University of Zurich[UZH]




# Introduction to Monte Carlo and the Monte Carlo propagation of uncertainties in DN and L to gain and offset

2019

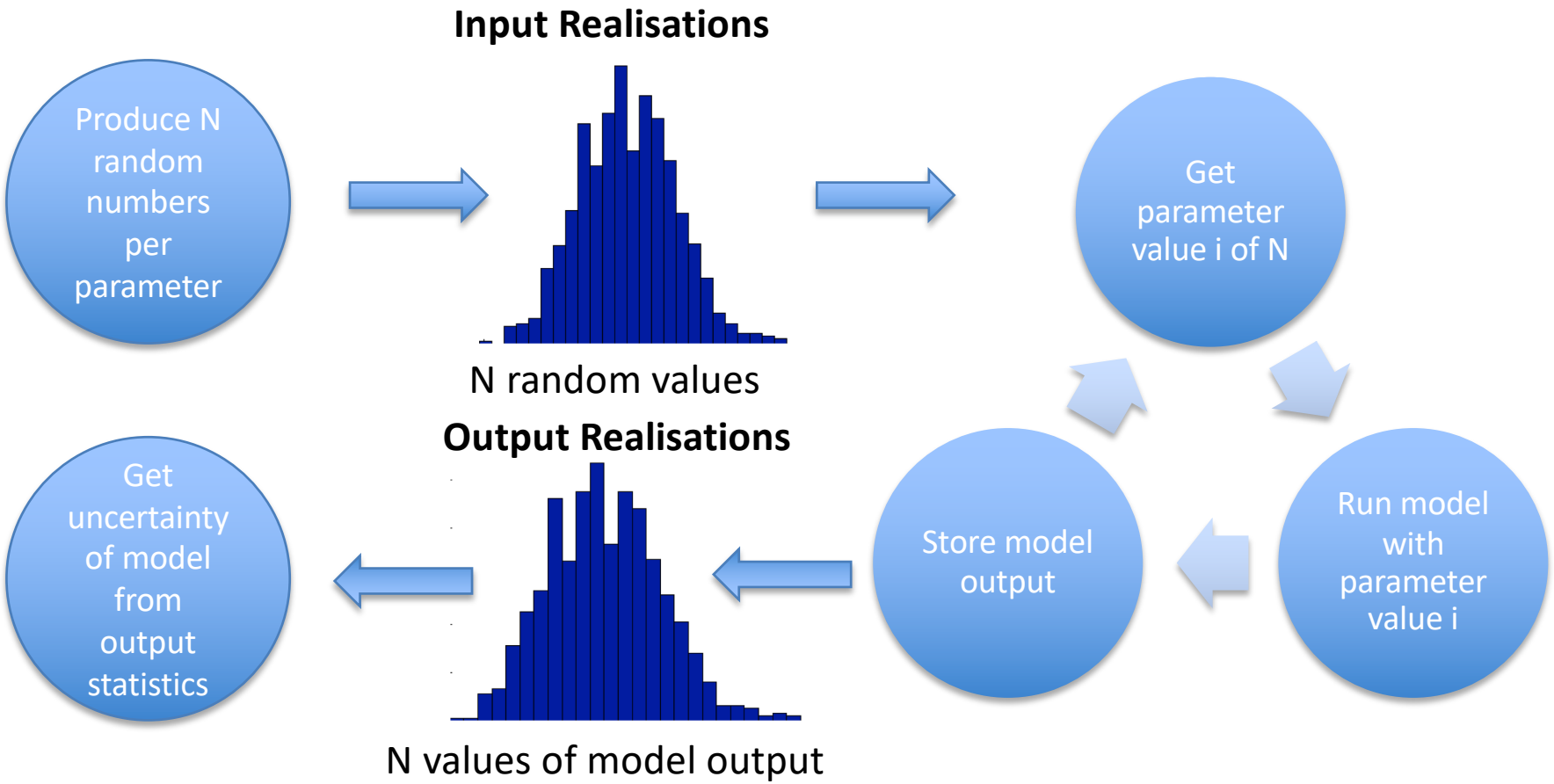
WG4

ahueni

University of Zurich[UZH]

RSL measurements | products | policy

# How Monte Carlo works: Computational Brute-Forcing



University of Zurich[UZH]

# How to generate Realisations

RSL measurements | products | policy

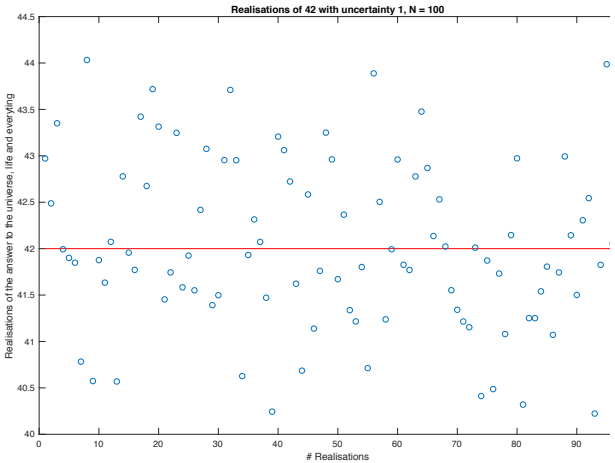
Stddev = 1, Rel. Stddev = 2.3810 %

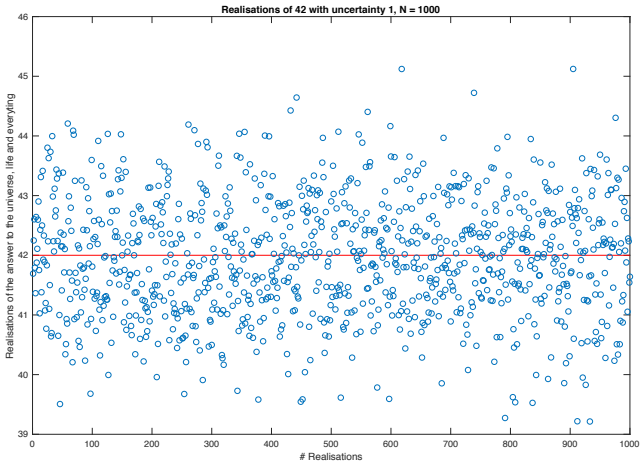
```
% creates realisations, either using a new random generated number
array or
% using the supplied random number array
function [realisations, random_numbers] =
get_realisations_gauss_dist(numSamples, mu, sigma, random_numbers)

    if nargin == 3
        random_numbers = randn(numSamples, length(mu));
    end

    for i=1:numSamples
        realisations(:,i) = mu + sigma.*random_numbers(i,:)';
    end

end
```

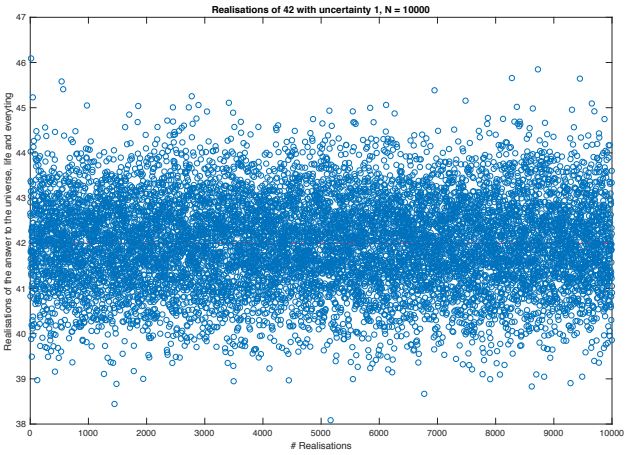
University of Zurich[UZH]

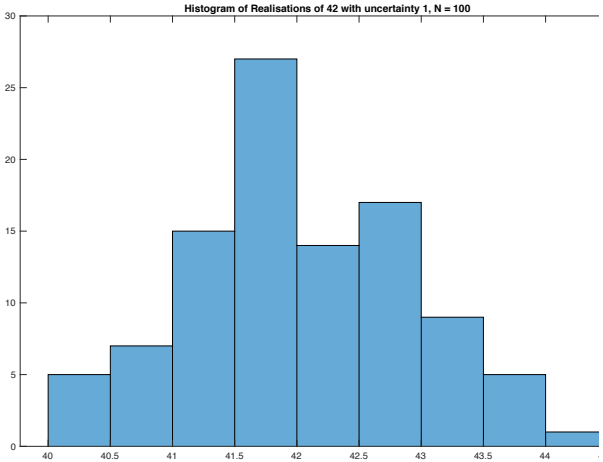
# Testing the Realisation Generation: Is it Gaussian?

