

How it works in principle

The idea is to reduce risks when changing AAPS settings. I first test the impact of such changes offline on a PC by emulating in a python script what the loop would do. That python is a translation of the original AAPS „determine-basal“ java version.

The code scans an existing AAPS logfile and extracts the results of the original run. Also the inputs for "determine_basal" like last glucose, IOB array or profile definition are extracted. Here profile is not the usual daily pattern of basals etc. but a collection of settings like which SMB feature is ON or OFF; for details go to the „SMB“ tab in AAPS and scan the section named „Profile“.

After collecting all these inputs the emulator will also read your alternative AAPS settings like

- different temp target
- different SMB settings
- different basal safety factors
- different ISF (here named „sens“ !) or IC values

Settings like those related to meals already eaten or insulin already dosed are handled in other modules of AAPS and are therefore out of reach here. For the same reason changes to Autosense cannot be handled and the only meaningful change you can do is set it to „1“ which means switch it off.

Then determine_basal.py is called to rerun the analysis and produce results in the working folder for comparing the original situation with the emulated scenario.

Comments and current limitations

This is is not a complete virtual loop! Please keep in mind that the variant shows a new and independent decision at each timestep. It is not a sequence of decisions building on each other because there is no forecasting what a different decision would have lead to at the next time step. Therefore you need to carefully consider the results at each timestep separately and consider whether it is safe enough to be activated in AAPS. Once this emulator gives me promising hints and the parameter is tuned I go to quasi "open loop", e.g. for a few days I manually apply an increase in SMB as a correction bolus before going live.

The code still contains dead sections because it is a spinoff from scanAPSlog.py, a more complete analysis of the logfiles. I use that to populate my own local database for treatments, sgvs, etc. independant of Nightscout.

The code contains many (in)active print statements for debugging purposes. I am still learning python and have not yet taken the lesson on proper debugging.

If the logfile starts in the middle of preparing for the loop decision this event lacks required json definitions and may lead to a crash. If you absolutely need that logfile append it to its predecessor logfile and use the combined one.

Be aware that maxBolus=0 in the tabular output file means maxBolus was high enough to not limit the SMB and was therefore not listed explicitly in the logfile by AAPS.

Glucose values are assumed to be in mg/dl.

Tested system settings

- Windows10, 64 bit
- python 3.7
- some python libs like matplotlib (see the import statements)
- AAPS versions for logfile: orefl & SMB for 2.3 / 2.5 / 2.5.1
- AAPS versions for determine_basal: orefl & SMB for 2.3 / 2.5 / 2.5.1

Installation

Create folder containing

- "determine_basal.py" as the emulation of the original java version
- "vary_settings.py" which manages the process
- "<AAPS logfile>" (unzipped) for the period to be investigated
- "<variant_label>.dat" which contains the definition of the settings changes

Define your variant

Create or edit "<variant_label>.dat" to define the changes for your what-if scenario. You have access to the relevant lists or dictionaries as shown in the tab „SMB“, sections „Glucose-data“ through to „Autosense-data“. The general format per row is:

```
.  
.   
<array> <item in array> <new value or formula> <optionally: ### any comment>  
.   
.
```

The entries may be separated by several BLANKS and aligned for better readability. Their meaning is as follows:

- *<array>*
name of python dictionary; the available names are
 - autosens_data do not change; gets calculated inside AAPS elsewhere
 - glucose_status not really meaningfull
 - currenttemp not really meaningfull
 - iob_data do not change; gets calculated inside AAPS elsewhere
 - meal_data do not change; gets calculated inside AAPS elsewhere
 - profile most variations will happen here

If array is not a recognised name the row will be skipped.

Must not contain BLANKS

- *<item in array>*
Item in <array>
Must not contain BLANKS
- *<new value or formula>*
Boolean or numeric value or expression that evaluates accordingly
May contain BLANKS but must not contain '###'

optionally also:

- *<### any comment>* an optional comment after a block of '###'

For detailed content of the arrays and their correct spelling see the original logfile and look for rows

containing "[DetermineBasalAdapterSMBJS.invoke():" or check the appropriate sections in the AAPS "SMB" tab.

CAUTION:

Whereas AAPS will check for limits and deactivate some SMB settings this emulator does not know about them and just goes ahead. Therefore it is your responsibility to check whether AAPS would accept the settings from your variant definition file.

