# Create Poly Correction Tables

#### Research Work Notes

## Checking results with Broad Correction - try 52 – 26/1/25

Clipping “delta tab” with 0.0001 make grad more reasonable…

Apparently, no values are clipped from Grad – the clip may be skipped.

The computed delta table is strongly clipped… may try less clipping there

## Checking results with Broad Correction - try 51 – 25/1/25

Look stage by stage – save intermediary images

One issue: Apparently central rings influence external rings –

Possible work-around: Go step by step from center to margins…

Apparently, clipping “delta tab dev” (with minPos = 0.000001) did nothing – and was not even used…

Dev Grad was already in too-wide range…

## The process of calibrated correction of the first coefficient – 25/1/25

Input:

* Prev Tab Dev
* Current Tab Dev
* Delta Table

Compute

* Delta Tab Dev (Current – Prev)
* Gradient 🡨 Delta Tab Dev / Delta Table
  + Change in dev for change in table
* Delta Table for full correction 🡨 (-) Tab Dev / gradient
  + == (-) (Current Tab Dev / Delta Tab Dev) \* Delta Table
* Delta Table to use 🡨 Delta Table for full correction \* fraction

For example:

* Prev Tab Dev was 20
* Current Tab Dev is 15
* Delta Table was -0.01
* Delta Tab Dev was -5
* Gradient is -5 / -0.01 🡺 500
* Delta Tab for Full Correction -15 / 500 🡺 -0.03

#### Where there could be problems?

1. In computing gradients, if the delta table was very small – we may have very big gradient
   1. This would lead to small new delta tab 🡪 No problem
2. While computing “Delta Tab for Full Correction,”
   1. If the gradient was very small, WE MAY HAVE VERY HIGH DELTA – PROBLEM
   2. Small gradient is probably the result of almost no change during last correction
      1. It happens
         1. where the current tube has no influence
         2. On the verge from positive to negative correction
   3. Possible Corrections:
      1. Limit the absolute value of change
      2. Where the change was small, do not correct OR limit correction

## Back to broad convergence – 25/1/25

Still working on the second correction of Tube 1

The general concept is working for the first step…

In the second step, which is based on gradient, there are some destructive extreme values…

#### Mitigation:

1. Identify problematic
2. Clip up small (positive and negative) values before division

#### Auxiliary

1. Show some global scores to help overall assessment of progress
2. Save different types of matrices to special directories

## More broad convergence – 20/1/25

#### First, save also:

1. The relevant poly tables…
2. The computed average deviance
3. The tables’ deltas

#### Second thought:

1. Compare
   1. previous deviance
   2. current deviance
   3. table delta
2. Assuming derivative is [current dev – prev dev] / tab-delta
3. Compute new delta by: -dev / (last\_dev\_delta / tab\_delta)

#### For example:

prevDev was -20

tabDelta was 0.001

newDev is -18

lastDevDetla = 2

Gradient = 2 / 0.001 = 2000

Full Correction = -(-18) / 2000 = 9/1000 = 0.009

Applied correction = fraction \* full correction

## Try broad convergence first – 19/1/25

#### First experiment:

1. Work on Tube 1 only – as it covers the whole range
2. For each table value – compute the average deviation over the 4 target points in the deviation map.
3. Correct each table value by proportion of the deviation
4. Compute Dev Map
5. Repeat

#### Computing target points from CSV

Input:

D:\PolyCalib\ImpulseResponseTab\Tube1\_imMargined\_width688\_height192\_zoom2.float.rtab

D:\PolyCalib\ImpulseResponseTab\Tube1\_radMargined\_width688\_height192\_zoom2.float.rtab

## Check original results of IR for Tube 0 – 18-19/1/25

Prepared new version of ImageR (1.6.7) that searches for maximum value…

Looking at tube 0 –

Poli\_AI\_t0\_r10\_d340\_width256\_height256\_zoom2\_r\_10.float.rvol

It has maximum effect at page 106!

## Looking again at IR data – 15/1/25

Data in “D:\PolyCalib\ImpulseResponseTab”

2 Tables:

Tube0\_imMargined\_width688\_height192\_zoom2.float.rtab

Tube1\_imMargined\_width688\_height192\_zoom2.float.rtab

The table for Tube0 starts with 97 for 1st row… Why?

(The 2 “rad” tables there are almost the same.)

