Obrada audio signala u stvarnom vremenu

Fran Jelavić

Voditelj: *Ivan Đurek*

Fakultet elektrotehnike i računarstva

Uvod

- DSP (Obrada digitalnog signala) u stvarnom vremenu
  - kompresija i prijenos informacija
  - poništavanje buke i uklanjanje šuma
  - prepoznavanje glasa
  - obrada i produkcija glazbe

- Kontinuirani signal → diskretni signal → obrada signala

Osnovni koncepti DSP-a

Kvantizacija

FFT

Konvolucija

Kvantizacija

- Analogni signal → digitalni signal
  - uzorkovanje, zatim kvantizacija
- **Problem**: nedostatak beskonačne memorije za pohranu signala

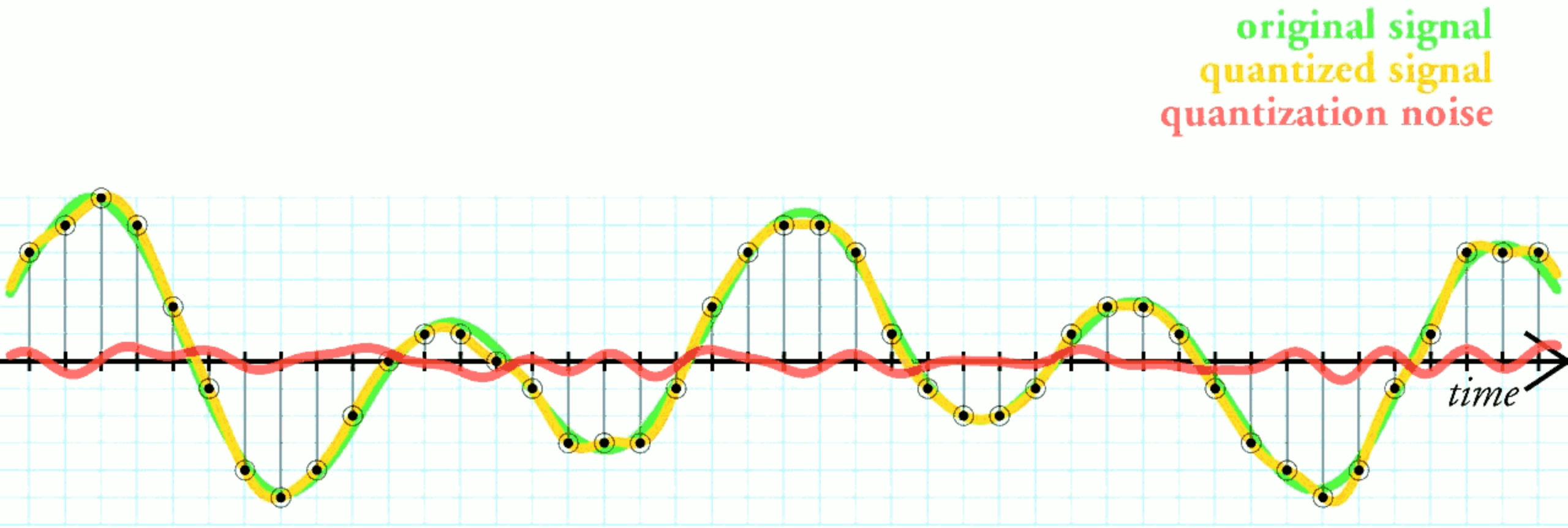
- Mapiranje vrijednosti amplitude:
  - **kontinuiranih** vrijednosti → **konačan skup diskretnih** razina
- Dijeljenje kontinuiranog raspona vrijednosti amplituda na konačan broj razina
  - uz preslikavanje tih vrijednosti na najbližu razinu

Kvantizacija

- Broj razina određen brojem bitova
  - npr. signal predstavljen sa 16 bitova → 2^{16}(65,536) različitih razina amplitude

