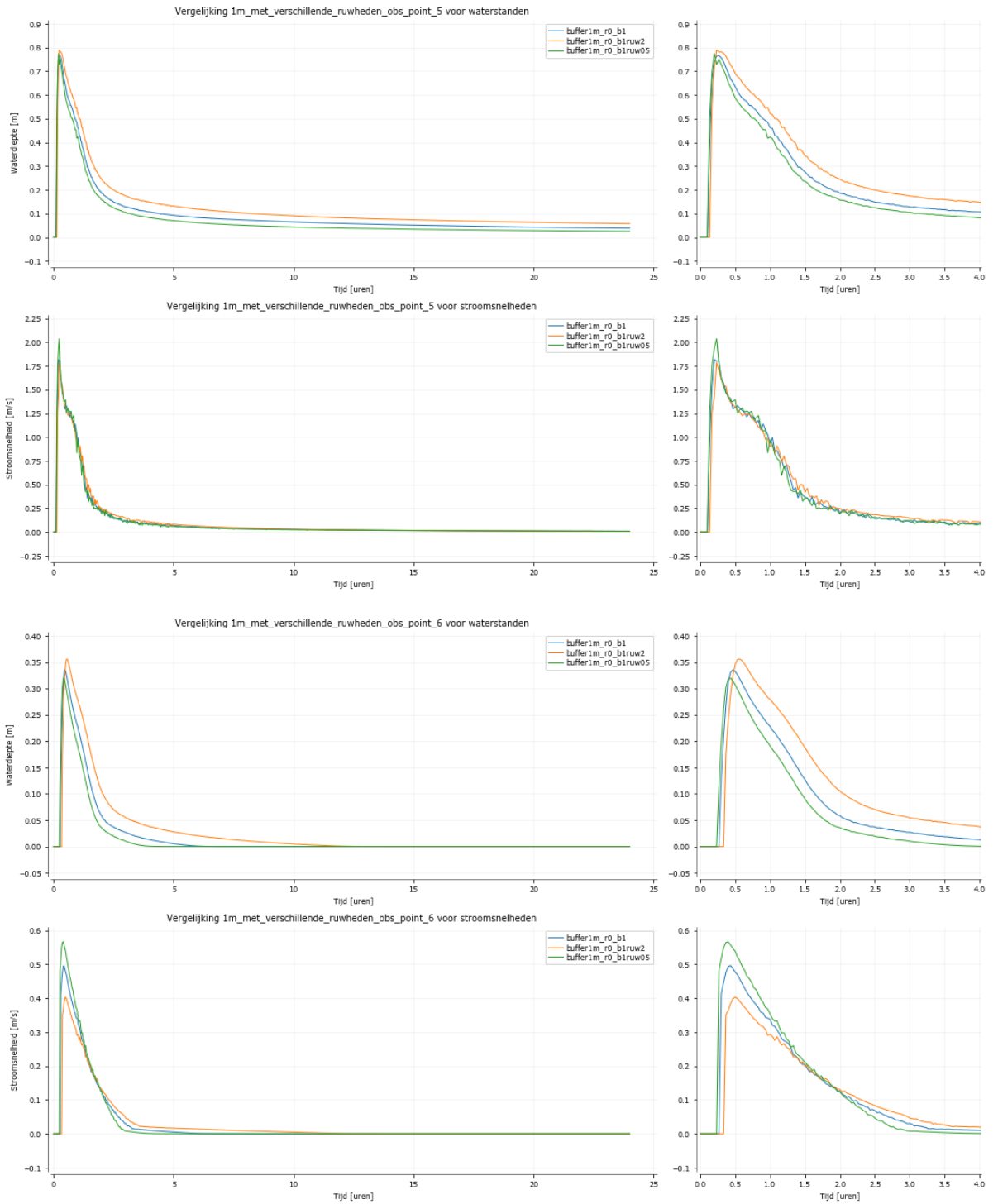
De gevoeligheid van de simulatieresultaten voor de ruwheid van het maaiveld is geëvalueerd door de oorspronkelijke ruwheidswaarde te halveren en deze te verdubbelen. De aanpassingen werken door zoals verwacht. Hoe hoger de ruwheidswaarde, hoe lager de stroomsnelheid en hoe groter de vertraging in de waterstandspiek. De vertraging van de piek is het best te zien in de wat verder vanaf de doorbraak gelegen observatiepunten. Het effect van de aanpassing van de ruwheidswaarde op de waterdiepte is kleiner dan het effect op de stroomsnelheid. Verder valt op dat het effect van de aanpassing van de ruwheid voor de eerste observatiepunten relatief klein is. Dit komt doordat de waterdiepte hier nog relatief groot is en de ruwheidswaarde een kleinere invloed heeft dan voor de verder weg gelegen observatiepunten met een kleinere waterdiepte.

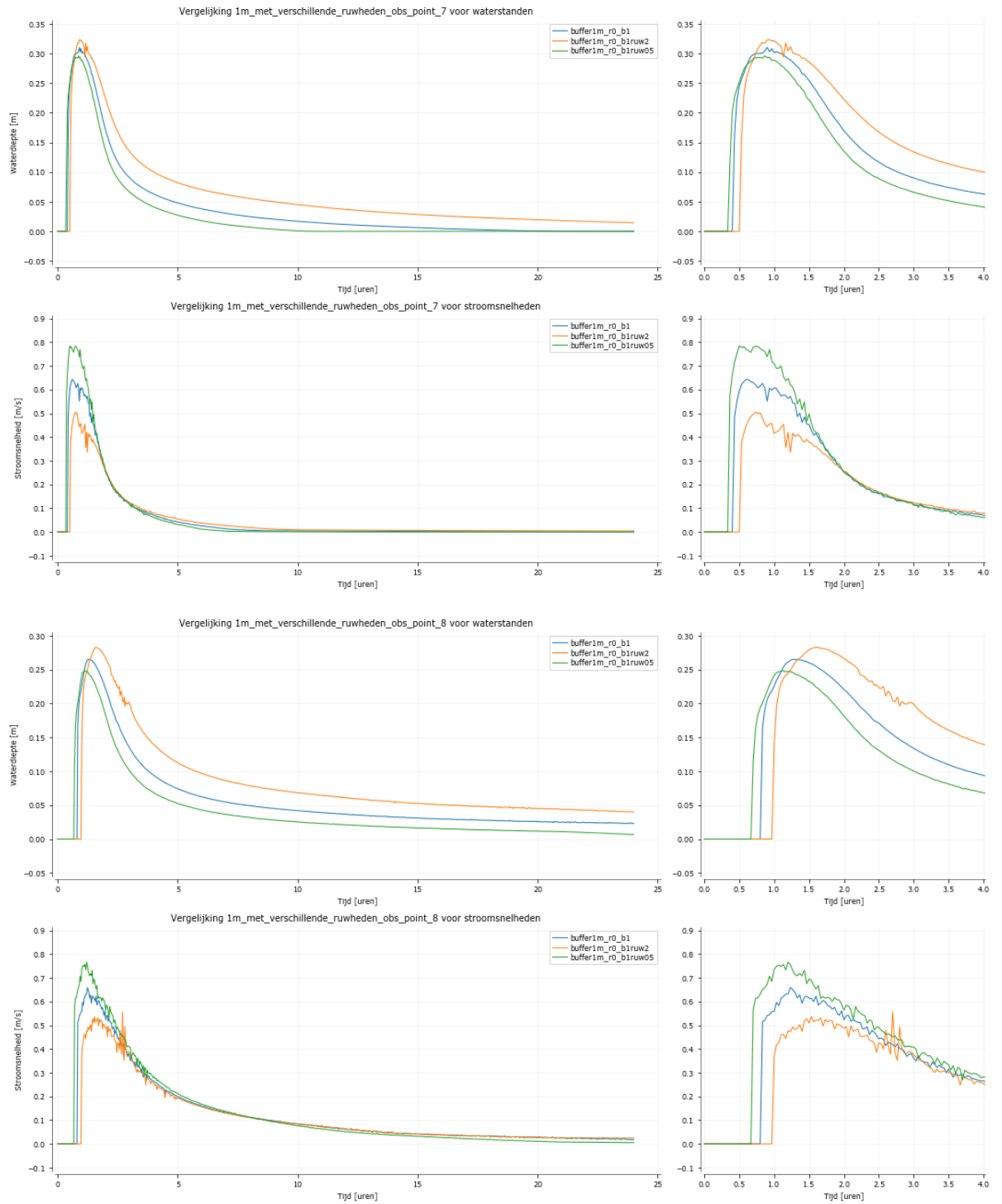
```
In [43]: plot_his_results(loc1=7, loc2=5, loc3=6, save ='No',name_output= '1m_met_versc  
hillende_ruwheden_obs_point_')
```

