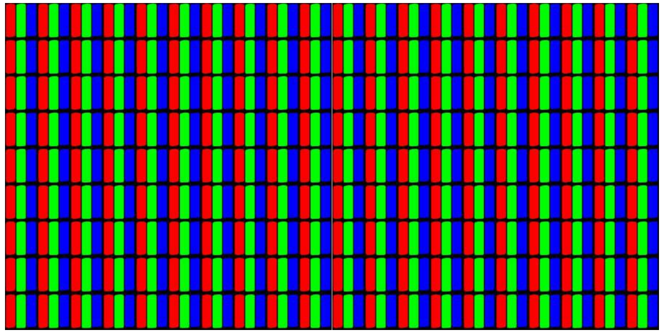
Chromatic Analysis of Numerical Program

David DEFOUR, LAMPS, Univ. of Perpignan

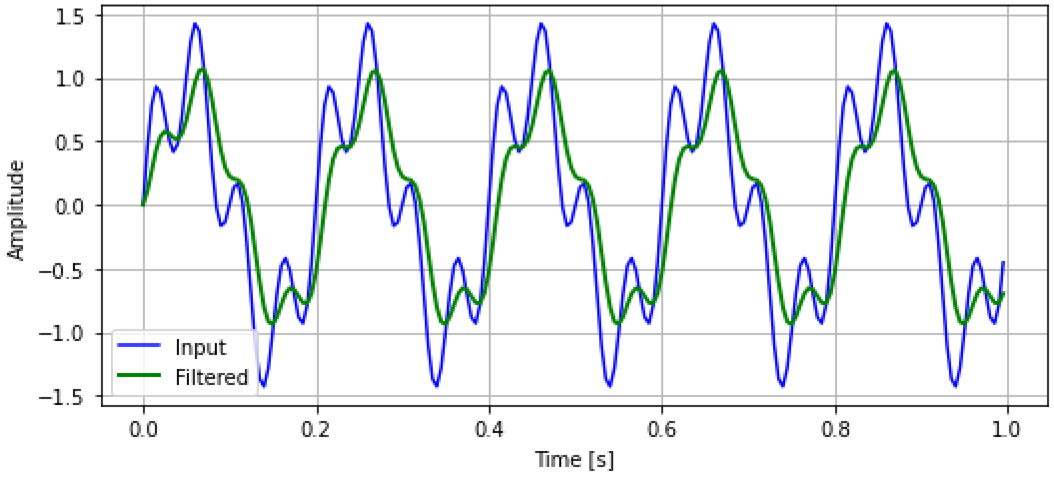
Franck Vedrine, Univ. Paris-Saclay CEA List



“A picture is worth a thousand words”

