# UV Hygrometer WECAN

## 2019-01-07

Fit UV Hygrometer signal to reference mixing ratio measurements (Aerodyne or Picarro).

Aerodyne used as H2O mixing ratio truth for most flights. Picarro2401 is two-second data rate. Using data from Teresa's R0 submission to the archive (after her calibration).

**Using ATH1 for ambient temperature.**

## Initialization

In[139]:=

```
ClearAll[Evaluate[$Context <> "*"]]
AppendTo[$Path, "~/bin/MathematicaPackages/"];
SetDirectory[NotebookDirectory[]];
<< MurphyKoop`
```

In[143]:=

```
kB = QuantityMagnitude[UnitConvert[Quantity["BoltzmannConstant"]]];
```

In[144]:=

```
(* Data file names: 1 Hz netcdf, exported calculated mixing ratio (ppmv),
and calibrated Aerodyne/Picarro file for water vapor reference. *)
datafile = "/Users/Shared/BigStuff/WecanData/WECANrf01.nc";
exportFile = "rf01.txt";
dataFileH2OReference =
  "/Users/Shared/BigStuff/WecanData/AerodyneData_R0_2018-11-06/WECAN-
    CON2O_C130_20180724_R0.ict";
```

In[147]:=

```
(* Cut off first 1000-2000 seconds after takeoff. Seems to take a while
  before the UVhygrometer and the reference aerodyne converge. The fit is
  poor during this time and I don't want to bias the rest of the flight.
   Some noisy flights required skipping even longer time. *)
dropAtStart = 2000;
```

In[148]:=

```
(* The linear decay is used to account for window dirtying and lamp drift,
either up or down. It is applied to the recorded voltage signal as a voltage
 change per second, starting from some point (linearDecayStartTime). *)
linearDecayRate = -1.0 × 10^-6;
```

In[149]:=
```
(* The model equation for the fit and its inversion. *)
model = a + b Exp[-σl x];
densityEqn[v_] = 10^22/(-σl) Log[(v - a)/b];
```

# 1 Hz Flight Data

time = Seconds from midnight = UTC seconds.
ggalt = GPS altitude (m)
tasx = Air speed (m/s)
psx = Ambient pressure PSXC, converted to Pa.
at = Ambient temp from heated Rosemont ATH1 (converted to Kelvin)
xsigv = UVH voltage signal.
xcellpres = Cell pressure (converted to Pa)
xcelltemp = Cell temperature (converted to Kelvin)

In[151]:=
```
(* Get the one sample/second data from the low-rate netcdf file. *)
oneHzDataMatrix = Transpose[
   {Import[datafile, {"Datasets", "Time"}],
    Import[datafile, {"Datasets", "GGALT"}],
    Import[datafile, {"Datasets", "TASX"}],
    Import[datafile, {"Datasets", "PSXC"}] * 100, (*Convert to Pascal *)
    Import[datafile, {"Datasets", "ATH1"}] + 273.15, (* Convert to Kelvin *)
    Import[datafile, {"Datasets", "XSIGV_UVH"}],
    Import[datafile, {"Datasets", "XCELLPRES_UVH"}] * 100, (* Pascal *)
    Import[datafile, {"Datasets", "XCELLTEMP_UVH"}] + 273.15}];

(* iTIME, etc are column labels into the oneHzDataMatrix. *)
{iTIME, iGGALT, iTASX, iPSX, iAT, iXSIGV, iXCELLPRES, iXCELLTEMP} =
  Range[Dimensions[oneHzDataMatrix][[2]]];
```

In[153]:=
```
(*Read the ICARTT file. First number in the file is the number
 of header lines, so drop these, then multiply H2O ppmv by 10^-6.
  Note that the Picarro data for RF10 has the mixing ratio in
 the 5th column rather than the 4th, and more header lines. *)
h2oReference = Import[dataFileH2OReference, "CSV"];
h2oReference = Drop[h2oReference, h2oReference[[1, 1]]];
h2oReference =
  Transpose[{h2oReference[[All, 1]], h2oReference[[All, 4]] * 10^-6}];
h2oMR =
  2;
```

# Match file times

Drop elements from data matrix with earlier start time to match start time of the other matrix. Then repeat with the end times. An if statement which chooses one of the two matrices to drop values from.