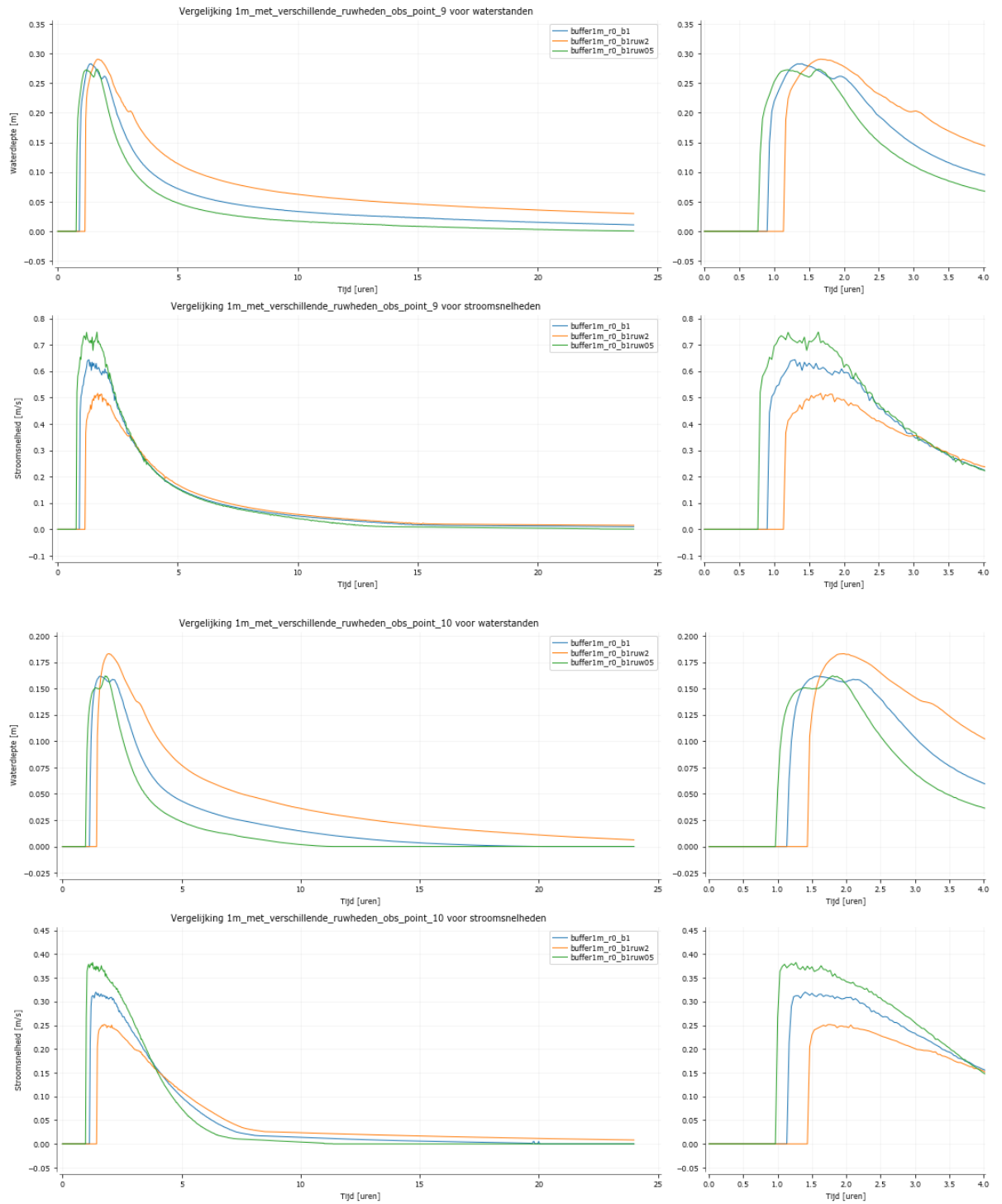
Number of observation points in his.nc file is: 12