ChatGPT created

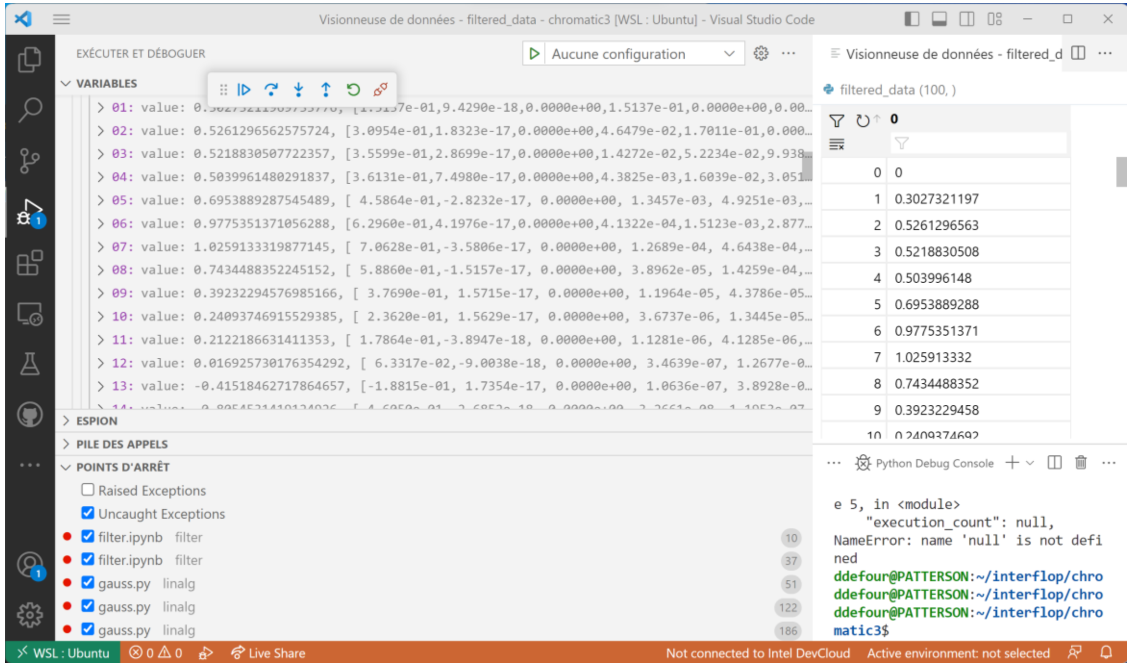
```python
import numpy as np
from scipy.signal import butter, lfilter, freqz
import matplotlib.pyplot as plt

# Create a low-pass Butterworth filter
def butter_lowpass(cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

# Apply the filter to the input signal
def butter_lowpass_filter(data, cutoff, fs, order=5):
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

# Example usage
# Generate some random input data
fs = 100.0       # Sample rate (Hz)
t = np.linspace(0, 1, int(fs), endpoint=False)
data = np.sin(2 * np.pi * 5 * t) + 0.5 * np.sin(2 * np.pi * 20 * t)

# Filter parameters
order = 6
cutoff_freq = 10.0   # Desired cutoff frequency (Hz)

# Apply the filter to the input data
filtered_data = butter_lowpass_filter(data, cutoff_freq, fs, order)
```



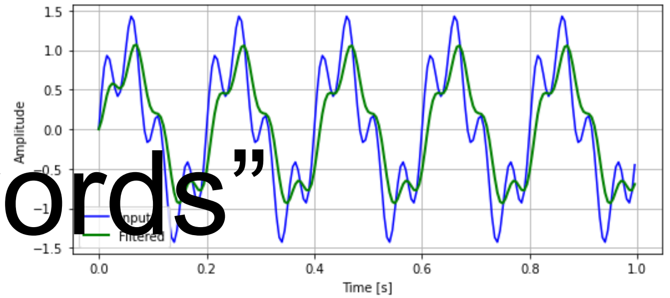
“A picture is worth a thousand words”

- Question:
  - How to analyze the relationship between input values, output values, coefficient, error ?
  - Usage: debugging, optimizing, teaching

