06/27/16 02:46:18 Untitled

```
/*
  Determine Basal
  Released under MIT license. See the accompanying LICENSE.txt file for
  full terms and conditions
  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
  THE SOFTWARE.
*/
var determine_basal = function determine_basal(glucose_status, currenttemp, iob_data, profile, autosens_data, meal_data, setTempBasal) {
    var rT = { //short for requestedTemp
    };

    if (typeof profile === 'undefined' || typeof profile.current_basal === 'undefined') {
        rT.error ='Error: could not get current basal rate';
        return rT;
    }
    var basal = profile.current_basal;
    if (typeof autosens_data !== 'undefined' ) {
        basal = profile.current_basal * autosens_data.ratio;
        basal = Math.round(basal*100)/100;
        if (basal != profile.current_basal) {
            console.error("Adjusting basal from "+profile.current_basal+" to "+basal);
        }
    }

    var bg = glucose_status.glucose;
    if (bg < 30) {  //Dexcom is in ??? mode or calibrating, do nothing. Asked @benwest for raw data in iter_glucose
        rT.error = "CGM is calibrating or in ??? state";
        return rT;
    }

    var max_iob = profile.max_iob; // maximum amount of non-bolus IOB OpenAPS will ever deliver

    // if target_bg is set, great. otherwise, if min and max are set, then set target to their average
    var target_bg;
    var min_bg;
    if (typeof profile.min_bg !== 'undefined') {
            min_bg = profile.min_bg;
    }
    if (typeof profile.target_bg !== 'undefined') {
        target_bg = profile.target_bg;
    } else {
        if (typeof profile.min_bg !== 'undefined' && typeof profile.max_bg !== 'undefined') {
            target_bg = (profile.min_bg + profile.max_bg) / 2;
        } else {
            rT.error ='Error: could not determine target_bg';
            return rT;
        }
    }


    if (typeof iob_data === 'undefined' ) {
        rT.error ='Error: iob_data undefined';
        return rT;
    }

    if (typeof iob_data.activity === 'undefined' || typeof iob_data.iob === 'undefined' || typeof iob_data.activity === 'undefined') {
        rT.error ='Error: iob_data missing some property';
        return rT;
    }

    var tick;

    if (glucose_status.delta >= 0) {
        tick = "+" + glucose_status.delta;
    } else {
        tick = glucose_status.delta;
    }
    var minDelta = Math.min(glucose_status.delta, glucose_status.avgdelta);
    //var maxDelta = Math.max(glucose_status.delta, glucose_status.avgdelta);

    var sens = profile.sens;
    if (typeof autosens_data !== 'undefined' ) {
        sens = profile.sens / autosens_data.ratio;
        sens = Math.round(sens*10)/10;
        if (sens != profile.sens) {
            console.error("Adjusting sens from "+profile.sens+" to "+sens);
        }
    }

    //calculate BG impact: the amount BG "should" be rising or falling based on insulin activity alone
    var bgi = Math.round(( -iob_data.activity * sens * 5 )*100)/100;
    // project positive deviations for 15 minutes
    var deviation = Math.round( 15 / 5 * ( minDelta - bgi ) );
    // project negative deviations for 30 minutes
    if (deviation < 0) {
        deviation = Math.round( 30 / 5 * ( glucose_status.avgdelta - bgi ) );
    }

    // calculate the naive (bolus calculator math) eventual BG based on net IOB and sensitivity
    if (iob_data.iob > 0) {
```

```
        var naive_eventualBG = Math.round( bg - (iob_data.iob * sens) );
    } else { // if IOB is negative, be more conservative and use the lower of sens, profile.sens
        var naive_eventualBG = Math.round( bg - (iob_data.iob * Math.min(sens, profile.sens) ) );
    }
    // and adjust it for the deviation above
    var eventualBG = naive_eventualBG + deviation;
    // calculate what portion of that is due to bolussnooze
    var bolusContrib = iob_data.bolussnooze * sens;
