P.Kiedron 02/26/2007

Automated radiometric calibration of RSS105

In this document I describe the process that I was performing off-line with the help of four Igor programs since May 2003. Hopefully information contained here will suffice for anybody to replicate the calibration process and write automated or semi-automated calibration software and database. The document should be used with the files containing results of intermediate and final calculations for two cases of calibration (use archives *AutoRadCalibFiles.tgz* that contain 22 text files).

Objectives

From calibration file we want extract the following

(1) Coefficients *k0, k1, k2, C0, DrkSlope*, that go to *to_jrss.txt* file

Note 1: *k0=k2=0*. Only *k1* is non-zero and it's calculated from calibration data. *k1* is entered into *to_jrss.txt* file only after passed through a filter that takes into account previously obtained *k1* values.

and

(2*) Responsivity* in Hz/(W/m2/nm) that goes to *resp_jrss.txt* file.

Files *to_jrss.txt* and *resp_jrss.txt* are inputs to *jrss* program that produces *C1* files and subsequently Langley files from *C1* direct normal irradiances.

Calibrators

Currently two lamp sources are used: PortCal (currently P128) and Licor (currently with bulb ORL787). The number 128 is hard wired and shows up in calibration file

RSS105.20061211_200952.Calibration.PC128.SGP as text

PORTABLE CALIBRATOR = 128

In case of Licor 5-digit number shows up in the file:

RSS105.20061211_201859.Calibration.PC65533.SGP as text

PORTABLE CALIBRATOR = 65533

Note 2: For Licor the number is determined by one of 4 combinations of two switches on the "fake calibrator" attachment. See *InstructionPortCalRSS.pdf* and *InstructionLicorOrielRSS.pdf* at http://www.arm.gov/publications/tech_reports/handbooks/rss/manuals/

The operator at SGP (RSS105 location) can make a mistake resulting in a wrong number in the file. Occasionally the operator my run Oriel spectral Hg-Cd lamp and make the mistake when selecting the switch position. For this reason the program should be able to spot Hg-Cd lamp measurement from data quality. However there is only one Licor and thus it is prudent that any 5-digit number may indicate that a possible Licor calibration took place. The Hg-Cd calibration will be weeded out by the program.

Note 3: We no longer perform routine spectral lamp (Hg-Cd) measurements. The wavelength-to-pixel registration is obtained from Fraunhofer correlation routine that analyzes each solar scan and assigns wavelength-to-pixel registration to it. Since the deployment in May 2003 the spectrum gradually shifted over 2.5 pixels.

Calibrators irradiance scale

A calibrator (lamp) has its irradiance scale assigned to it as a table of irradiance in $W/m^2/nm$ versus wavelength in nm. The irradiance scale may change in time. The lamps in calibrators may be changed. If this would occur we will assign a new irradiance table.

So far the same P128 and the same Licor lamp (#787) were used since May 2003. We monitor once every two months the ratio between the two irradiances when both calibrators are used in a quick succession. The ratios remained constant over 3.5 years to within ±1% (1-sigma) at all wavelengths. (This is an astonishing outcome!)

When a new lamp will be placed in PortCal then, most likely, we will change its internal code and a new number, say P139 would appear. The further complication is that PortCal does not have independently assigned irradiance scale. Its scale is based on Licor and it's transferred via the RSS105 during the first several calibrations.

On the other hand, Licor provides irradiance scale with each of its lamps. If a new lamp is placed in Licor, there will be no change in the 5-digit code. For these reasons we must keep track of lamps and their scales in the software or a database to know which irradiance scale is applicable.

Digression: RSS105 is basically calibrated from a single NIST traceable Licor lamp ORL787.

Exposures and number of scans during calibration

Number of exposures and cycles during calibration is governed by the J command that is part of initialization in *xtty* program. From RSS (Lee Harrison's) firmware manual *UserInfo(21).pdf* available at http://www.arm.gov/publications/tech_reports/handbooks/rss/manuals/

> "J" provides a user command to set the Portable Calibration control variables in the following order: # cycles of the exposure sequence, minimum exposure, exposure increment, and maximum exposure (in hundredths of second). The cycles are limited to 1 ... 50 and all the other numbers must be 10 ... 1791 ... the latter upper limit somewhat arbitrary but this tends to limit really CRAZY total burning times and also prevents endless loops.

The current standard for ASRC's RSS (and what is set by default on reset) is:
J 10 20 20 120 \ ten cycles of 0.20, 0.40, 0.60, 0.80, 1.00, 1.20 sec

We ran " J5 50 50 250" up to about 10/21/2004 and "J 3 20 20 240" thereafter. Currently, when P128 is measured, there are 38 scans in the following sequence of exposures:

240, 240, and 3 times this (20,40,60,80,100,120,140,160,180,200,220,240)

Each scan generates in the file a header and two columns of 1040 numbers. The first column is measured counts with an open shutter (*sig*) and second is with closed shutter (*drk*). Exception are the first two exposures (for PortCal only). They are done with closed shutter to measure a stray light. We haven't seen any stray light, so we do not use the first two exposures. THE FIRST TWO EXPOSURES (SCANS) ARE IGNORED WHEN PROCESSING P128 FILE! So we process only 36 exposures.

When Licor is measured, there are 36 scans in the following sequence of exposures:

3 times this sequence (20,40,60,80,100,120,140,160,180,200,220,240). All of them should be processed.

Parsing calibration file

When calibration file is read and parsed the usual precautions should be used and relied upon: (1) number of expected exposures should be verified, (2) number of rows (1040) should be verified, (3) appropriate sequence of exposures should occur, (4) time stamps should proceed in semi linear way (after taking into account the delay due to exposure value), (4) only two columns should be detected.

Any violation could be a ground for rejection of the calibration file, however one can imagine fixing some departures from the ideal file. This could be done but rather only manually.

Correlation test

This test allows weeding out mislabeled Hg-Cd lamp and some very bad scans. It can be used during file parsing.

Procedure: Correlation

For all scans do the following

```
net=sig-drk
net_shifted=net(pix+30)   //pix are pixels

a=average(net in 100pix-900pix)
b= average(net_shifted in 100pix-900pix)
c=average(net* net_shifted in 100pix-900pix)
```

```
        Correlation=c/(a*b)
End
```

In Figure 1 example of correlation for all 38 scans of P128 is given.