#### Same issue seen in preceding CSV file

Tube0\_IR\_grid.csv

## Where is my data about the global IR and IIR functions? – 14/1/25

In:

D:\PolyCalib\InverseIRTables

There are 4 tables:

Tube0:

InverseIR\_Tube0a\_row\_per\_image\_and\_radius\_width260\_height280\_dzoom2.short.rtab

InverseIR\_Tube0a\_col\_per\_image\_and\_radius\_width260\_height280\_dzoom2.short.rtab

Tube 1:

InverseIR\_Tube1b\_row\_per\_image\_and\_radius\_width260\_height280\_dzoom2.short.rtab

InverseIR\_Tube1b\_col\_per\_image\_and\_radius\_width260\_height280\_dzoom2.short.rtab

#### Question about the IIR table

The col per Image-Radius seems pretty straight-forward

– mostly higher radius gives lower (or higher) col.

The “row per Image-Radius” tables are not symmetric – and hence require farther investigation.

With Tube 0, first row (image) with real input-row value (1) is 97 –

Going up to row-index 191 at image index 279…

With Tube 1, first image indicates first row and last image indicates last row…?

#### Try to re-read (manually) the IR data

## New approach – work on big features first – 14/1/25

Looking at images – the HU/Radius function is pretty smooth and easy to identify.

Try to work on the whole volume –

Smooth the dev map(?)

#### Before going forward – what is the best result so far?

At “D:\PolyCalib\Exp44\_mult8lines\_targetCSV” –

Train\_metaset\_t0\_r67\_74\_d171\_386 - best is 6.58 at 16.xlsx

## Experiment 45 - 'mult8lines\_posGrad\_LR02' – 8/1/25

Use only positive grad

Sel LR to 0.2

Sample 256

Results for simple run are significantly worse!

## Add Target CSV class – 8/1/25

Follow inner rings in image 163

Results are in “D:\PolyCalib\Exp44\_mult8lines\_targetCSV”

With:

LRFraction = 0.08

nSamplesPerRadius = 128

Results are somewhat improved:

Best: “score at step 16: 6.581170”

## What is the next step? – 8/1/25

#### Generally

It seems that correcting each point separately is problematic – as some point form clear rings.

Try to do major corrections first and minor corrections only later.

#### Looking deeper into specific target points

Select some output points and build CSV files for all the relevant changes for them

#### Get rid of random sampling?

It adds inconsistency.

Is it possible to work on full volume?

#### Big trial – try to change general level to target

Use smooth changes on whole area

#### Add alternative “competing” scores?

## Evaluation of results with 8 input lines (67 – 74) – 7/1/25

Best score is a bit better and faster than with 4 lines input.

See result learning curve at: “D:\PolyCalib\Exp41\_multisets4lines\_wide”

File “Train\_metaset\_t0\_r67\_74\_d171\_386 - learning curve.xlsx”

Note that name “multisets4lines” is misleading, it should have been “multisets8lines”

## Try wider scope – 6/1/25

I run Impulse Response on 8 lines: 67 – 74

Results are in: “D:\PolyCalib\Impulse\Impulse\_r67\_1\_74\_c170\_1\_386”

File “Tube0\_IR\_grid\_r67\_d170.csv”

Line 67 expands rows “2 – 218”

Detectors per line “170 🡪 386” – altogether (386 – 169) 217 detectors

## Evaluation of “Exp40\_multisets4lines\_wide” - 5/1/25

Best score: “<CTabValSets::Train> 37: 8.055171”

#### Follow scores in log

In “Train\_metaset\_t0\_r67\_70\_d171\_346.csv”

Best is at line 37 (explicitly indexted)

There are 16 sets there.

At first set best is at index 37 (of 50, line 38) – value 9.05353625796058

Looking at “Train\_set\_set1\_t0\_r67\_d171\_343.csv” – last column is “all”

There are 52 lines (indexed by excel 2-53) – Best is at “40” - 9.05353625796058

## Some measures to make things faster and cleaner – 30/12/24

Check timing through log. Is it working? “GPolyTrainer.log” – add log of new features.

Check what files are saved and how to avoid some.

May restrict some activities to relevant images.

## Analyzing first run-results on 4 table rows around image 163 – 30/12/24

Data is in “D:\PolyCalib\Exp39\_multisets4lines\Vol”

There were 100 epochs going over all TV sets.

Volumes were saved every 10 epochs.

Technical Comment: First volume index should be “00” in order to allow 2D leafing. - Done

#### Best image 163 is at:

BP\_PolyAI\_Output\_width256\_height256\_zoom2\_save20\_Multiset\_Training.float.rvol

Or

BP\_PolyAI\_Output\_width256\_height256\_zoom2\_save30\_Multiset\_Training.float.rvol

From “30” there start a new kind of thin-rings artifacts, that get worse!

Question: What is the target value in relation to the images that I see?