    // and add it back in to get snoozeBG, plus another 50% to avoid low-temping at mealtime
    var naive_snoozeBG = Math.round( naive_eventualBG + 1.5 * bolusContrib );
    // adjust that for deviation like we did eventualBG
    var snoozeBG = naive_snoozeBG + deviation;


    var expectedDelta = Math.round(( bgi + ( target_bg - eventualBG ) / ( profile.dia * 60 / 5 ) )*10)/10;

    if (typeof eventualBG === 'undefined' || isNaN(eventualBG)) {
        rT.error ='Error: could not calculate eventualBG';
        return rT;
    }

    // min_bg of 90 -> threshold of 70, 110 -> 80, and 130 -> 90
    var threshold = min_bg - 0.5*(min_bg-50);

    rT = {
        'temp': 'absolute'
        , 'bg': bg
        , 'tick': tick
        , 'eventualBG': eventualBG
        , 'snoozeBG': snoozeBG
    };

    var basaliob;
    if (iob_data.basaliob) { basaliob = iob_data.basaliob; }
    else { basaliob = iob_data.iob - iob_data.bolussnooze; }

    // net amount of basal insulin delivered over the last DIA hours
    var hightempinsulin = iob_data.hightempinsulin;

    var wtfAssist=0;
    var mealAssist=0;
    var mealAssistPct = 0;
    //var wtfAssistPct = 0;
    // if BG is high (more than DIA hours of basal above max_bg, i.e. above about 220mg/dL) and rising, wtf-assist
    var high = profile.max_bg + ( basal * (profile.dia) * sens );
    if ( bg > high && minDelta > Math.max(0,bgi) ) {
        wtfAssist=1;
    }
    // minDelta is > 12 and devation is > 50 wtf-assist and meal-assist
    var wtfDeviation=50;
    var wtfDelta=12;
    if ( deviation > wtfDeviation && minDelta > wtfDelta ) {
        wtfAssist=1;
        mealAssist=1;
    } else {
        // phase in mealAssist, as a fraction
        mealAssist = Math.max(0, Math.round( Math.min(deviation/wtfDeviation,minDelta/wtfDelta)*100)/100 );
    }
    var remainingMealBolus = Math.round( (1.1 * meal_data.carbs/profile.carb_ratio - ( meal_data.boluses + Math.max(0,hightempinsulin) ) )*10)/1
        // if minDelta is >3 and >BGI, and there are uncovered carbs, meal-assist
    if ( minDelta > Math.max(3, bgi) && meal_data.carbs > 0 && remainingMealBolus > 0 ) {
        mealAssist=1;
    }
    // when rising with carbs or rising fast for no good reason, meal-assist (ignore bolus IOB)
    if (typeof(meal_data.carbs) == 'undefined') {
        var wtfAssist=0;
        var mealAssist=0;
    }
    if (mealAssist > 0) {
        // ignore all covered IOB, and just set eventualBG to the current bg
        mAeventualBG = Math.max(bg,eventualBG) + deviation;
        eventualBG = Math.round(mealAssist*mAeventualBG + (1-mealAssist)*eventualBG);
        rT.eventualBG = eventualBG;
        //console.error("eventualBG: "+eventualBG+", mAeventualBG: "+mAeventualBG+", rT.eventualBG: "+rT.eventualBG);
    }
    // lower target for meal-assist or wtf-assist (high and rising)
    wtfAssist = Math.round( Math.max(wtfAssist, mealAssist) *100)/100;
    if (wtfAssist > 0) {
        min_bg = wtfAssist*80 + (1-wtfAssist)*min_bg;
        target_bg = (min_bg + profile.max_bg) / 2;
        expectedDelta = Math.round(( bgi + ( target_bg - eventualBG ) / ( profile.dia * 60 / 5 ) )*10)/10;
        mealAssistPct = Math.round(mealAssist*100);
        wtfAssistPct = Math.round(wtfAssist*100);
        rT.mealAssist = "On: "+mealAssistPct+"%, "+wtfAssistPct+"%, Carbs: " + meal_data.carbs + " Boluses: " + meal_data.boluses + " ISF: " + s
    } else {
        rT.mealAssist = "Off: Carbs: " + meal_data.carbs + " Boluses: " + meal_data.boluses + " ISF: " + sens + ", Target: " + Math.round(target
    }

    rT.reason="";
    if (bg < threshold) { // low glucose suspend mode: BG is < ~80
        rT.reason += "BG " + bg + "<" + threshold;
        if ((glucose_status.delta <= 0 && minDelta <= 0) || (glucose_status.delta < expectedDelta && minDelta < expectedDelta) || bg < 60 ) {