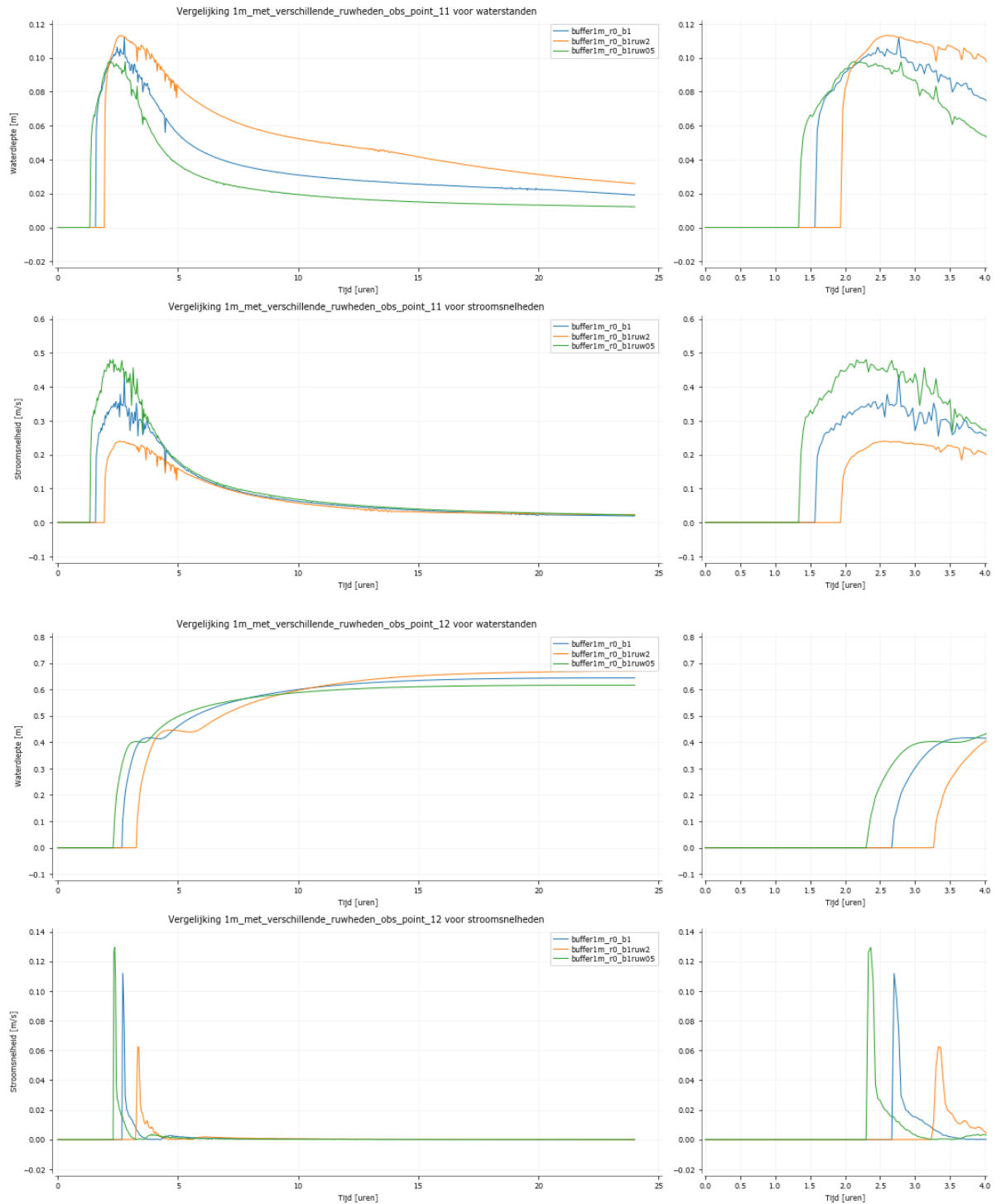










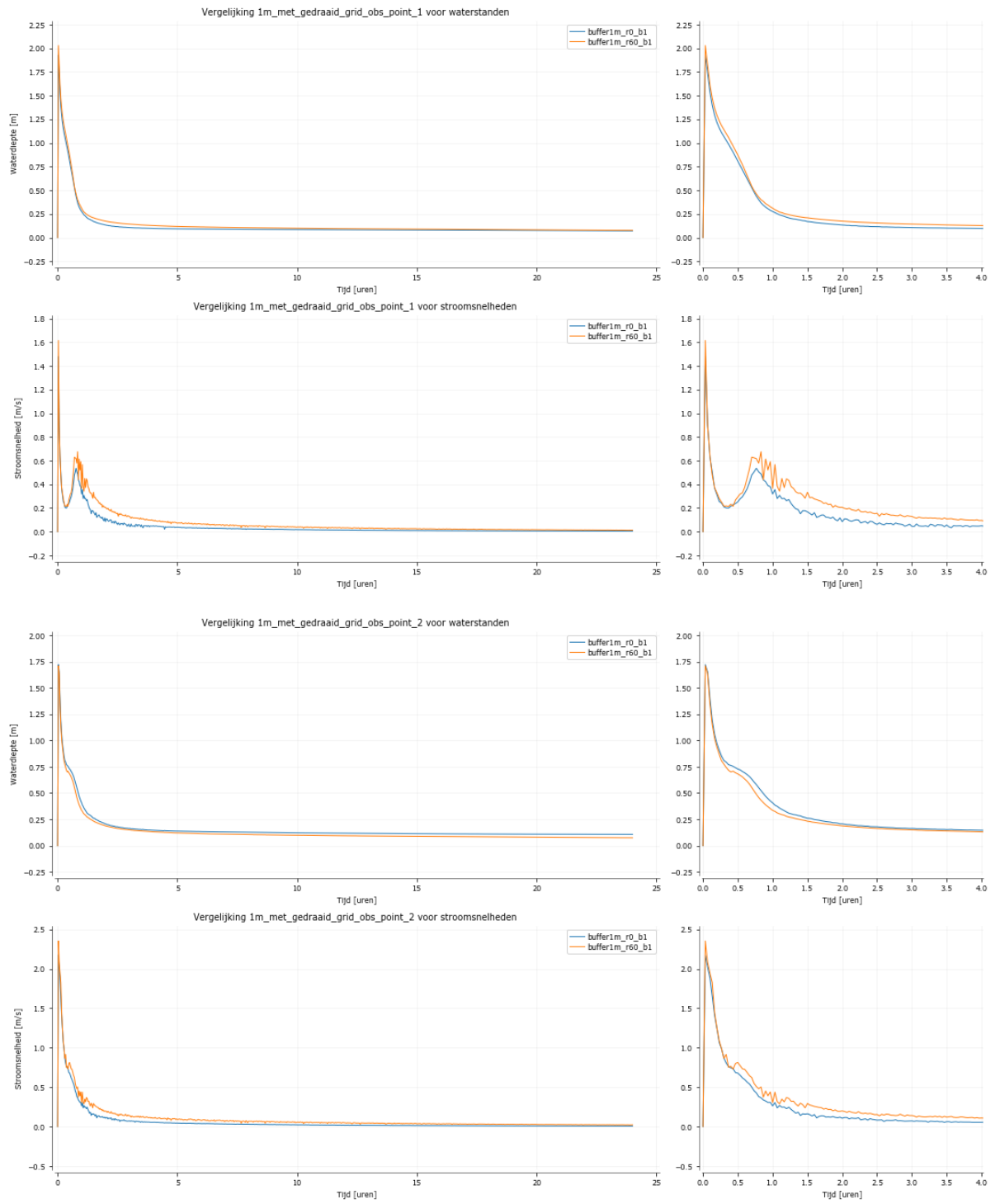


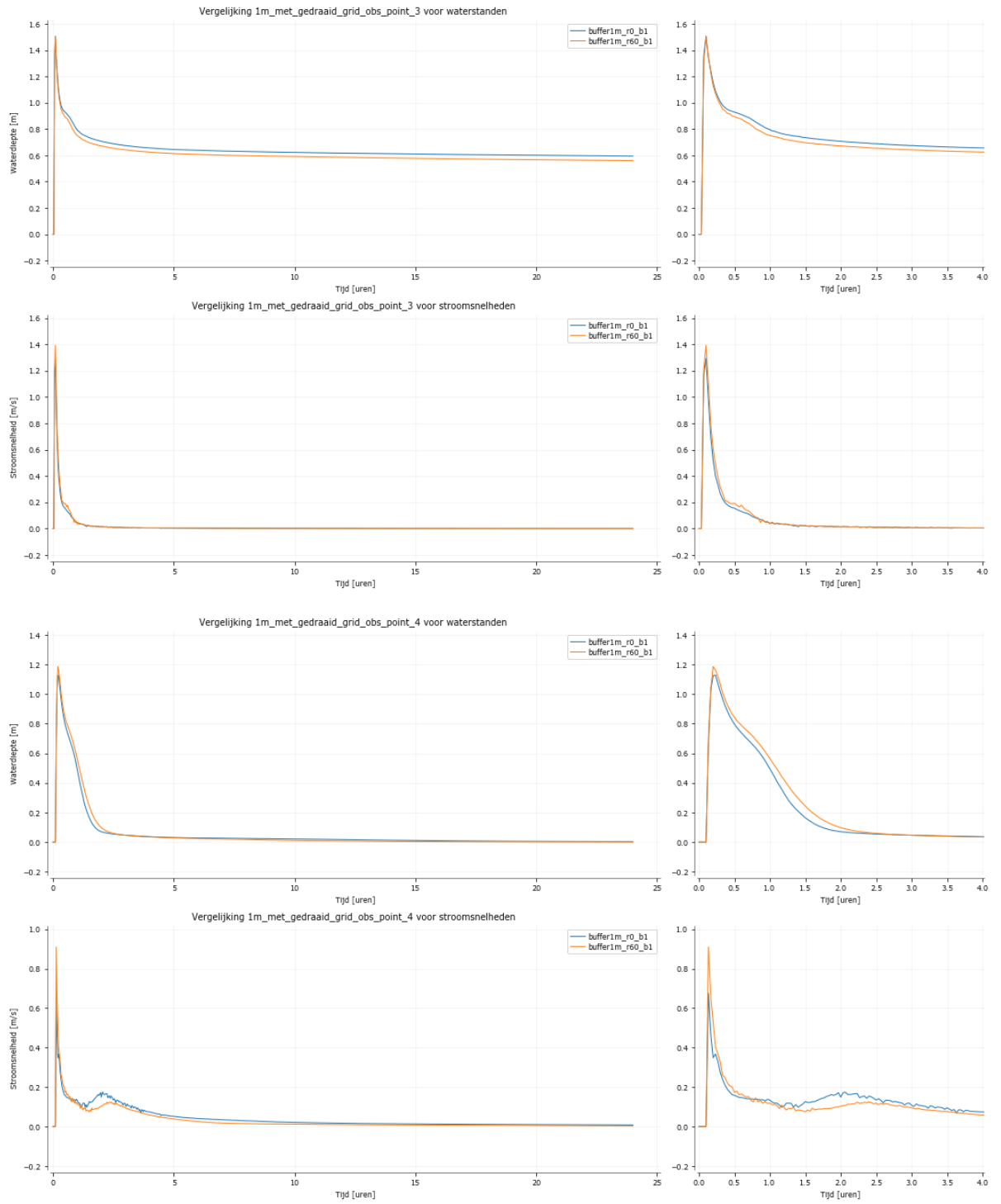
Invloed van grid rotatie:

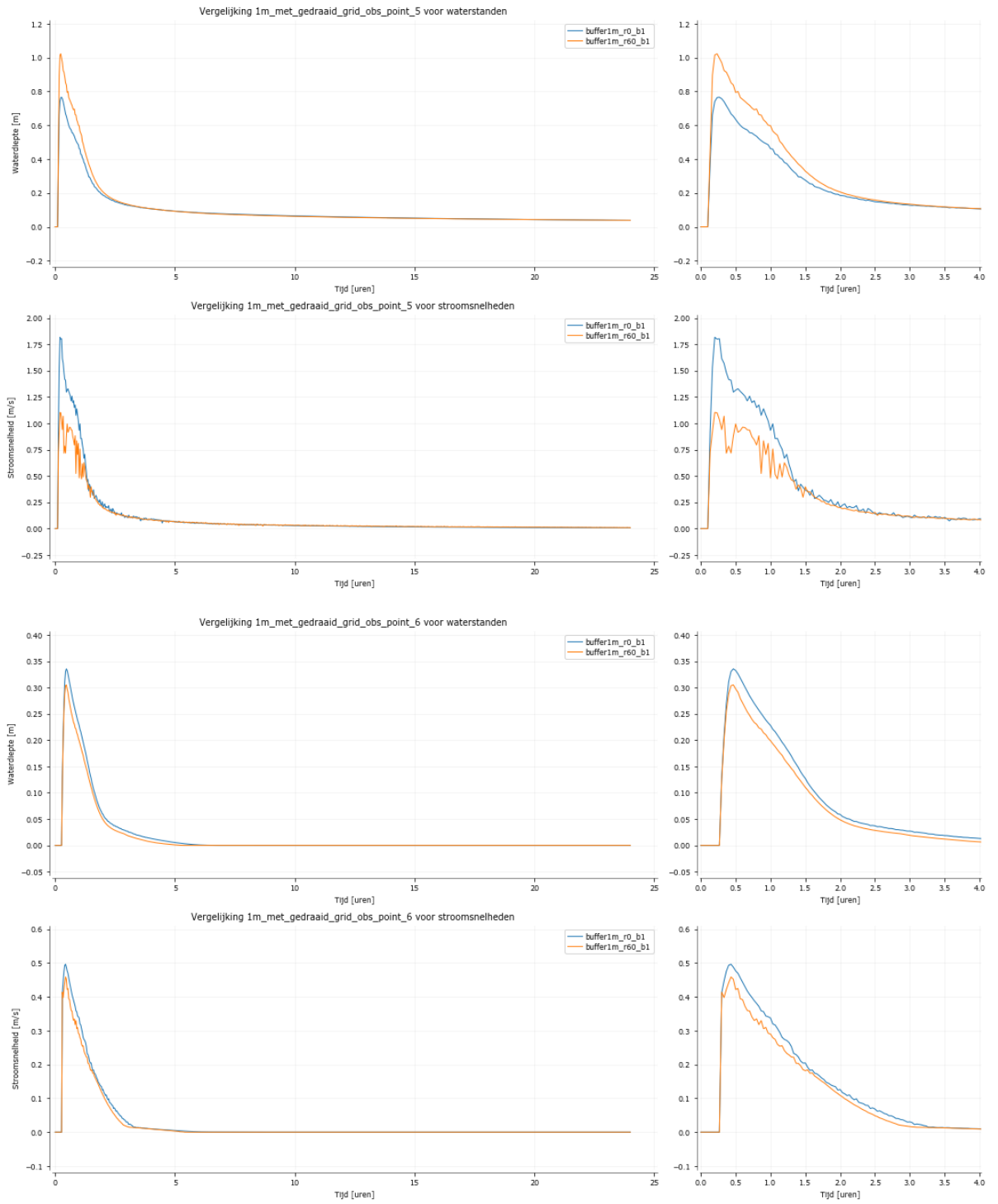
De invloed van grid rotatie is verwaarloosbaar voor de eerste vier observatiepunten. Bij de smalle steeg ter hoogte van observatiepunt 5 is te zien dat de gridrotatie zorgt voor hogere waterstanden. Dit is te verklaren doordat de steeg door de rotatie niet meer in de hoofdstroomrichting ligt. Hierdoor passen er minder cellen naast elkaar in de steeg. Dit effect werd verder benedenstrooms ook door in de overige observatiepunten.

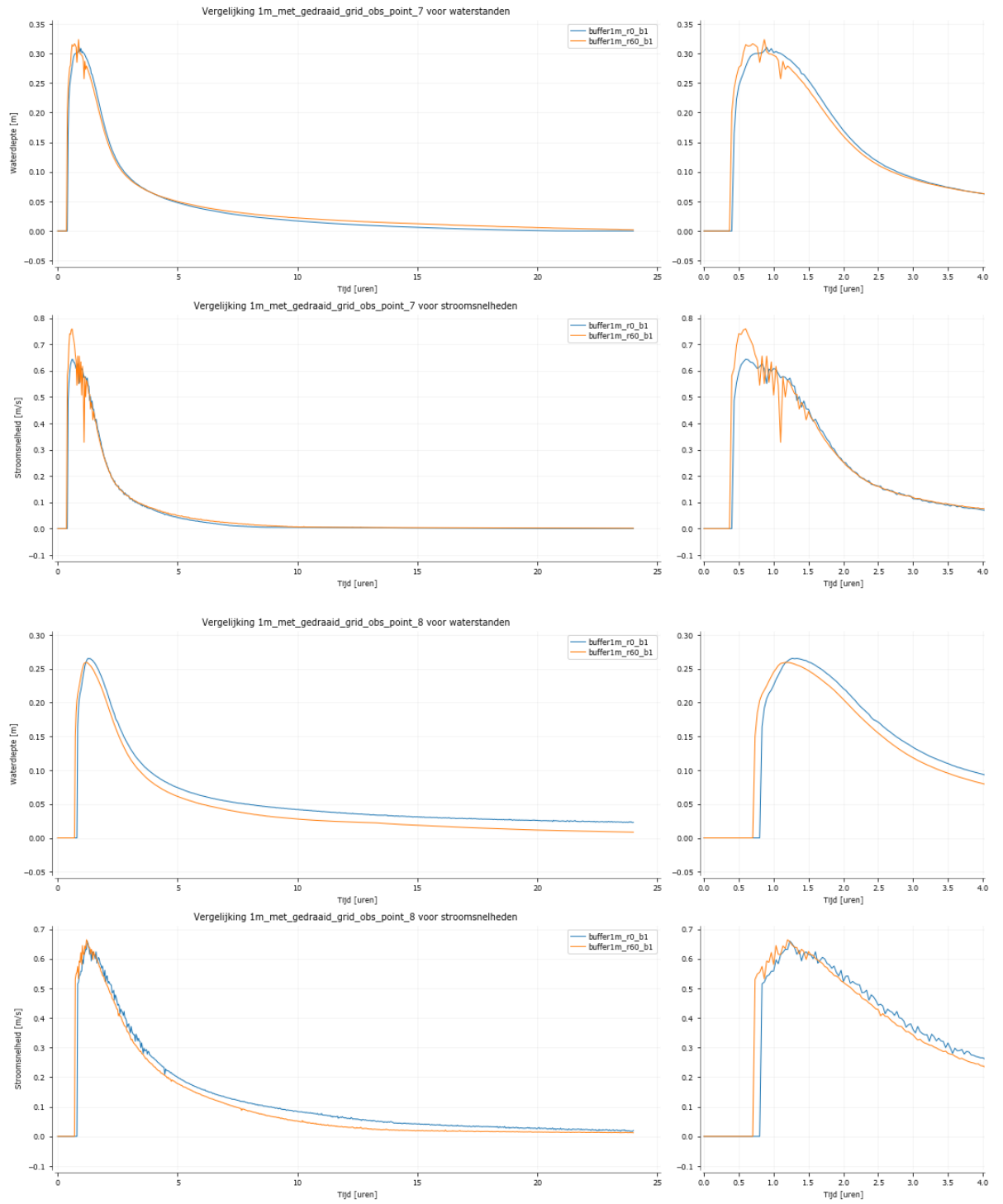

```
In [44]: plot_his_results(loc1=7, loc2=-3, loc3=999, save = 'No', name_output= '1m_met_ge  
draaid_grid_obs_point_')
```