- Greška (**šum**) **kvantizacije**
  - razlika između izvorne kontinuirane i kvantizirane vrijednosti amplitude
  - povećanje broja razina:
    - smanjuje šum
    - usporava proces obrade signala; memorijski zahtjevnije

Kvantizacija



FFT (*Fast Fourier Transform*)

- Digitalni signal → reprezentacija signala u frekvencijskoj domeni
  - skup brojeva iz kojeg se lako raspoznaju najzastupljenije frekvencije

- $fft_{izlazni\ niz}$= [23, 43, 65, 443, 321, 54, 56 ...]
  - svaki indeks (***bin***) – određen raspon frekvencija
  - vrijednost *bin*-a – intenzitet frekvencija u rasponu
- Raspon frekvencija ovisi o rezoluciji:
  - $rezolucija = \frac{f_{uzorkovanja}}{fft_{veličina\ spremnika}}$

Konvolucija

- Množenje dva signala i integriranje rezultata kroz vrijeme
  - tj. kontinuirana suma umnoška amplituda signala A i signala B kroz vremensku domenu
- Konvolucija s impulsnim odzivom: $y(t) = \int_{-\infty}^{\infty} x(t) \cdot h(t-\tau) d\tau$
- Implementacija FFT-om:
  - množenje u frekvencijskoj domeni i vraćanje rezultata u vremensku domenu inverznim FFT-om (IFT)
- Računalno zahtjevno za duže signale