Example: This repo contains the variant definition file „Demo_Sports_Adaptations.dat“. It will be used for a logfile covering 2-3 hours of playing badminton. I like to stay below 140mg/dl because otherwise I lose concentration and react too slow. In this case it worked quite well but what if I tweak things a bit?

```
profile    min_bg          101                ### just above the temp target threshold for SMB
profile    max_bg          profile['min_bg']      ### redefine target
#profile   target_bg       int((profile['min_bg'] + profile['max_bg'])/2)  ### example of numerical expression
profile    tempTargetset   True                ### spelling error
profile    temptargetSet   True                ### i.e. not using NS profile target
profile    allowSMB_with_high_temptarget False        ### was True at the time
```

1. The first row changes the upper target from 100 to 101 to move it just outside of the „enableSMB_with_temptarget“ range.
2. The second row sets the upper target value the same as the lower one with a simple expression. It needs to be the same because I want to simulate a temp target which does not allow for a finite bandwidth if defined in the AAPS GUI.
3. The third row is ignored because „#profile“ is not a recognised dictionary. But just look at the rather complex expression assignment possible. As far as targets are concerned the „target_bg“ is defined as the average of „min_bg“ and „max_bg“ anyway inside determine_basal meaning this row is not required.
4. The fourth row tries to redefine the flag for „tempTargetset“ as True. However, that spelling is not really correct because there are 2 swaps in lower case and upper case spelling.
5. The fifth row keeps the flag for „temptargetSet“ as True and this time in correct spelling. There is no check to verify the target bandwidth is zero in such a case.
6. Finally the sixth row disables SMB for higher TempTargets, i.e. above 100.

In summary the target increases from 100 to 101 and following a general recommendation for sport SMB is now disabled.

For first time use I recommend to leave things unchanged and verify that the results agree with the original. Minor numerical deviations are probably due to roundings in the original logfile.

Execute the analysis

Open a DOS command window, navigate to the working folder containing "<variant_label>.dat" and enter

```
DOS-prompt>python vary_settings.py <AAPS_logfile_name> <plot_options> <variant_label>
```

The meaning of the arguments is as follows:

- <AAPS_logfile_name>
the logfile (unzipped) downloaded from Android
- <plot_options>
"All" or a slash separated list of things to be shown so it does not overload the plot.
The list is any subset from "insReq/maxBolos/SMB/basal/target/bg/pred" where
 - insReq is the total insulin required at the current time step

- `maxBolus` is the upper limit of allowed SMB
- `SMB` is the actual SMB to be delivered
- `basal` is the actual basal insulin to be delivered
- `target` is the lower and upper target
- `bg` is the original glucose value
- `pred` plots the prediction curves per timestep of the emulated scenario including hints if SMB needed to be disabled; you get a quasi animation by using forward/backward keys in the pdf-file created with one page per loop execution

With the special flag „-“ you can suppress individual `<plot-options>`. I normally use „All/-pred“ which results in everything apart from „pred“ and runs faster.

- `<variant_label>`

A label for the result file names and plot. It is also the base name of the variant definition file "`<variant_label>.dat`".

or alternatively:

- `<variant_label>.dat`
this alternative method of giving the filename directly may be easier especially when using the DOS type ahead method

The process will create the following files:

- `<AAPS_logfile_name><variant_label>.log`
echo of parameter changes as requested by `<variant>.dat`
- `<AAPS_logfile_name>orig.txt`
extract of original logfile with entries relevant for the loop result; for the same logfile this will always be the same name and be independent of the variant; therefore there is no harm when a different variant is emulated and it gets overwritten.
- `<AAPS_logfile_name><variant_label>.txt`
equivalent, emulated subset for the variant loop result
- `<AAPS_logfile_name><variant_label>.tab`
comparison of key values showing original versus emulated run; a summary line at the bottom compares total basal and total SMB delivered
- `<AAPS_logfile_name><variant_label>.pdf`
plot of the tabular results; if the previous file with that name is opened in pdf-viewer it will crash

The comparison table file is also output to the screen and always contains all columns.

The plot is saved in a predefined resolution(200dpi) and size(9x12 inches, portrait). Without the „pred“ option it is also displayed interactively and things like aspect ratio or margins can be modified. After such modifications the graph can be saved manually by clicking the disk symbol, preferably in jpeg or png format which makes it easier to later include it in other documents. With the „pred“ option you can inspect it in a pdf-viewer and transverse the history of the logfile with the forward and backward keys.

A file compare between the two txt-files is possible, but mainly the formatting is still different and lists too many lines although numerically equivalent. Personally I find the table and plot more useful.

Example: What do the result files look like for the example from above?