In[157]:=
```
If[oneHzDataMatrix[[1, iTIME]] > h2oReference[[1, iTIME]], h2oReference =
   Drop[h2oReference, oneHzDataMatrix[[1, iTIME]] - h2oReference[[1, iTIME]]],
  oneHzDataMatrix = Drop[oneHzDataMatrix,
    h2oReference[[1, iTIME]] - oneHzDataMatrix[[1, iTIME]]]];
```

In[158]:=
```
If[oneHzDataMatrix[[-1, iTIME]] > h2oReference[[-1, iTIME]],
  oneHzDataMatrix = Drop[oneHzDataMatrix,
    -(oneHzDataMatrix[[-1, iTIME]] - h2oReference[[-1, iTIME]])],
  h2oReference = Drop[h2oReference,
    -(h2oReference[[-1, iTIME]] - oneHzDataMatrix[[-1, iTIME]])]];
```

In[159]:=
```
If[Length[oneHzDataMatrix] ≠ Length[h2oReference], "Mismatched Data lengths!"]
```

# Remove invalid data

In[160]:=
```
(* Delete slow airspeeds (takeoff and landings) from fit. *)
validPositions = Position[oneHzDataMatrix[[All, iTASX]], _?(#1 > 70 &)];

(* Report number of lines of entire data matrix, number of valid lines. *)
{Length[oneHzDataMatrix[[All, iTASX]]], Length[validPositions]}

(* Extract the valid lines from both matrices. *)
OneHzDataMatrix = Extract[oneHzDataMatrix, validPositions];
h2oReference = Extract[h2oReference, validPositions];
```

Out[161]=
```
{21004, 20915}
```

In[164]:=
```
 (* Only keep Aerodyne / Picarro when concentration >
  0. (Invalid data flagged as -99999) *)
validPositions = Position[h2oReference[[All, h2oMR]], _?(# > 0 &)];
Length[validPositions]
oneHzDataMatrix = Extract[oneHzDataMatrix, validPositions];
h2oReference = Extract[h2oReference, validPositions];
```

Out[165]=
```
19978
```

In[168]:=

```
(* These two matrices should be the same length if all went right. *)
If[Length[oneHzDataMatrix] ≠ Length[h2oReference],
 "Mismatched Data lengths!", Length[oneHzDataMatrix]]
```