Audio efekti

Delay

Reverb

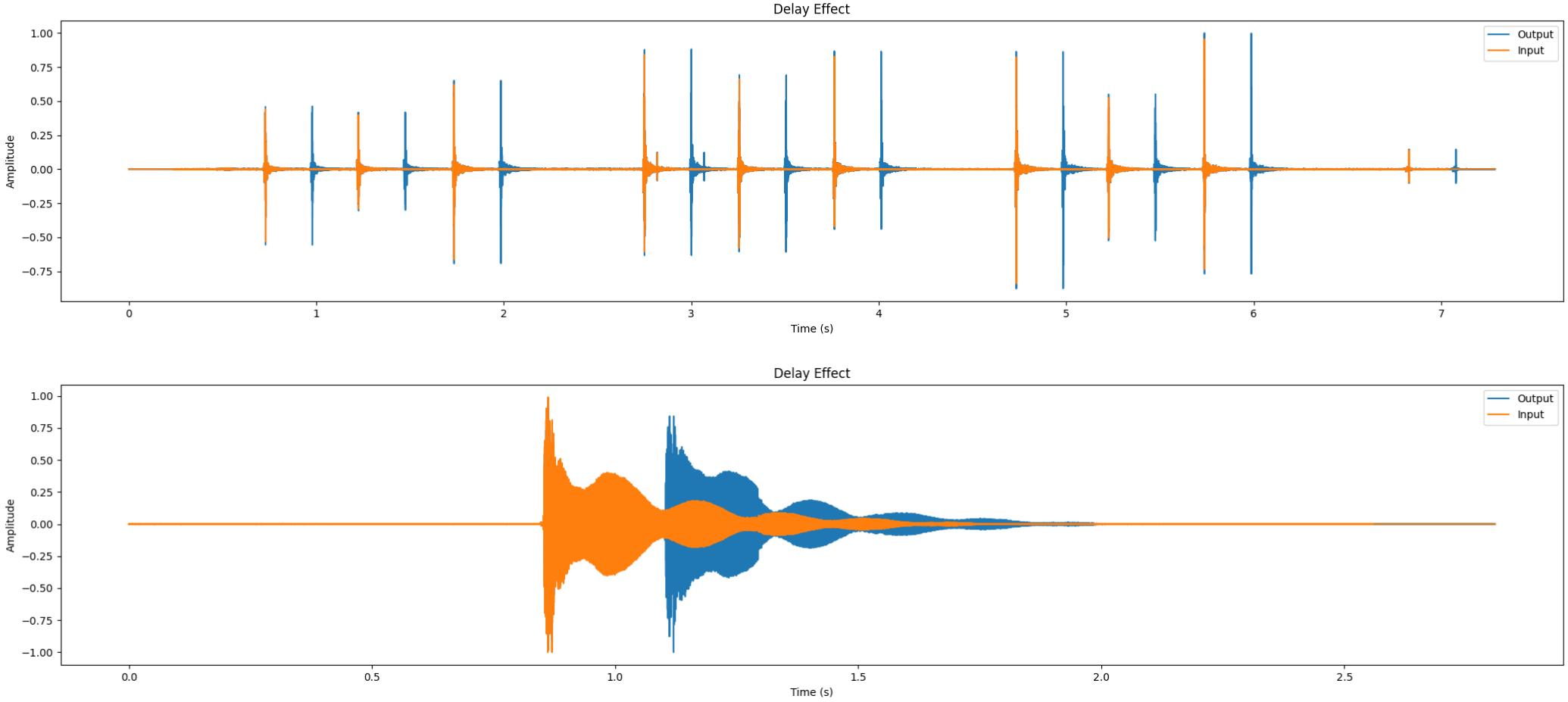
Chorus

Delay

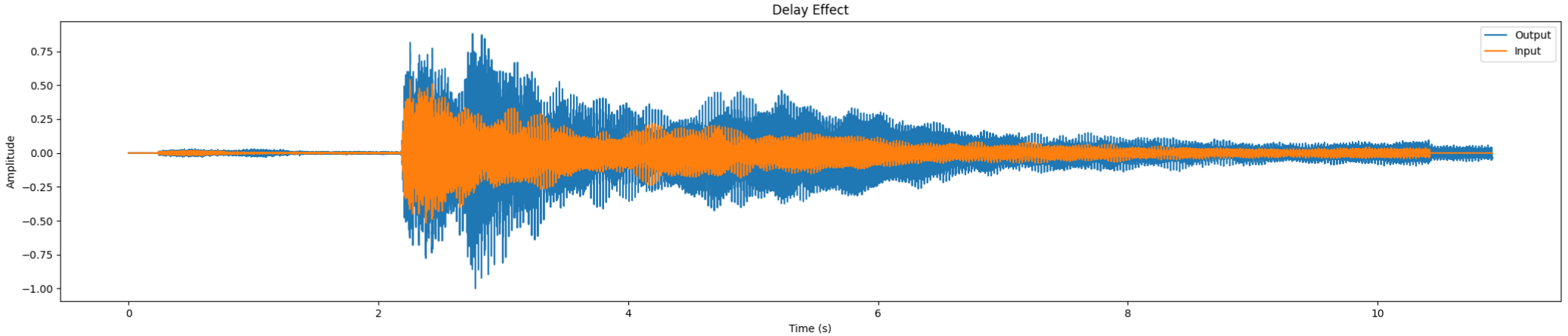
- Reprodukcija odgođene verzije izvornog audio signala
- Dojam ponavljanja zvuka ili jeke