First let us check „AndroidAPS. 2019-11-13 00-00-00 .7.Demo Sports Adaptations.log“:

```
Echo of what-if definitions actioned for variant Demo_Sports_Adaptations created on Sun, 08 Mar 2020 16:36:09 +0100

loop execution in row=2050 at= 2019-11-13T18:23:30Z
edited old value of 100 in profile with min_bg=101
edited old value of 100 in profile with max_bg=101
not actioned: [#rofile], [target_bg], [int((profile['min_bg'] + profile['max_bg'])/2)]
appended new entry to profile with tempTargetset=True
edited old value of True in profile with temptargetSet=True
edited old value of True in profile with allowSMB_with_high_temptarget=False

loop execution in row=5288 at= 2019-11-13T18:28:30Z
edited old value of 100 in profile with min_bg=101
edited old value of 100 in profile with max_bg=101
not actioned: [#rofile], [target_bg], [int((profile['min_bg'] + profile['max_bg'])/2)]
appended new entry to profile with tempTargetset=True
edited old value of True in profile with temptargetSet=True
edited old value of True in profile with allowSMB_with_high_temptarget=False
...
```

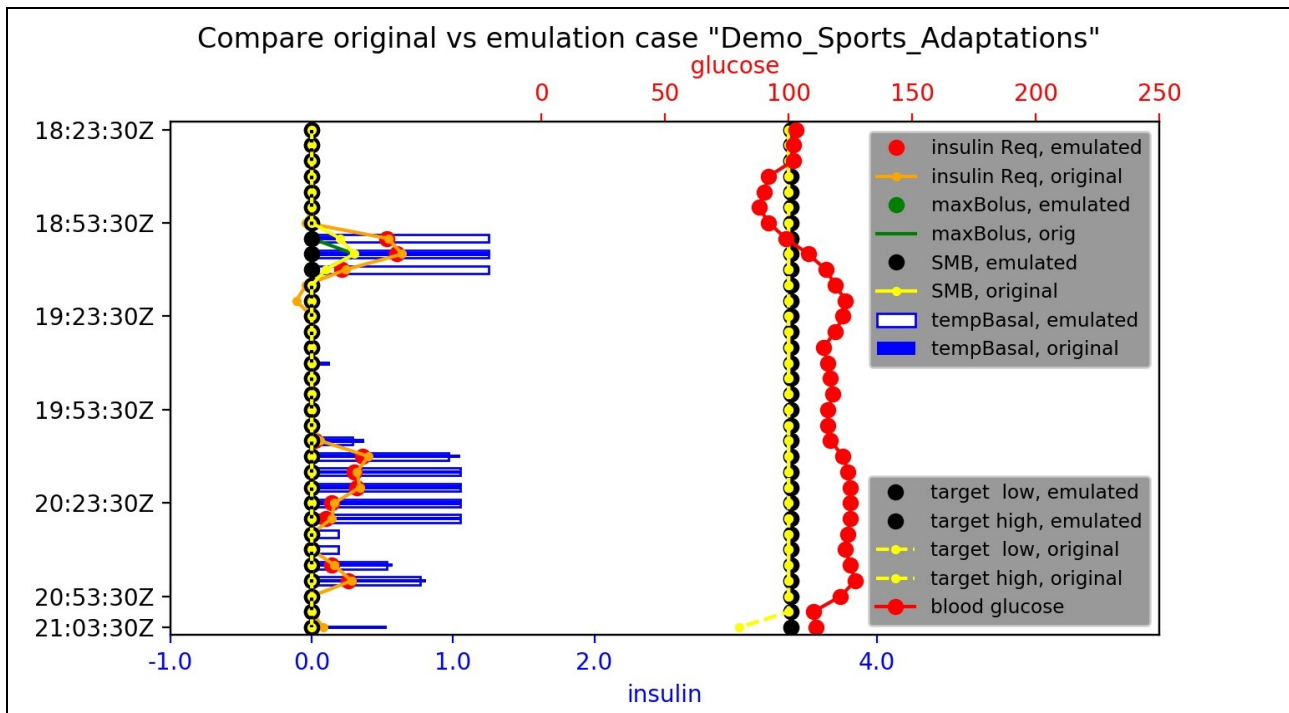
This is an echo of the changes assigned at each timestep. It is useful to check results of complex expressions for values and verify that there were no spelling errors like in the „appended ...“ rows

Next check „AndroidAPS. 2019-11-13 00-00-00 .7.Demo Sports Adaptations.tab“:

		-----target-----		insulin Req		-maxBolus-		---SMB---		---tmpBasal---	
id	time	--UNIXtime--	bg	orig	emul	orig	emul	orig	emul	orig	emul
0	18:23:30Z	1573669410.3	103	100-100	101-101	0	0	0	0	0	0
1	18:28:30Z	1573669710.2	102	100-100	101-101	0	0	0	0	0	0
2	18:33:30Z	1573670010.3	102	100-100	101-101	0	0	0	0	0	0
3	18:38:30Z	1573670310.3	92	100-100	101-101	0	0	0	0	0	0
4	18:43:30Z	1573670610.3	90	100-100	101-101	0	0	0	0	0	0
5	18:48:30Z	1573670910.3	88	100-100	101-101	0	0	0	0	0	0
6	18:53:30Z	1573671210.3	92	100-100	101-101	-0.04	0	0	0	0	0
7	18:58:30Z	1573671510.3	99	100-100	101-101	0.54	0.53	0	0	0.18	1.25
8	19:03:30Z	1573671810.6	108	100-100	101-101	0.64	0.6	0.3	0	1.25	1.25
9	19:08:30Z	1573672110.3	115	100-100	101-101	0.24	0.21	0.1	0	0	1.25
10	19:13:30Z	1573672410.3	119	100-100	101-101	-0.04	0	0	0	0	0
11	19:18:30Z	1573672710.3	123	100-100	101-101	-0.11	0	0	0	0	0
12	19:23:30Z	1573673010.3	122	100-100	101-101	0	0	0	0	0	0
13	19:28:30Z	1573673310.3	119	100-100	101-101	0	0	0	0	0	0
14	19:33:30Z	1573673610.2	114	100-100	101-101	0	0	0	0	0	0
15	19:38:40Z	1573673920.6	116	100-100	101-101	0	0	0	0	0.13	0
16	19:43:30Z	1573674210.4	117	100-100	101-101	0	0	0	0	0.03	0.01
17	19:48:30Z	1573674510.4	118	100-100	101-101	0	0	0	0	0	0
18	19:53:30Z	1573674810.3	116	100-100	101-101	0	0	0	0	0	0
19	19:58:30Z	1573675110.4	116	100-100	101-101	0	0	0	0	0	0
20	20:03:30Z	1573675410.6	117	100-100	101-101	0.06	0.02	0	0	0.37	0.29
21	20:08:30Z	1573675710.4	122	100-100	101-101	0.4	0.36	0	0	1.05	0.97
22	20:13:30Z	1573676010.3	124	100-100	101-101	0.32	0.3	0	0	1.05	1.05
23	20:18:30Z	1573676310.4	125	100-100	101-101	0.34	0.32	0	0	1.05	1.05
24	20:23:30Z	1573676610.3	125	100-100	101-101	0.16	0.14	0	0	1.05	1.05
25	20:28:30Z	1573676910.8	125	100-100	101-101	0.14	0.1	0	0	1.05	1.05
26	20:33:30Z	1573677210.3	124	100-100	101-101	0	0	0	0	0	0.19
27	20:38:30Z	1573677510.4	123	100-100	101-101	0	0	0	0	0	0.19
28	20:43:30Z	1573677810.3	125	100-100	101-101	0.16	0.14	0	0	0.57	0.53
29	20:48:30Z	1573678110.7	127	100-100	101-101	0.28	0.26	0	0	0.81	0.77
30	20:53:30Z	1573678410.3	121	100-100	101-101	0	0	0	0	0	0
31	20:58:30Z	1573678710.3	110	100-100	101-101	0	0	0	0	0	0
32	21:03:30Z	1573679010.7	111	80- 80	101-101	0.08	0	0	0	0.53	0
Totals:								0.6	0.0	0.76	0.91

This table lists the main comparisons between the original case and the emulated case. In the target columns the last row shows that the original target was reduced from 100 to 80 because sport was over. As the variant continued with 101 just ignore that last timestep in the final judgement. The Insulin required is slightly lower as expected due to a slightly higher target. The SMB columns confirm that all of the original SMBs were disabled. The tempBasal now has to offset that lack of insulin around those timesteps and is otherwise slightly reduced because of reduced requirements. The totals row at the bottom gives an indication of the changes but the timing of the major differences is important. While the SMBs of 0.6U were delivered the basals added up to 0.12U in those 15 minutes giving a total of 0.72U. In the variant case the total basal insulin adds up to 0.31U in that short period. So in summary this difference is too small to justify a change in settings which worked well on that day and the next week will be different anyway because the body will be in a different condition. But the process could be repeated for the logfiles of the equivalent activities on other days.