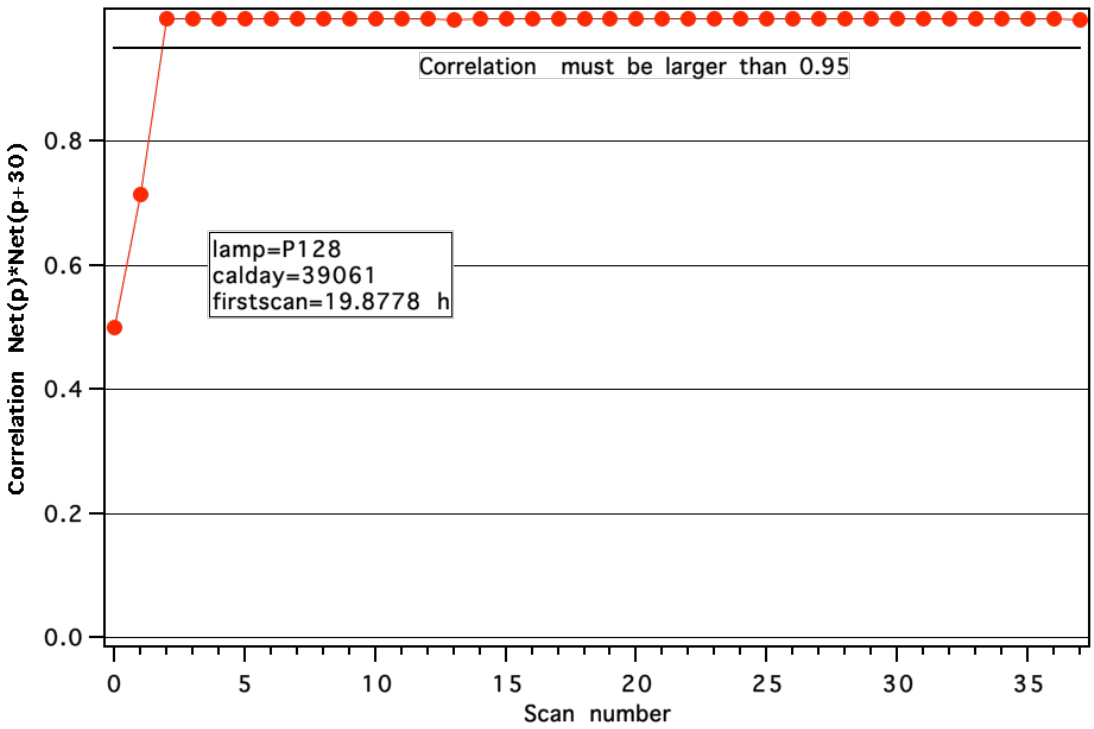

Figure 1. Example of correlation for all 38 scans from P128 measurement. (In the case of P128 correlation for the first two scans do not need to be calculated.)

QC plots

Through this document various plots will be generated that could be used as plots in calibration QC to help the operator in making a decisions. Ultimately the calibration, will not be 100% automated. An operator will have to press the button to accept or reject the calibration on the basis of plots and some numbers. Figure 1 should be a QC plot.

All examples will be based on two calibration files:

RSS105.20061211_200952.Calibration.PC128.SGP

RSS105.20061211_201859.Calibration.PC65533.SGP

All intermediate calculation results are included in the archives: *AutoRadCalibFiles.tgz* They should be used in the process when developing new software.

Another example of QC plot is a plot of all *net=sig-drk* scans (see Fig. 2). One can see quality of data and certainty Figure 2 can help to spot bad scans or mislabeled Hg-Cd file right away.

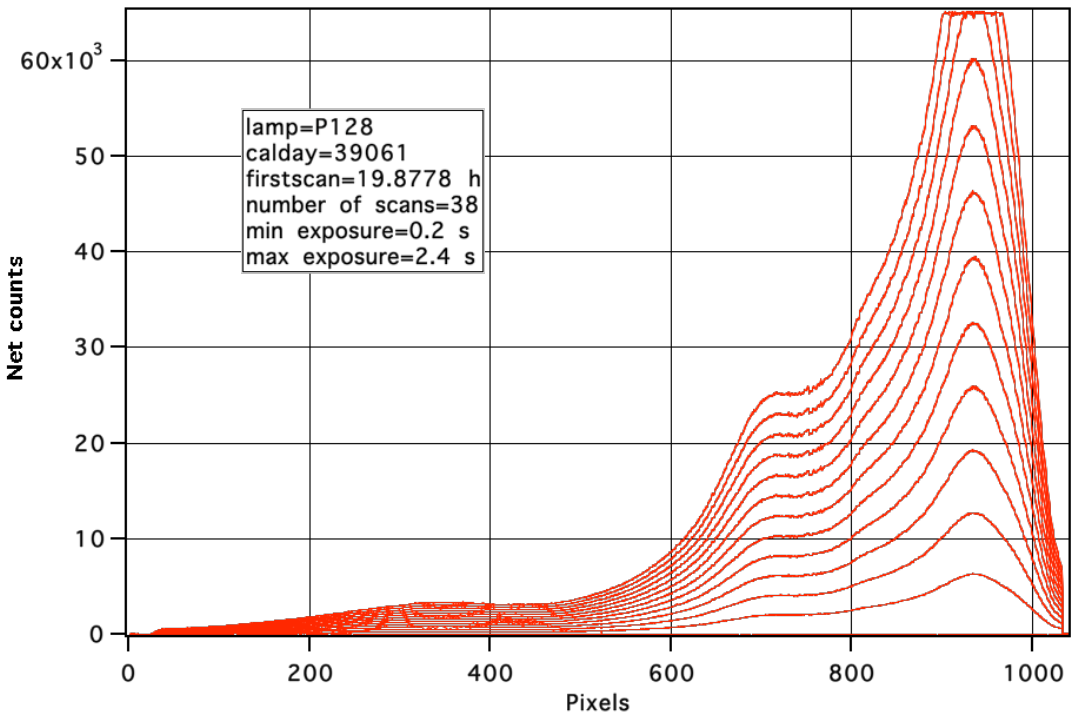

Figure 2. Net counts for all scans. (In the case of P128 the first two scans do not need to be plotted.)

Multiple calibrations per day

When more than one calibration per day is performed, then the resulting responsivities are averaged and only single responsivity per day is assigned in *resp_jrss.txt* file. Also other parameters like *C0*, *DrkSlope* and *k1* are averaged.

Note 4: Currently, SGP operator performs semi-simultaneous PortCal and Licor calibrations once every two months.

Calibration processing: Part 1

The following is based on Igor *ReadCalib_Rss105.pxp* that reads and processes calibration file to the point of generating average counts per second and parameters *k1*, *C0* and *DrkSlope*. The following is a rather free transcription of Igor routines.

Note 5: All arrays in Igor are counted with its index starting at 0.

Note 6: Igor has extended arithmetic with NaN values. All Igor routines (when calculating averages or fits) ignore NaN values in arrays. This is very convenient and it simplifies programming. In other languages this must be handled in other ways and may turn out to be more cumbersome.

Procedure1: Dark Intercept and Slope

For all scans do the following

```
Exposure= Exposure/100   //exposures in seconds
Drk_avg= average(drk in 100pix-900pix)

Plot Drk_avg  vs. Exposure
Fit straight line                  //y=intercept+slope*x
Get C0=intercept   //(in cts)
Get DrkSlope=slope  //(in cts/sec)

//(the fit can be repeated to remove largest residuals)
```

End

In Figure 3 the plot of average dark values and fit results are presented.