Proposal: Do whole circle for radius – not random samples, to reduce uncertainty

Task: Resume save of best tables and allow continuous training

Change: Should be easier to see which TV leads to what Image-Rad - Done

Efficiency: Do not copy basic volumes each time - Done

Try:

1. Use only positive gradients
2. Score only main targets (alternative score)
3. For changing general level – change section of TV together

Transparency: I need to see the target level more clearly

Try Bigger Learning Steps

Tracing: Follow by counters how many times each target is used for correcting

#### Inspecting one central TV:

Train\_tab\_value\_t0\_r68\_d343 - individual case of non-convergence.xlsx

High value is consistently pushed down by lower TV –

But somebody else is pushing up even stronger!

#### Missing time after end of recon:

2024-12-30 22:55:30.331181: End Recon - Elapsed 3.765 seconds

2024-12-30 22:55:32.234933: Start ComputeNewScoreOfVolume1

## Errors in running 4 rows – 29/12/24

<ComputeGrad2> WARNING: deltaTable was 0!

Check it later…

## Start to work on several sets interleaved – 28/12/24

1. Prepare several sets in the same row
2. Later work on several rows
3. Save volume and dev map every while
4. Add csv for each set with scores from all TVs
5. Add score and csv for the multi-set

## More tests – huge error – 28/12/24

Huge error corrected by “skipping step” in case the main target as deviation |d| < 1

## Working according to highest derivation is not good enough! – 28/12/24

Example of non-convergence:

Dev:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 17.23244 | 24.14173 | 2.285382 | 17.03021 | 15.17244 |
|  |  |  |  |  |

Grad:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1968.897 | 976.7303 | 11927.4 | -3036.29 | 2 | -0.00019 |  | 999.9923 |

Next Dev:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 17.20246 | 24.13029 | 2.242912 | 17.15274 | 15.1821 |

If I want the score to improve with each iteration –

The expected effect for each point is grad \* deltaPos

Sign of sum of grad over all tv may decide direction of change

Also – derivation changed strongly on small changes in tv!

File saved in: “D:\PolyCalib\Exp37\_try\Train\_tab\_value\_t0\_r67\_d339\_not\_converging.xlsx”

## Should I do extra-recon after retrace of first fixed step? – 28/12/24

From the point of view of the individual tab value I have all the information regarding current and previous table positions (only the order was reversed)

If I do the next step on the same group – the global prev dev is not up-to-date for them!

One more recon after table update will close this issue.

As there could be crosstalk – better to update dev of back-tracked TVs.

## How to start training? 27/12/24

Start each set separately

1. Add constant to all elements
2. Run recon and score once
3. Set first derivative/gradient from constant diff
4. Select starting point – first or second (current or previous)

#### Complication with retracing first fixed step

1. Only some values are retraced (if any)
2. Better to do extra recon when some values were changed
3. Table values that were not changed should not update their grad, etc., after retrace by others

#### How to identify which values should update on retrace?

There could be a retrace flag per table value

And then it is easy to update only if retrace!

## Work on several table positions in parallel – 26-27/12/24

Just had to make the CSV internal member for each C Tab Value.

Adjust training procedure.

#### Next

First work on a whole row-segment, segmented modulo 4

Then work on 4 consecutive rows, interleaved

Create a combined score of all managed table-values, and print only it

#### Computing gradient

When a tab value is incremented – the gradient is computed between the previous and result dev map.

Save volume every “n” full cycles. Start with “n = 1”.

#### Selecting the area to handle for first trials

I have available:

D:/PolyCalib/Impulse/Impulse\_r67\_1\_70\_c170\_1\_346

Start with row 67, detectors 335 – 346 (12 detectors)

Check the effect

## Check better on single point convergence – 25/12/24

Add prints to special CSV file…

## Check C Tab Value – why a single point is not converging? – 24/12/24

The single point starts converging well – until it changes direction.

Should check more.

## New classes for geometric orientation and convergence – 23/12/24

C Tab Value - For a single point

C Tab Values – matrix of “C Tab Value”

First check convergence through a single “C Tab Value”

Later use array of values…

## Compute again IR and IIR for a small sub-region for image 163, tube 0 – 18/12/24

Define the relevant region by current IIR tables –

Rows 67 – 70 (4 rows)

Columns 1 – 170 🡪 170 – 345 (176 columns, detectors)

Run IR loop on 2 rows (67 and 70) and every 4 detectors starting from 170

#### RE-visit my IR loop

It is now working nicely on bands…

#### Problem due to strongly negative values

At some point – with a ring just around a black hole in the very center,

There are very strong negative values in the middle –

At r67 with d346 it is:

1. -8695 (min) in the middle
2. +6239 (max) at the corners of the small ring

Adding the “min” to all the volume creates artificially high values, where there is none.