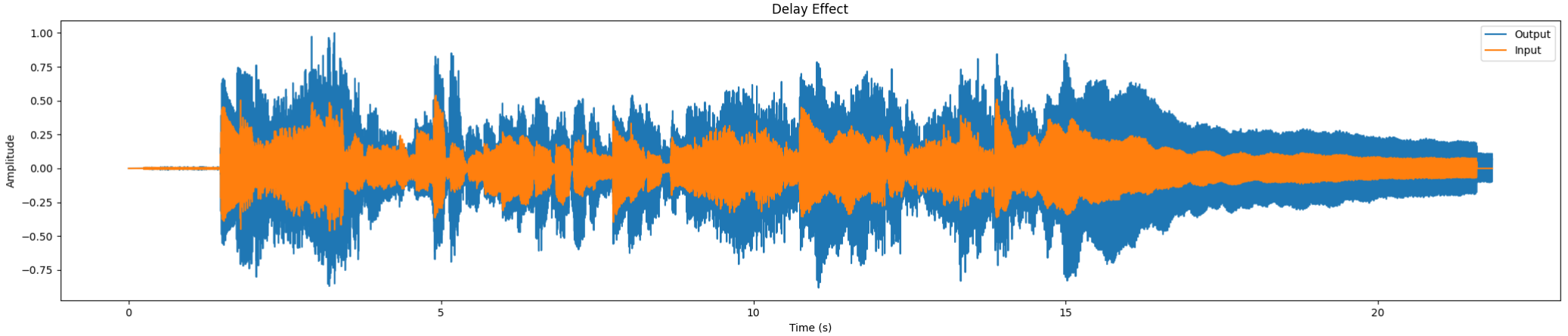
- Temelj za stvaranje drugih efekata s konceptom kašnjenja
  - repeticijom kašnjenja:
    - *reverb* – efekt koji daje dojam prostora pomoću jeke
  - modulacijom vremena kašnjenja:
    - *flanger* – efekt poput zvuka kroz PVC cijevi
    - *chorus* – efekt koji simulira zvuk od više glasova ili instrumenata s istom ulogom

Delay


Delay Effect
Output
Input
Amplitude
Time (s)
Delay Effect
Output
Input
Amplitude
Time (s)

Delay


Delay Effect
Output
Input
Amplitude
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
Time (s)


Delay Effect
Output
Input
Amplitude
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
5
10
15
20
Time (s)

Delay

```
def delay(input_signal, fs, delay_time):
    # Izračun broja zakašnjelih uzoraka preko vremena kašnjenja
    delay_samples = int(fs * delay_time)

    # Stvaranje filtra kašnjenja pomakom ulaznog signala za broj zakašnjelih uzoraka
    delay_filter = np.pad(input_signal, (delay_samples, 0), 'constant')
    # Proširenje ulaznog signala do duljine filtra kašnjenja
    input_signal = np.pad(input_signal, (0, delay_samples), 'constant')

    output_signal = input_signal + delay_filter

    # Normalizacija izlaznog signala kako bi se spriječilo odrezivanje
    output_signal /= np.max(np.abs(output_signal))

    return output_signal
```