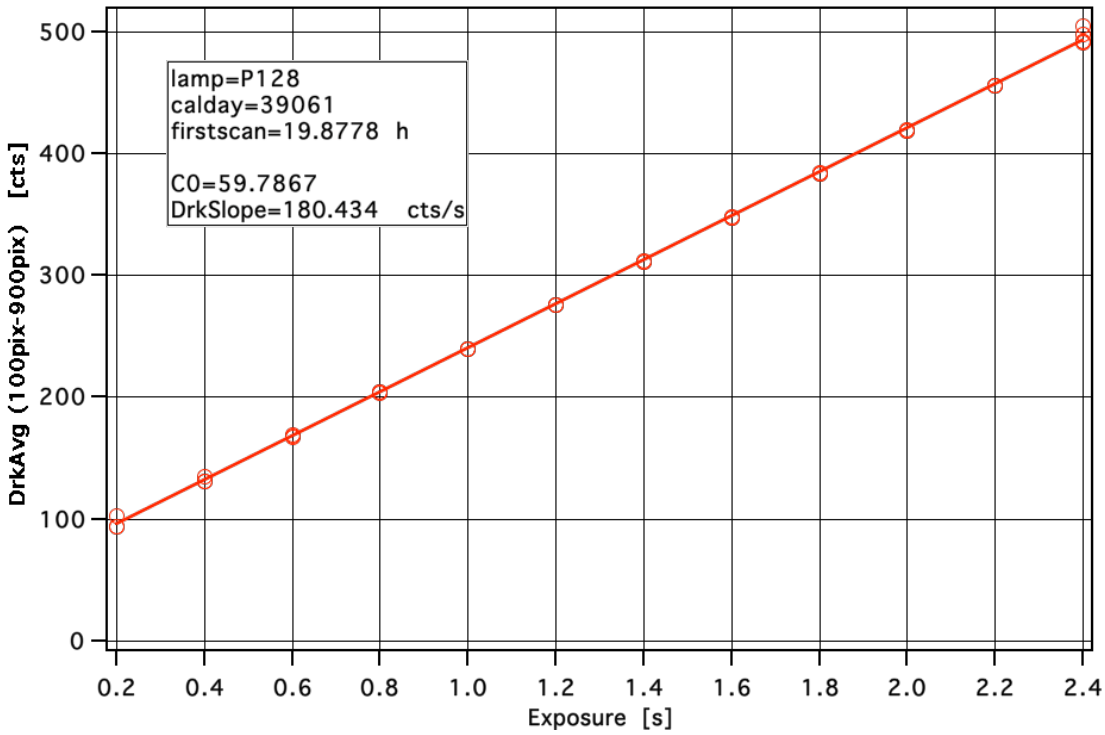

Figure 3. Dark average versus exposure (In the case of P128 dark for the first two scans do not need to be plotted or used in calculations.)

Procedure2: Get averages header parameters

```
Avg_TimeStamp   (Lee jd1900)        //this will be used to get pixel shifts
32 averages for each header entry
Avg_Header[j]   j=0,…,31            //j=1 is CCD temperature
```

End

Output sample Proc1 and 2: See files *Output_P128.txt* and *Output_Licor.txt*

Row 2:	*Avg_TimeStamp*
Row 9:	*C0*
Row 10:	*DrkSlope*
Row 26-57:	*Avg_Header* (32 values)

Procedure3: Get Net cps 2-D Array

```
                                    //Nmax=60000  THIS IS INPUT PARAMETER.
                                    // nscn  of scans  (nscn=38 or 36  currently)

scn=0                               //loop through all the scans
do
        pix=0
        do
                A_net[pix][scn]= (sig[pix]-drk[pix])/Exposure[scn]   //cts per sec
                if(sig[pix]>=Nmax)
                        A_net[pix][scn]= NaN    //saturated CCD
                endif

        pix=pix+1
        while(pix<1040)
        A_net[523][ scn]=(A_net[522][scn]+A_net[524][scn])/2   //bad pixel=523
scn = scn +1
while(scn < nscn)
```

End

Explanation: Usage of this array is for convenience of further programming. It could be generated when ingesting the calibration file.

Output sample Proc3: See files A_*net_P128.txt* and *A_net_Licor.txt*

Procedure4: Get Filtered Net 2-D Array

```
If Pxxx then nk=2 else nk=0 endif   //to exclude first two scans for Pxxx (P128)

make/o/n=(nscn-nk) tempor    //nscn=38 or 36 currently

pix=0
do
        scn=nk
        do


                tempor[scn-nk]=A_net[pix][scn]   //extract rows

        scn=scn+1
        while(scn<nscn)

        wavestats/q tempor    //get stats of tempor  (average V_avg  and location
                              // location of max V_maxloc and
                              //min values V_minloc
        a=V_avg
        tempor[V_maxloc]=NaN
        tempor[V_minloc]=NaN

        wavestats/q tempor   //  V_npnts is number of points that are not NAN's
        if(V_npnts==0)
                tempor=a
        endif

        scn=nk
        do

                Afilt_net[pix][scn]=tempor[scn-nk]

        scn = scn +1
        while(scn<nscn)

pix=pix+1
while(pix<npix)
```

End

Explanation: Afilt_net has min and max removed for each pixel. Thus *Afilt_net* will have 2040 more NaN's than *A_net*.

Output sample Proc4: See files *Afilt_net_P128.txt* and *Afilt_net_Licor.txt*

Procedure5: Get Weighted Average Non-Linearized Net cps

```
If Pxxx then nk=2 else nk=0 endif   //to exclude first two scans for Pxxx (P128)

make/o/n=1040  AvgNet

Numer=0
Denom=0

pix=0
do
        scn=nk
        do

                b= Afilt_net[pix][scn]

                if( b  IS NOT   NaN)          //NaN's are excluded

                        a=sqrt(Exposure[scn])
                        Denom = Denom +a
                        Numer = Numer +a*b
                endif

        scn= scn+1
        while(scn<nscn)

        AvgNet[pix]= Numer / Denom          //average weighted by sqrt of exposures

pix=pix+1
while(pix<1040)
```

End

Explanation: the average is weighted by square roots of exposures because standard deviation of Poisson noise is proportional to square roots of exposure.

Output sample Proc5: See files *Output_P128.txt* and *Output_Licor.txt* The *AvgNet* is in rows: 59-1098. In the file *OutArray_111CalsTill39106.txt* there 111 processed calibrations in the same format as the above files.

Procedure6: Get Non-Linearity Coefficient k1

```
make/o/n=(nscn-nk) tempor, temporx

make/o/n=(15*10)  gfun, cfun   //150 long arrays; may contain NaN's

k=0
do
        scn=nk
        do

                pix=0
                do
                        net[pix]=Afilt_net[pix][scn]

                pix=pix+1
                while(pix<npix)
                net=net+DrkSlope   // adding drk, i.e.,  DrkSlope from Procedure1
                temporx[scn-nk]=Exposure[scn]
                wavestats/q/R=(100+50*k,150+50*k) net   //to get avg in 50 pixel intervals
                tempor[scn-nk]=V_avg* Exposure[scn]      // avg times exposure (tot counts)

        scn=scn+1
        while(scn<nscn)

        sort tempor, tempor, temporx    //this sorting (ascending) is to weed out NaN's  (NaN's will
                                        //be at the end Igor sorting)
        wavestats/q tempor              //this return  V_npnts which  is number of non NaN points
        redimension/n=(V_npnts) tempor, temporx   //cuts off NaN's
                                //last three lines are not necessary in Igor

        wavestats/q temporx         //gets min and max values
        a1=V_min
        a2=V_max
        CurveFit/q poly 3,  tempor /X=temporx /D      //fits trinomial  to counts  vs. exposure
                                                      // return wc[0], wc[1], wc[2] coefficients


        //the following prepares gfun and cfun to obtain k1
        i=0
        do
                xx=a1+i*(a2-a1)/9
                yy=(xx*1.05-xx*.95)/xx
                yy=yy*poly(wc,xx)/(poly(wc,xx*1.05)-poly(wc,xx*.95))
                cfun[10*k+i]=poly(wc,xx)
                gfun[10*k+i]=yy-1   // g(C)= g(C) –1 substitution

        i=i+1
        while(i<10)
        // where poly(wc,xx)=wc[0]+wc[1]*xx+wc[2]*xx^2

k=k+1
while(k<15)

Do linear fit gfun=k1* cfun with intercept set to 0   //intercept must be 0!  (See Appendix I)
```