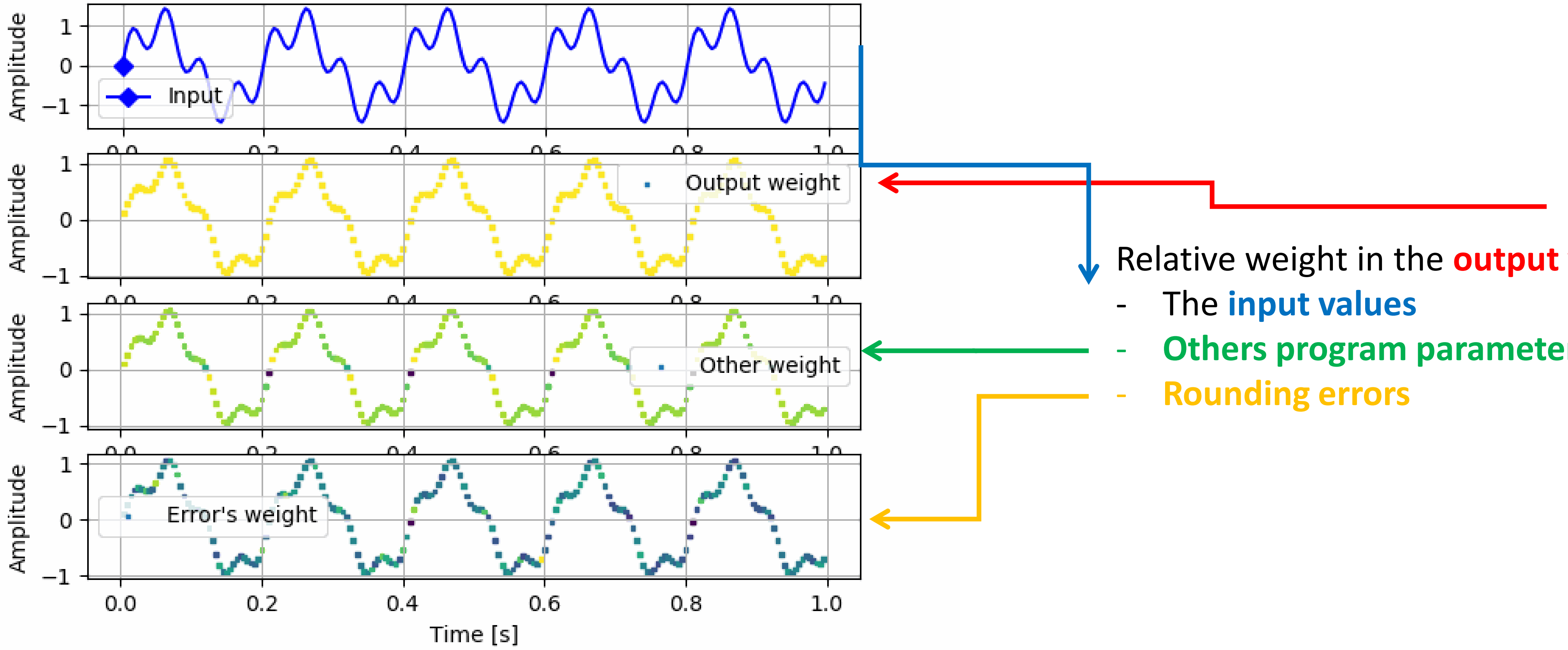
- Call the specialist…



“A picture is worth a thousand words”

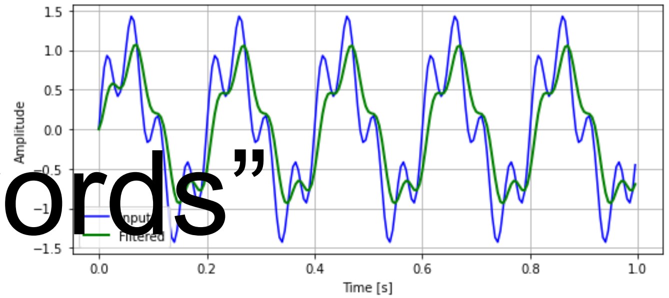


- … or conduce a chromatic analysis



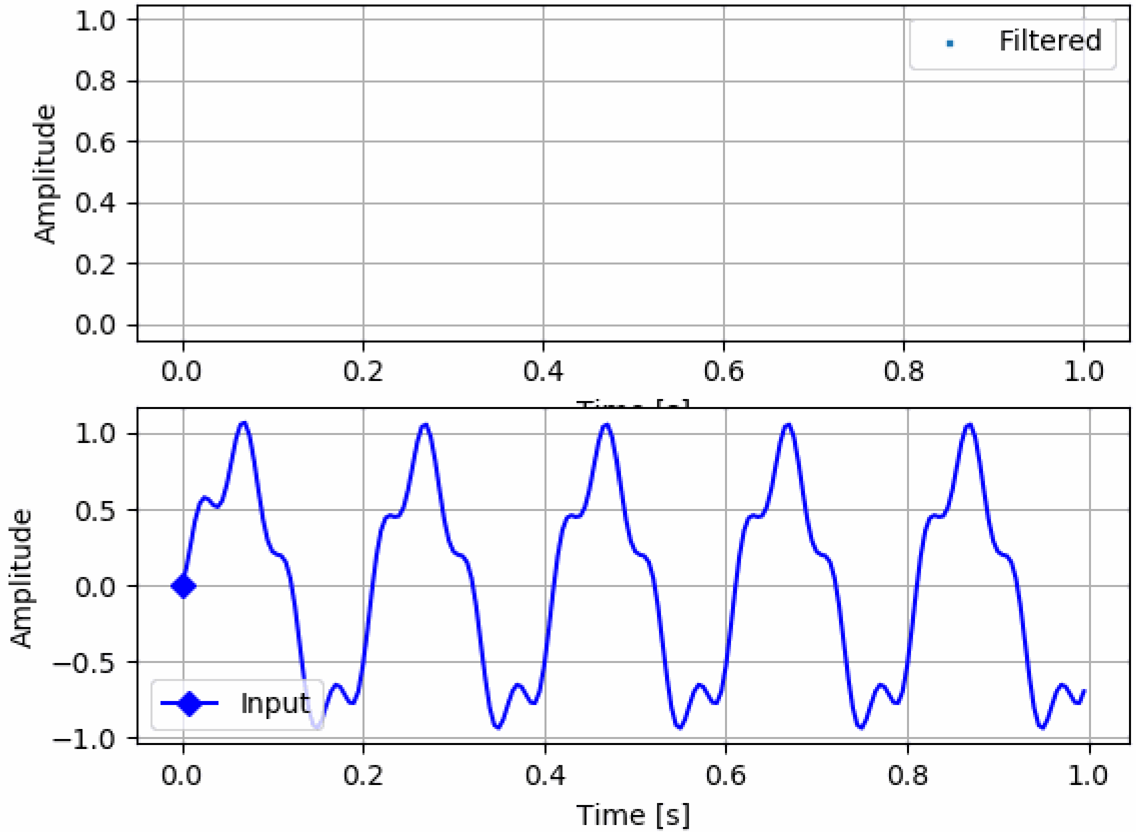
Relative weight in the **output value** of:

- The **input values**
- **Others program parameters**
- **Rounding errors**



“A picture is worth a thousand words”

- … or conduce a chromatic analysis



Given an output, what input value account for it

A few word about colors…. In RGB

RED (122,0,0) + **BLUE** (0,0,122) =

GREEN (0,122,0) + **BLUE** (0,0,122) =

RED (122,0,0) + **GREEN** (0,122,0) =

A few word about colors…. In RGB

RED (122,0,0)	**+**	**BLUE (0,0,122)**	**=**	**PURPLE (122,0,122)**
GREEN (0,122,0)	**+**	**BLUE (0,0,122)**	**=**	**CIAN (0,122,122)**
RED (122,0,0)	**+**	**GREEN (0,122,0)**	**=**	**YELLOW (122,122,0)**

A few word about colors…. In RGB

RED (122,0,0)	+	BLUE (0,0,122)	=	PURPLE (122,0,122)	+	PURPLE (122,0,122)	=	PURPLE (254,0,254)
GREEN (0,122,0)	+	BLUE (0,0,122)	=	CIAN (0,122,122)	+	CIAN (0,122,122)	=	CIAN (0,254,254)
RED (122,0,0)	+	GREEN (0,122,0)	=	YELLOW (122,122,0)	+	YELLOW (122,122,0)	=	YELLOW (254,254,0)

A few word about colors…. In RGB

RED (122,0,0)	+	BLUE (0,0,122)	=	PURPLE (122,0,122)	+	PURPLE (122,0,122)	=	PURPLE (254,0,254)
GREEN (0,122,0)	+	BLUE (0,0,122)	=	CIAN (0,122,122)	+	CIAN (0,122,122)	=	CIAN (0,254,254)
RED (122,0,0)	+	GREEN (0,122,0)	=	YELLOW (122,122,0)	+	YELLOW (122,122,0)	=	YELLOW (254,254,0)

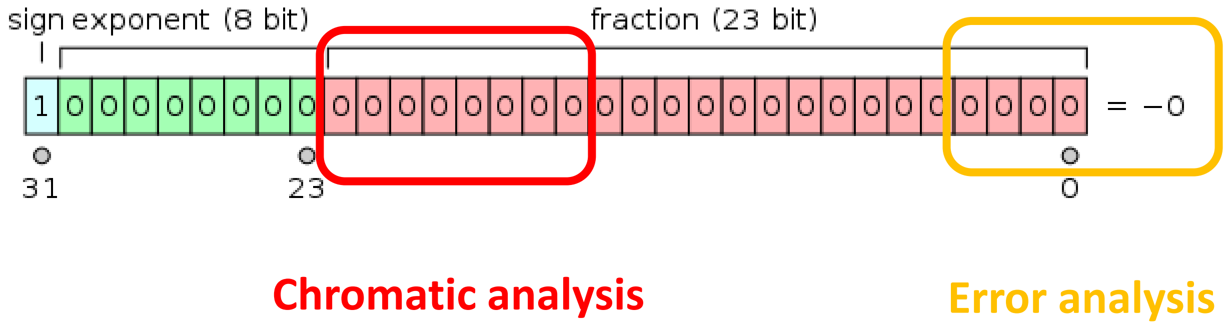
Colors naturally provides visual information under **additive property**