Reverb

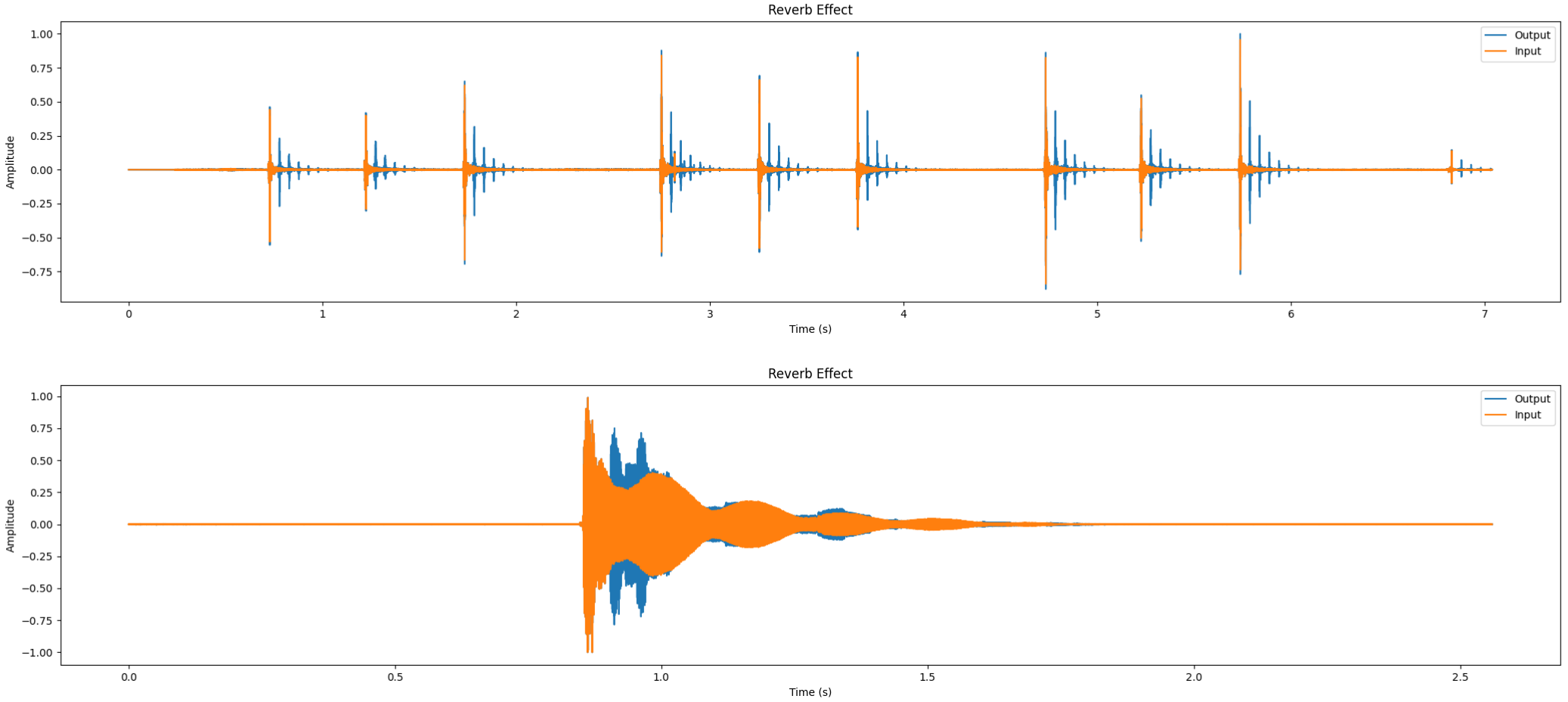
- Simulacija akustičnih karakteristika prostora:
  - odbijanje zvučnih valova od površina i slabljenje tijekom vremena
  - površinska apsorpcija i raspršenje

- Konvolucija audio signala s impulsnim odzivom
  - impulsni odziv – refleksija praska zvuka u prostoru i njegovo slabljenje
  - dulji impulsni odziv = izraženiji dojam prostora
  - različite karakteristike – *room reverb*, *hall reverb*