End

Explanation: First we obtain averages of counts in 50 pixel intervals. These counts have *DrkSlope* added and are multiplied by exposure. There is 15 intervals (100, 150), (150,200),…,(800,850). For each interval we expect to have *nscn-nk*=36 averages for various exposures unless some interval's average is NaN (like for large exposure). This may happen in regions where CCD saturates. For each interval we fit a trinomial to counts vs. exposure. We end up with 15 trinomials.

For each interval we use the trinomial function to build *gfun* and *cfun* arrays. The first is dimensionless and the second is in counts. Each array is 10 long. The exposure range is divided by 10 equidistant points *xx* and the counts *cfun* are equal to the value of the trinomial at *xx*.

The *gfun* array is more tricky. The approach stems from the solution to the difference/differential equation. It is explained by equations (2-4) in the publication *SPIE02_4815_13_kiedron.pdf* available at:
http://www.arm.gov/publications/tech_reports/handbooks/rss/publications/

The reasoning is repeated here. Let's *C(I)* denotes counts at a given pixel or pixel interval due to irradiance *I*. There exist a function *f(C)=I* that for any *C* returns *I*. This function is an inverse of *C(I)*. Let us define the function *g(C)* as follows from a difference equation:

$$g(C) = C(I_k) \approx \frac{I_k - I_{k+1}}{I_k} \frac{C(I_k)}{C(I_k) - C(I_{k+1})}$$

that we change to the differential equation

$$g(C) = \frac{dI}{I}\frac{C}{dC} = \frac{df(C)}{f(C)}\frac{C}{dC} = \frac{C}{f(C)}\frac{df(C)}{dC}$$

This differential equation has the following solution with respect of *f(C)*:

$$f(C) = A \cdot C \cdot \exp\left(\int \frac{g(c)-1}{c} dc\right)$$

If in the last equation we set *g(C)-1=k1*C*, then

$$f(C) = A \cdot C \cdot \exp(k1 \cdot C)$$

This is the linearizing function *CountsCorrected(cts, YesOrNo, k0,k1,k2)* with *k0=k2=0* that is used by the *jrss*. It is in the original document *RSS105SignalNoise_Oct04_1pk.pdf* that gave basic equations used by *jrss*. (see Appendix I or
http://www.arm.gov/publications/tech_reports/handbooks/rss/manuals/)

It has to be emphasized that irradiances I_k are not available but the exposures E_k can serve as their proportional surrogates (keep in mind that $g(C)$ is a dimensionless function):

$$\frac{E_k - E_{k+1}}{E_k} = \frac{I_k - I_{k+1}}{I_k}$$

Output sample Proc6: See *cfun_gfun_P128.txt* and *cfun_gfun_Licor.txt*. The value of *k1* is in *Output_P128.txt* and *Output_Licor.txt* in row 18.

In Figure 4 a fit between gfun and cfun is presented.

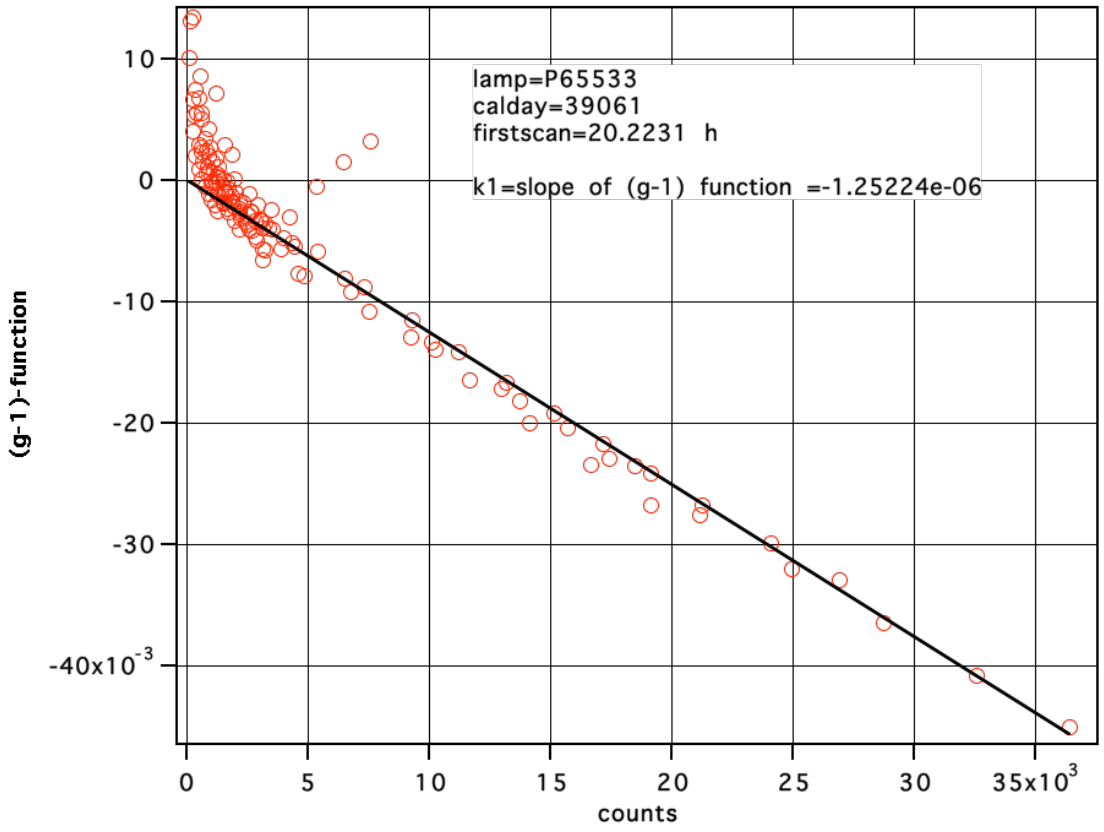

Figure 4. *g(C)-1* plot against C counts. The slope of the linear fit is k1 coefficient.

The outliers between 5k-10k cts are from 750-800 pixel interval. Apparently in that region linearity behavior might be different. This is more pronounced for P128 calibrator that has different color temperature and thus different count distribution in this interval. Nevertheless we use single non-linearity correction for all pixels. It seems to work and trying to make different corrections in different pixel regions would generate an impossible amount of testing.