Introduction

- Assessment
  - For some applications (DNN), we are more concerned by understanding the resulting value than by the propagation of errors
- Objective
  - Estimate the relations between input and output variables under additive property
- Proposed solution
  - Propose the concept of chromatic number to tint scalar or set of scalars
  - Each scalar is decomposed as the sum of tinted values

Background

- Differences between chromatic analysis and error analysis
  - Example:
    - $f(x, y) = x + \exp(y)$ with $x = 10\ 000$ and $y = 8$
    - Error analysis => a small perturbation on y has a relatively stronger impact on f than one on x
    - Chromatic analysis => the weight of x (10 000) is far greater than the weight of y (2 980) in f(12 980)



Background

1. Sensitivity analysis
   - Evaluate how variations in input parameters affect the output
   - Identify which input parameters have the greatest effect on the output
   - Issues:
     - Curse of dimensionality, inability to handle correlated input, difficult to interpret variation on multiple input
2. Componentwise analysis
   - Condition number is a global measure that does not consider the input structure and dilute precise information into a global number.
3. Automatic Differentiation
   - Compute the gradient at each step
   - Forward or backward according to the input/output dimensionality
   - Possible implementation:
     - Each number X is replaced by a Dual Number $\langle x|x'\rangle$ where x' is the derivative such that $X = x + x'\varepsilon$ with ε an abstract number such that $\varepsilon^2 = 0$.

Chromatic number: Definition

- A **Chromatic Number** consists in associating a color to scalar or set of scalar in order to track them during computation
  - It correspond to a pair $\langle x|V_x\rangle$:
    - x is the floating-point number
    - V_x is a vector of n floating-point numbers representing the weight of the n tint within x
  - Additive property
    - $x \approx \sum_{i=0}^{n} V_x[i]$
- Property
  - V_x Corresponds to a component-wise decomposition of numerical values
  - Multiple scalars can be set with the same tint (helps tracking multiple values at the same time and helps reduces the dimensionality of the problem)

Chromatic number: Operations

- Set a new arithmetic on chromatic numbers:
  - Addition: $< \ x, V_x > \ + < y \, , V_y > = \ < x + y \, , V_x + V_y >$
  - Subtraction: $< \ x, V_x > \ - < y \, , V_y > = \ < x - y \, , V_x - V_y >$
  - Multiplication: $< x \, , V_x > . < y \, , V_y > = \ < x . y \, , \frac{y.V_x + x.V_y}{2} >$
  - Division: $\frac{<x \, , V_x>}{<y \, , V_y>} \ = < \frac{x}{y}, \frac{\frac{x}{V_y} + \frac{V_x}{y}}{2} > = < \frac{x}{y} \, , (\frac{x}{y^2} . V_y \ + \ \frac{V_x}{y})/2 >$
  - Sqrt(x): $\sqrt{< x \, , V_x >} = \ < \sqrt{x}, \frac{V_x}{\sqrt{x}} >$
  - Any functions: $f\left(< \ x, V_x >, < y \, , V_y >\right) = \ < f\left(x + y\right), \frac{f\left(x, V_y\right) + f\left(V_x, y\right)}{2} >$

Chromatic number: Extentions

- Garbage element
  - Set a specific element in V_x to collect contributions of non-chromatic numbers to preserve additive property.
  - Optional element if every computation were done without rounding error ($x = \sum_{i=0}^{n} V_x[i]$)
- Error element
  - Set an element to track rounding errors performed on x in $\langle x | V_x \rangle$
  - Accumulate rounding error similarly to compensated algorithm (use of EFT & extended precision)

Chromatic number: Implementation

- Space and time complexity grows linearly with the number of tinted values.
  - Example: A chromatic analysis on a 8 Mb dense matrix will lead to 8 Tb of intermediate representation.
  - C++/Python implementation with V_x stored either as a vector or dictionary
- Optimization 1: Fusion of small contributions
  - Discard tinted element which are becoming too small compared to others and accumulate them in the garbage element ($\left|\frac{V_{x[i]}}{V_{x[j]}}\right| \geq C$ with C a tunable parameter typically set to 2^{53} for double precision). Particularly useful when used when V_x is a dictionary structure.