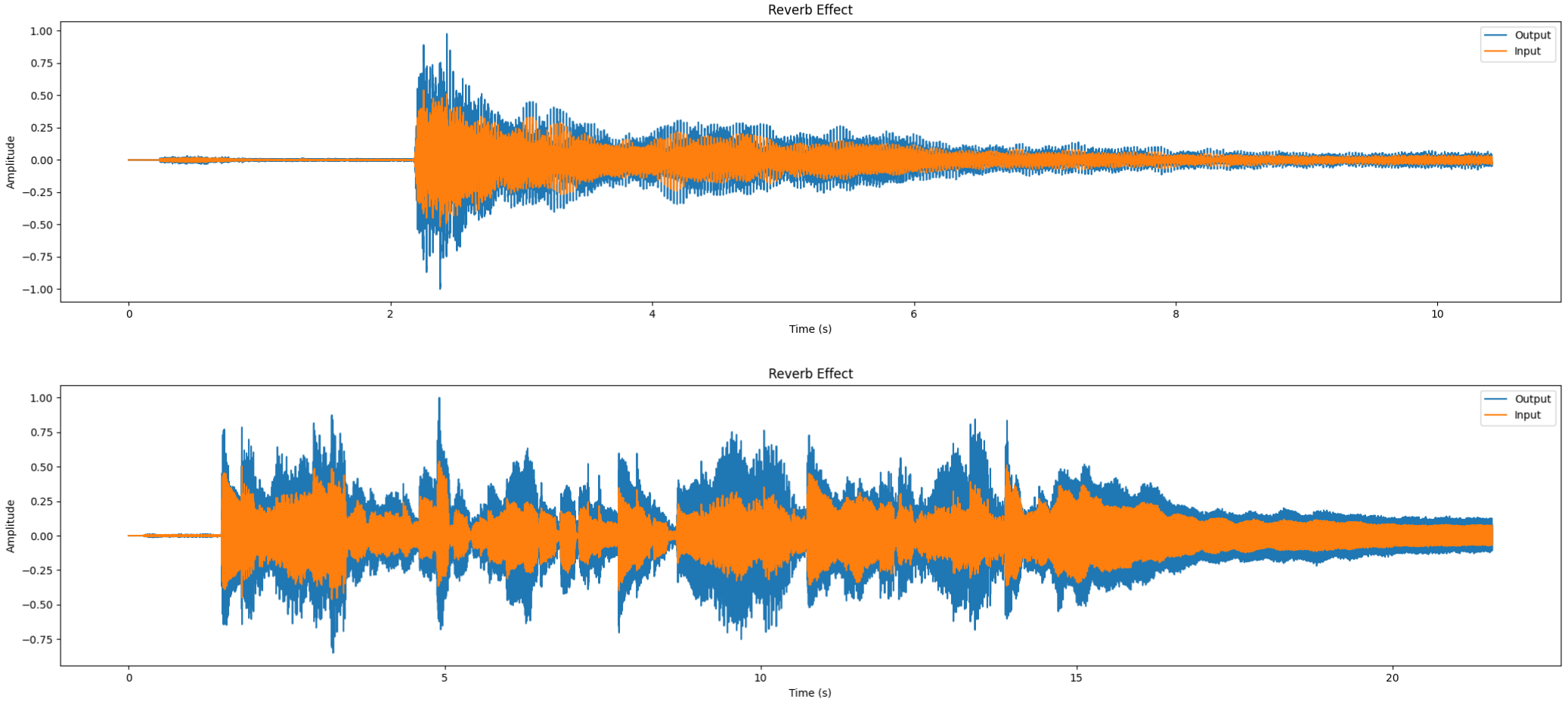
Reverb

- Parametri:
  - *decay* (slabljenje) – vrijeme potrebno zvuku da prestane odjekivati
    - kontrolira se podešavanjem brzine kojom je zvučna energija apsorbirana
    - eksponencijalna funkcija slabljenja – vrijeme da se zvuk smanji za 60dB
    - $slabljenje = e^{-\frac{1}{f_{uzorkovanja} \cdot t_{slabljenje}}}$
  - veličina (virtualnog) prostora
  - *pre-delay* (pred-kašnjenje)
  - *diffusion* (difuzija)
  - *damping* (prigušivanje)
  - *wetness/dryness* (udio učinka efekta na signal)

Reverb


Reverb Effect
Output
Input
Amplitude
Time (s)
Reverb Effect
Output
Input
Amplitude
Time (s)

Reverb


Reverb Effect
Output
Input
Amplitude
Time (s)
Reverb Effect
Output
Input
Amplitude
Time (s)

Reverb

```
def reverb(input_signal, fs, decay_time, delay_time):
    # Stvaranje izlaznog signala kao niza nuli
    output_signal = np.zeros_like(input_signal)

    # Izračun faktora slabljenja preko vremena slabljenja
    decay_factor = np.exp(-1 / (fs * decay_time))
    # Izračun broja zakašnjelih uzoraka preko vremena kašnjenja
    delay_samples = int(fs * delay_time)

    # Primjena reverb efekta uz dodatan feedback faktor
    feedback_gain = 0.5
```