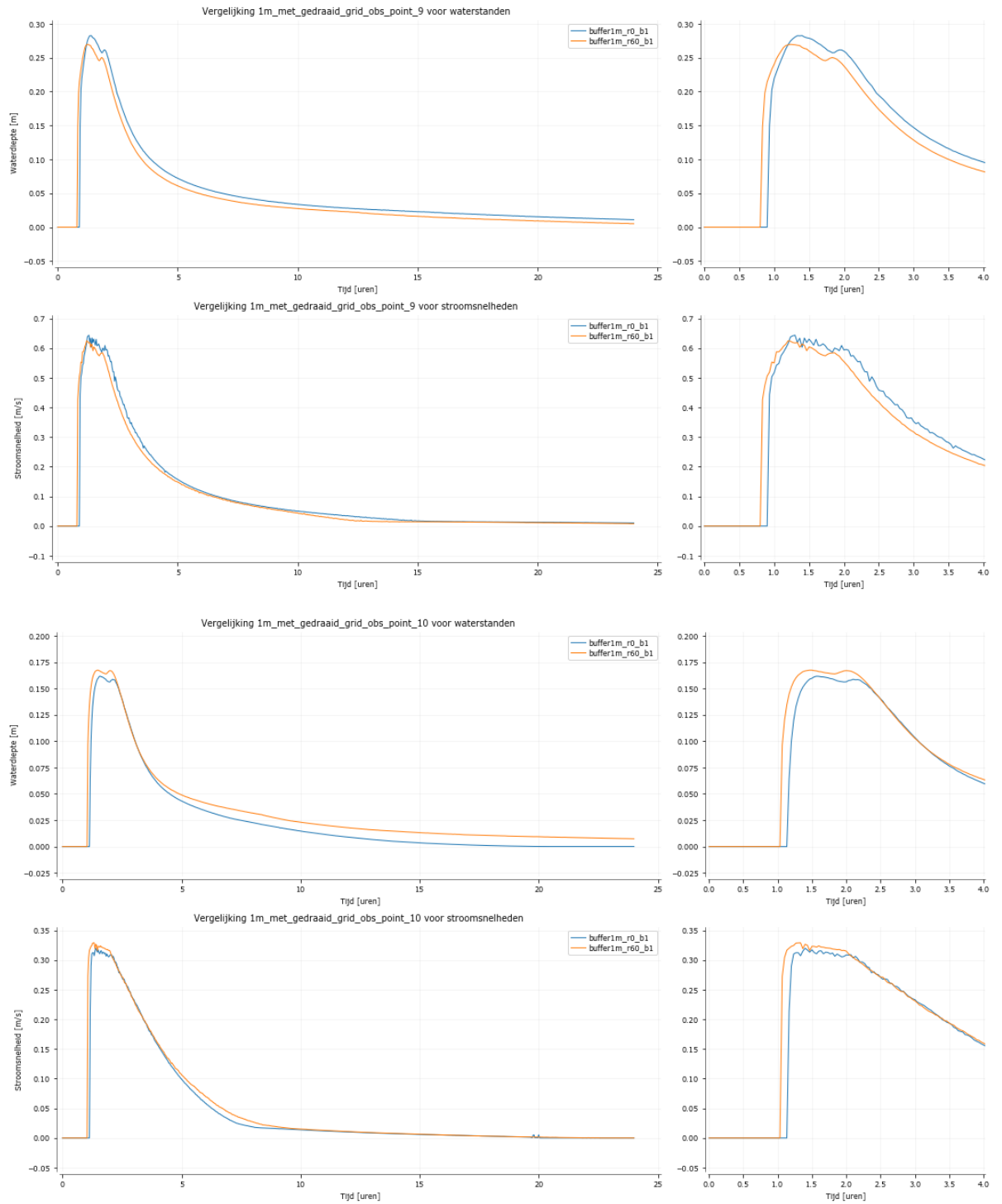
Number of observation points in his.nc file is: 12