Chromatic number: Implementation

- Optimization 2: Refinement algorithm
  - Start the chromatic analysis by aggregating the maximum number of value under the same tint in order to minimize the size of V_x .
  - Detect which tint account for the most and restart the computation by subdividing the selected tint, while detecting under-approximation (cancellation within a tint)

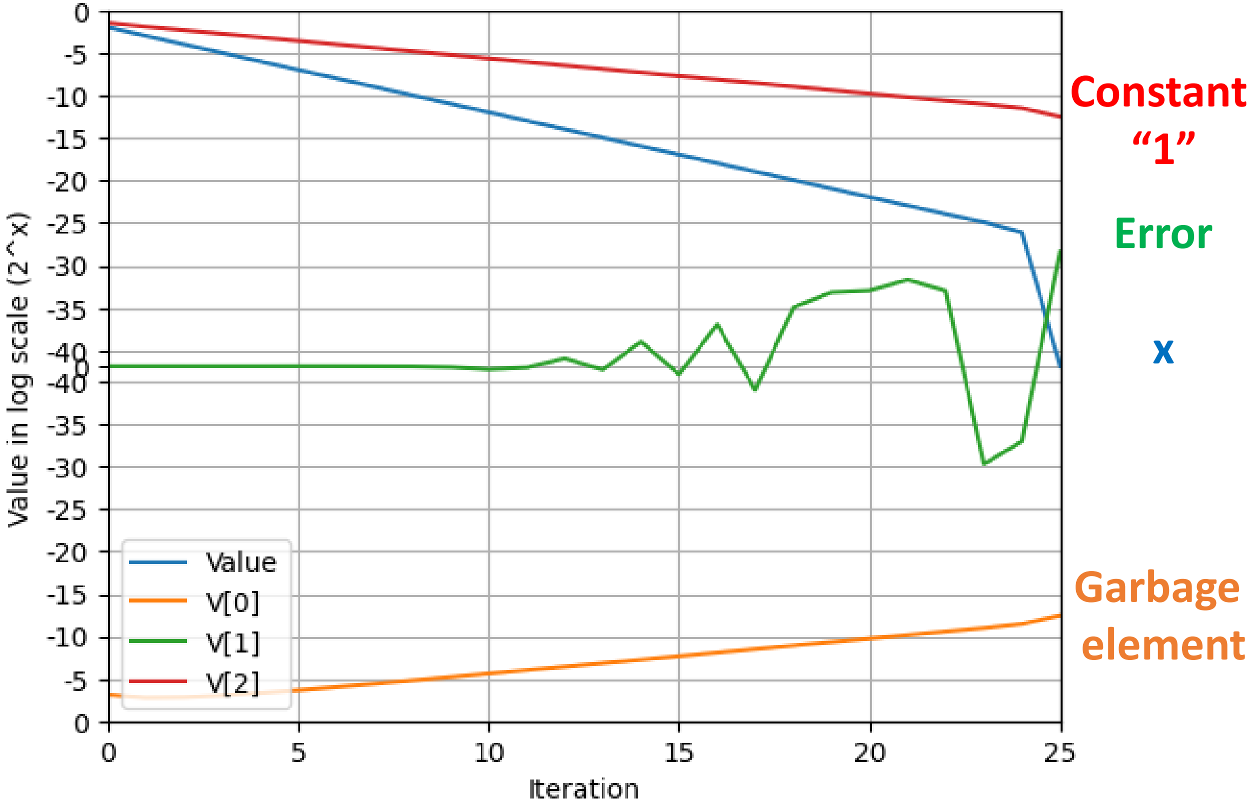
Algorithm 1 Contribution refinement subdivision algorithm

Require: *O=func(I)* the function to analyse
Require: I the set of scalar to track
Require: $card(I) = N$, and $O = \langle o, V_o \rangle$
  I'=split(I) ▷ Initial Spliting
  do
    O'=func(I')
    S =False
    for i in I' **do**
      if $|o'| > k_0|V_{o'}[1]|$ and $|V_{o'}[i+2]| > k_1|V_{o'}[1]|$ and card(I'[i])$>$ 1 **then**
        I'=split(I'[i])
        S =True
      end if
    end for
  while S