This kills the “threshold” concept, and cause false radius identification:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| row | det | max | image | rad |
| 67 | 342 | 7658.18 | 159.967 | 1 |
| 67 | 346 | 14934.47 | 159.99 | 97.527 |

#### Problems in the very center – negative radius

The high values at the very center had radius “0” and hence were not counted…

As no high values were found – radius defaulted to “-1”

This special case is now corrected by adding offset of “1” to the radius value while identifying points above threshold – and subtracting this offset later from the average.

## New design for spaced-steps for setting multi-point derivations – 18/12/24

Every input point influences several output points –

The influence of every point is concentrated in some “neighborhood” around its top-influence

We can change a “grid” of input points so that the level of “crosstalk” would be limited.

When changing a grid – we can compute the gradient for the points that were changed…

Computation of the gradient would be by accumulating the influence on several output points in the relevant neighborhood.

#### Correcting the coordinates

For the last experiments I had: table [0,70,300] 🡨 🡪 Dev [163,30]

In the inverse table:

table [0,70,130] 🡨 🡪 Dev [163,30]

table [0,70,129] 🡨 🡪 Dev [163,31]

Apparently I have to add 170 to the detector index

## Try to flatten a whole image – 17/12/24

First must get the input-output geometric function for a whole image…

I used to work on table [0,70,300] 🡨 🡪 Dev [163,30]

#### Where is my data?

“D:\PolyCalib\ImpulseResponseTab/Tube0\_IR\_grid.csv”

“D:\PolyCalib\InverseIRTables”

Create “D:\PolyCalib\InverseIRTables\Image163\_sources.csv”

## Compute real derivation creates real convergence – 16/12/24

It seems to be working…

## Manual convergence on a single point – 16/12/24

Hold all the relevant information:

1. Table value
2. Deviation
3. Table value delta
4. Deviation delta
5. Real gradient is: deviation delta / table delta”
6. In general – higher table value should cause higher (signed) deviation value

Each time try to reduce a fraction (LR) of the variation – start with 0.1

To start:

1. Measure variation on a flat table
2. According to the direction (sign) of the deviation – try a small (constant) table step.
3. Loop training using the last two readings

## Back to single point – convergence is not consistent – 15/12/24

With LR at 0.1, sometimes it converges very well,

BUT:

1. Sometimes it reaches asymptotic convergence to high loss level
2. When gradient is negative, it still “correct” down and away from the target

#### Try another optimizer

I was using Adam.

Try single value input.

Try SGD – it does not seem to improve 😊

#### Try new approach – manual point-by-point convergence

I have all the information, at least for the “linear” case with a single volume.

Do manual computations per table point first.

It might be defined as home-made optimizer later.

## Single image train – 15/12/24

Use all radiuses for a single image.

What are their sources in the tables?

Start with a single table – than try 2 tables together.

## Single point train – continued – 15/12/24

1. Loss should return single value for single point
2. Gradient for a single point
3. Change only single point in table

Loss values are completely wrong. Check – through dump:

1. Tables
2. Images’ Volumes
3. Dev Maps
4. Loss

## Single point train by class CPolyDLTrainer0 – 10/12/24

1. First run on flat table –
   1. Save initial deviation map
   2. Identify target (for flatness)
2. **Second, try to change only one point**
3. Check difference
4. Identify hot point in dev diff
5. Save new dev map

Hottest point on diff is:

<data\_x> 30.106918 </data\_x>

<data\_y> 163.031235 </data\_y>

Fully consistent with computation

3) Define loss for a single point

## Make the training faster – 9/12/24

Do not need to repeat first and second recon each time…

May load ready target

#### Single point mode –

To make things simple – start with “impulse response” to some point…

Find the relevant point in the dev map

Give loss function only for this point – and optimize it!

#### Tube 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| row | det | max | im | rad |
| 70 | 300 | 2159.782 | 163 | 30 |

#### Tube 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| row | det | max | im | rad |
| 70 | 300 | 3805.391 | 67 | 30.218 |

## Working with Recon as external Function and Back Propagation – 9/12/24

C Ex Recon is working within training –

#### How to create meaningful gradients?

Start with a single point in the tables – and the corresponding single point in the Dev Map.

## Problems in integrating Recon as external function – 2/12/24

Work with C Ex Recon

It includes “forward” and “backward”

BUT – the backward format is not accepted!

## Try small input – 1/12/24

Try to use a small vector of ones as input!