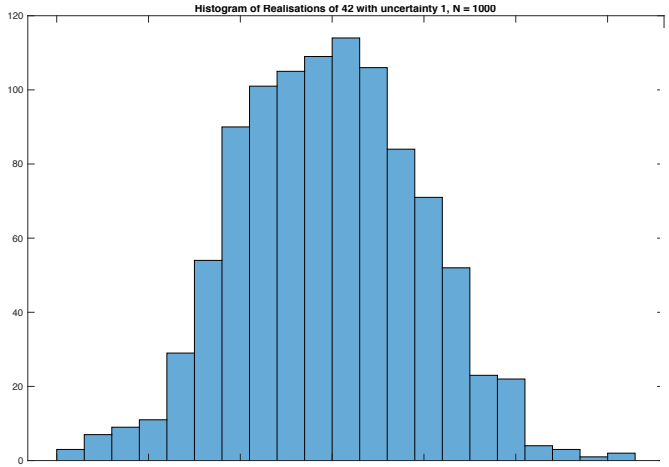
measurements | products | policy

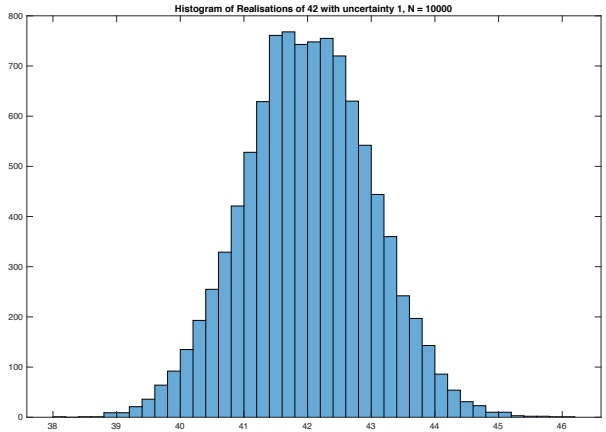


University of Zurich[UZH]

RSL measurements | products | policy

# Uncertainty Propagation: Creating the Output Realisations

A squared log transformation of the answer to the universe, life and everything …

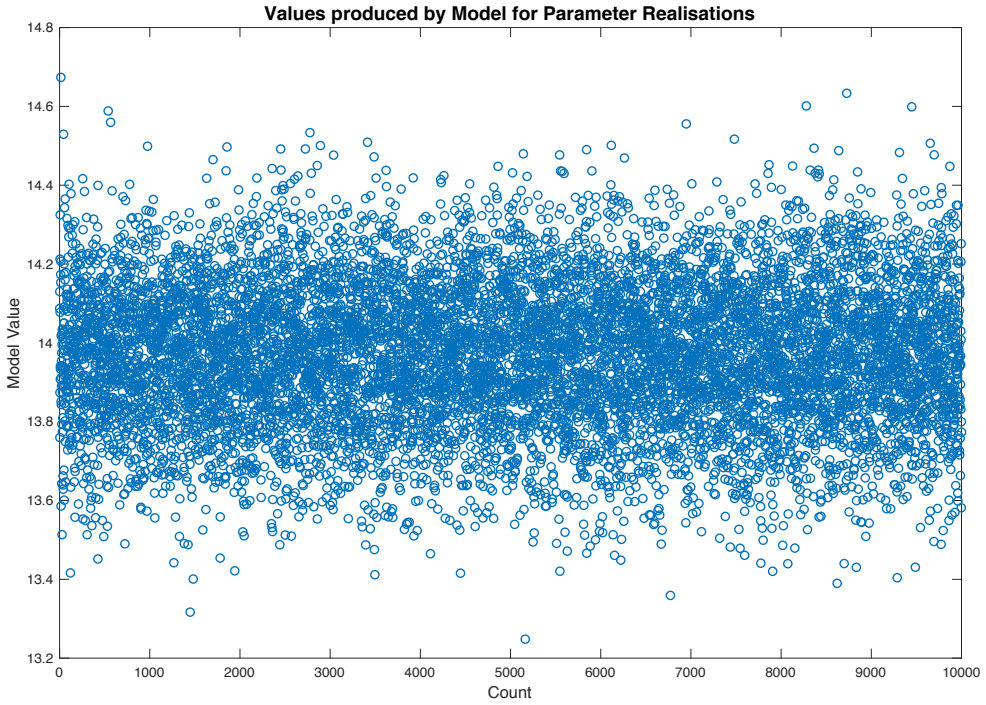
```
% simple model to draw from the generated realisations

output = zeros(size(realisations)); % allocate output vector

% loop over all realisations and calculate the model output
for i=1:length(realisations)

    parameter = realisations(i);

    output(i) = (log(parameter))^2;

end
```

University of Zurich[UZH]

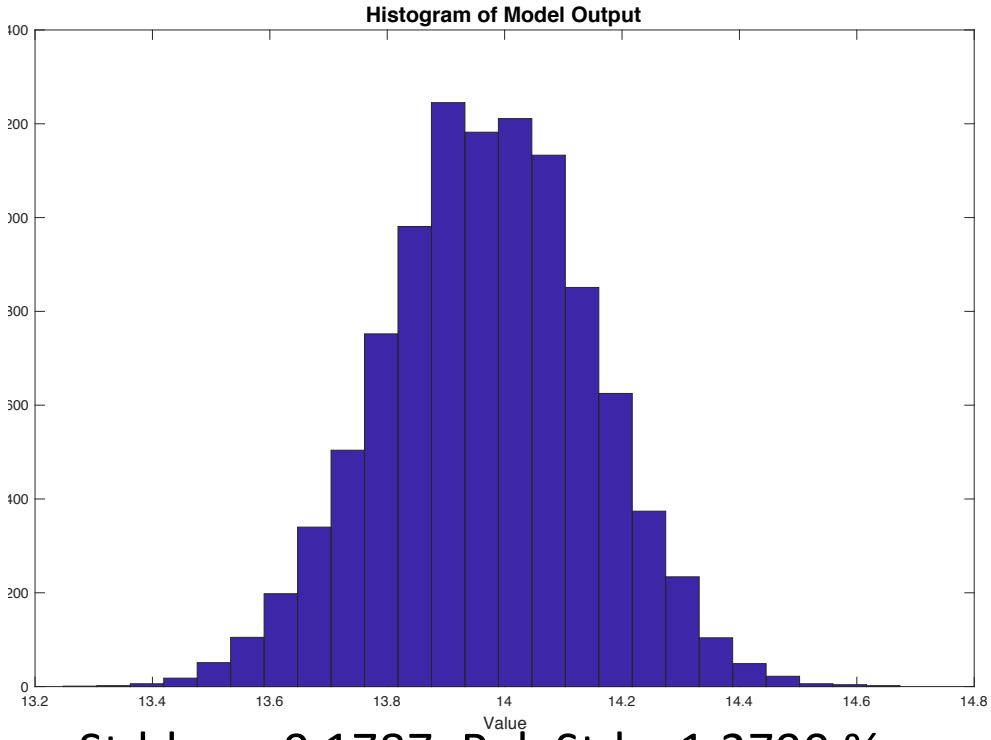
RSL measurements | products | policy

# Uncertainty Propagation: Creating the Output Realisations

A squared log transformation of the answer to the universe, life and everything …




Mean = 13.9694

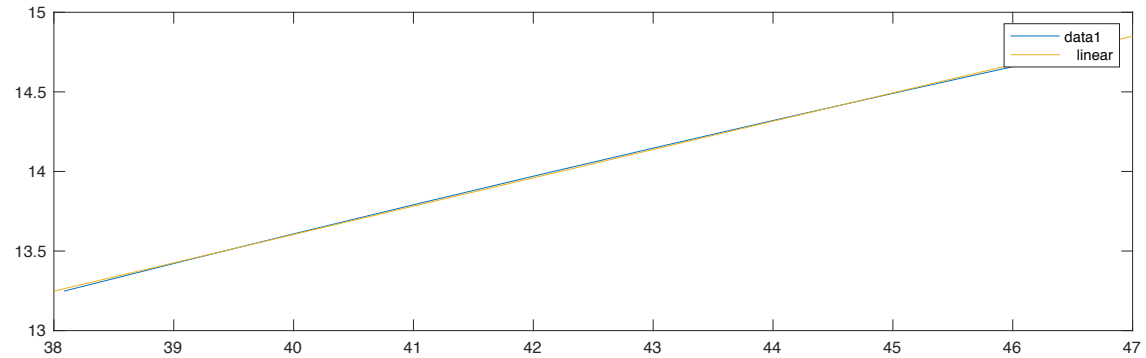


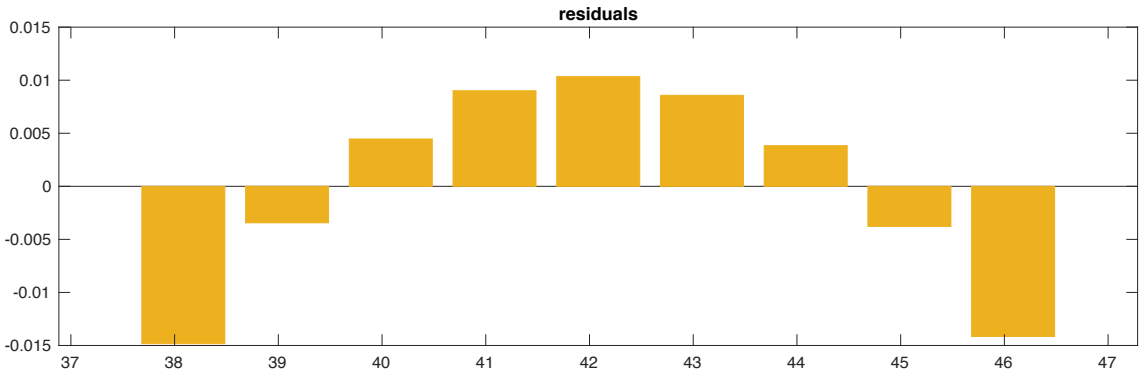

Stddev = 0.1787, Rel. Std = 1.2790 %

University of Zurich[UZH]

RSL measurements | products | policy

# Analytical Deduction of Uncertainty




```
c = (2*log(mu)/mu);

u2_y = c^2 * sigma^2;
u_y = u2_y ^ (0.5)


u_y = 0.1780
```

University of Zurich[UZH]