Out[168]=

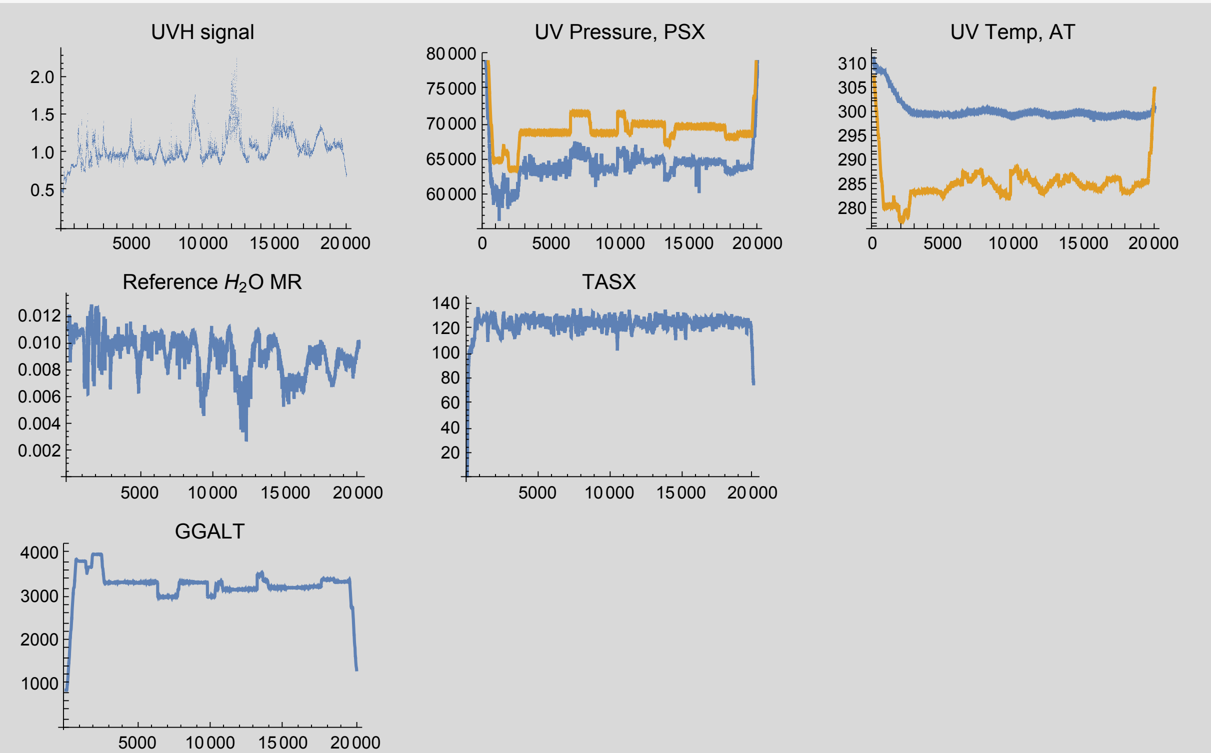
```
19 978
```

# Plot Data

In[169]:=

```
GraphicsGrid[{{ListPlot[oneHzDataMatrix[[All, iXSIGV]],
     PlotLabel → "UVH signal", PlotRange → All],
    ListPlot[{oneHzDataMatrix[[All, iXCELLPRES]], oneHzDataMatrix[[ All, iPSX]]},
     PlotLabel → "UV Pressure, PSX", Joined → True],
    ListPlot[{oneHzDataMatrix[[All, iXCELLTEMP]], oneHzDataMatrix[[All, iAT]]},
     PlotLabel → "UV Temp, AT", Joined → True]},
   {ListPlot[h2oReference[[All, 2]], PlotLabel → "Reference H2O MR",
     Joined → True, Joined → True],
    ListPlot[oneHzDataMatrix[[All, iTASX]], PlotLabel → "TASX",
     Joined → True, PlotRange → All]},
   {ListPlot[oneHzDataMatrix[[All, iGGALT]], PlotLabel → "GGALT",
     Joined → True, PlotRange → All]}}, ImageSize → Full]
```

Out[169]=



# Calculate number density in cell

Find H2O number density $(m^{-3})$ in the sample cell from the reference water vapor mixing ratio, adjusting for the conditions in the sample cell.

In[170]:=

```
densityCell =
  h2oReference[[All, h2oMR]] * oneHzDataMatrix[[All, iXCELLPRES]] / (oneHzDataMatrix[[All, iXCELLTEMP]] * kB);

(* SomethingPlot variables are created for
 convenience in plotting data. Consists of {time,value}. *)
densityCellPlot = Transpose[{oneHzDataMatrix[[All, iTIME]], densityCell}];
```

# Adjust signal voltage for window and O2 absorption

In[172]:=

```
(* Adjust xsigv for lamp decay & window contamination.
  LinearDecayRate (set in Initialization) determined by trial and error.
  LinearDecayStartTime is the first time in the trimmed files. It's needed
 because the output calculations may have a different start time. *)
linearDecayStartTime = oneHzDataMatrix[[1, iTIME]]

(* Temporary variable *)
temp = Table[oneHzDataMatrix[[i, iXSIGV]] + linearDecayRate * i,
   {i, 1, Length[oneHzDataMatrix]}];

(* Atmospheric oxygen also absorbs the UV light,
so adjust xsigv for cell pressure using simple Beer's law. The
 value (-0.25E-5 V/Pascal) was determined by trial and error. *)
xsigvAdj = temp / Exp[-0.25 * oneHzDataMatrix[[All, iXCELLPRES]] * 10^-5];
```

Out[172]=

```
68 756
```

# Fit number density to signal voltage

In[175]:=
```
(* Create data array for fit. Factor out 10^22
 in molecular density so values are near unity. *)
fitArray = Transpose[{10^-22 * densityCell, xsigvAdj}];

(* Remove the first 'dropAtStart' points which never fit well. *)
fitArray = Drop[fitArray, dropAtStart];
Length[fitArray]

(* Calculate least-squares fit. I've modified some of these by hand
  (see below) because the fit is biased by some portions. Also there
  is substantial correlations between the variables such that they can
  differ greatly from the rest of the flights so I'll manually adjust
  them so that the values are similar between different flights. *)
fit = FindFit[fitArray, model, {{a, 0}, {b, 5}, {σl, 0.2}}, x]
```