            // BG is still falling / rising slower than predicted
            return setTempBasal(0, 30, profile, rT, currenttemp);
        }
        if (glucose_status.delta > minDelta) {
            rT.reason += ", delta " + glucose_status.delta + ">0";
```

```
        } else {
            rT.reason += ", avg delta " + minDelta.toFixed(2) + ">0";
        }
        if (currenttemp.duration > 15 && basal < currenttemp.rate + 0.1 && basal > currenttemp.rate - 0.1) {
            rT.reason += ", temp " + currenttemp.rate + " ~ req " + basal + "U/hr";
            return rT;
        } else {
            rT.reason += "; setting current basal of " + basal + " as temp";
            return setTempBasal(basal, 30, profile, rT, currenttemp);
        }
        /*
        if (currenttemp.rate > basal) { // if a high-temp is running
            rT.reason += ", cancel high temp";
            return setTempBasal(0, 0, profile, rT, currenttemp); // cancel high temp
        } else if (currenttemp.duration && eventualBG > profile.max_bg) { // if low-temped and predicted to go high from negative IOB
            rT.reason += ", cancel low temp";
            return setTempBasal(0, 0, profile, rT, currenttemp); // cancel low temp
        }
        rT.reason += "; no high-temp to cancel";
        return rT;
        */
    }
    // if there are still carbs we haven't bolused or high-temped for,
    // and they're enough to get snoozeBG above min_bg
    //if (remainingMealBolus > 0 && snoozeBG + remainingMealBolus*sens > min_bg && minDelta > Math.max(0,expectedDelta)) {
    if (remainingMealBolus > 0 && snoozeBG + remainingMealBolus*sens > min_bg && minDelta > expectedDelta) {
        // simulate an extended bolus to deliver the remainder over DIA (so 30m is 0.5x remainder/dia)

        //var insulinReq = Math.round( (0.5 * remainingMealBolus / profile.dia)*100)/100;
        var basalAdj = Math.round( (remainingMealBolus / profile.dia)*100)/100;
        if (minDelta < 0 && minDelta > expectedDelta) {
            var newbasalAdj = Math.round(( basalAdj * (1 - (minDelta / expectedDelta)) ) * 100)/100;
            console.error("Reducing basalAdj from " + basalAdj + " to " + newbasalAdj);
            basalAdj = newbasalAdj;
        }
        rT.reason += remainingMealBolus+"U meal bolus remaining, ";
        // by rebasing everything off an adjusted basal rate
        basal += basalAdj;
        basal = Math.round( basal*100 )/100;
        //rT.reason += ", setting " + rate + "U/hr";
        //var rate = basal + (2 * insulinReq);
        //rate = Math.round( rate * 1000 ) / 1000;
        //return setTempBasal(rate, 30, profile, rT, currenttemp);
    //} else if (snoozeBG > min_bg) { // if adding back in the bolus contribution BG would be above min
    }
    if (eventualBG < min_bg) { // if eventual BG is below target:
        if (mealAssist > 0) {
        //if (mealAssist === true) {
            rT.reason += "Meal assist: " + meal_data.carbs + "g, " + meal_data.boluses + "U";
        } else {
            rT.reason += "Eventual BG " + eventualBG + "<" + min_bg;
            // if 5m or 15m avg BG is rising faster than expected delta
            if (minDelta > expectedDelta && minDelta > 0) {
                if (glucose_status.delta > minDelta) {
                    rT.reason += ", but Delta " + tick + " > Exp. Delta " + expectedDelta;
                } else {
                    rT.reason += ", but Avg. Delta " + minDelta.toFixed(2) + " > Exp. Delta " + expectedDelta;
                }
                if (currenttemp.duration > 15 && basal < currenttemp.rate + 0.1 && basal > currenttemp.rate - 0.1) {
                    rT.reason += ", temp " + currenttemp.rate + " ~ req " + basal + "U/hr";
                    return rT;
                } else {
                    rT.reason += "; setting current basal of " + basal + " as temp";
