LINEARIZATION : FROM A CURVED SHAPE TO A STRAIGHT SHAPE

PREFACE

The linearization step prepares data for the next step, the XYZ conversion, which is a linear transformation. This transformation uses matrix computation that requiring linear data in input.

WHAT IS LINEAR AND NON-LINEAR ?

Linear ? Non-Linear ? What's the difference ?

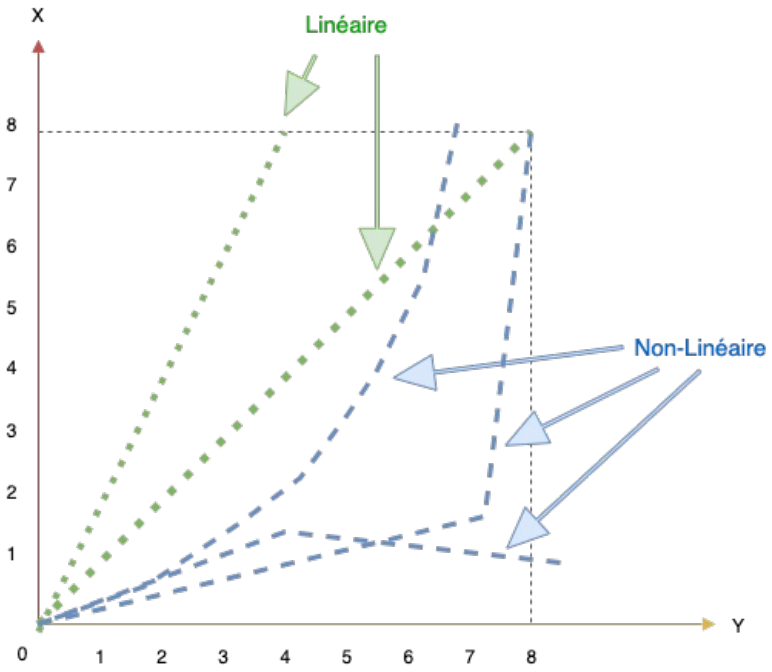
If you are not familiar with the mathematical concept of linearity, it's (rather) easy : a linear system is the proportionality of each value.

An example of linear system : one loaf of bread costs 1€ [1], if you buy two loaf of bread, the cost will be 2€. If you buy 10 loaf of breads, the cost will be now 10€, and so on : this is a linear system.

In a non-linear system, it's exactly the opposite : there is no proportionality for each value. A logical progression may not exist.

An exemple of non-linear system : the baker applies a discount based on the quantity of loaves of bread purchased. For each dozen, the discount will increase. This is a non-linear system : the continuity of the next value in this sequence may not be continuous.

Some few visuals examples for linear and non-linear :



In the world of imagery, we have to deal with linear systems and non-linear systems, especially with logarithmic. Don't be afraid, it will be easy :)

But why are we working with both systems ? Because in computer system, it's better to work with linear data, while in the real world, we are often in non-linear systems. We need to switch between them without any problem.

THE LINEARIZATION PRINCIPLE

Before going to complex computations (such as XYZ conversion), you must linearize each colorimetric value : computers (and matrixes) work better with :)

According to the documentation **SMPTE RP-431-2-2011 - D-Cinema Quality - Reference Projector and Environment**, the linearization process involves the following two steps [2] :

- Removing the gamma from the input image (unless if the gamma has already been removed, so the gamma equals 1.0, meaning the image is already linearized)
- Converting the colorimetric values to a floating-point range between 0.0 and 1.0. Even if it is possible to do without it, the norm recommands computing within a [0..1] range.

Concretely, the linearization aims to removing the gamma - or to bring back to a neutral gamma of 1.0 - before proceeding to a XYZ conversion, which absolutely requires a linear input to work correctly.

If you don't apply a linear conversion to your input, the XYZ conversion will still work (it's just matrix computation), but the result could be awful :)

QUICK EXAMPLE OF LINEARIZATION

Here is a quick example of R'G'B' linearization [3] to a linear RGB in the [0..1] range.
These concepts and examples will be studied in more depth in the following chapters especialy in the Gamma and Bitdepth chapters.

In our example, our input gamma is 2.6.
You must adapt this example based on the gamma from your input image.
For example, for an sRGB image, the gamma would be 2.2.

We suppose that our bitdepth is 12 bits per component with a 2.6 input gamma :

```
# SMPTE Equation :
# Channel = (Channel/4095)^2.6

# Red value equals 0x7ff (12 bits)
>>> Red = 0x7FF

# We convert the red value 0x7FF in the [0..1] range
# Why divide by 4095 ? We will see on the following chapters Gamma and Bitdepth
# But, to sum up, to convert a value 0x7FF in a value between 0 and 1,
# we must divide the value 0x7FF by the maximal value in this bitdepth,
# here, with a 12-bit bitdepth, the maximal value is 0xFFF (with all 12 bits set to 1)
>>> Red = (Red/4095)   # 4095 == 0xFFF
>>> print(Red)
0.4998778998778999
# Red is now within a [0..1] range.

# Now, we can remove the gamma which is non-linear.
# The gamma transfert function uses the power law for its computation.
# We remove its gamma using the pow() function :
>>> pow(0.4998778998778999, 2.6)
0.16483378645422764
```

Our RED value is now linear and equals 0.164833(...). We just need to apply the same computation to the other components Green and Blue to obtain a linear RGB.

With theses linear values, we can now move to the next step : the XYZ conversion (which requires a linear input)

NOTES

1. Or another price, it's just an example :) ↩

2. Linearization : ↩

   - *« The digital files were linearized (applying a gamma of 2.6), then a 3x3 matrix was applied to convert RGB to XYZ, followed by application of the (1)/2.6 gamma function. The finished color-corrected files were stored as 12-bit X'Y'Z' data in 16-bit TIFF files. »* -- Color and Mastering for Digital Cinema (Glenn Kennel)
   - *« Color conversion from R'G'B' to X'Y'Z' requires a three-step process which involves linearizing the color-corrected R'G'B' signals (by applying a 2.6 gamma function), followed by their passage through a linear 3x3 transform matrix. The resultant linearized and coded XYZ signals are then given an inverse 2.6 gamma transfer characteristic whose output is quantized to 12 bits. »* -- SMPTE RP-431-2-2011 - DCinema Quality Reference Projector and Environement - Chapitre « Color Conversion to XYZ »

3. `R'G'B` == `RGB + Gamma` ↩