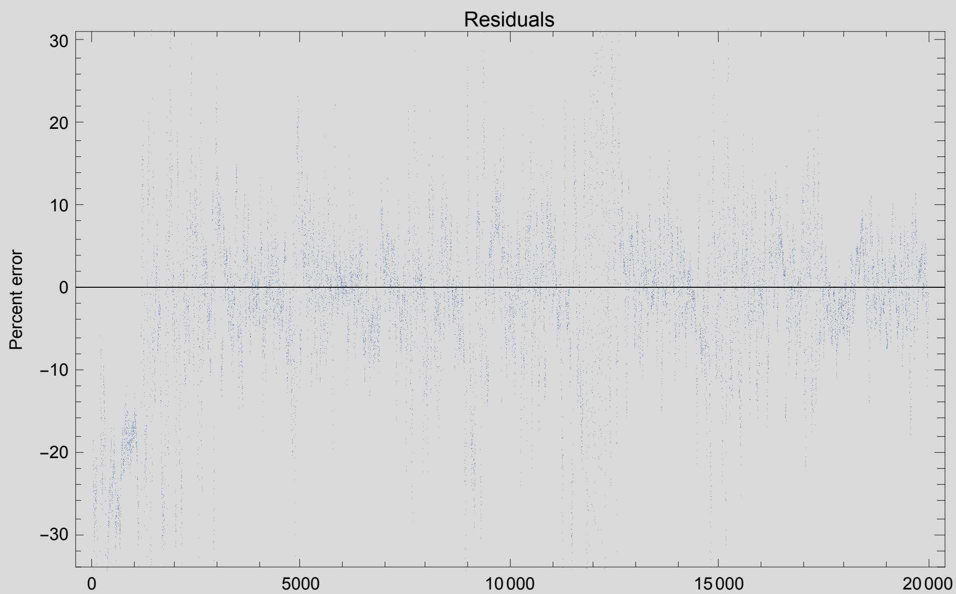
Out[177]=
```
17978
```

Out[178]=
```
{a → -0.40473251, b → 3.149831, σl → 0.045373484}
```

In[179]:=
```
(* Hand-tuned values to use in generating output. See above comments. *)
fit = {a → 0.43, b → 3.28, σl → 0.0989};
```

In[180]:=
```
ListPlot[100 * (densityCell - densityEqn[xsigvAdj] /. fit) / densityCell,
 Frame → True, FrameLabel → {"", "Percent error"},
 PlotLabel → "Residuals", ImageSize → Scaled[0.8]]
```

Out[180]=




# Invert for number density from Voltage

Reread the .nc file for time and uvh values to calculate mixing ratio without any gaps.

In[181]:=
```
calcDataMatrix = Transpose[{Import[datafile, {"Datasets", "Time"}],
    Import[datafile, {"Datasets", "XSIGV_UVH"}],
    Import[datafile, {"Datasets", "XCELLPRES_UVH"}] * 100,
    Import[datafile, {"Datasets", "XCELLTEMP_UVH"}] + 273.15}];
```

In[182]:=
```
(* Drop drop from calcDataMatrix if
 it starts earlier than the h2oReference. *)
If [h2oReference[[1, 1]] > calcDataMatrix[[1, 1]],
 {"Dropping data at start", calcDataMatrix =
   Drop[calcDataMatrix, h2oReference[[1, 1]] - calcDataMatrix[[1, 1]]]}]
```

Out[182]=
```
{Dropping data at start, {{68 809, 0.49843997, 82 711.792, 309.11419},
  {68 810, 0.49917513, 82 864.569, 309.16384}, ⋯ 21 189 ⋯,
  {90 000, -0.011142594, 11 589.89, 352.41116}}}
```

large output | show less | show more | show all | set size limit...