                    return setTempBasal(basal, 30, profile, rT, currenttemp);
                }
            }
        }

        if (eventualBG < min_bg) {
            // if we've bolused recently, we can snooze until the bolus IOB decays (at double speed)
            if (snoozeBG > min_bg) { // if adding back in the bolus contribution BG would be above min
                rT.reason += ", bolus snooze: eventual BG range " + eventualBG + "-" + snoozeBG;
                //console.log(currenttemp, basal );
                if (currenttemp.duration > 15 && basal < currenttemp.rate + 0.1 && basal > currenttemp.rate - 0.1) {
                    rT.reason += ", temp " + currenttemp.rate + " ~ req " + basal + "U/hr";
                    return rT;
                } else {
                    rT.reason += "; setting current basal of " + basal + " as temp";
                    return setTempBasal(basal, 30, profile, rT, currenttemp);
                }
            } else {
                // calculate 30m low-temp required to get projected BG up to target
                // use snoozeBG to more gradually ramp in any counteraction of the user's boluses
                // multiply by 2 to low-temp faster for increased hypo safety
                var insulinReq = 2 * Math.min(0, (snoozeBG - target_bg) / sens);
                if (minDelta < 0 && minDelta > expectedDelta) {
                    // if we're barely falling, newinsulinReq should be barely negative
                    rT.reason += ", Snooze BG " + snoozeBG;
                    var newinsulinReq = Math.round(( insulinReq * (minDelta / expectedDelta) ) * 100)/100;
                    //console.log("Increasing insulinReq from " + insulinReq + " to " + newinsulinReq);
                    insulinReq = newinsulinReq;
                }
                // rate required to deliver insulinReq less insulin over 30m:
                var rate = basal + (2 * insulinReq);
                rate = Math.round( rate * 1000 ) / 1000;
                // if required temp < existing temp basal
                var insulinScheduled = currenttemp.duration * (currenttemp.rate - basal) / 60;
```

```
                if (insulinScheduled < insulinReq - 0.2) { // if current temp would deliver >0.2U less than the required insulin, raise the rate
                    rT.reason += ", "+currenttemp.duration + "m@" + (currenttemp.rate - basal).toFixed(3) + " = " + insulinScheduled.toFixed(3)
                    return setTempBasal(rate, 30, profile, rT, currenttemp);
                }
                if (typeof currenttemp.rate !== 'undefined' && (currenttemp.duration > 5 && rate > currenttemp.rate - 0.1)) {
                    rT.reason += ", temp " + currenttemp.rate + " ~< req " + rate + "U/hr";
                    return rT;
                } else {
                    rT.reason += ", setting " + rate + "U/hr";
                    return setTempBasal(rate, 30, profile, rT, currenttemp);
                }
            }
        }
    }

    // if eventual BG is above min but BG is falling faster than expected Delta
    if (minDelta < expectedDelta) {
        if (glucose_status.delta < minDelta) {
            rT.reason += "Eventual BG " + eventualBG + ">" + min_bg + " but Delta " + tick + " < Exp. Delta " + expectedDelta;
        } else {
            rT.reason += "Eventual BG " + eventualBG + ">" + min_bg + " but Avg. Delta " + minDelta.toFixed(2) + " < Exp. Delta " + expectedDelt
        }
        if (currenttemp.duration > 15 && basal < currenttemp.rate + 0.1 && basal > currenttemp.rate - 0.1) {
            rT.reason += ", temp " + currenttemp.rate + " ~ req " + basal + "U/hr";
            return rT;
        } else {
            rT.reason += "; setting current basal of " + basal + " as temp";
            return setTempBasal(basal, 30, profile, rT, currenttemp);
        }
    }