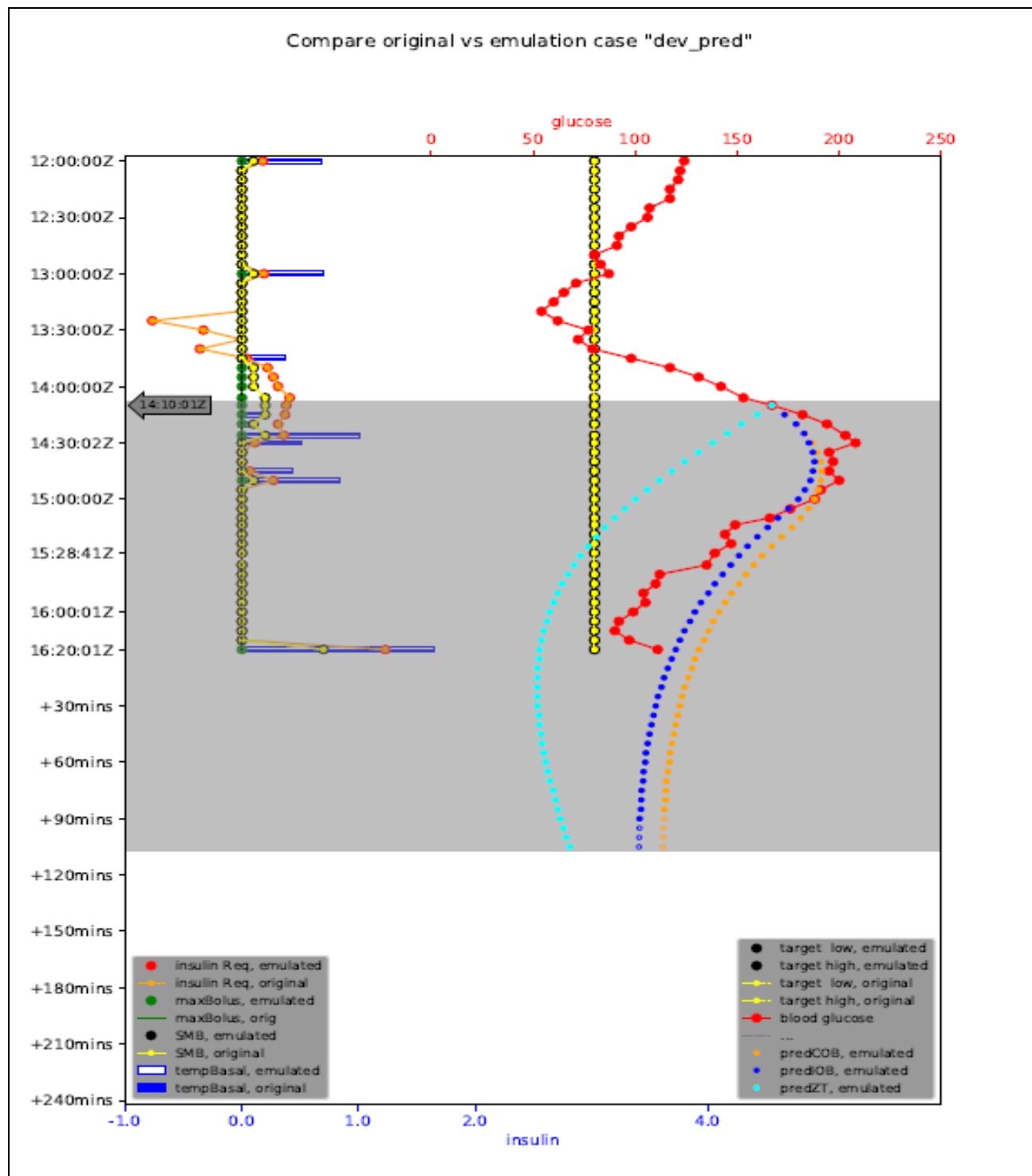
The graphical representation of the comparison tells the same story. The best evaluation method is to look at both, the graph and the table when assessing the comparison. The graph for the example looks like this:



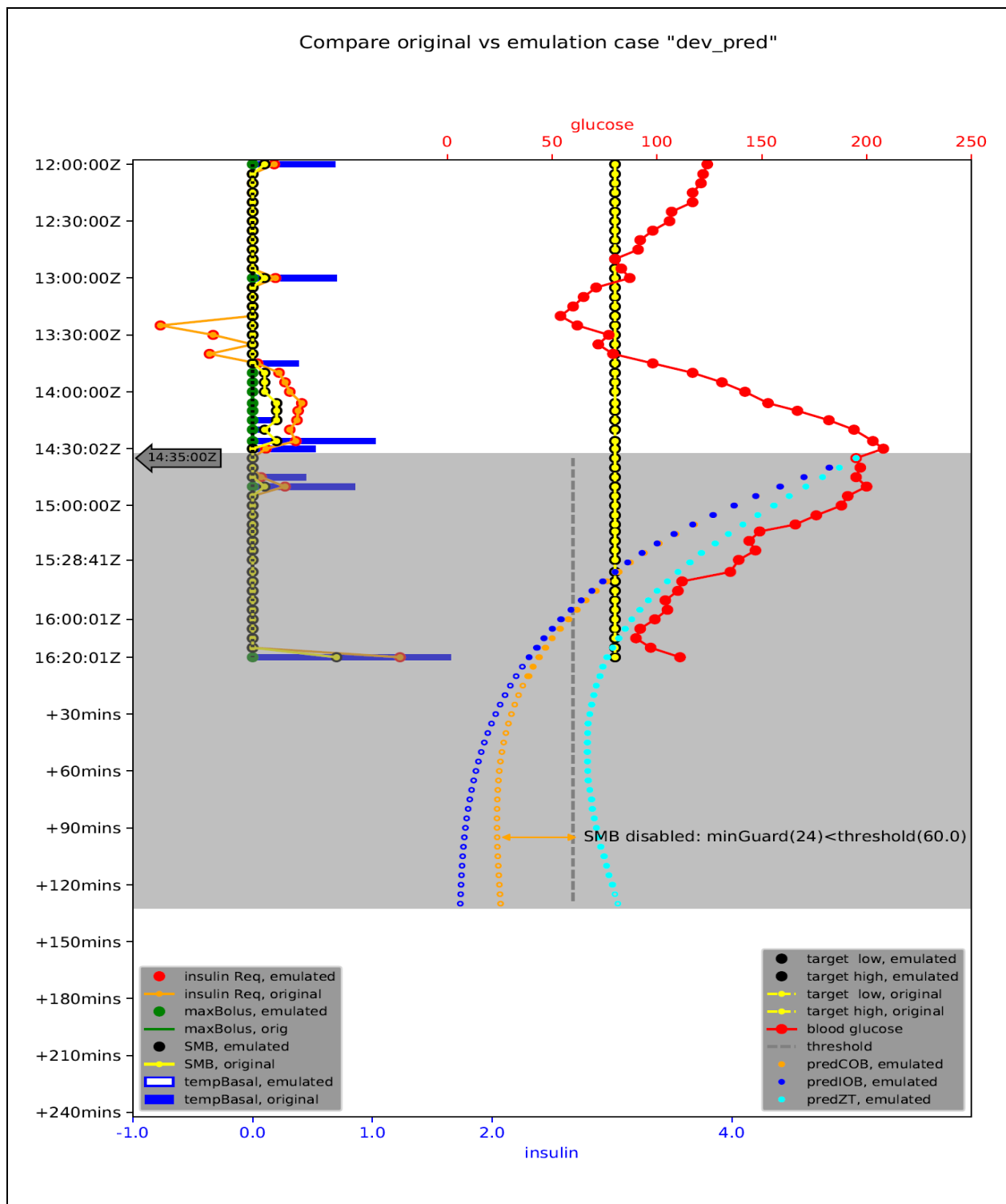
The new „pred“ output option - an early release

It overlays the graph with the prediction lines known from the AAPS “Home“ tab. With the backward and forward arrows in the pdf-viewer you can get a slow motion animation of the history. The hope is that this will improve the understanding of how the loop works. This overlay only shows the emulated case.

The example below is from a different logfile and shows a moment when the predictions matched the future glucose quite well. This means the settings matched real life quite well for more than one hour and that there was no outside disturbance like new carbs or target changes.



Some timesteps later the loop reports in its „SMB“ tab a reason why SMB was disabled. Such a situation is included in the graph whenever it applies to help understanding of the reasoning. What I find interesting is that the loop compares the predictions against the threshold rather than the lower target. That threshold is defined as 50% of the lower target plus 20mg/dl, i.e. always below target!



This graphical sketch of special situations is still under development and needs more debugging.

By the way, open circles in the predictions show the initial calculations which later get capped, truncated and rounded in AAPS and those final values are shown as filled circles.