## Start training on small matrices – 27/11/24

Questions:

1. How to select a small region where the dev map and poly table are both relevant?
2. What is the preferred artificial derivation for recon?

Thinking:

Start with central area:

1. Limited number of detectors
2. Limited number of rows
3. Limited number of images
4. Only small radiuses

In the central area I can easily use “Linear” with full connectivity…

Target is flat deviation map – flat value is taken from empty map

#### Data dimensions

Dev Map is 280 images \* 128 radiuses 🡺

## Now I know how to integrate external function in loss function – 27/11/24

Example is in “D:\SW\TrainDL - ExFunc example”

## Ideas for using DL to create poly tables – 26/11/24

Input can be one or several Deviation Maps.

The connections can be spatially oriented – for each location use only relevant input space.

#### Gradual approach

1. Do some training for fully connected
2. Use central area of a few images for small input and output matrices
3. Use BP results for loss function
4. …

## Create real DL prototype – 25/11/24

New Spyder project at “D:\SW\PolyDL”

INPUT for DL:

The Dev Map – 280\*260

NN:

All 2 All, few layers

Output – the fraction part of the 1st coefficient – valued between -0.01 and +0.01

Output size is 192 \* 668

Try training initially for flat table!

## After discussion – 22/11/24

1. Where are the edges of the water phantom container?
   1. They are well seen in FOV 450
2. Galit suggest to separate data by FOV – set new sets of data
   1. New data from Galit is in “h:\Poly Calibration by AI\From Galit 2”
3. I promise to try to create a prototype with real DL where recon is used only as LOSS

## Prepared presentation for Physics team – 21/11/24

Presentation is under GIT in:

D:\SW\PyIP\Docs

Result Driven Preparation of Poly Correction Tables - Initial Research Report.pptx

## Use IIR to correct poly tables – 16/11/24

Optimizing to target is working – but poorly!

1. The target itself is usually not improving - First check why?
2. Make more bold steps

#### Check how the target is missed…

The correction is not at the exact position to improve the target!

First make the errors apparent – check their direction!

Find location and direction of most change in the new [image, radius] raster

#### New flow: Several steps per target

First select a target to correct

Then take several steps until it is significantly improved (at least 40%)

For each target there are 4 spatial options: Each of the 2 tubes, Left & Right

One of the tubes may not be effective.

## Use IIR to correct poly tables – 2/11/24

Some ideas:

1. Separate the “abs diff from target” score from all other scores
   1. Try accepting steps only by “average abs diff”
   2. Make sure that the max point that was targeted really improved
2. Maybe try correcting both tubes – and see which one is better?
3. Select patches of different radius for deviations of different size
   1. Prepare several initial patches
4. Avoid traps of repeatedly trying to correct failed corrections

## Use IIR to correct poly tables – 2/11/24

Add global log to help understand how steps are selected and what is the result

## Use IIR to correct poly tables – 30/10/24

Set a single target-level and strive to bring all rings to this level.

Find point with biggest deviation from target – and select relevant correction.

Direction of correction (up or down) is evident from the selected deviation.

After correction & recon, consider both local and global deviation score.

Later this method will also help to decide width and amplitude of correction.

Avoid marginal spaces – at least initially.

#### Create new “score” – distance from flat target at ring

Dump Deviation [Image, Radius] as displayable matrix

Apparently, there are very strong deviations on the margins –

Try to cut out the margins! Peel function seems to work OK.

#### First verify that problems are correctly identified

Even after peel – result seems to be wrong!

self.avgDev=-1.0128281116485596, iImage=92, iRad=0, maxDev=tensor(-458.4808)

Second run even worse:

self.avgDev=-0.9682614803314209, iImage=92, iRad=0, maxDev=tensor(-999.0014)

The inner rings should be deterministic – as there are only 4 pixels!

Something is basically wrong with the average per radius computations!

## Check IR and IIR in a single automatic loop – 29/10/24

In Identify Ring Source.py

First activate new recon option:

Set config file “d:\Config\Poly\Impulse.txt”: Tube, row, detector

Set BP dump name at: “d:\Config\Poly\BPDumpFileName.txt”

To something like: “d:/PolyCalib/Impulse\Poli\_AI\_t1\_r70\_d300\_width256\_height256\_zoom2.float.rvol”

## Correcting table by Reverse-IR function – 23-27/10/24

Work with new “impulse response values” to check full loop of IIR

D:\PolyCalib\Impulse

Poli\_AI\_t1\_r70\_d300\_width256\_height256\_zoom2.float.rvol

Load it and analyze it – to find source of ring!