    if (eventualBG < profile.max_bg || snoozeBG < profile.max_bg) {
        rT.reason += eventualBG+"-"+snoozeBG+" in range: no temp required";
        if (currenttemp.duration > 15 && basal < currenttemp.rate + 0.1 && basal > currenttemp.rate - 0.1) {
            rT.reason += ", temp " + currenttemp.rate + " ~ req " + basal + "U/hr";
            return rT;
        } else {
            rT.reason += "; setting current basal of " + basal + " as temp";
            return setTempBasal(basal, 30, profile, rT, currenttemp);
        }
    }

    // eventual BG is at/above target:
    // if iob is over max, just cancel any temps
    var basaliob;
    if (iob_data.basaliob) { basaliob = iob_data.basaliob; }
    else { basaliob = iob_data.iob - iob_data.bolussnooze; }
    rT.reason += "Eventual BG " + eventualBG + ">=" + profile.max_bg + ", ";
    if (basaliob > max_iob) {
        rT.reason += "basaliob " + basaliob + " > max_iob " + max_iob;
        if (currenttemp.duration > 15 && basal < currenttemp.rate + 0.1 && basal > currenttemp.rate - 0.1) {
            rT.reason += ", temp " + currenttemp.rate + " ~ req " + basal + "U/hr";
            return rT;
        } else {
            rT.reason += "; setting current basal of " + basal + " as temp";
            return setTempBasal(basal, 30, profile, rT, currenttemp);
        }
    } else { // otherwise, calculate 30m high-temp required to get projected BG down to target

        // insulinReq is the additional insulin required to get down to max bg:
        // if in meal assist mode, check if snoozeBG is lower, as eventualBG is not dependent on IOB
        var insulinReq = (Math.min(snoozeBG,eventualBG) - target_bg) / sens;
        if (minDelta < 0 && minDelta > expectedDelta) {
            var newinsulinReq = Math.round(( insulinReq * (1 - (minDelta / expectedDelta)) ) * 100)/100;
            //console.log("Reducing insulinReq from " + insulinReq + " to " + newinsulinReq);
            insulinReq = newinsulinReq;
        }
        // if that would put us over max_iob, then reduce accordingly
        if (insulinReq > max_iob-basaliob) {
            rT.reason += "max_iob " + max_iob + ", ";
            insulinReq = max_iob-basaliob;
        }

        // rate required to deliver insulinReq more insulin over 30m:
        var rate = basal + (2 * insulinReq);
        rate = Math.round( rate * 1000 ) / 1000;

            var maxSafeBasal = Math.min(profile.max_basal, 3 * profile.max_daily_basal, 4 * basal);
        if (rate > maxSafeBasal) {
            rT.reason += "adj. req. rate:"+rate.toFixed(1) +" to maxSafeBasal:"+maxSafeBasal.toFixed(1)+", ";
            rate = maxSafeBasal.toFixed(1);
        }

        var insulinScheduled = currenttemp.duration * (currenttemp.rate - basal) / 60;
        if (insulinScheduled > insulinReq + 0.1) { // if current temp would deliver >0.1U more than the required insulin, lower the rate
            rT.reason += currenttemp.duration + "m@" + (currenttemp.rate - basal).toFixed(3) + " = " + insulinScheduled.toFixed(3) + " > req " +
            return setTempBasal(rate, 30, profile, rT, currenttemp);
        }

        if (typeof currenttemp.duration == 'undefined' || currenttemp.duration == 0) { // no temp is set
            rT.reason += "no temp, setting " + rate + "U/hr";
            return setTempBasal(rate, 30, profile, rT, currenttemp);
        }

        if (currenttemp.duration > 5 && rate < currenttemp.rate + 0.1) { // if required temp <~ existing temp basal
            rT.reason += "temp " + currenttemp.rate + " >~ req " + rate + "U/hr";
            return rT;
```

```
387         }
388
389         // required temp > existing temp basal
390         rT.reason += "temp " + currenttemp.rate + "<" + rate + "U/hr";
391         return setTempBasal(rate, 30, profile, rT, currenttemp);
392     }
393
394 };
395
396 module.exports = determine_basal;
```