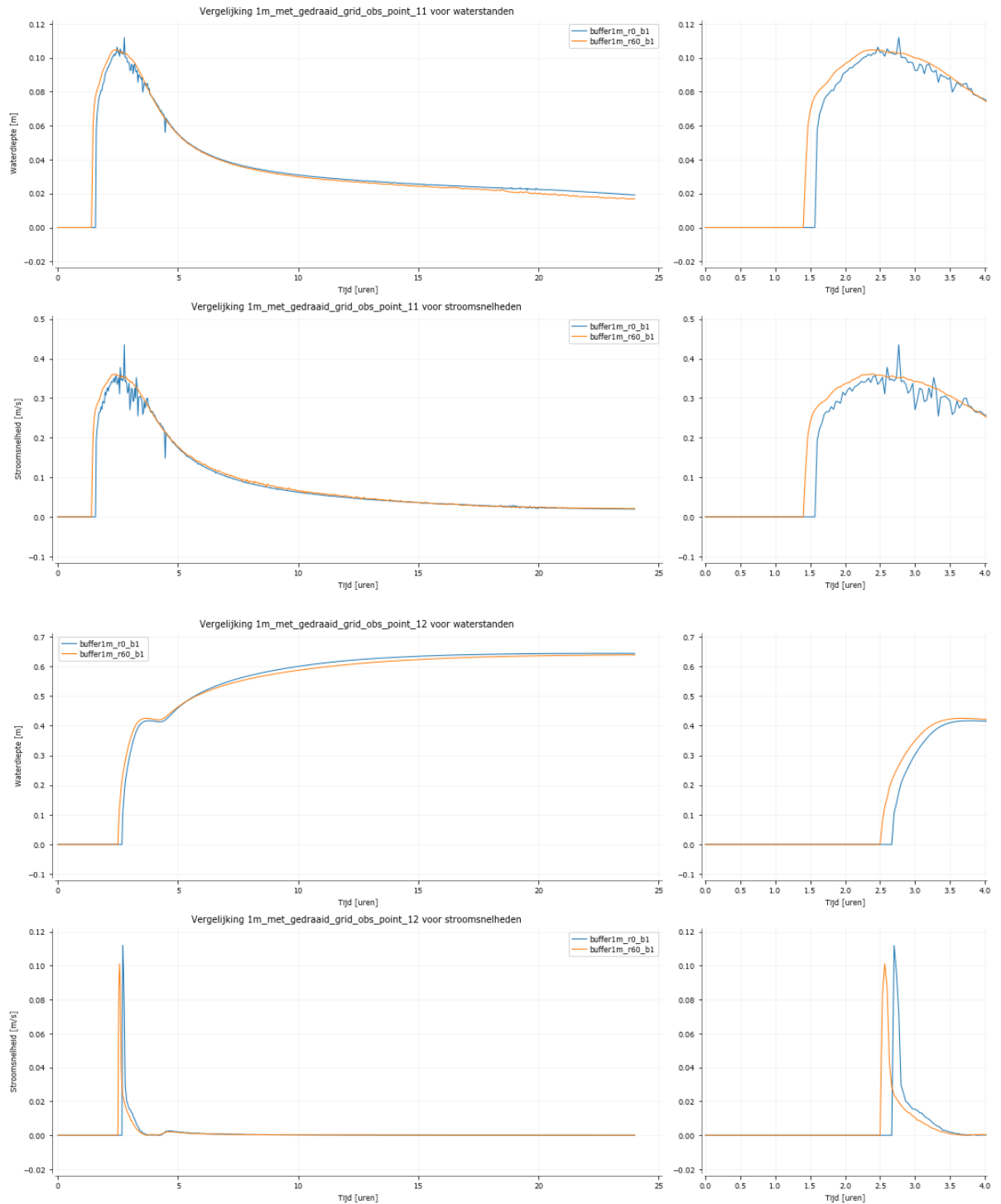










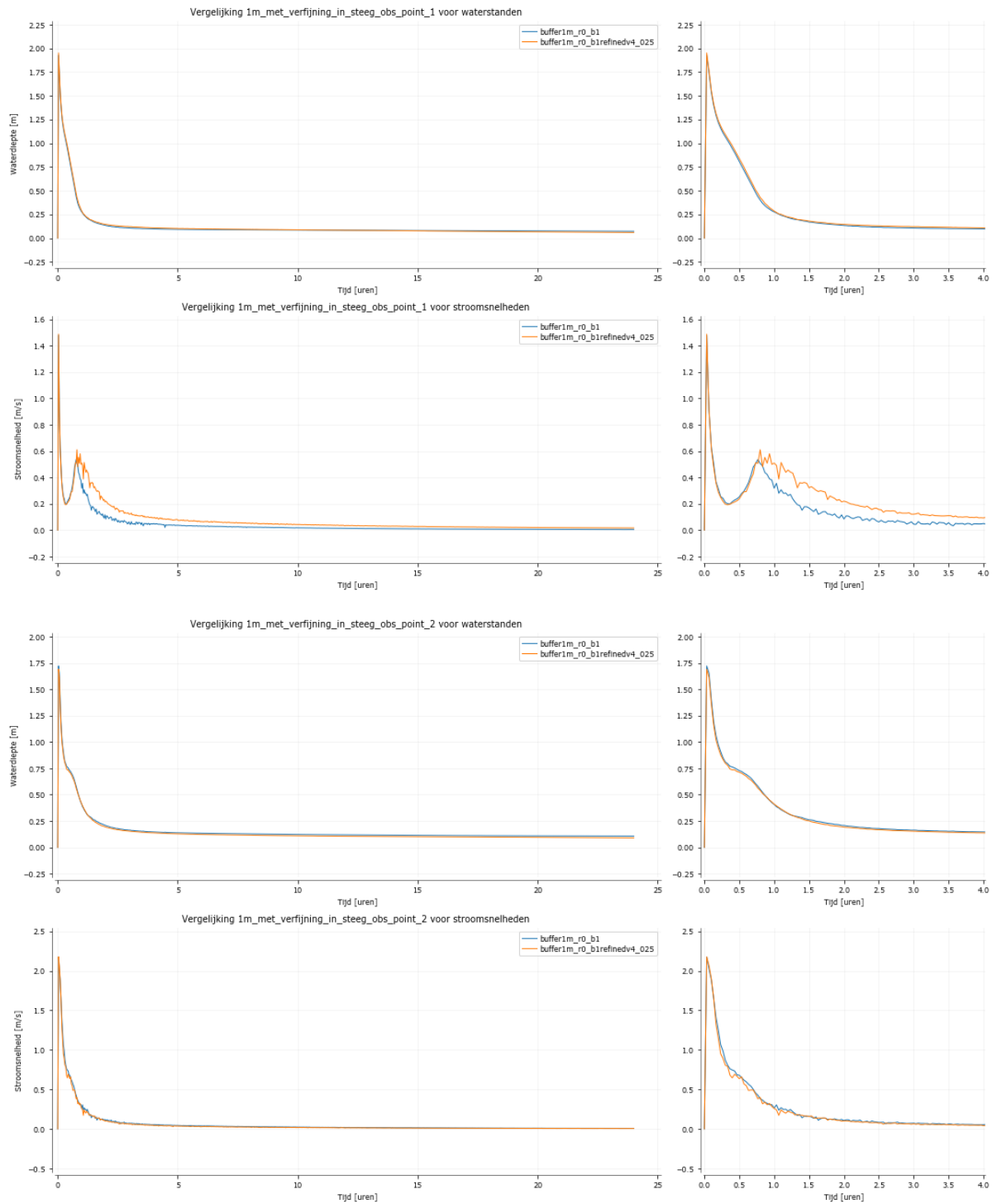


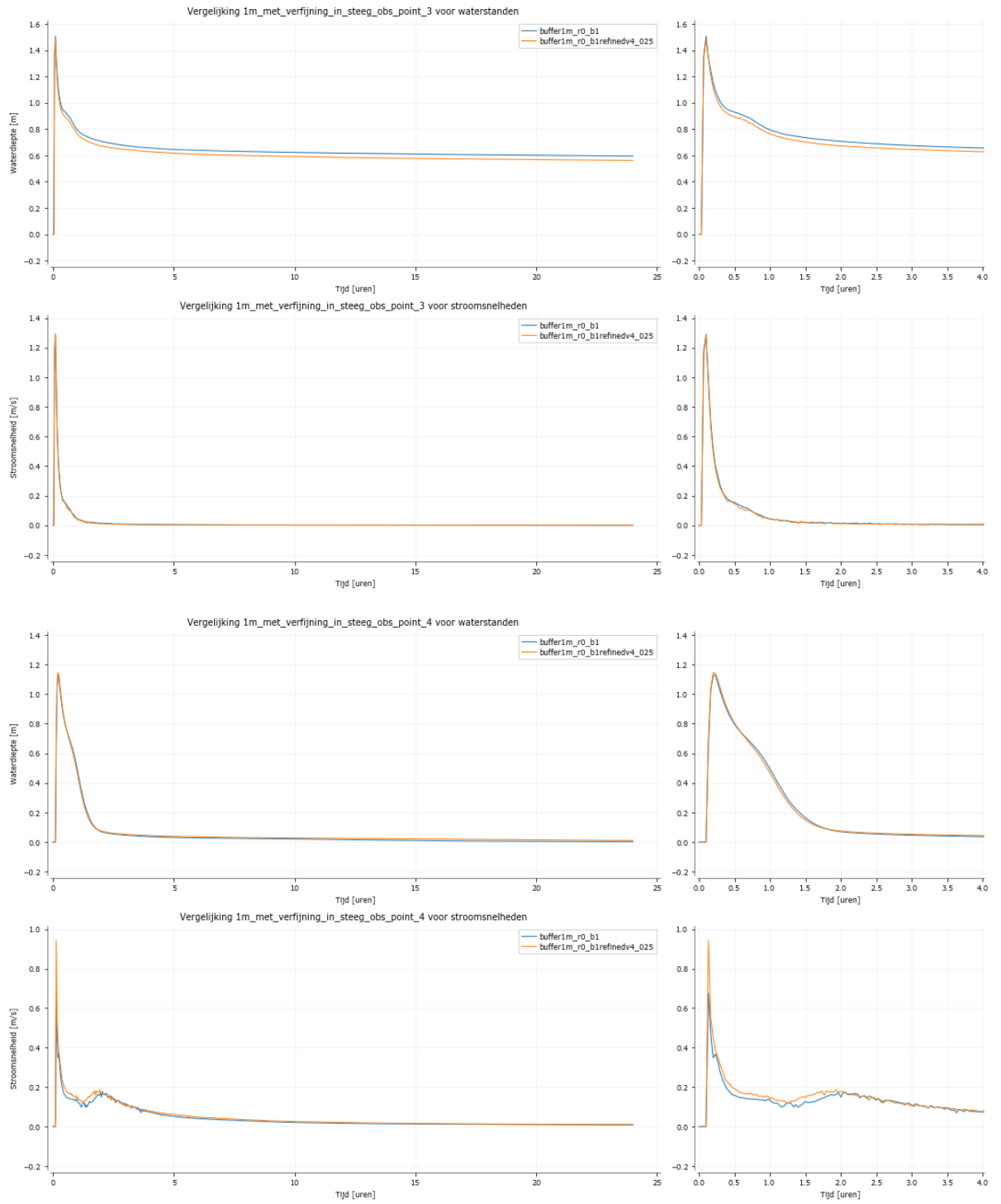
Grid verfijning

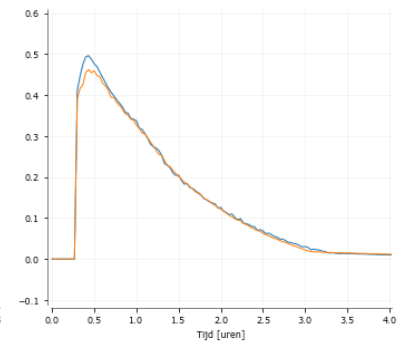
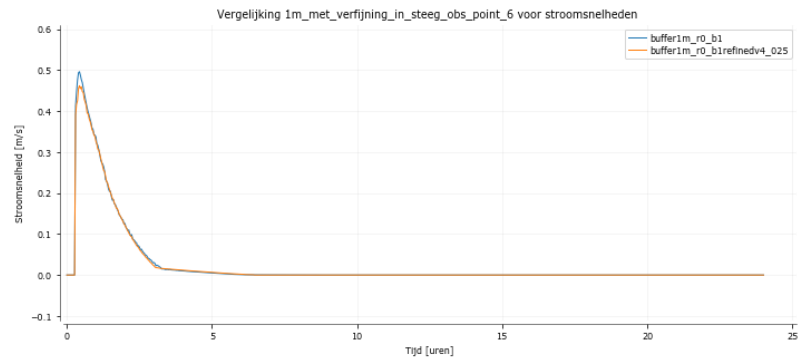
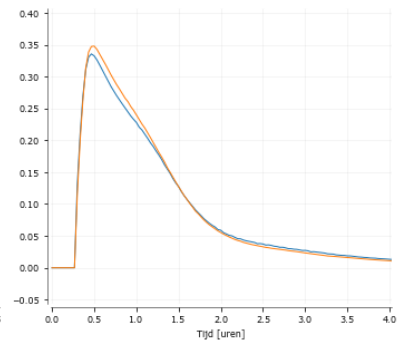
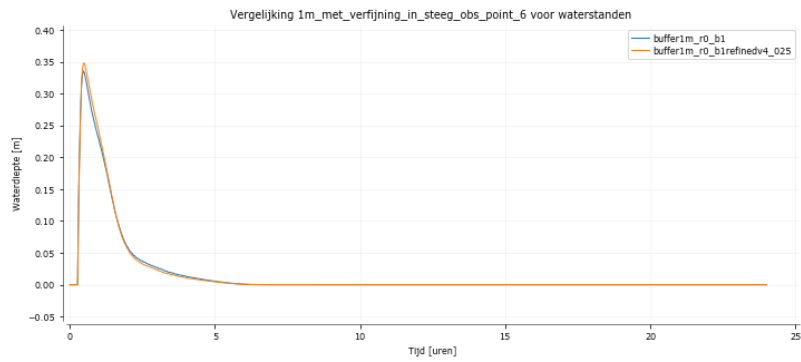
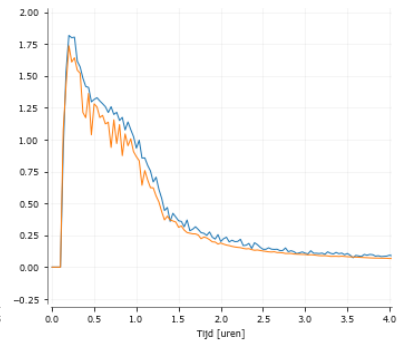
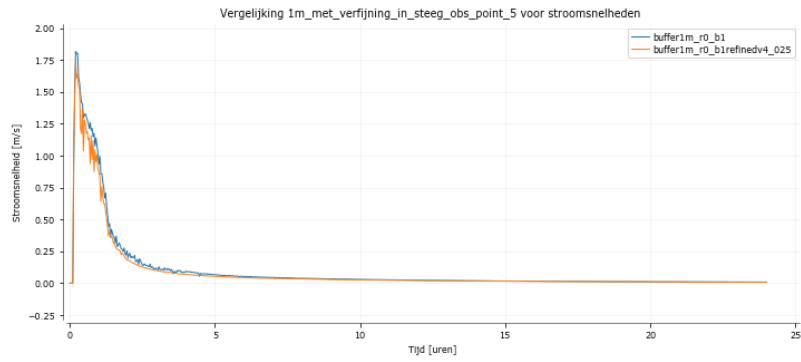
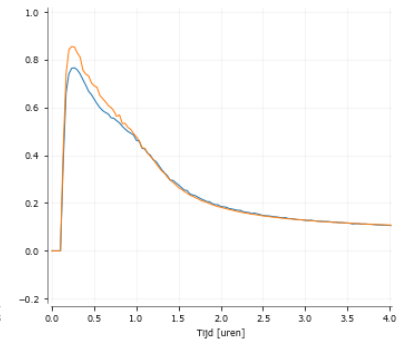
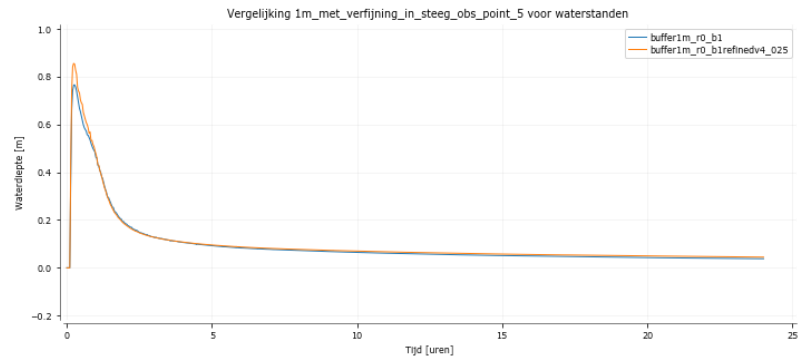
Tijdens het onderzoek kwam de vraag naar voren of een roosterverfijning tot 0.5 m in de smalle steeg bij