RSL measurements | products | policy

# Analytical Deduction of Uncertainty: Comparison with Monte Carlo

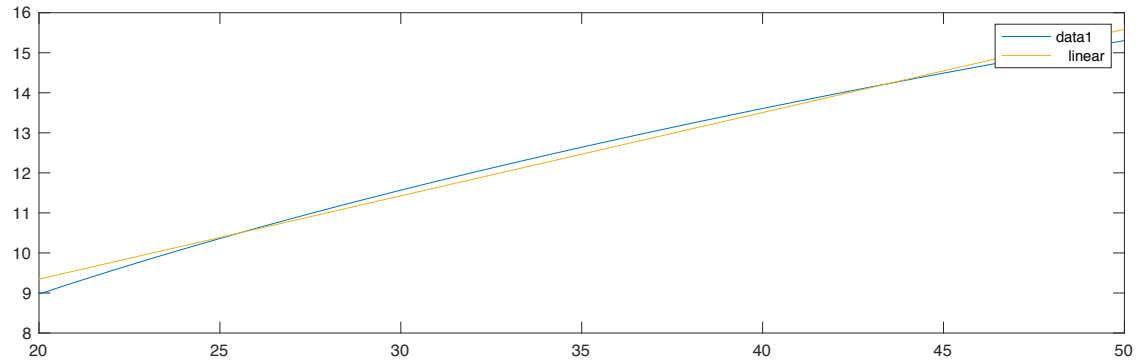
| N | MC u(y) | Delta to Analytical |
|---|---|---|
| 100 | 0.1540 | -0.0240 |
| 1000 | 0.1686 | -0.0094 |
| 10000 | 0.1763 | -0.0017 |
| 100000 | 0.1785 | 5.5467e-04 |

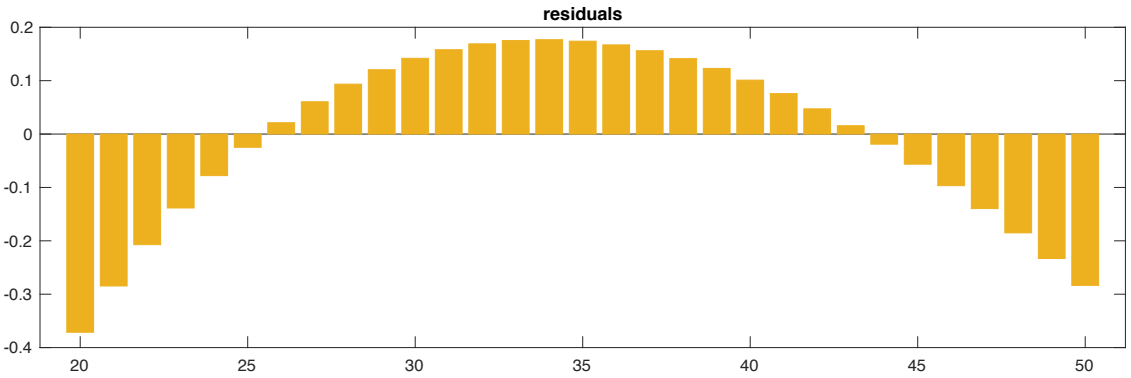
```
Analytical u(y) = 0.1780
```

University of Zurich[UZH]

RSL measurements | products | policy

# Limits of Analytical Deduction of Uncertainty



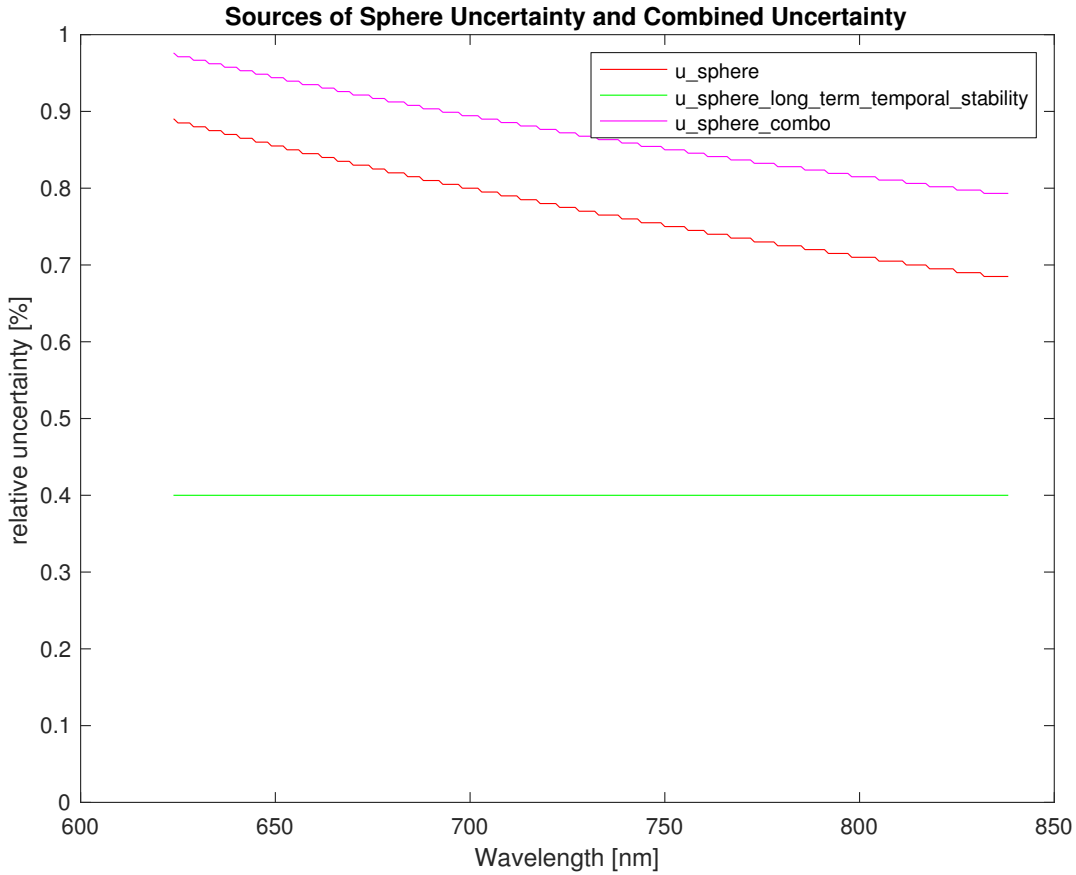




Partial derivatives assume linearity over the range of values.

University of Zurich[UZH]

RSL measurements | products | policy

# Combined Uncertainties of Sphere



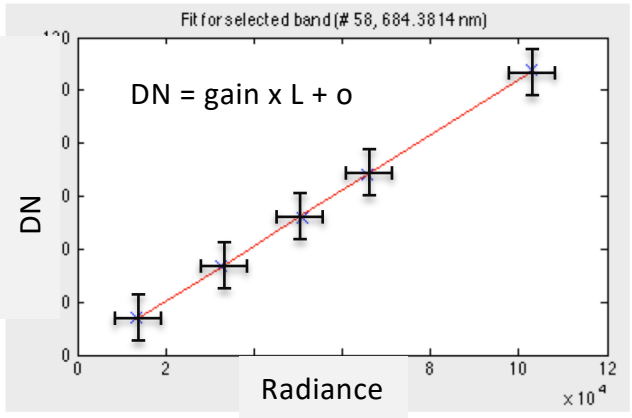

Added in quadrature as error factors are multiplicative

University of Zurich[UZH]

RSL measurements | products | policy

# MC applied to straight line calibration

u(DN)

