# Temat

**„Projekt i implementacja adaptacyjnego systemu korekcji parametrów nagrań dźwiękowych”**

Celem pracy jest zaprojektowanie i implementacja adaptacyjnego systemu korekcji parametrów nagrań dźwiękowych z wykorzystaniem metod uczenia maszynowego.

## Zadania:

### Przygotowanie szkicu teoretycznego i raportu literaturowego

Literatura:

* Everest F. – Podręcznik akustyki

### Projekt aplikacji

Aplikacja ma wyznaczać:

* Barwę dźwięku – ulepszona spectral centroid
* Klarowność – energia pierwszych 50ms vs reszta
* Czas pogłosu – T60/T30/Early Decay time
* Pierwsze odbicie

### Przygotowanie środowiska pracy (instalacja oprogramowania i dodatkowych bibliotek programistycznych)

Python

### Implementacja kodu

Pycharm

### Testy aplikacji

Testy subiektywne oprogramowania – ocena działania programu w porównaniu z nagraniami wzorcowymi – ocena wierności odtworzenia charakteru nagrania, ocena naturalności brzmienia (osobno dla algorytmu machine learning).

# Wstęp

Celem pracy dyplomowej jest projekt oraz implementacja systemu analizy akustyki pomieszczenia, jako próbki nagrania audio w celu ekstrakcji parametrów akustycznych charakterystycznych dla tego nagrania, takich jak wpływ pomieszczenia na barwę dźwięku, czas pogłosu, czy też analiza odbić fali dźwiękowej. Parametry te nadają nagraniu charakterystyczne brzmienie.

Projektowany system dokonuje korekty nagrania, rozumianej jako manipulacja badanych wcześniej parametrów dopasowując je według danych założeń do wzorca. Badana jest również zmiana tychże parametrów, która może się okazać kluczowa do późniejszego rozwoju programu o uczenie maszynowe do tworzenia trenowania sieci. nałożenie zmierzonych parametrów na czyste nagranie studyjne.

System taki znajdzie zastosowanie między innymi jako wtyczka/rozszerzenie do programów obróbki dźwięku i będzie narzędziem efektowym wykorzystywanym do realizacji utworów muzycznych. Narzędzie to byłoby w stanie zarówno nakładać efekty pogłosowe dopasowane do pożądanego pomieszczenia, jak i również usuwać niechciany wpływ pomieszczenia na nagranie – usuwanie pogłosu z nagrania.

Nagranie poddawane analizie w początkowej fazie projektowania algorytmu jest odpowiedzią impulsową pomieszczenia, co dokładniej opisane zostanie w dalszej części pracy.

Analiza próbek nagrań dźwiękowych w następnych rozdziałach odbywa się w kontekście ludzkiego postrzegania zachodzących zjawisk, aniżeli ich opisu w zakresie fizyki. Dobór parametrów charakteryzujących badany sygnał ma więc na celu odzwierciedlenie elementów, które dobrze reprezentują postrzeganie zmiany zawartości sygnału po jego cyfrowej obróbce, a także zmiany sygnału dźwiękowego w zależności od cech pomieszczenia.

Parametryzacja nagrań w dalszym etapie pracy dyplomowej zostanie rozszerzona o uczenie maszynowe będące elementem eksperymentalnym oraz mające na celu usprawnienie obliczeń. Wpływ uczenia maszynowego na rozpoznanie charakteru nagrania (barwy itp.) zostanie zadany poprzez poddanie go ocenie subiektywnej przez słuchaczy.

# Teoria

## Czas pogłosu

Czas pogłosu definiuje się jako zanik energii o 60dB. Jest to tysiąckrotny zanik poziomu ciśnienia akustycznego, co zapisuje się:

.

Parametr T30 jest to czas, po jakim poziom energii spadnie o 60dB względem stanu początkowego, wyznaczany na podstawie nachylenia krzywej zaniku poziomu energii w zakresie 30dB, co umożliwia wyznaczenie czasu pogłosu nawet w przypadku, gdy mamy do czynienia z szumami.