Experiments N°1: Archimedes’ computation of Pi

- Goal:
  - Track the weight of the initial constant 1

```
import math
from chnbr import ChNbr

ChNbr.setNumIdx(1)
t1 = ChNbr(1/math.sqrt(3), idx=2)
for i in range(1, N):
    t2 = (ChNbr.sqrt(t1*t1+1.0) - 1.0)
        /t1
    t1 = t2
```



Experiments N°2.1: inference DNN MNIST

- Goal
  - Track the weight of pixels during inference by assigning tint to each pixel of an image
  - MNIST 28x28 pixel images, 10 output class
  - Possible usage: adversarial attack to alter output probability classification (Fig. 2)
- Ouput
  - 10 chromatics numbers for each output class

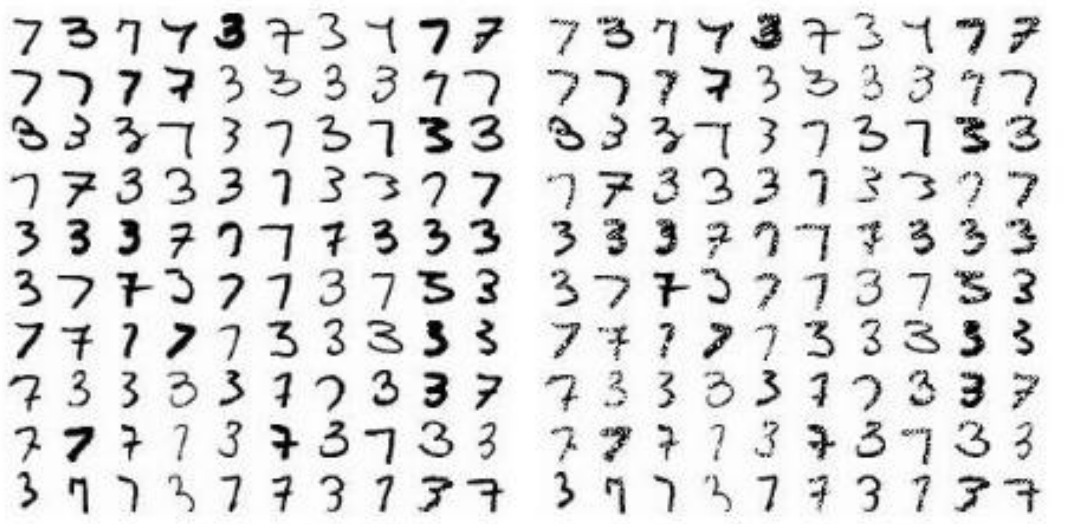


Fig. 2. Adversarial construction on MNIST dataset of 3s and 7s such that each example has a minimal number of pixels alterted to mislead the discrimination between the two sets among the ten classification bins.

Experiments N°2.2: Training DNN MNIST

- Goal
  - Track the weight of image class during learning phase
  - MNIST 28x28 pixel images, each pixel of an image tinted according to its classification (0 to 9)
  - Possible usage: understand the network numerical behavior
- Ouput
  - Resulting networking made of chromatic numbers tinted according to the input images class

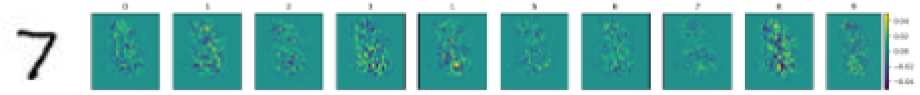


Fig. 3. Example of absolute pixel weight generated to classify image "7" with a given network trained with chromatic numbers, where image pixels are indexed according to the class to which they belong (index between 0 and 9). On each image, the color of the pixel corresponds to the contribution weight of the pixel.