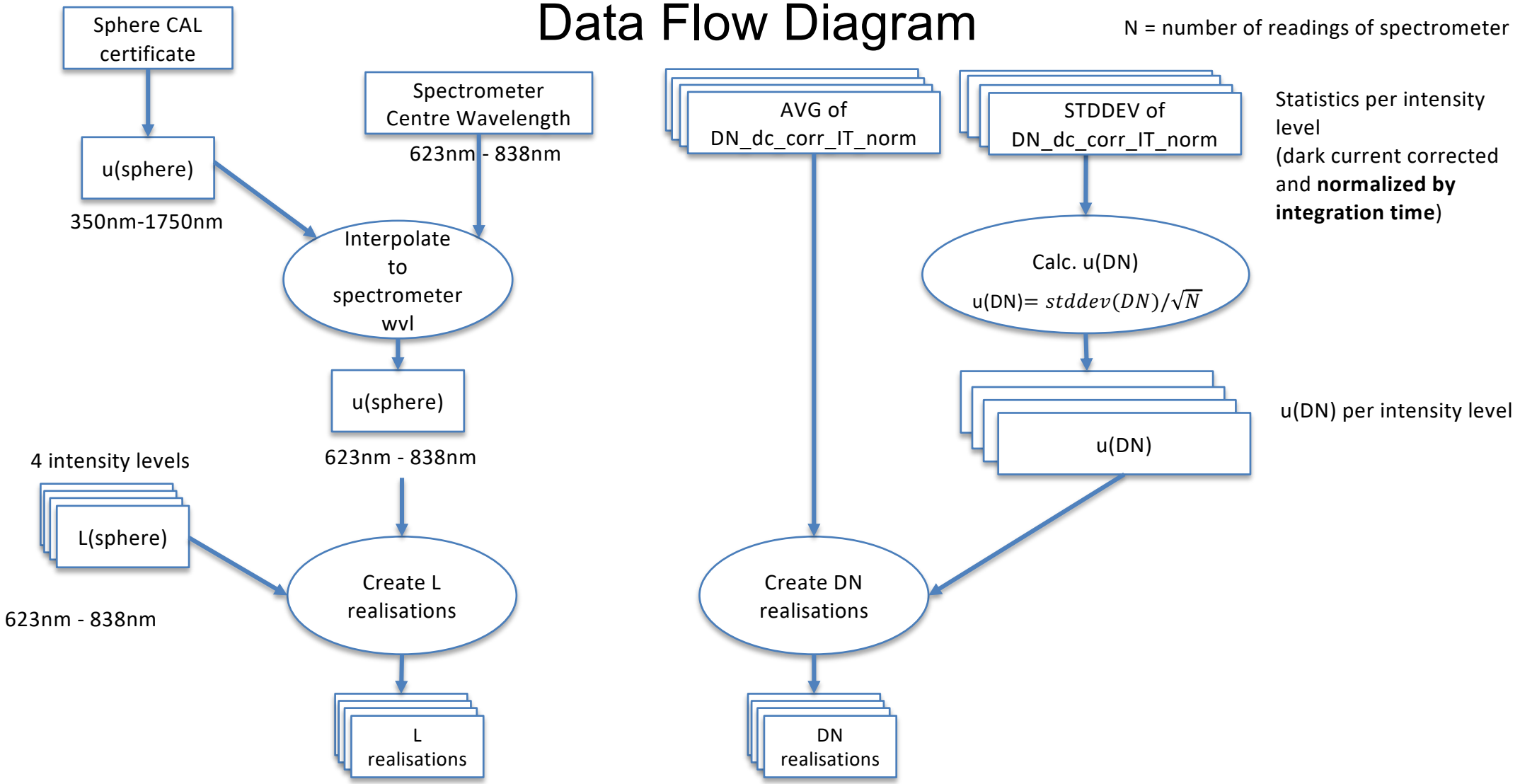



u(L)

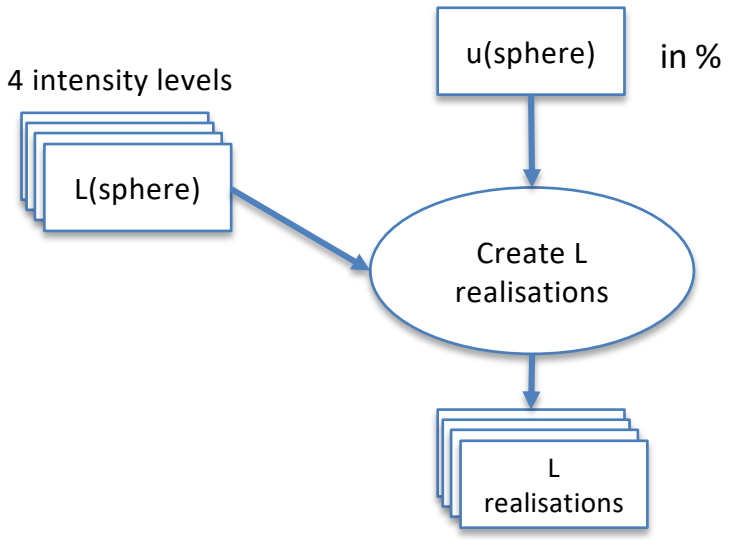
DN = gain x L + o

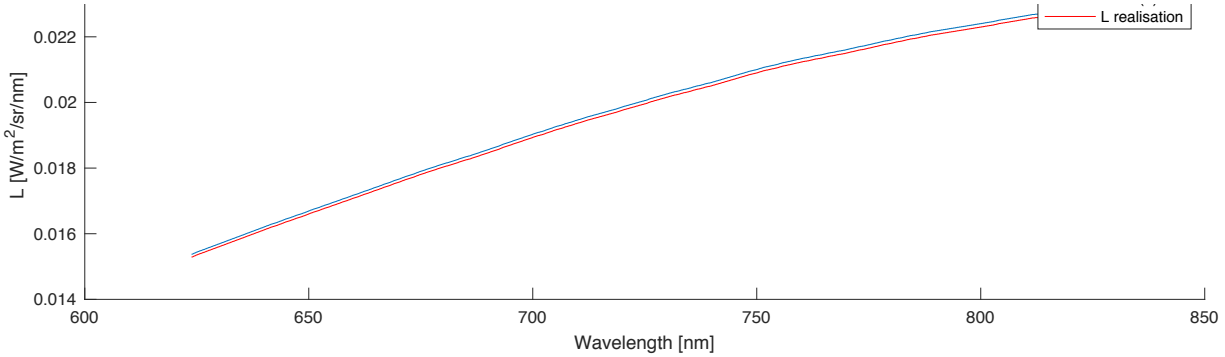
L = (DN – o) / gain

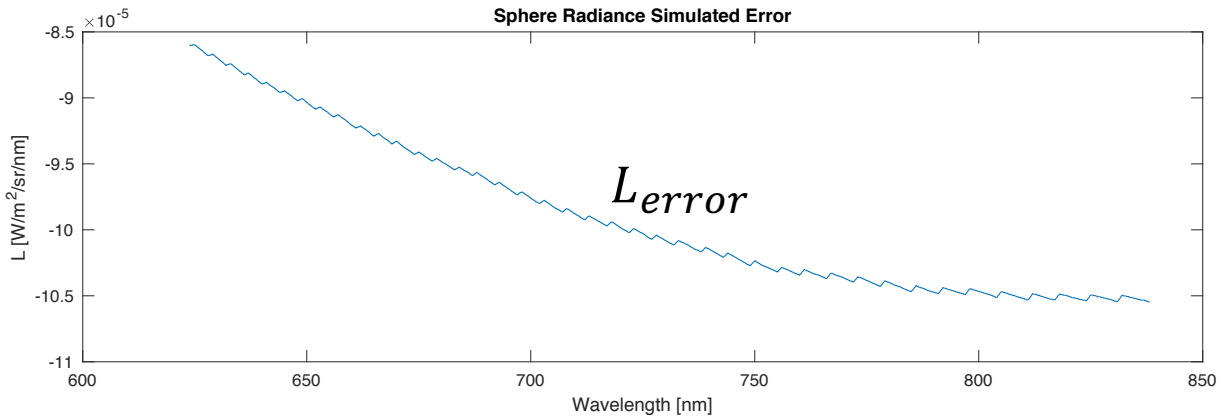
# Data Flow Diagram



# Data Flow Diagram: Create L Realisations




$$L_{realisation} = L_{sphere} + L_{error}$$

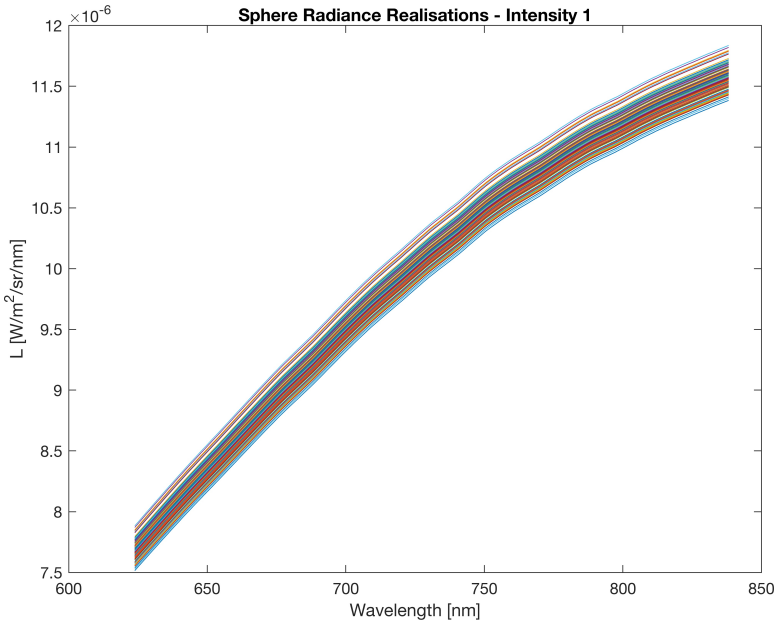
$$u\_abs_{sphere} = \frac{u(sphere) \cdot L_{sphere}}{100}$$

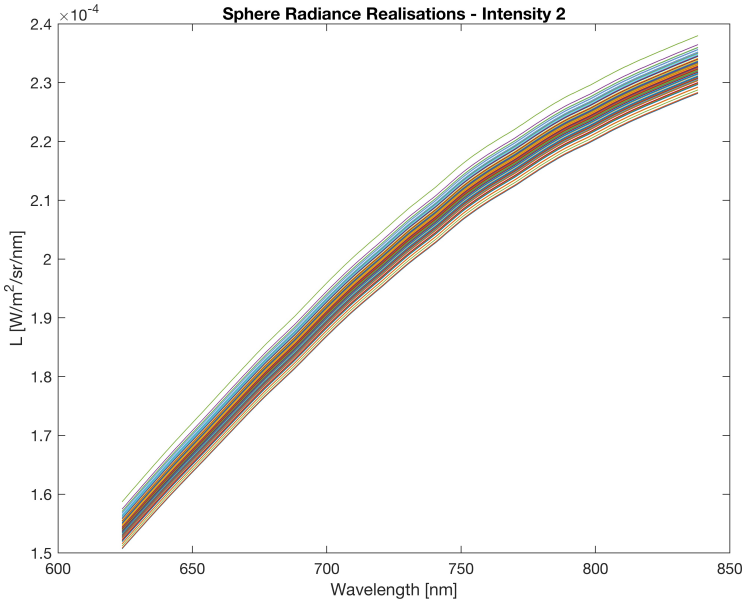
$$L_{error}(\lambda) = u\_abs_{sphere}(\lambda) \cdot rand$$