In[183]:=
```
(* Ditto for end of data. *)
If [h2oReference[[-1, 1]] < calcDataMatrix[[-1, 1]],
 {"Dropping lines at end", calcDataMatrix =
   Drop[calcDataMatrix, h2oReference[[-1, 1]] - calcDataMatrix[[-1, 1]]]}]
```

Out[183]=
```
{Dropping lines at end, {{68 809, 0.49843997, 82 711.792, 309.11419},
  {68 810, 0.49917513, 82 864.569, 309.16384},
  ··· 20 878 ···, {89 689, 0.68709797, 86 212.61, 301.74239}}}
```

large output | show less | show more | show all | set size limit...

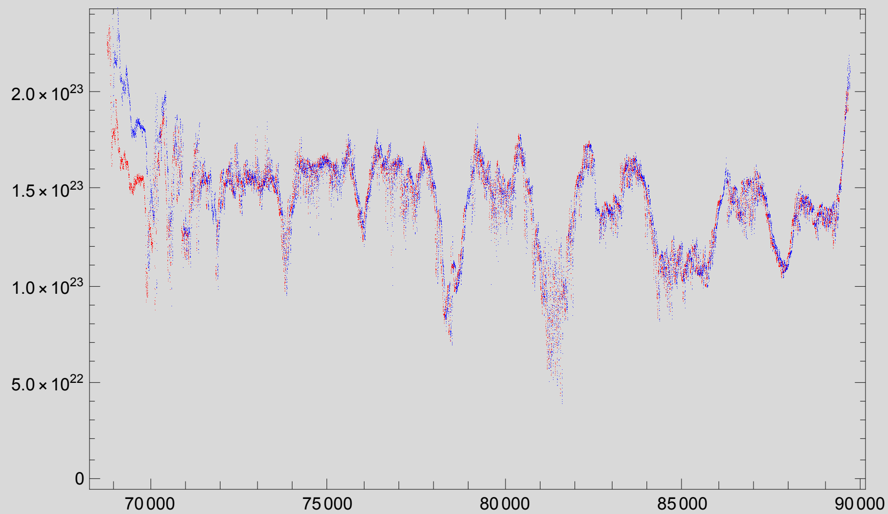
In[184]:=
```
(* Adjust xsigv for lamp decay & window contamination. *)
temp = Table[calcDataMatrix[[i, 2]] +
     linearDecayRate * (calcDataMatrix[[i, 1]] - linearDecayStartTime),
    {i, 1, Length[calcDataMatrix]}];

(* Adjust xsigv for pressure (O2 adsorption). *)
calcXsigvAdj = temp / Exp[-0.25 * calcDataMatrix[[All, 3]] * 10^-5];
```

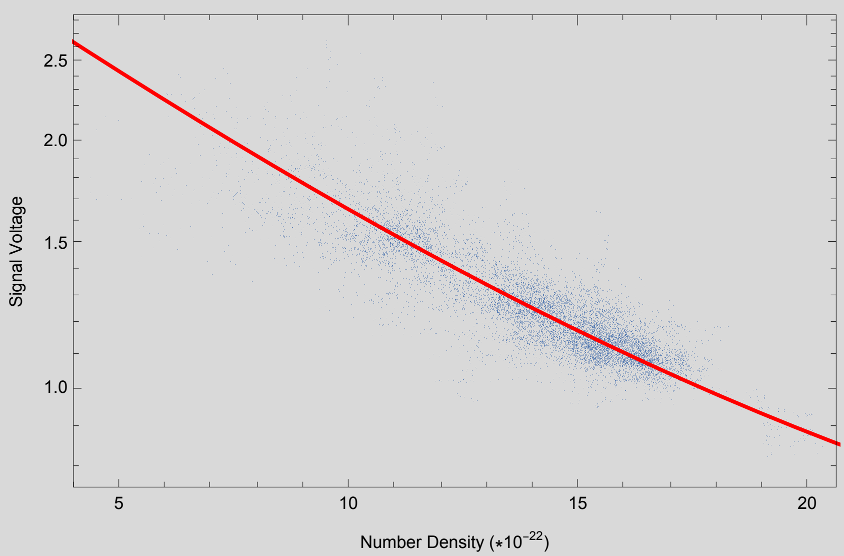
In[186]:=
```
(* Calculate H2O number density in the UVH cell. *)
densityCalc = densityEqn[calcXsigvAdj] /. fit;
densityCalcPlot = Transpose[{calcDataMatrix[[All, 1]], densityCalc}];
```