observatiepunt 5 tot andere resultaten zou leiden dan het model met een resolutie van 1 m. Hiertoe is het rooster in de steeg lokaal verfijnd tot 0.5 m. De verschillen ter hoogte van de smalle steeg zijn rond de 6 centimeter op een waterdiepte van ongeveer 0.80 m. Het verschil zit binnen de verwachte modelonzekerheid waardoor lokaal verfijnen van 1 m naar 0.5 m niet als een toegevoegde waarde wordt gezien.

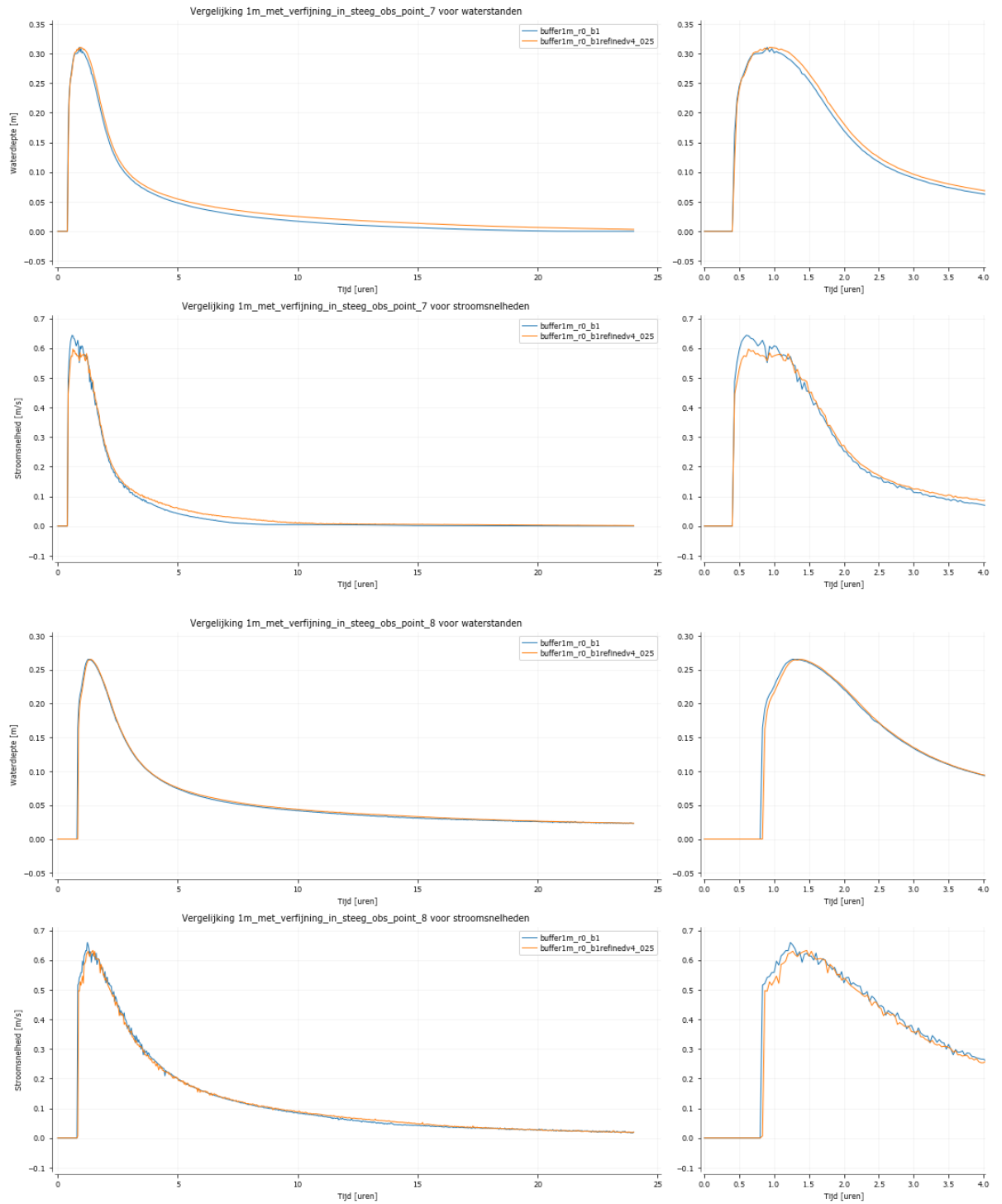
```
In [60]: plot_his_results(loc1=7, loc2=2, loc3=999, save = 'No', name_output= '1m_met_ver  
fijning_in_steeg_obs_point_')
```