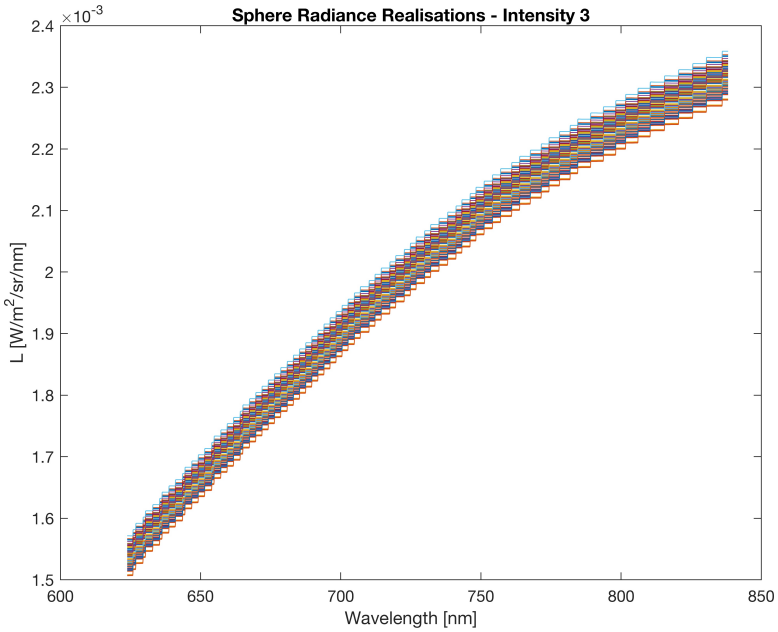
Note: use same random number for all spectral bands to achieve full correlation of uncertainty

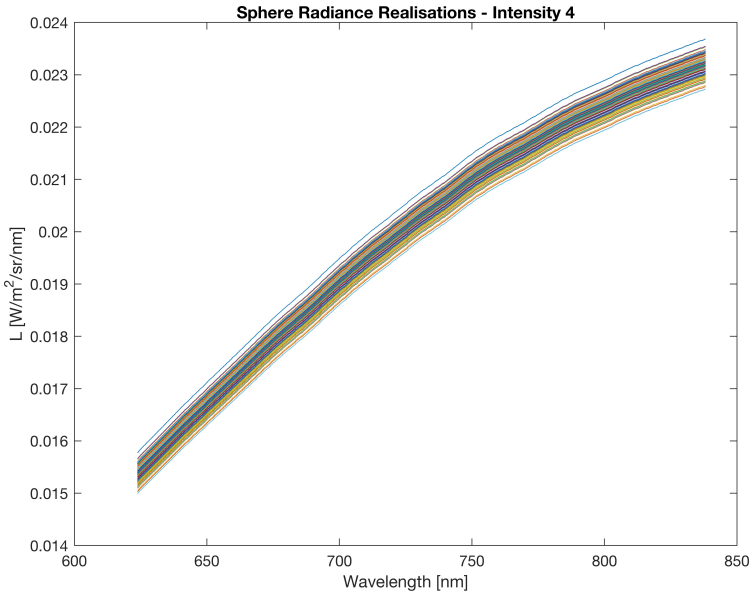
# Creating Sphere Radiance Realisations

Trick: use same random number to draw from PDF. This then assumes full correlation between the spectral bands.

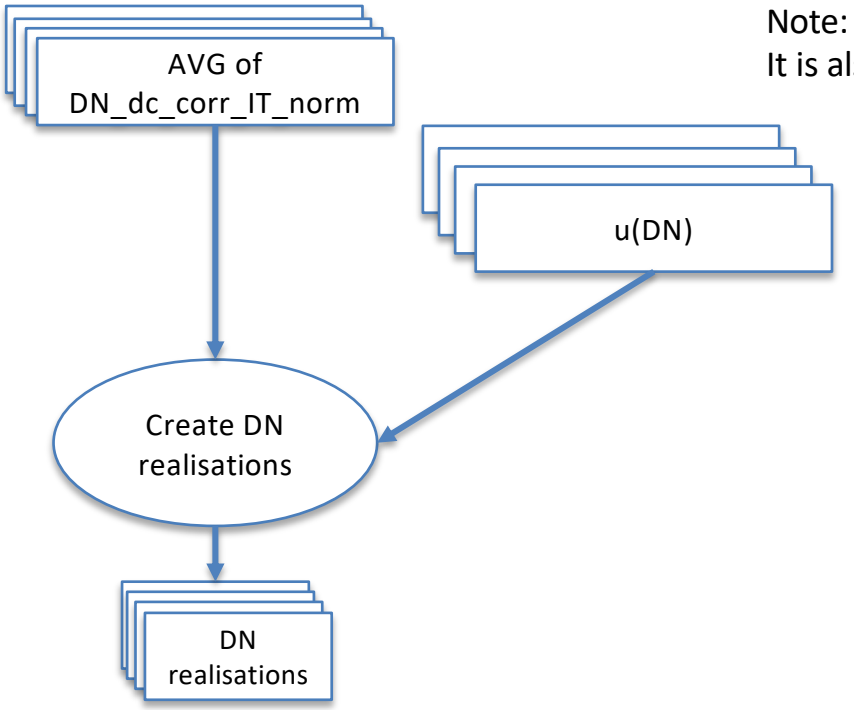


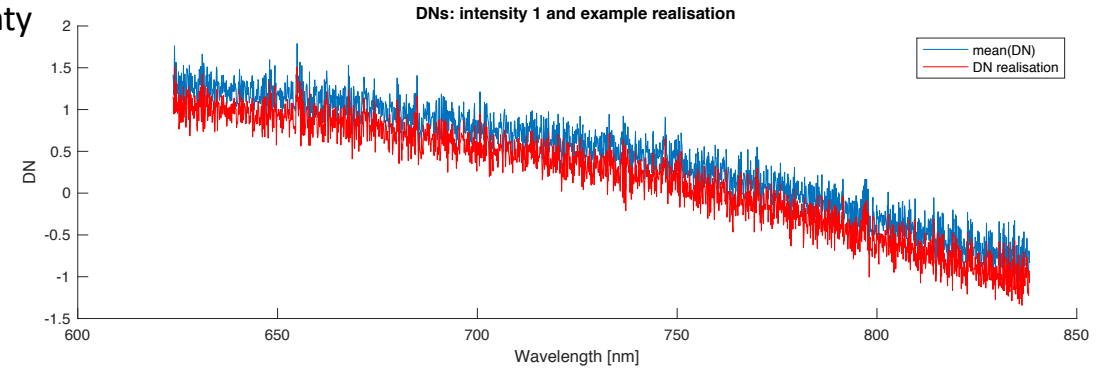



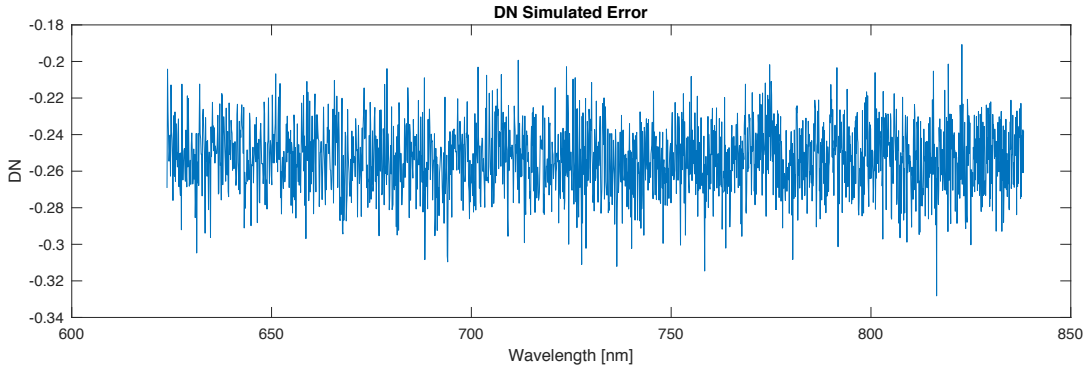
# Data Flow Diagram: Create DN Realisations




Note: the u(DN) are already absolute as they were estimated from the measurements.
It is also important to get the standard deviation of the DNs normalized by integration time.