Procedure7: Get Linearized Net Array

```
scn=nk
do
        pix=0
        do

                net[pix]=Afilt_net[pix][scn]

        pix = pix +1
        while(pix <npix)

        net=CountsCorrected((net+DrkSlope)*Exposure[scn],1,0,k1,0)  //see Appendix I for
                                                                    //this  function

        net=net- DrkSlope * Exposure[scn]
        net=net/ Exposure[scn]

        pix=0
        do

                Alin_net[pix][scn]=net[pix]

        pix = pix +1
        while(pix<npix)

scn = scn +1
while(scn <nscn)
```

End

Procedure8: Get Weighted Average Linearized Net cps

```
        If Pxxx then nk=2 else nk=0 endif   //to exclude first two scans for Pxxx (P128)

        make/o/n=1040  AvgNetLin

        Numer=0
        Denom=0

        pix=0
        do
                scn=nk
                do
                        b= Alin_net[pix][scn]

                        if( b  IS NOT   NaN)          //NaN's are excluded

                                a=sqrt(Exposure[scn])
                                Denom = Denom +a
                                Numer = Numer +a*b
                        endif

                scn= scn+1
                while(scn<nscn)

                AvgNetLin[pix]= Numer / Denom      //average weighted by sqrt of exposures
        pix=pix+1
        while(pix<1040)
```

End

Note 7: Procedure 8 is procedure 5 where instead of *Afilt_net Alin_net* is used

Output sample Proc7 and 8: See files *Output_P128.txt* and *Output_Licor.txt.* The *AvgNetLin* is in rows: 1100-2139.

To demonstrate that linearization works we generate two plots in Figure 5 where we plot ratios of *net/AvgNet* and *net_linearized/ AvgNetLin* for all exposures, where net are columns in *Afilt_net* and *net_linearized* are columns in *Alin_net.* Then the ratios are smoothed with 25 pixel moving average. The Figure 5 is the proverbial pudding.

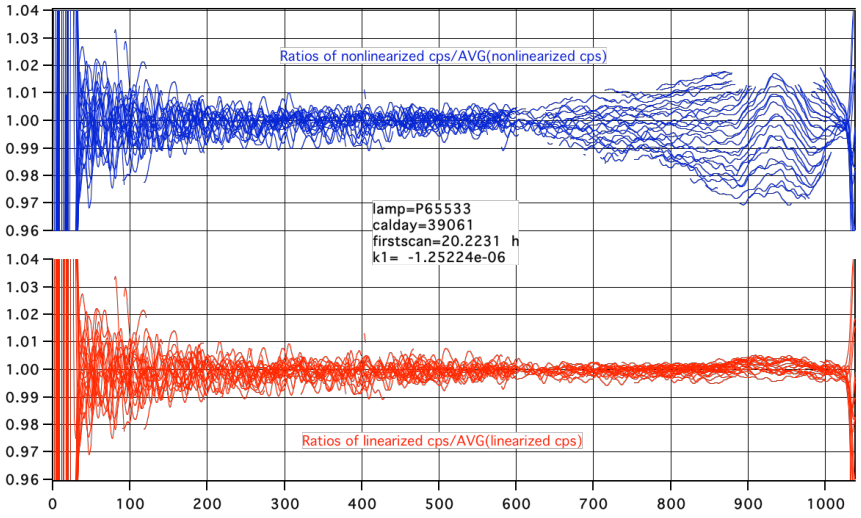

Figure 5. Rations of *net/AvgNet* and *net_linearized/ AvgNetLin.*

Also it might be useful to plot as part of QC *AvgNet* and *AvgNetLin* and their ratio (see Figure 6).

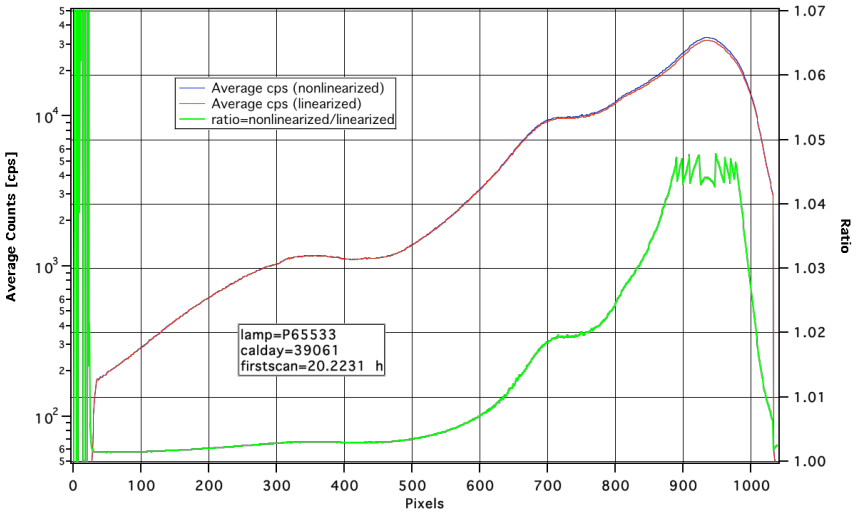

Figure 6. *AvgNet*, *AvgNetLin* and ratio *AvgNet/ AvgNetLin*

Note 8: *AvgNet* will not be used hereafter! It's value has no effect on C1 files.

Calibration processing: Part 2

The following is based on three Igor programs

Process1_nmCorrect.pxp
Process2_Trnds.pxp
Process3_ApplyIrrad.pxp

However we introduced some modifications that may results that the new resposivities might be negligibly different from the old ones.

Inputs

The following inputs are necessary:

AvgNetLin, Avg_TimeStamp, nm_precursor, LampIrrad, dpixBlue, dpixRed

Note 8: Irradiance files are prepared so *LampIrrad* is sampled in *nm_precursor*. Obviously this is not the way irradiance scale comes from Licor, but for convenience I prepared them that way.

Obtaining responsivity

Step 1: From *TimeStamp* obtain pixel shifts *dpixBlue, dpixRed*. This program already exists in *jrss*.

Step 2: Calculate *NewNM* from *nm_precursor* and *dpixBlue, dpixRed* using wavelength interpolating program *getNewNM(dpixBlue, dpixRed, "nm_precursor")* (see Appendix I)

NewNM is wavelength-to-pixel assignment during calibration.

Input data are from: *Irrad_Licor787.txt*, *Irrad_P128.txt*, *nm_precursor.txt*. The following pixel shifts were used in examples:

(*dpixBlue, dpixRed)*=(-2.5187,-2.7404) for Licor
(*dpixBlue, dpixRed)*=(-2.6248,-2.9032) for P128

Step 3: from *AvgNetLin* (it is in *NewNM* grid) obtain *AvgNetLin(nm_precursor)* in *nm_precursor* grid. Use equivalent of Igor linear *interp* function:

AvgNetLin(nm_precursor)=interp(nm_precursor, NewNM, AvgNetLin) where

AvgNetLin is from Output_Licor.txt and *Output_P128.txt* files.

Step 4: Calculate *Responsivity* in *nm_precursor* grid:

Responsivity = AvgNetLin(nm_precursor)/LampIrrad

The results are in *Results_Part2.txt* where there are six 1040 long arrays. Among them: *Responsivity* and *NewNM* for both Licor and P128 and the average responsivity. The average goes to the *resp_jrss.txt*.