In[188]:=

```
GraphicsColumn[{ListPlot[{densityCellPlot, densityCalcPlot},
   PlotStyle → {Red, Blue}, Frame → True],
  Show[ListLogPlot[fitArray, Frame → True,
    FrameLabel → {"Number Density (*10^-22)", "Signal Voltage"}, PlotRange → All],
   LogPlot[a + b Exp[-σl x] /. fit, {x, 0.2, 40}, PlotStyle → {Red, Thick}]]},
 ImageSize → Scaled[0.8]]
```

Out[188]=




# Convert Number Density to Mixing Ratio (ppmv)

Calculate mixing ratio present in the sample cell.

In[189]:=

```
(* Need total gas density in the cell. *)
calcCellDensity = calcDataMatrix[[All, 3]] / (calcDataMatrix[[All, 4]] * kB);
mr = densityCalc / calcCellDensity; (* NOT yet in ppm here! *)
mrPlot = Transpose[{calcDataMatrix[[All, 1]], mr}];
Dimensions[mr]
```

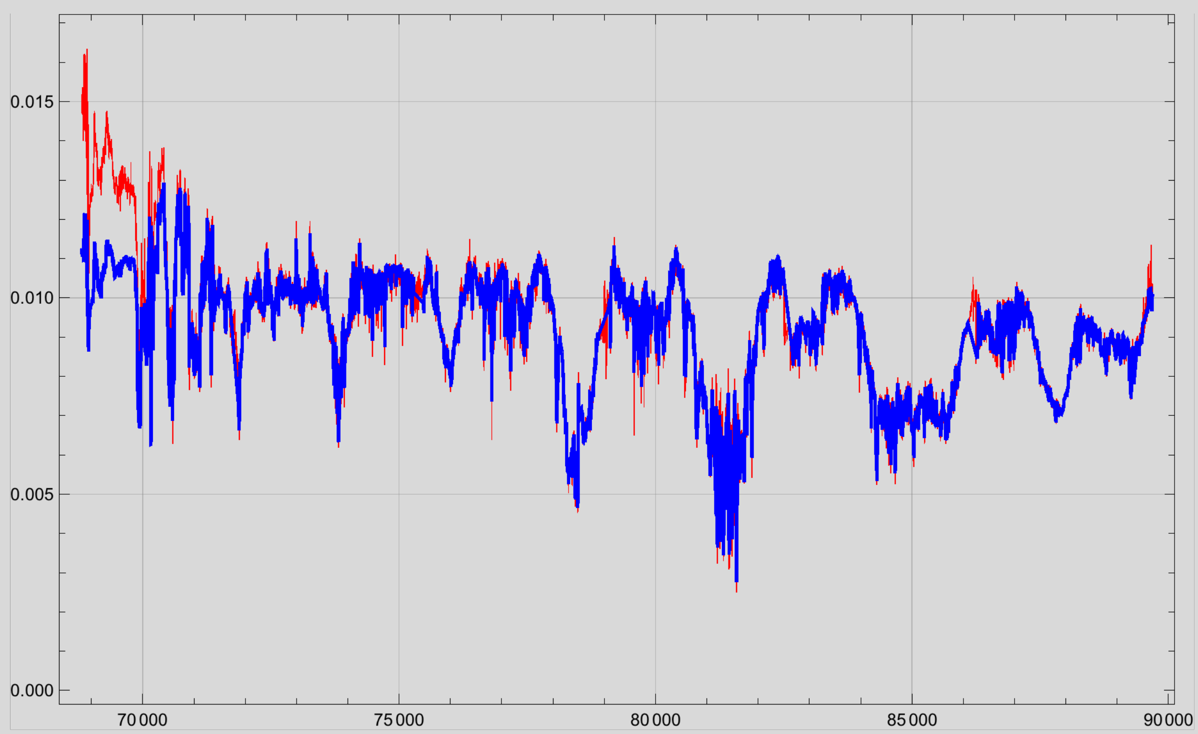
Out[191]=

```
{20 881}
```

In[192]:=

```
ListPlot[{mrPlot, h2oReference}, PlotRange → Automatic, ImageSize → Full,
 Joined → {True, True}, Frame → True, GridLines → Automatic,
 PlotStyle → {{Red, Thickness[0.001]}, {Blue, Thickness[0.003]}}]
```