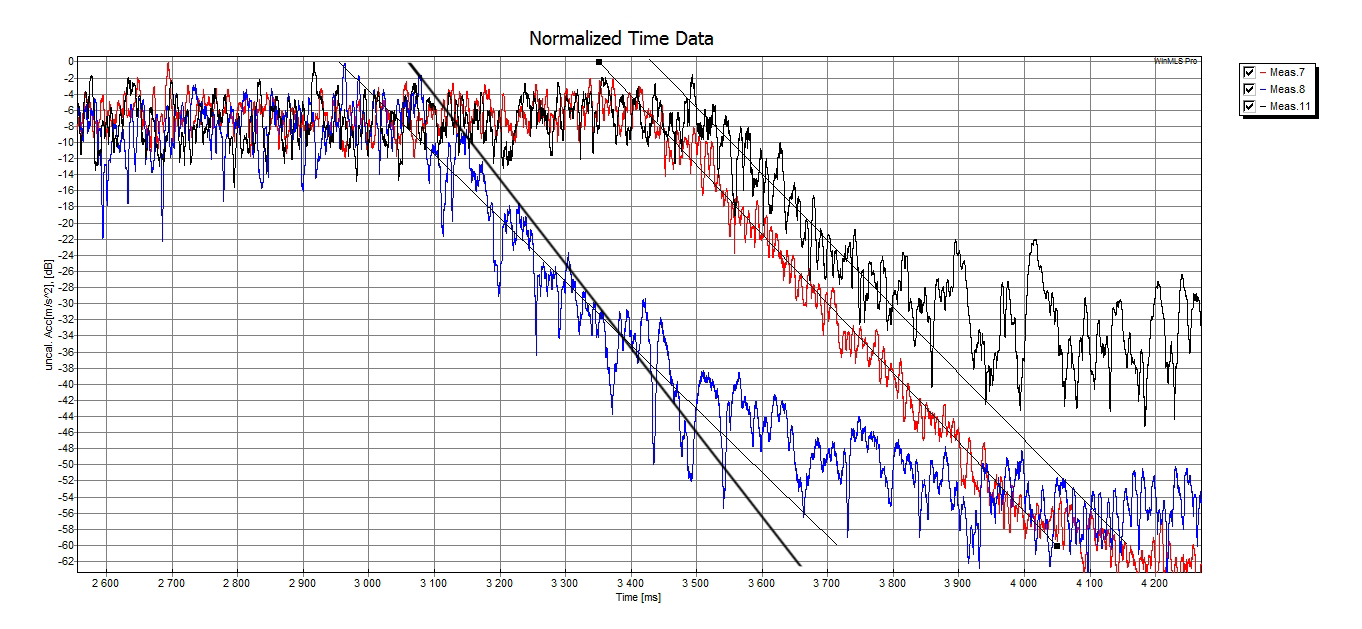
### Przykładowy pomiar zaniku poziomu energii w pomieszczeniu.

Tabela 1 Pomiar wymiarów pomieszczenia.

|  |  |
| --- | --- |
| wymiary pomieszczenia [m] | |
| długość | 5,92 |
| szerokość | 5,40 |
| wysokość | 3,40 |

Tabela 2 Współczynniki pochłaniania wybranych elementów pomieszczenia.

|  |  |  |
| --- | --- | --- |
|  | wsp. pochłaniania | powierzchnia [m2] |
| drzwi | 0,18 | 2,03 |
| meble | 0,17 | 7,00 |
| okno | 0,18 | 7,16 |
| ściany | 0,02 | 59,04 |
| drzwi | 0,17 | 1,76 |
| podłoga | 0,31 | 31,97 |
| sufit | 0,05 | 31,97 |



Wykres 1. Krzywe zaniku poziomu energii [czerwona - 2kHz, niebieska - 500Hz, czarna – bez filtra]

## Charakterystyka częstotliwościowa w zależności od długości impulsu.

Do pomiaru czasu pogłosu metodą impulsową należy uwzględnić rodzaj źródła tegoż sygnału. W poniższych przykładach zaprezentowano sygnał impulsowy z generatora odtwarzany na głośnikach. Źródłem takiego sygnału mogą także być: pękający balon, wystrzał z pistoletu.