Step 5: If more then one calibration in one day get averages of: *C0, DrkSlope, k1, Responsivity*. Also the average of *NewNM* can be performed. (*NewNM* is not used by *jrss*!)

Step 6: The filtered *k1* goes to *to_jrss.txt* file. Currently the filter consist of straight line fit to the current and the nearest (in time) four *k1* values vs. time. The results are in Figure 6A in Appendix 2. It seem that a simple box car 2 or 3 point average can do a good job as well, particularly when there is no steep trends as during the first 3-4 months after the deployment (see Figure 6A).

Step 7: Generate various QC trend plots. It is the last chance to reject calibration. (See Appendix II for the trend plots)

Step 8: Generate *to_jrss.txt* and *resp_jrss.txt* files. In *resp_jrss.txt* set the middle column to 0; first column is *NewNM* (not used) and the last column is *Responsivity*; in the header there are pixel shifts. All values are averages (including shifts and *NewNM*) in case of multiple calibrations in one day. Keep in mind there is only one entry per day in the file.

Since some changes were introduced when transcribing the Igor procedures we obtained slightly different results. The differences are negligible (see Figure 7).

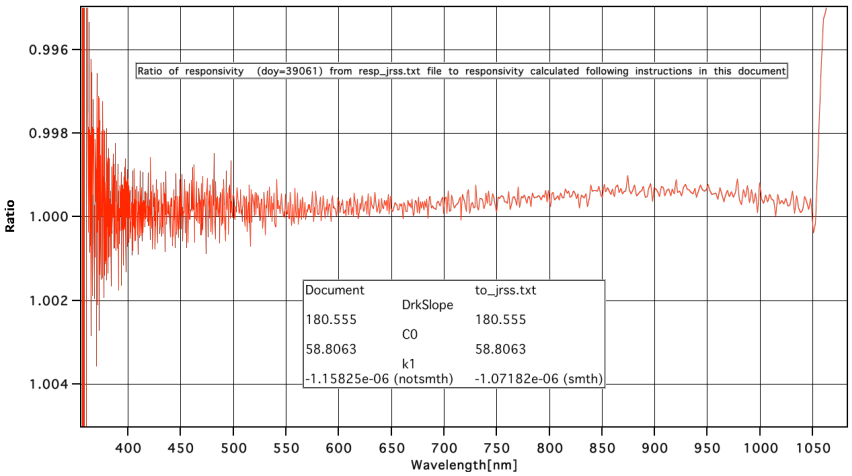

Figure 7. Comparison of results following this document and those in existing *resp_jtss.txt* and *to_jrss.txt* files

Appendix I: Some functions

Linear fit with fixed intercept

Say we have two arrays $\{y_i\}$ and $\{x_i\}$ and fit straight line $y=a*x+b$ *where* b has a fixed value.

To minimize the equation

$$f(a,b) = \sum \left(y_i - a \cdot x_i - b\right)^2$$

with respect of a we get derivative

$$\frac{\partial f}{\partial a} = -2\sum \left(y_i - a \cdot x_i - b\right) \cdot x_i$$

that leads to the following solution for a

$$\frac{\partial f}{\partial a} = 0 \quad \Rightarrow \quad \sum (y_i - b) \cdot x_i - a \sum x_i^2 = 0$$

$$a = \frac{\sum (y_i - b) \cdot x_i}{\sum x_i^2}$$

Linearizing function

```
//This func returns corrected counts
function  CountsCorrected(cts, YesOrNo, k0,k1,k2)   //Example of parameters for rss104 would be {1,0,-12.5e-07,0}
variable cts, YesOrNo, k0,k1,k2

variable r

r=cts

if(YesOrNo==1)
        if(cts>0)  // This is so because cts^k0 can get flaky for cts<0 and even for cts=0

                r=cts*cts^k0*exp((k1+k2*cts)*cts)

        endif

endif

return r

End
```

Wavelength interpolating function

```
function  getNewNM(dp1,dp2, s_OldNM)
          variable dp1,dp2                    // shifts at 0 (Blue) and 1039 (Red) pixels
          string   s_OldNM              // name of oldNM wave as string

          wave  OldNM=$s_OldNM    // passing global wave

          make/o/n=1040  pix, dpix    // local wave

          pix=p                                 // pix=0,1,...,1039
          dpix=pix-((dp2-dp1)/1039*pix+dp1)          // dpix  is pix shifted ( There is a "minus"
                                                     //sign because of the nature of
                                                     //Fraunhofer! Verified empirically!)

          make/o/n=1040  NewNM

          NewNM=interp(dpix,pix, OldNM)  //Igor linear interpolating function

End
```

Example of usage:

getNewNM(dpixBlue, dpixRed "nm_precursor")