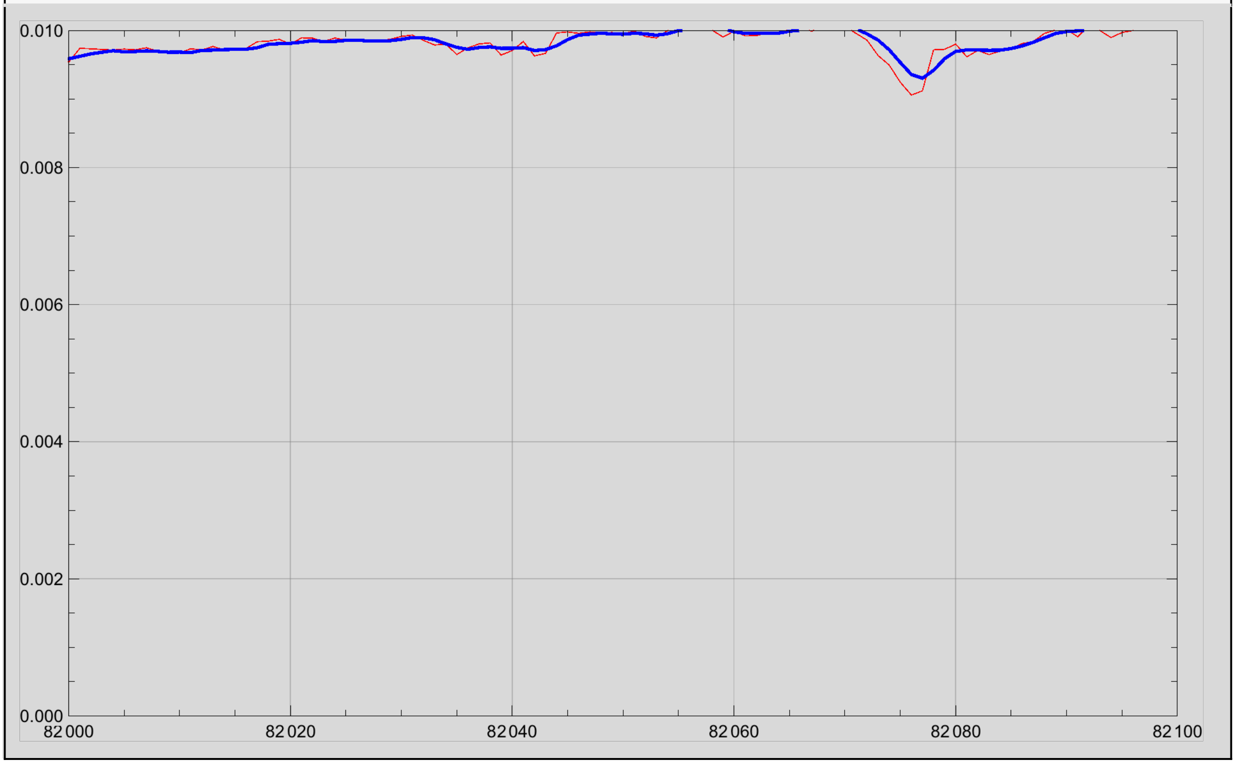
Out[192]=



In[193]:=

```
ListPlot[{mrPlot, h2oReference}, PlotRange → {{82 000, 82 100}, {0.00, .01}},
 ImageSize → Full, Joined → {True, True}, Frame → True, GridLines → Automatic,
 PlotStyle → {{Red, Thickness[0.001]}, {Blue, Thickness[0.003]}}]
```

Out[193]=




# Export data as ascii values of mixing ratio

Export time to two digits past decimal point. Multiply mixing ratio by $10^6$ to get ppmv and export to at least 4 digits.

In[194]:=

```
exportData = Transpose[{mrPlot[[All, 1]], mrPlot[[All, 2]] * 1 000 000.0}];
```

```
Export[exportFile, NumberForm[exportData, {6, 1}], "Table"]
```