W zależności od charakteru źródła, możemy mieć do czynienia z bardzo wąskim pasmem użytecznych częstotliwości, bądź też niskim poziomem sygnału, gdy sygnał ma bardzo krótki czas trwania (co zauważyć można na poniższych przykładach)



* Rysunek 2 Charakterystyka częstotliwościowa dla impulsu 50 μs

![](data:None;base64,)

Rysunek 3 Charakterystyka częstotliwościowa dla impulsu 75 μs



Rysunek 4 Charakterystyka częstotliwościowa dla impulsu 100 μs



Rysunek 5 Charakterystyka częstotliwościowa dla impulsu 200 μs



Rysunek 6 Charakterystyka częstotliwościowa dla impulsu 250 μs



Rysunek 7 Charakterystyka częstotliwościowa dla impulsu 500 μs



Rysunek 8 Charakterystyka częstotliwościowa dla impulsu 750 μs



Rysunek 9 Charakterystyka częstotliwościowa dla impulsu 1000 μs

## Barwa dźwięku

### **Spectal centroid**

Spectral centroid, czyli środek ciężkości charakterystyki częstotliwościowej sygnału jest parametrem, który został użyty w algorytmie w celu aproksymacji barwy dźwięku rozumianej jako częstotliwości dominujące w danym nagraniu w określonym odcinku czasu.

Do algorytmu wyznaczającego parametr centroidy częstotliwościowej dodano ważenie amplitudy zgodnie z założeniami prawa Webera Fechnera, ze względu na to, iż rozdzielczość FFT na wykresie wynosi 10Hz, co każdy punkt. Stąd też występuje zagęszczenie pomiarów dla wyższych częstotliwości dla skali logarytmicznej (która reprezentuje sposób postrzegana wrażeń dźwiękowych przez człowieka). Stąd, aby uzyskać bardziej naturalny dla ludzkiego ucha wynik działania tego algorytmu dodano ważenie wartości amplitudy FFT w odniesieniu do częstotliwości.

Przykłady

Przykład na wykresie [sinus 440Hz + szum biały] (kolejno FFT sygnału, watość SC ważone, wartość SC nieważone):



Kolejny przykład – utwór muzyczny

Kolejno:

* FFT w skali liniowej (X lin Y lin) [wartości Y nieważone]
* FFT w skali logarytmicznej (X log Y lin) [wartości Y nieważone]
* FFT X log i Y log [wartości Y nieważone]
* FFT X log i Y log [wartości Y ważone]

Legenda:

Poprawka ta widocznie udoskonala znajdowanie środka wagi widma.

# Projekt aplikacji – omówienie

Do implementacji projektu wybrano język Python w wersji 3,7, który charakteryzuje się szerokim zakresem zastosowań ze względu na wysokopoziomową składnię, pozwalającą na szybką implementację złożonych czy rozbudowanych aplikacji. Wsparcie w postaci różnorodnych i specjalistycznych bibliotek dla tego języka umożliwia znacząco szybsze rozwiązywanie zaawansowanych problemów, które na przykład w przypadku języka C++ wymagałyby wielokrotnie więcej nakładu pracy (co mogłoby być niewspółmierne do zalet płynących z zastosowania C++). Wersja 3,7 zapewnia wsparcie w możliwym późniejszym rozwoju aplikacji.

## Możliwości

Zrozumiałość mowy w danym pomieszczeniu najbardziej zależna jest od powstających wczesnych odbić, które definiują stosunek oryginalnego sygnału do jego ogona pogłosowego. Analizując odpowiedź impulsową pomieszczenia możemy więc analizować sygnał pod tym kątem.