Appendix II: Trend plots for QC

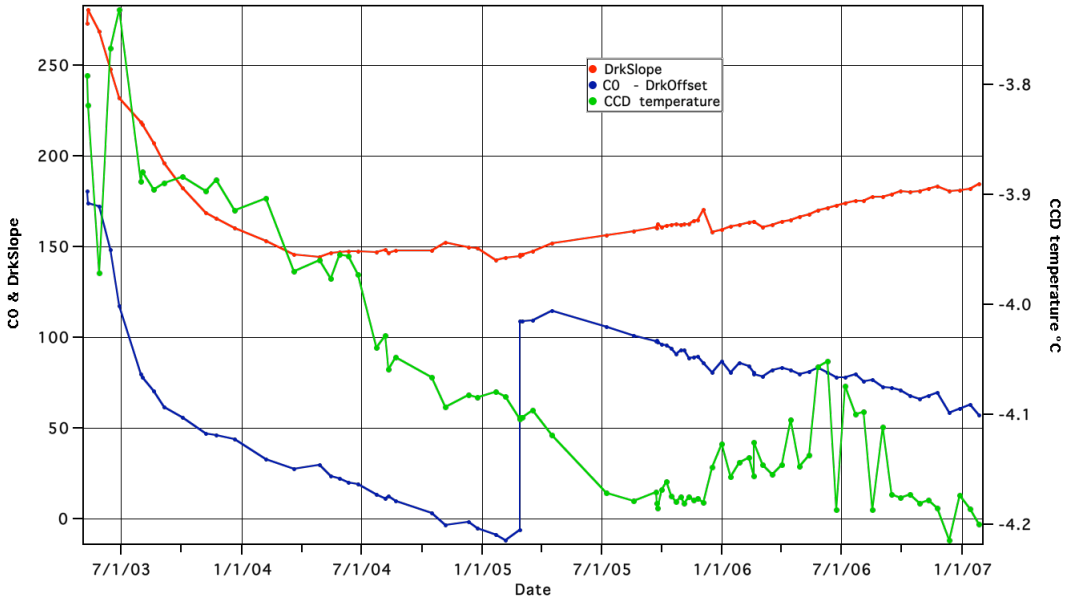

Figure 1A. Shows trend of C0, DrkSlope and CCD temperature

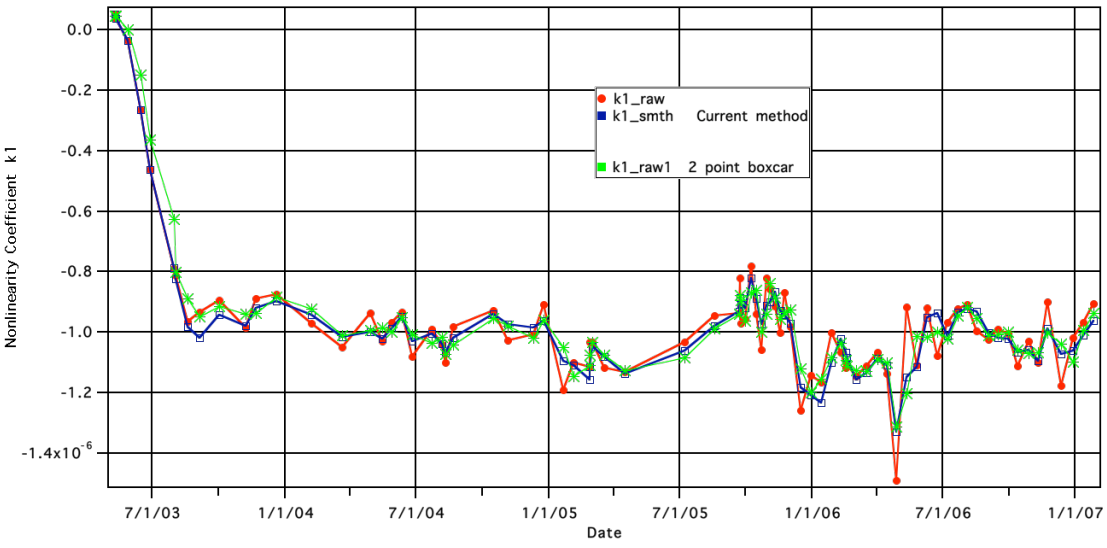

Figure 2A. Trend of k1 coefficient: raw data (red), smoothed with current method (blue), 2 pnts boxcar smth (green)

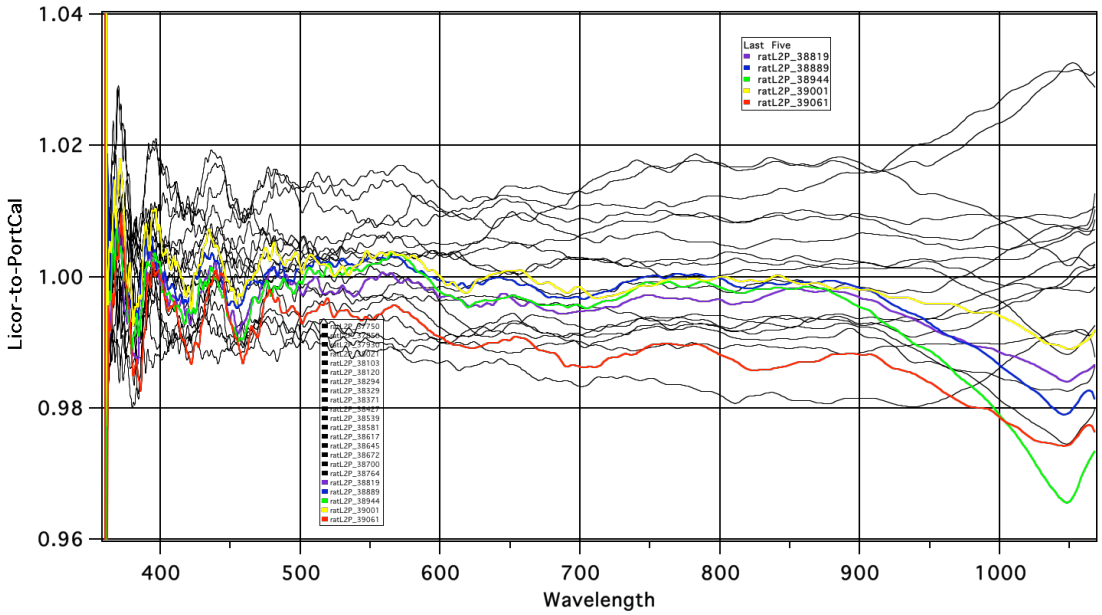

Figure 3A. Shows smoothed ratios of Licor-to-PortCal "coincidental" responsivities. Most recent are labeled in different colors.

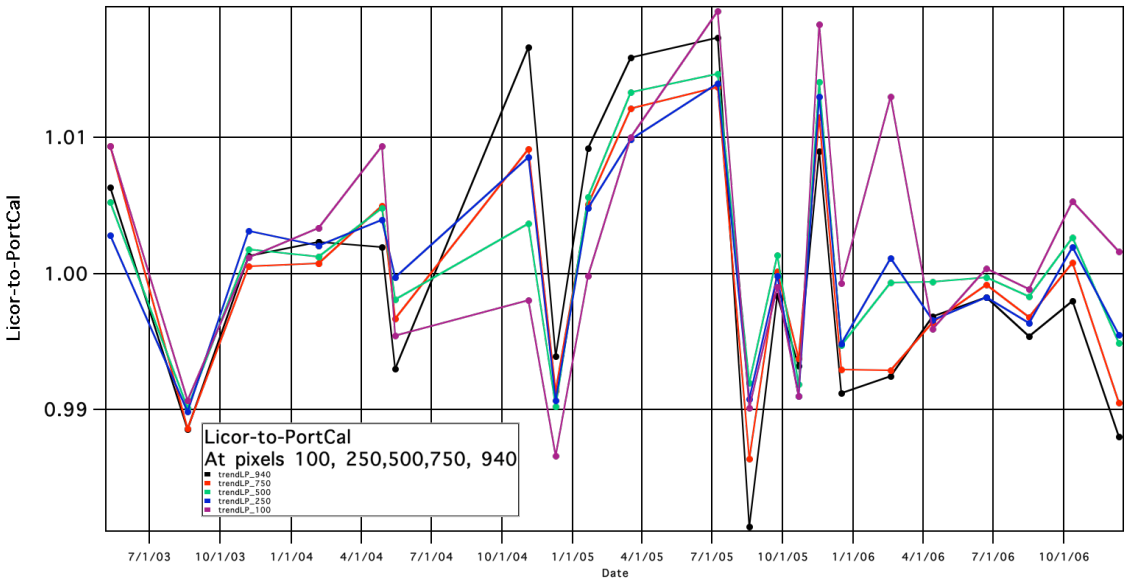

Figure 4A. Shows trend at several pixels smoothed (15 pixels) ratios of Licor-to-PortCal coincidental responsivities. (±10 pixel averages are taken)