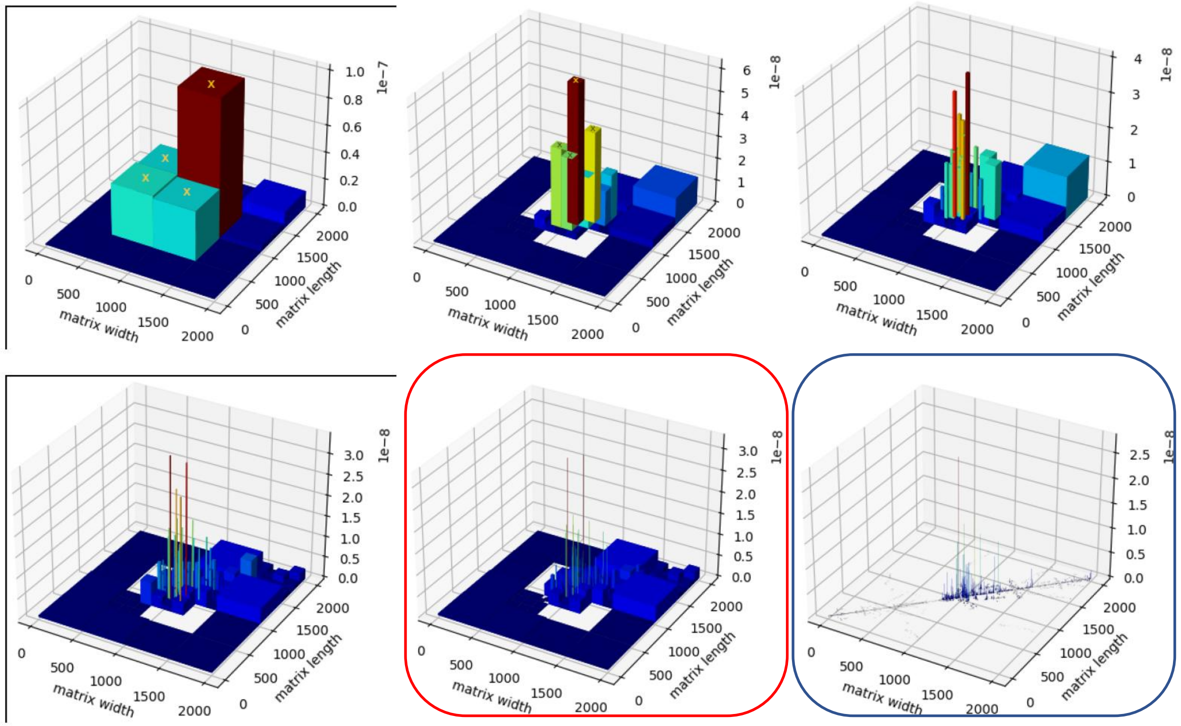
Experiment N°3: Sparse solver

- Matrix from MatrixMarket:
  - BCSSTK13: size 2003 x 2003; 42943 entries; estimated conditioned number 4.6 1010
  - BCSSTK14: size 1806 x 1806; 32630 entries; estimated conditioned number 1.3 1010
- Execution time in sec. and memory to solve BCSSTK14 between Python and C++ version.
  - 6-10x overhead in Python, 10-700x overhead in C++ (due to the sparsity of the system)
  - Memory usage grows linearly => x500-1000 on memory for 1000 tinted values

Number of tinted value followed	*no-instr*	1	16	32
Python	250s /70Mo	1634s /108Mo	2022s /125Mo	2500s /156Mo
C++	0.13s /21Mo	1.3s /26Mo	40s /135Mo	92s /253Mo

Experiment N°3: Sparse solver

- Iterative refinement algorithm , starting with a 4x4 subdivision according to the index in each direction of the matrix BCSSTK13.
- Stops after 5 iterations in 836 sec.



Reference
Analysis conduced while keeping the 128 most contributing tint in each cell. (2205 sec)
=> More time consuming and less precise than the iterative algorithm

Conclusion

- Chromatic analysis
  - Provide a numerical analysis based on the contribution of tinted scalars
  - Propose an additive decomposition of results
  - Allows fusion of input data to limit the dimensionality problem encountered with other analysis
  - Thanks to the additive property, it is possible to combine the process with an iterative refinement algorithm to reduce the memory overhead
  - Helps understand what is important among input values, constant, scalar
  - Cope with
- Future works
  - Combine chromatic analysis with others (global sensitivity analysis)
  - Investigate various tinting mechanism
    - According to data type, time, location (functions, MPI Process…),