Wykres – energia w pierwszych 50ms vs ogon

Dokładniejszą metodą analizy powstających odbić jest zastosowanie sygnału sinusoidalnego w postaci krótkiego impulsu, który można następnie przeanalizować dokonując korelacji z czystą reprezentacją tego sygnału.

Wykres – korelacja IR <-> sin

## Schemat działania aplikacji

## Opis działania

Można jeszcze wcisnąć moduł odczytu tkinter – filedialog A a końcu reprezentacja wyniku na wykresach, no bo przecież to jest projekt całej aplikacji

W pierwszym kroku algorytm dokonuje odczytu pliku w formacie wave, który umożliwia zapis nagrania bez kompresji. We wstępnej analizie badane są:

* ilość kanałów,
* częstotliwość próbkowania,
* ilość próbek.

Do wykonania tego etapu utworzono klasę WaveFile, która inicjowana jest adresem do pliku:

class **WaveFile**:  
 PATH = str()  
  
 def \_\_init\_\_(self, path):  
 self.PATH = path  
 self.FILE = wave.open(self.PATH)  
 [self.CHANNELS, self.SAMP\_WIDTH, self.FRAMERATE, self.NFRAMES, \*self.rest] = self.FILE.getparams()  
 self.RAW = np.fromstring(self.FILE.readframes(-1), dtype='int32')  
 self.RAW = self.RAW/max(self.RAW)

Przykład utworzenia instancji klasy:

orig\_file = WaveFile('./PathToFile/orig.wav')

Opis wykorzystanych zmiennych:

* PATH - ścieżka do pliku,
* FILE - obiekt modułu wave,
* CHANNELS - liczba kanałów,
* SAMP\_WIDTH - ilość bajtów dla jednej próbki (przeważnie jest to 2),
* FRAMERATE - częstotliwość próbkowania,
* NFRAMES - liczba próbek,
* RAW - nieprzetworzony zapis pliku wave.

Odczyt wartości próbek zapisany do zmiennej RAW (będącej jednowymiarową tabelą typu int32[[1]](#footnote-1)), która w następnym kroku algorytmu zostaje znormalizowana (w wyniku operacji dzielenia typ zmiennej ulega rzutowaniu na typ float).

Celem wyżej wymienionych analiz pliku wave jest przygotowanie odpowiednich kroków w dalszej części algorytmu.

### Pogłos - Badanie odpowiedzi impulsowej

Najpierw obliczany jest kwadrat dla każdej wartości próbki:

for i, val in enumerate(frames):  
 f\_squared[i] = do\_square(val)

Nastepnie odwrócono kolejność próbek:

f\_squared = np.flip(f\_squared)

Po czym dokonano całkowania i utworzono oś czasu X:

integral = integrate(f\_squared)  
time = np.linspace(0, len(integral)/FP, num=len(integral))

OPISAĆ:

db30o = x[y.searchsorted(-30, 'left')]  
db30 = round(db30o\*1000\*CHOP)  
db0o = x[y.searchsorted(-0.2, 'right')]  
db0 = round(db0o\*1000\*CHOP)  
print("0dB---> at: " + str(db0) + "ms")  
print("-30dB-> at: " + str(db30) + "ms")  
print("-60dB-> at: " + str(db30+(db30-db0)) + "ms [extrapolated]")  
print("T30------>: " + str((db30-db0)\*2) + "ms")  
  
x1 = int((db30o)\*FP) # spadek o 30dB  
x2 = int((db0o)\*FP) # odniesienie  
x3 = int((db0o+((db30o-db0o)\*2))\*FP) # spadek o 60dB szacowany

Pzenieść z opracowanego rozdziału pogłos

Na pliku reprezentującym odpowiedź impulsową pomieszczenia przeprowadzana zostaje operacja całkowania.