in DN, i.e. absolute uncertainty



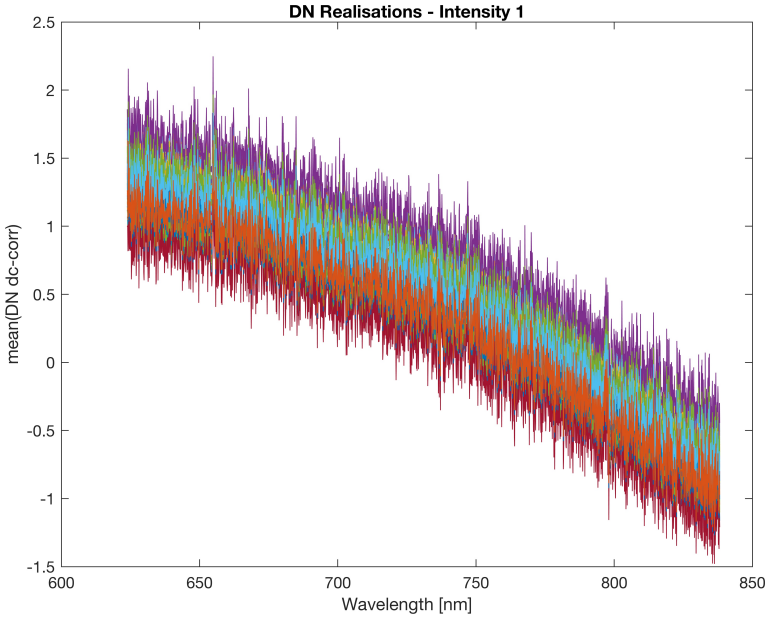

$$DN_{error}\ (\lambda) = u_{DN}(\lambda) \cdot rand(\lambda)$$

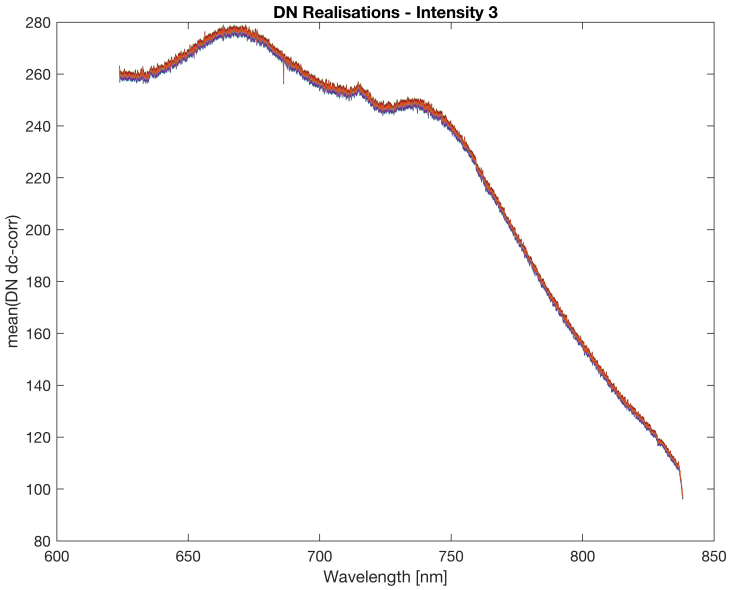
$$DN_{realisation} = DN_{dc\_corr\_IT\_norm} + DN_{error}$$

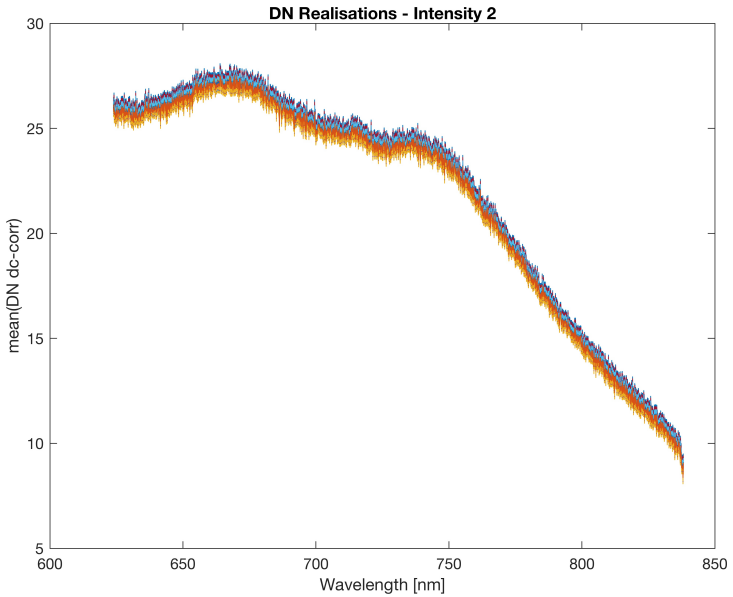
Note: individual random numbers for all spectral bands to achieve uncorrelated noise

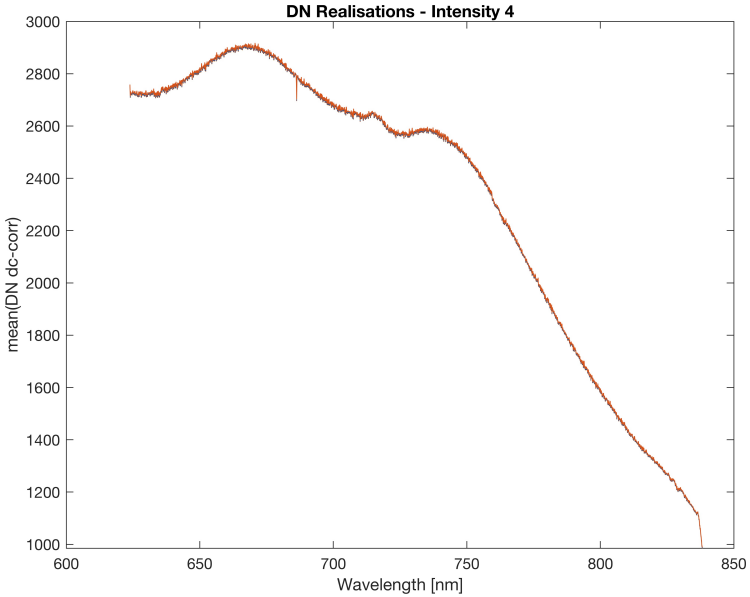
# Creating DN Realisations

Note: noise is random for the DNs

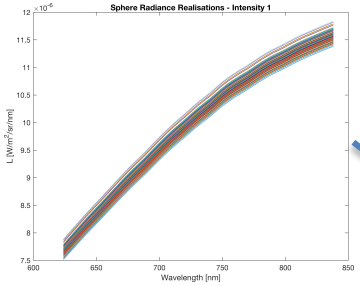


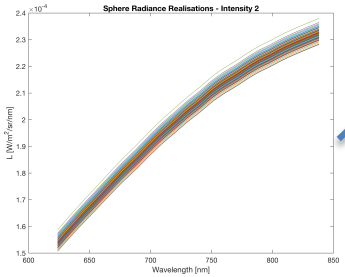



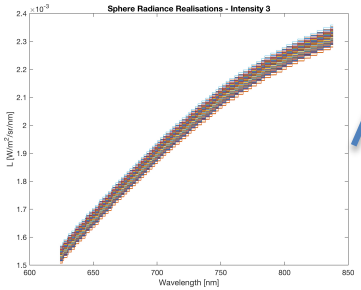



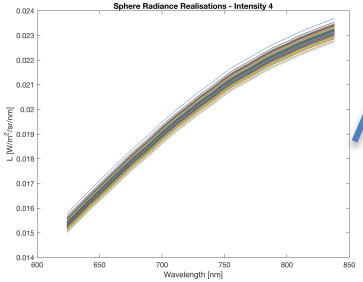



L realisations

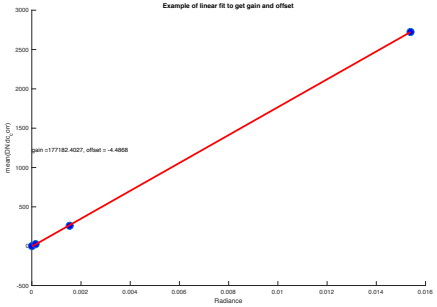


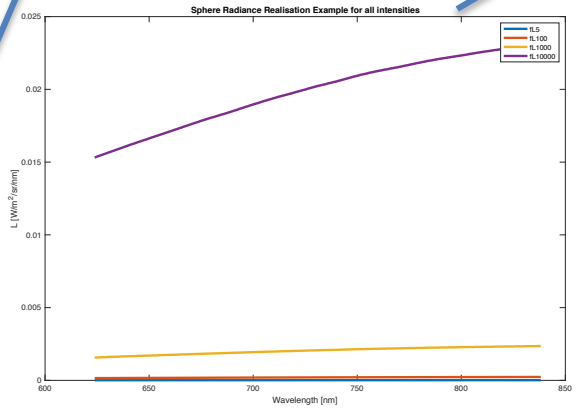

# Dataflow of a combined Monte Carlo Simulation for two PDFs

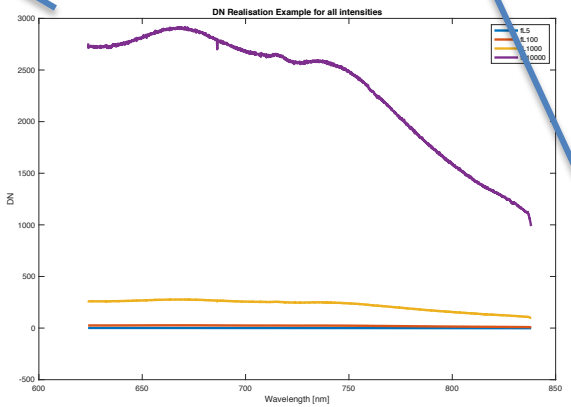



Select a realization from the L realisations

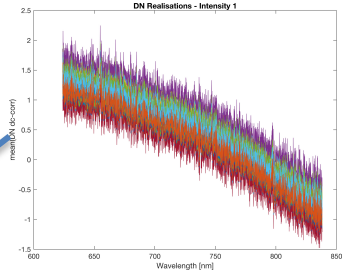
Do first order fitting to get gain and offset for these realisations for all bands

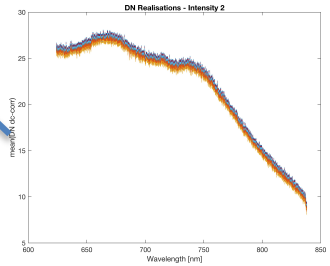
Select a realization from the DN realisations