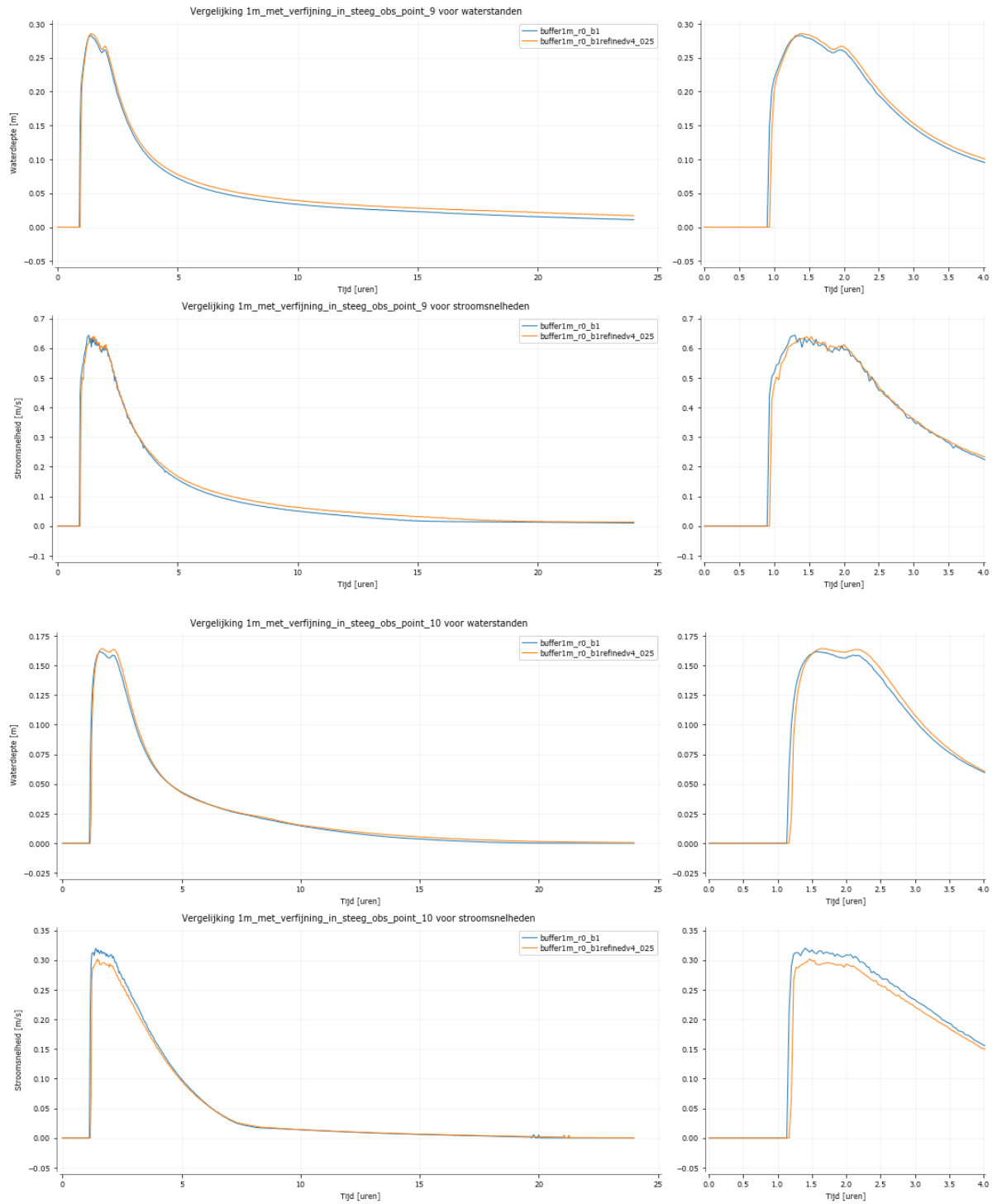

Number of observation points in his.nc file is: 12

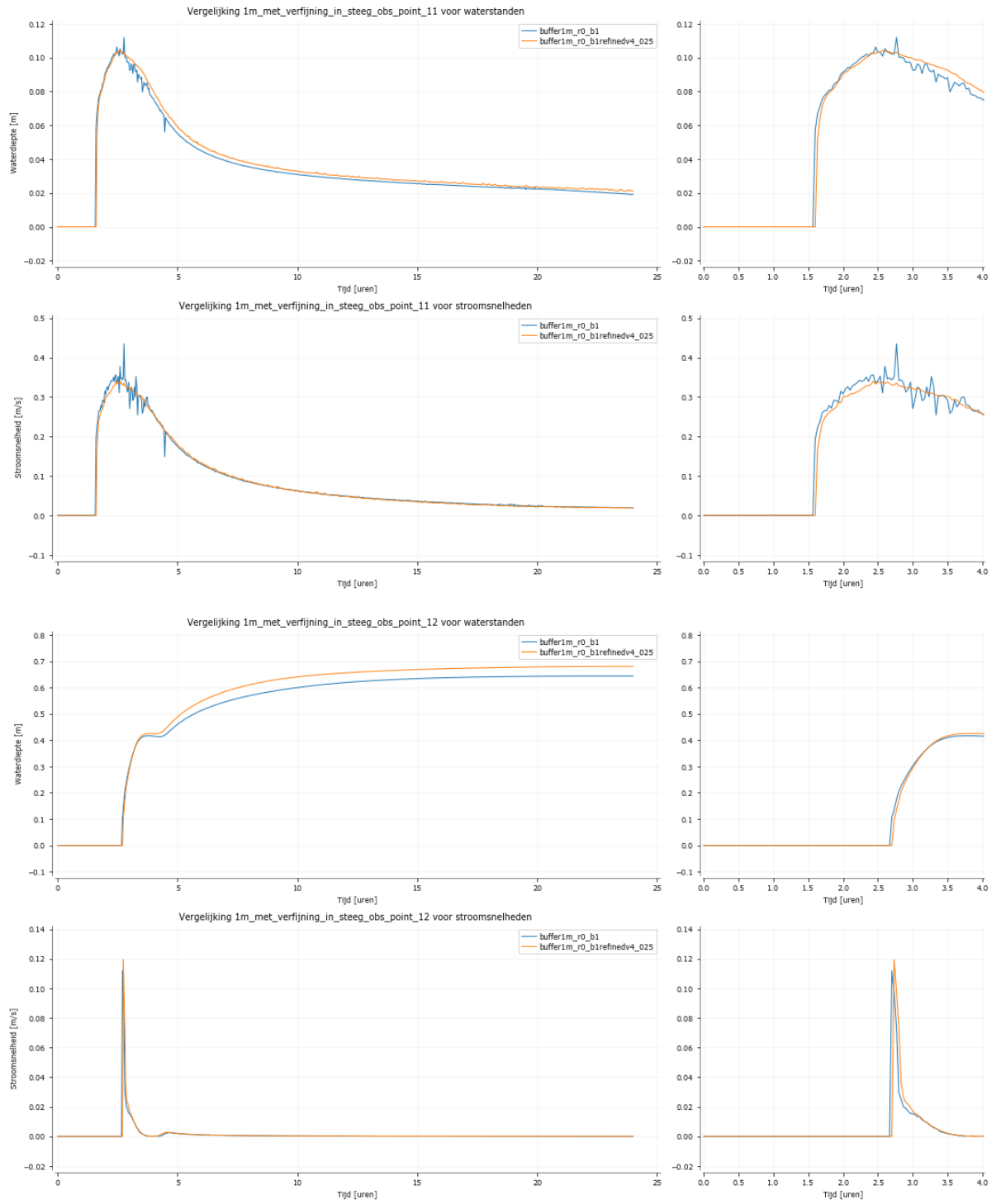












Invloed van bresgroei op de bufferdoorbraak

De bresgroei is ingesteld op een initiële bresbreedte van 3 m die groeit tot maximaal 5.54 meter. De bres wordt niet groter doordat de stoomsnelheid op dat moment alweer te laag is om de bres breder te maken. De buffer is dan namelijk al bijna leeg. De kritische snelheid voor erosie staat op 0.2 m/s ingesteld. De tijd die ingesteld is om in de diepte de bodem van de dam te bereiken is ingesteld op 1 minuut.

Het schematiseren van het falen van de kering met een bres is een minder extreem scenario dan wanneer de hele kering in een keer wordt weggehaald. Via de bresgroei loopt de regenwaterbuffer langzamer leeg en veroorzaakt minder hoge piekwaterstanden benedenstrooms. Het gebruik van een groeiende bres resulteert ook in een lichte vertraging van de afvoergolf.

```
In [46]: plot_his_results(loc1=7, loc2=0, loc3=999, save = 'No', name_output= '1m_met_en_zonder_bresv2_obs_point_')
```


Number of observation points in his.nc file is: 12