Równanie 1 PN-EN ISO 3382

Z wykresu krzywej zaniku poziomu energii szacowany jest spadek o 60dB na podstawie zakresu 30dB [parametr T30].

### Analiza zmiany barwy

Analiza plik wejściowy a wyjściowy

### Transformacja sygnału

Nałożenie zmierzonej odpowiedzi impulsowej na sygnał nagrany w warunkach zbliżonych do warunków studyjnych (niezauważalny pogłos oraz brak zniekształceń sygnału).

def nextpow2(L):  
 N = 2  
 while N < L: N \*= 2  
 return N  
  
  
def convolution(x, h):  
  
 L = len(h) + len(x) - 1 # linear convolution length  
 N = nextpow2(L)  
  
 H = np.fft.rfft(h, N) # Fourier transform of the impulse  
 X = np.fft.rfft(x, N) # Fourier transform of the input signal  
  
 Y = H \* X # spectral multiplication  
 y = np.fft.irfft(Y) # time domain again  
  
 y = np.array(y/(max(y)\*1.001), dtype='float32')  
 return y

splot = convolution(orig\_file.RAW, IR\_file.RAW)  
  
plot(orig\_file.RAW,IR\_file.RAW, splot)

REPREZENTACJA WYNIKU

def plot(toPlot, \*args):  
  
 print(len(args))  
 N\_PLOTS = 1 + len(args)  
  
 plt.subplot(N\_PLOTS, 1, 1)  
 plt.plot(timeLine(toPlot, 44100),toPlot)  
  
 for index, pl in enumerate(args):  
  
 plt.subplot(N\_PLOTS, 1, index+2)  
 plt.plot(pl)  
  
 plt.show()

def plot\_signals(\*args: ***Signal***):  
  
 for i, arg in enumerate(args):  
 plt.subplot(len(args),1, i+1)  
 plt.title(arg.title)  
 if arg.timedomain:  
 plt.xlabel('time [s]')  
 plt.plot(arg.time, arg.data)  
 else:  
 plt.plot(arg.data)  
 plt.xlabel('frequency [Hz]')  
 plt.show()

orig\_file = WaveFile(ORIG)  
IR\_file = WaveFile(IR)  
  
sig\_1 = Signal(file=orig\_file,  
 title="Original file",  
 timeDomain=True)  
sig\_2 = Signal(file=IR\_file,  
 title="Impulse response",  
 timeDomain=True)  
  
plot\_signals(sig\_1, sig\_2)

### Zapis nowego pliku

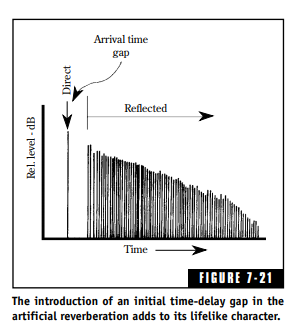
wav\_file = wave.open("file2.wav", 'w')  
   
nchannels = 1  
sampwidth = 2  
framerate = 44100  
nframes = len(splot)  
comptype = "NONE"  
compname = "not compressed"  
  
wav\_file.setparams((nchannels,  
 sampwidth,  
 framerate,  
 nframes,  
 comptype,  
 compname  
 ))

wav\_file.writeframes(np.array(splot, dtype='float32'))  
wav\_file.close()

## Pogłos

Oddanie naturalności brzmienia, w kontekście zjawiska pogłosu wymaga między innymi rozpatrzenia jednego z ważnych elementów, jakim jest opóźnienie pomiędzy źródłem dźwięku, a pierwszym odbiciem, które to nadaje wrażenie wielkości pomieszczenia.

Fala dźwiękowa pokonując bezpośrednią drogę dociera najszybciej do słuchacza, następnie docierają wczesne odbicia. Opóźnienie to mierzone jest przez algorytm [fragment kodu z omówieniem].