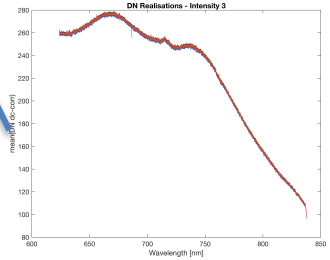


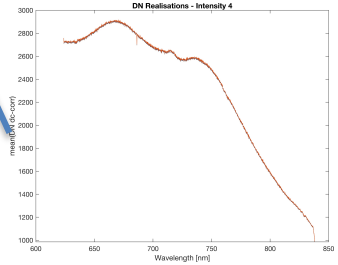




DN realisations



University of Zurich[UZH]

# Implementation Details: A nested Monte Carlo Run for two PDFs

measurements | products | policy

```
for n=1:N

    for m = 1:N

        for i=1:size(DN_l_dc,1)

            c = polyfit(L_realisations(:,i,n), DN_realisations(:,i,m), 1);
            gains(cnt,i) = c(1);
            offsets(cnt,i) = c(2);

        end

        cnt = cnt + 1;

    end

end
```

Combine realisations from both L and DN and then do linear fits for all spectral bands

University of Zurich[UZH]

RSL measurements | products | policy

# How many linear fits will we have to do?

- For N – number of realisations for both DN and L
- N = 100
- For 2047 spectral bands = n_bands

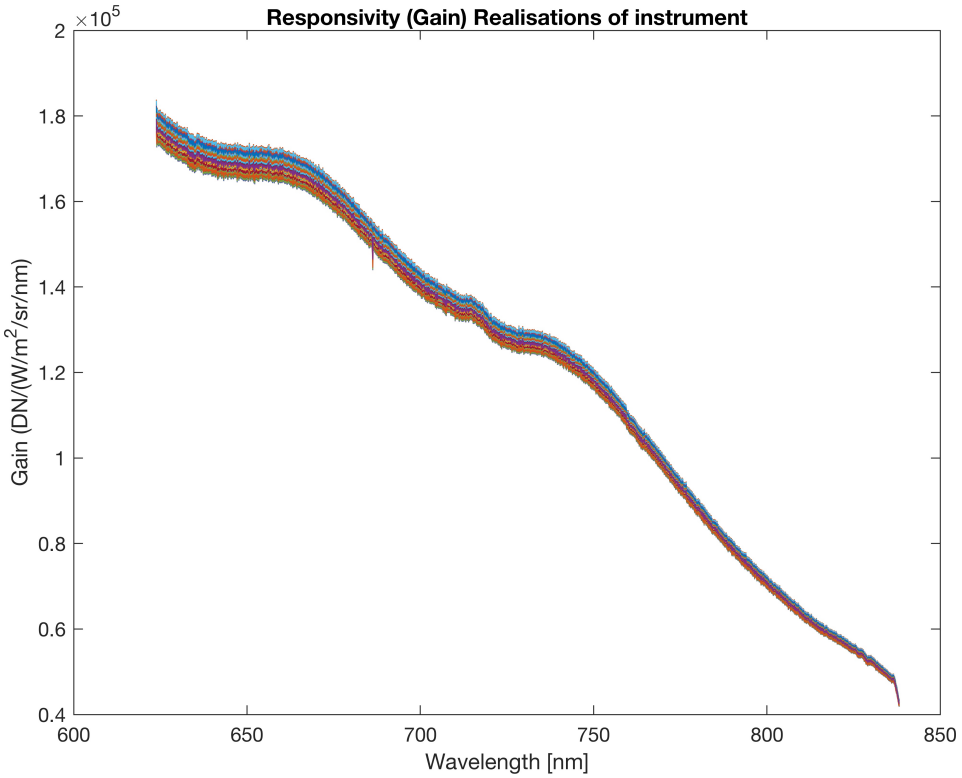
#linear_fits = N * N * n_bands = **20’470’000**

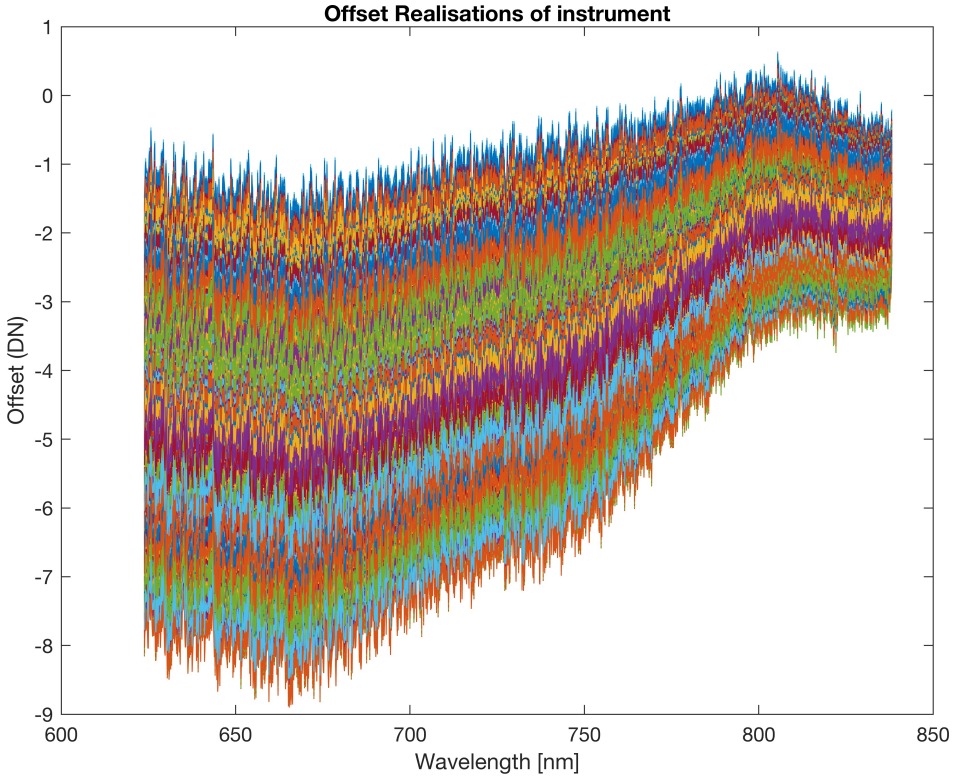
Runtime on a Macbook Pro: 1h 20’

University of Zurich[UZH]

RSL measurements | products | policy

# Realisations of Gains and Offsets

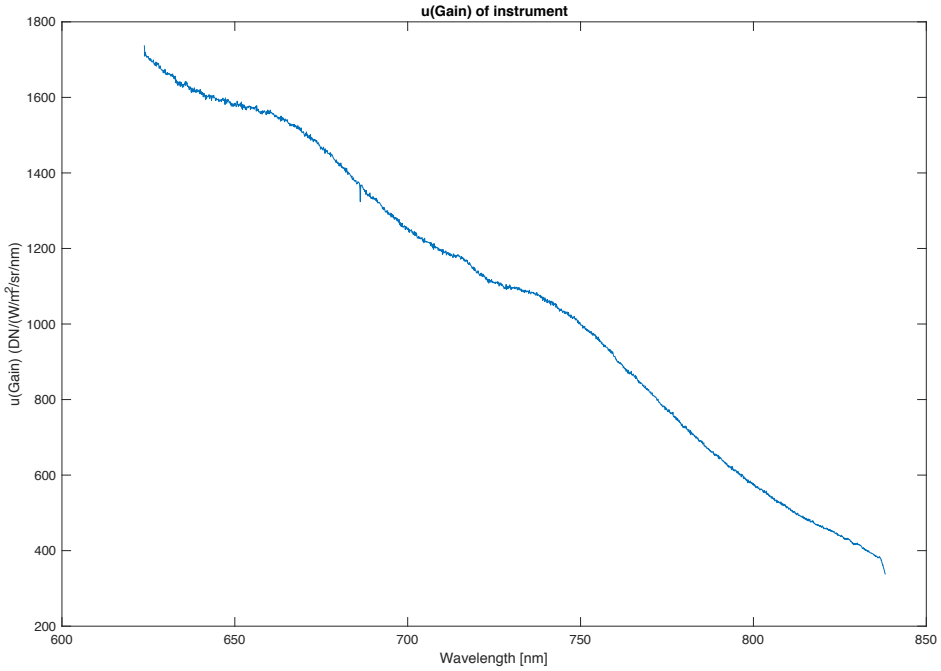


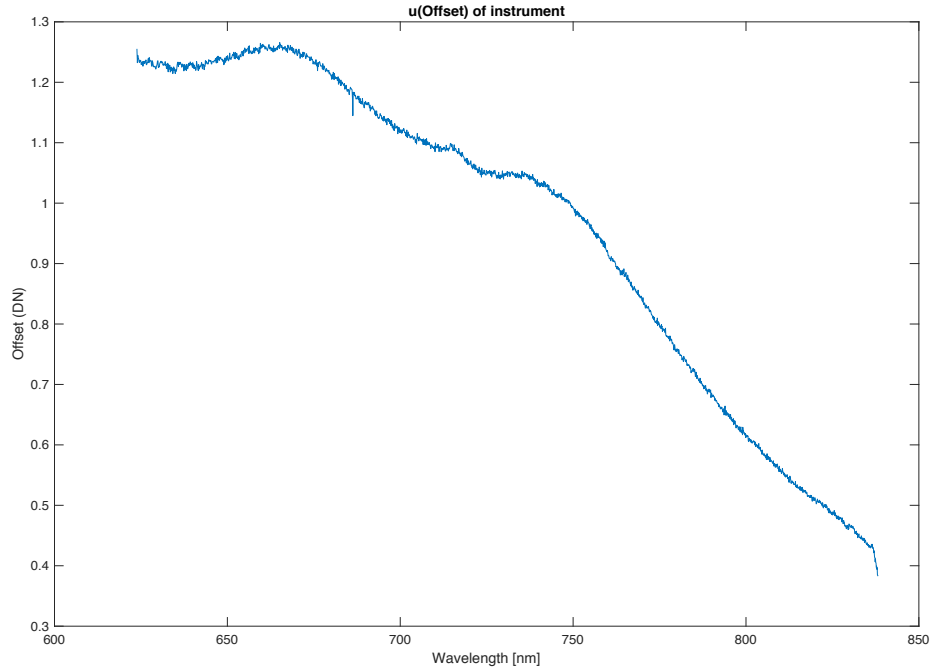



University of Zurich[UZH]