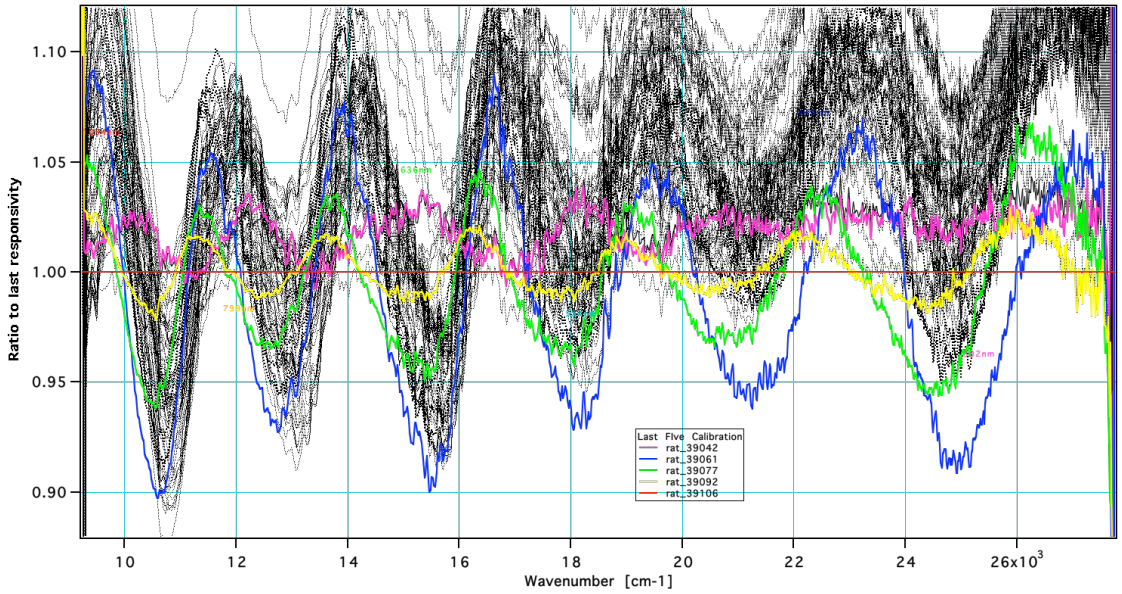

Figure 5A. Ratio of responsivities to the last responsivity. The last five are labeled with different colors. The ratios are plotted against wavenumbers that are obtained from nm_precursor as follows: 10^4/(nm/1000). The last one (the red one) is constant=1.

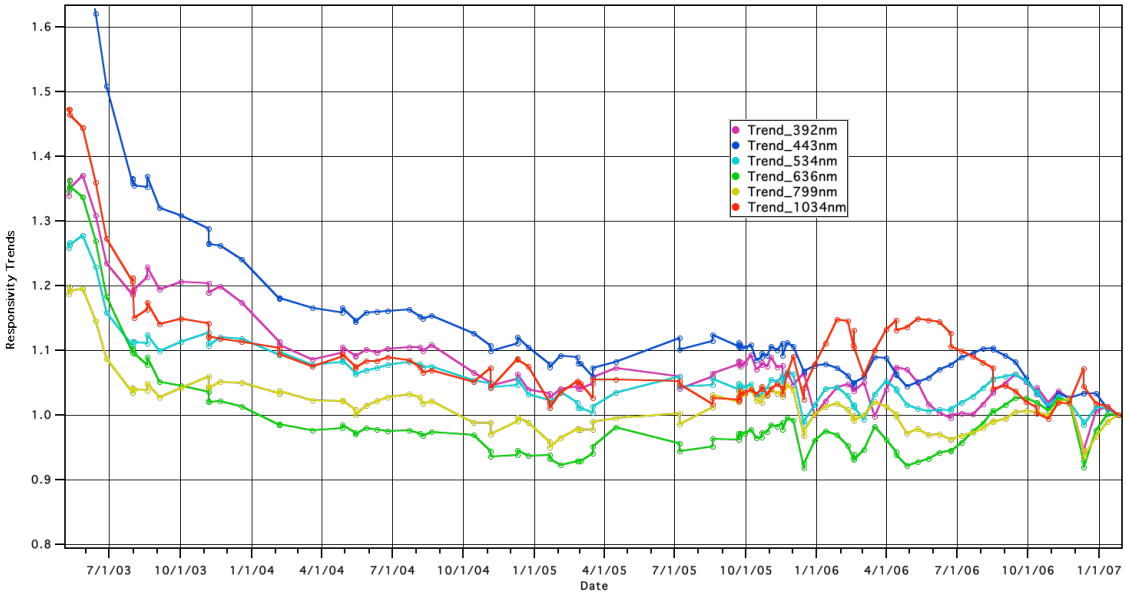

Figure 6A. Ratio of responsivities to the last responsivity for several selected pixels (wavelengths).

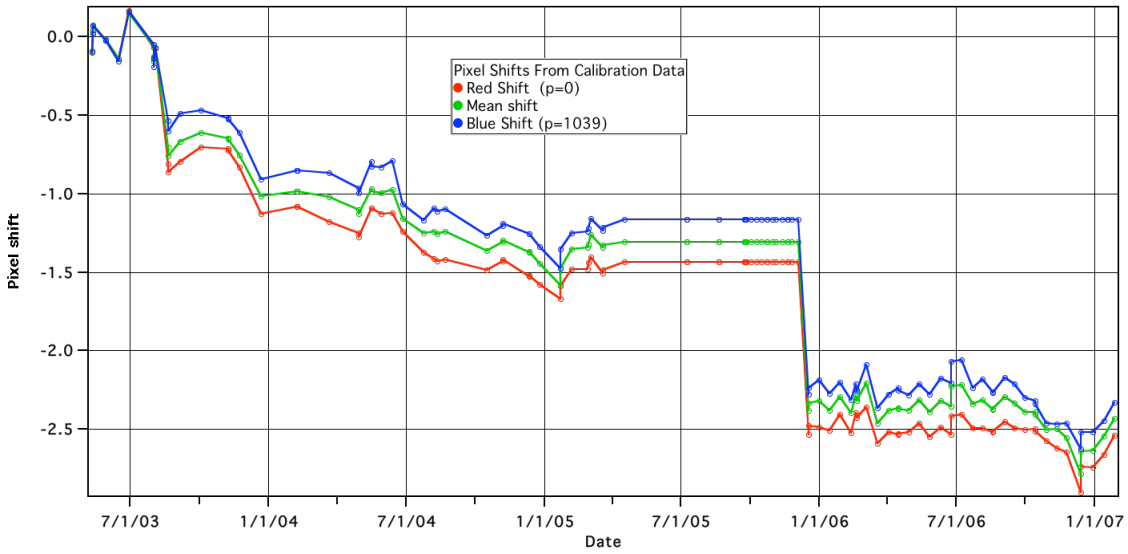

Figure 7A. Pixel shifts obtained from calibration times. (in 2005 the pixel shifts were not updated!)

Final remarks

Generating some QC plots (Figs. 5A, 6A, 7A and parts of 1A and 2A) could be done using existing *to_jrss.txt* and *resp_jrss.txt* files with temporarily added the most recent calibration data still before accepting them.

Some graphs need data that are not in the *to_jrss.txt* and *resp_jrss.txt* files.

Licor2PortCal_AllTill39061.txt file can be used to plot Figs 3A and 4A.

OutArray_111CalsTill39106.txt can be used to extract CCD temperature for Fig. 1A and the unsmoothed individual (no daily averages) *k1* values.

Other option is that after developing the new calibration processing program all calibrations could be reprocessed and then the old results would be discarded. This would be a hard task that would lead to regenerating C1 files with slightly different responsivities and parameters. I am not sure that we want to do it. The differences would be small. Then we would have to re-archive all files. This certainly would result in some confusion for users. Therefore, personally, I prefer that the calibration database should be populated with the past results, unless – of course – the new processing program would demonstrate that the past results are grossly incorrect.