Kolejnym elementem jest gęstość ech występujących w ogonie pogłosowym, która jest wymagana, aby dźwięk wydawał się naturalny i aby nie wystapił efekt flutter. Zgodnie z badaniami Schroedera szacuje się, że gęstość ta wynosi około 1000 ech na sekundę.

### Pomiar czasu pogłosu

Algorytm wyliczający czas pogłosu bazuje na metodzie impulsowej. Do wykonania nagrań odpowiedzi impulsowej jako użyto różnego źródła dźwięku różnego rodzaju, między innymi – pękający balon, trzaśnięcie, klaśnięcie, aby oszacować wpływ jakości nagrania odpowiedzi impulsowej na późniejszą operację splotu [całe badanie ma charakter subiektywnego postrzegania działania algorytmu, aniżeli odwzorowywania pomiarów zgodnie ze sztuką].

Do zminimalizowania poziomu tła w nagraniach dokonano analizy w pasmach częstotliwościowych [jakich? – pasma +spadki] przy użyciu filtrów cyfrowych, co pozwoliło na uzyskanie większego zakresu (około 30-50dB) zaniku poziomu dźwięku pozwalającego oszacować jego spadek o 60dB.

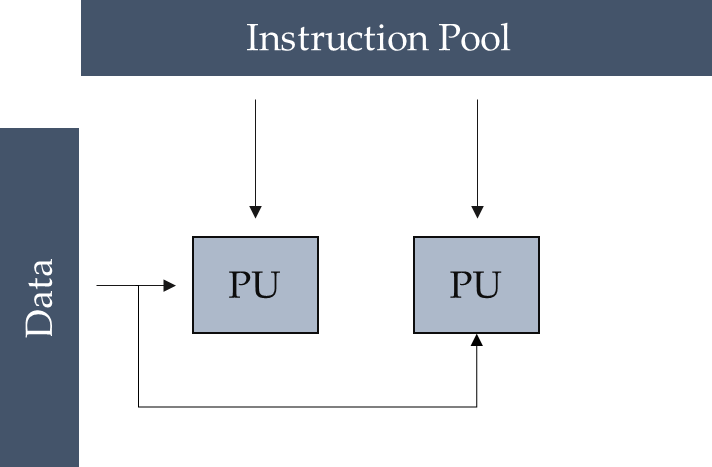
Nagrania wykonano w trzech miejscach położenia mikrofonu oraz trzech miejscach położenia źródła dźwięku (co daje razem 9 nagrań na jedno pomieszczenie).

Z założeń projektowych algorytmu, przy analizie nagrania pominięto częstotliwości mniejsze, jak 120Hz ze względu na dodatkowe trudności występujące w analizie tychże częstotliwości, takie jak rezonanse osiowe, styczne i skośne, a także wpływ jakości mikrofonu na pomiar częstotliwości w tym zakresie. Zjawiska te mogą wpływać na dodatkowe błędy, które utrudnią analizę działania projektowanego algorytmu, z tego powodu zostaną one pominięte, a analiza zostanie przeprowadzana w przypadkach „idealnych”, czyli niestwarzających dodatkowych problemów.

## Równoległość parametryzowania

W przypadku analizy plików o większej objętości czas obliczeń można skrócić stosując zrównoleglenie procesu parametryzacji. Obecnie równoległość przetwarzania danych jest standardem dla profesjonalnych aplikacji komputerowych, gdyż są one w stanie wykorzystać potencjał, który zapewnia wielordzeniowość dzisiejszych komputerów

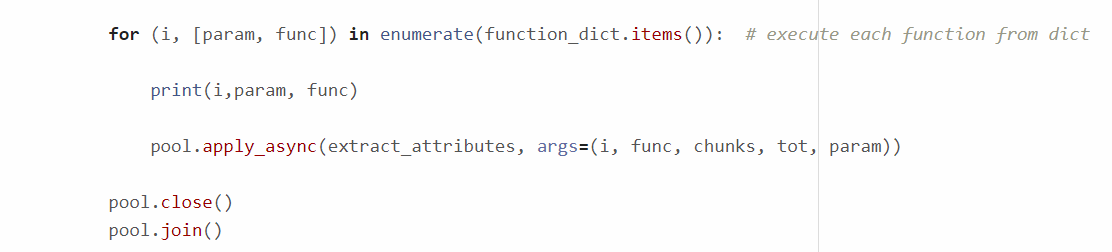
Propozycją wykorzystania równoległości jest architektura „Multiple Instructions, Single Data”.