RSL measurements | products | policy

# Uncertainty of Gains and Offsets (Standard deviation of output realisations)



University of Zurich[UZH]

RSL measurements | products | policy

# Propagating the uncertainty of gain and offset to calibrated radiance

$$u^2(L_{scene}) = \frac{u^2(DN)}{G^2} + \frac{u^2(Offset)}{(-G)^2} + \frac{{L_{scene}}^2 u^2(G)}{(-G)^2} + 2 \cdot u(G, Offset) \frac{L_{scene}}{G^2}$$

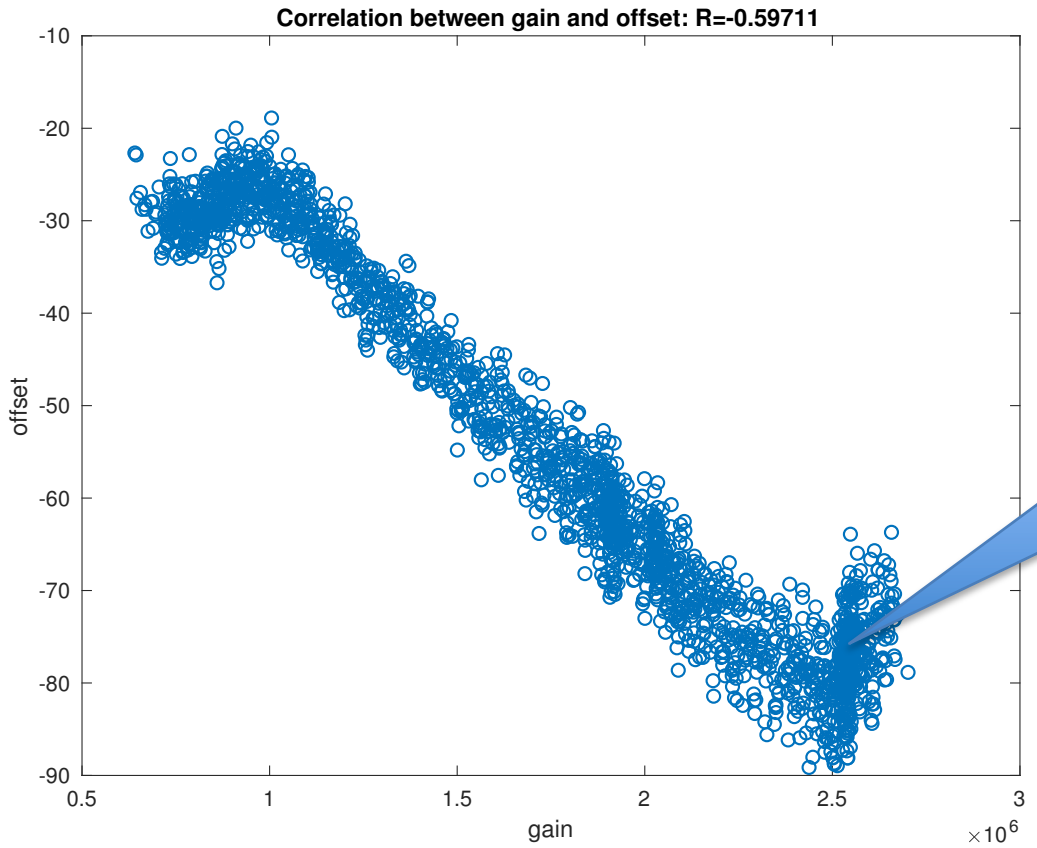
$$u(G, Offset) = u(G) * u(Offset) * corrcoeff(G, Offset)$$

For details see Uncertainty Textbook

University of Zurich[UZH]

RSL measurements | products | policy

# Correlation of gain and offset

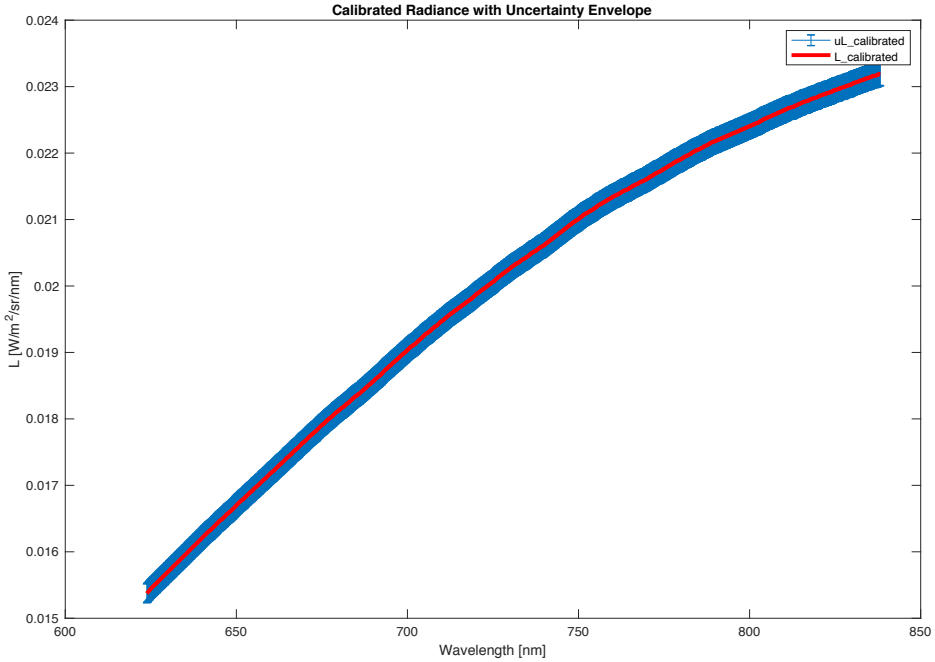


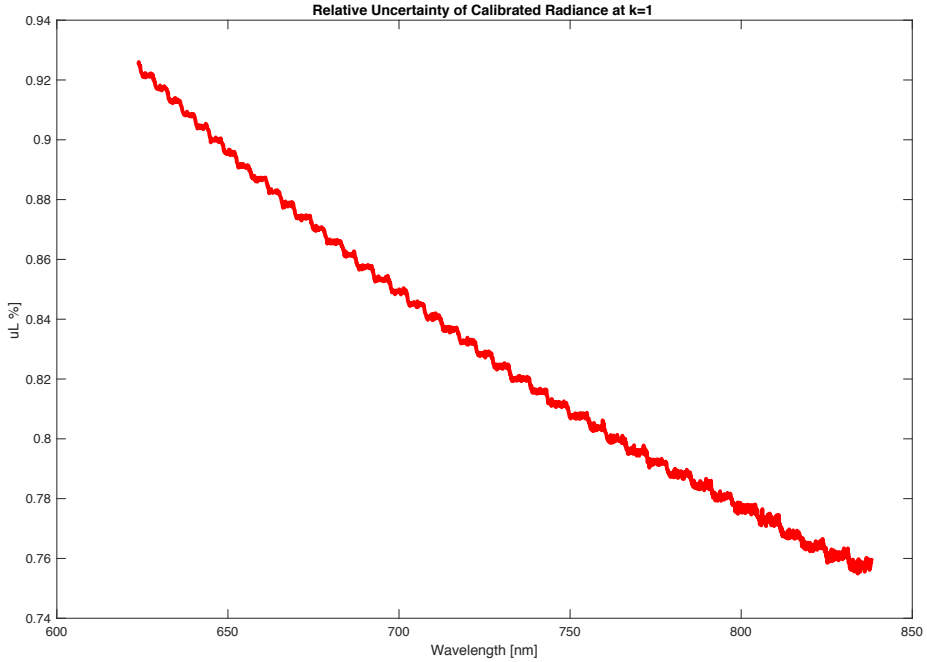

Likely an effect of lowest light setting (sensor non-linearity)

University of Zurich[UZH]

RSL measurements | products | policy

# Computed Uncertainty for a Calibrated Radiance Spectrum



University of Zurich[UZH]

# Example Code

RSL measurements | products | policy

| File | Description |
|---|---|
| RAD_CAL_with_Linear_Fit_and_uncertainty_estimation_with_Monte_Carlo.m | Main script |
| print_jpeg.m<br>print_pdf.m | Functions to export figure to JPEG or PDF |
| progressbar.m<br>gui_active.m | Functions for progress bar used to show progress during monte carlo run |
| L_Sphere.mat<br>STD_DN.mat<br>DN_L_CAL.mat<br>uL.mat | Input files for the code:<br>Sphere radiances, standard deviations of DNs, average of DNs the spectrometer recorded during calibration (corrected for dark current), uncertainty of sphere radiances |
| u_rad_coeffs.mat | Output of Monte Carlo run: uncertainties of gain, offset and uncertainty due to gain and offset correlation |
| offset_realisations.mat<br>gain_realisations.mat | Realisation matrices for both gain and offset |
| get_realisations_gauss_dist.m | Function to create realisations |