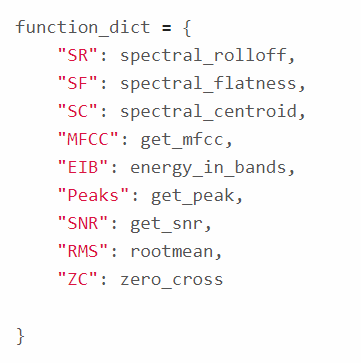
Wybór tej metody uzasadniam poprzez charakter projektowanej aplikacji, która analizuje w danym czasie wyłącznie jeden plik, stąd też nie możemy zastosować architektury „Single Instruction, Multiple Data”, która jest częściej spotykanym podejściem.

Algorytmy analizy sygnału w czasie, jak i w dziedzinie częstotliwości dokonują operacji i przekształceń wszystkich próbek, stąd też w większości mają te same czasy potrzebne na ich wykonanie. Dzięki zrównolegleniu uzyskujemy zamiast czasu analizy, po prostu .

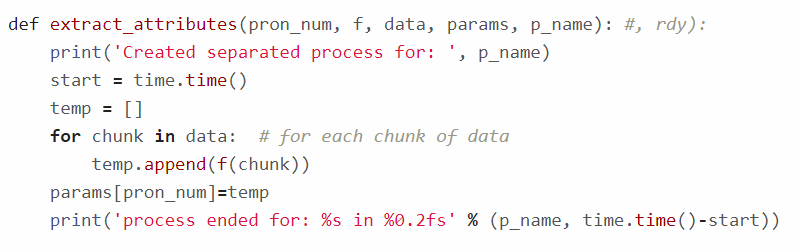
Przykład kodu zgodnego z powyższymi założeniami, realizującego równoległą parametryzację:



Dla słownika function\_dict, wszystkie jego przedmioty (ang. items), czyli funkcje wykonywane są na sygnale, ale w osobnych wątkach wywoływanych za pomocą instrukcji pool.apply\_async() powodujące asynchroniczne wykonanie funkcji parametryzującej.



Powyżej zaprezentowano przykład słownika zawierającego spis funkcji, które należy wykonać na sygnale. Nazwa funkcji w języku Python jest wskaźnikiem na adres tej funkcji, co umożliwia wywoływanie w pętli różnej funkcji przy tej samej zmiennej.



Ostatnim elementem jest sama funkcja bezpośrednio wywoływana w pętli i wykonująca daną instrukcję ze słownika funkcji parametryzującej. Pełni ona rolę „Wrappera” poprzez iteracyjne wykonywanie analizy i zapamiętywanie wyniku w tablicy i ostatecznie zapisywanie do pamięci współdzielonej reprezentowanej poprzez zmienną params.

# Podsumowanie

Np. zestawienie - wykres zmiany barwy wejście a wyjście

Ocena jakości działania programu

Coś o możliwym zastosowaniu uczenia maszynowego i możliwych korzyści

# Uczenie maszynowe

## Wstęp

W celach eksperymentalnych założono implementację algorytmów uczenia maszynowego do wcześniej omówionego programu.

Uczenie maszynowe pozwala na optymalizację oraz znajdywanie rozwiązań dla złożonych problemów w mniej konwencjonalny sposób, niż wierne sztuce metody opisywane w normach.

1. 32 bitowa liczba całkowita z zakresu od do . [↑](#footnote-ref-1)