Reverb

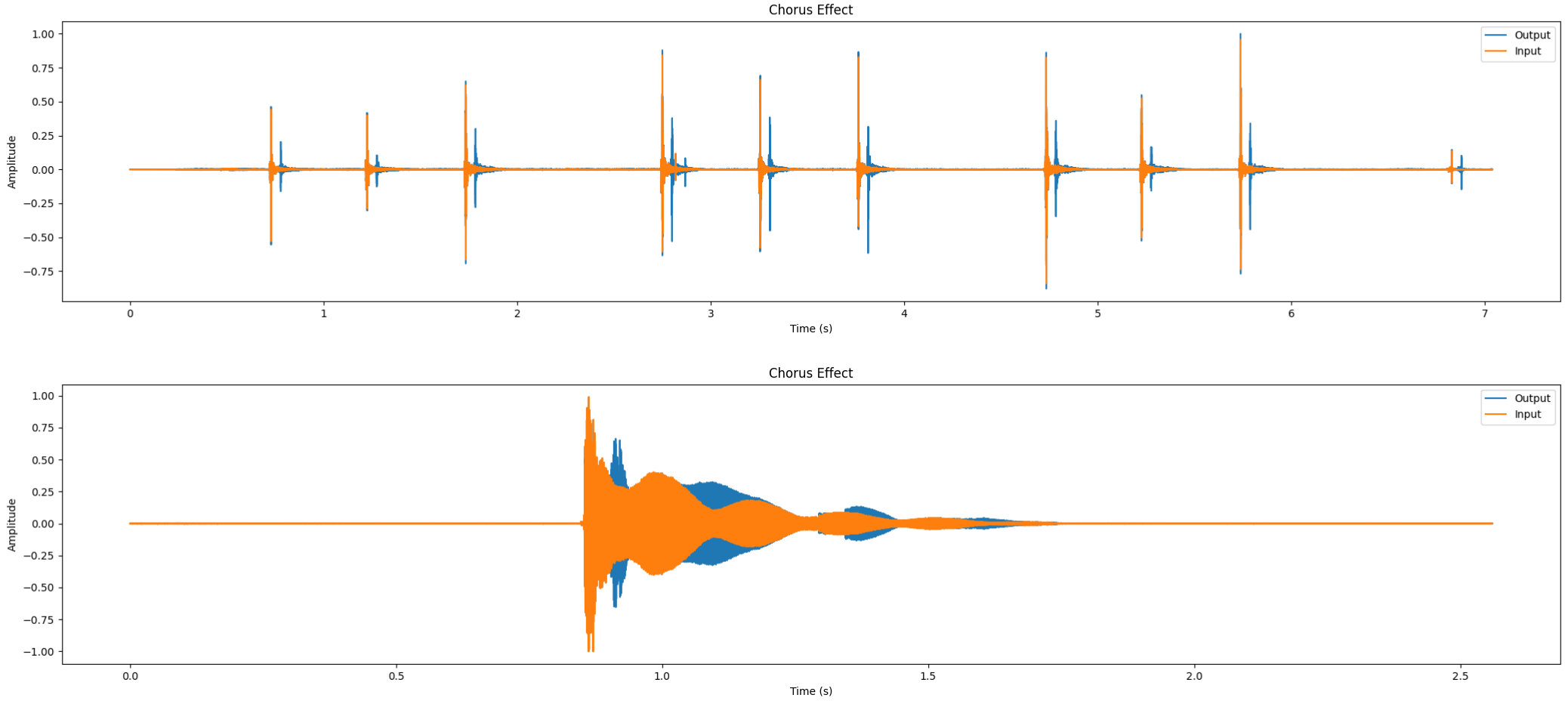
```
for i in range(len(input_signal)):
        # Suma ulaznog sa zakašnjelim i oslabljenim signalom za trenutni uzorak
        if i >= delay_samples:
                output_signal[i] = input_signal[i] + feedback_gain * decay_factor *
                    output_signal[i - delay_samples]
        else:
                output_signal[i] = input_signal[i]

# Normalizacija izlaznog signala kako bi se spriječilo odrezivanje
output_signal /= np.max(np.abs(output_signal))

return output_signal
```

Chorus

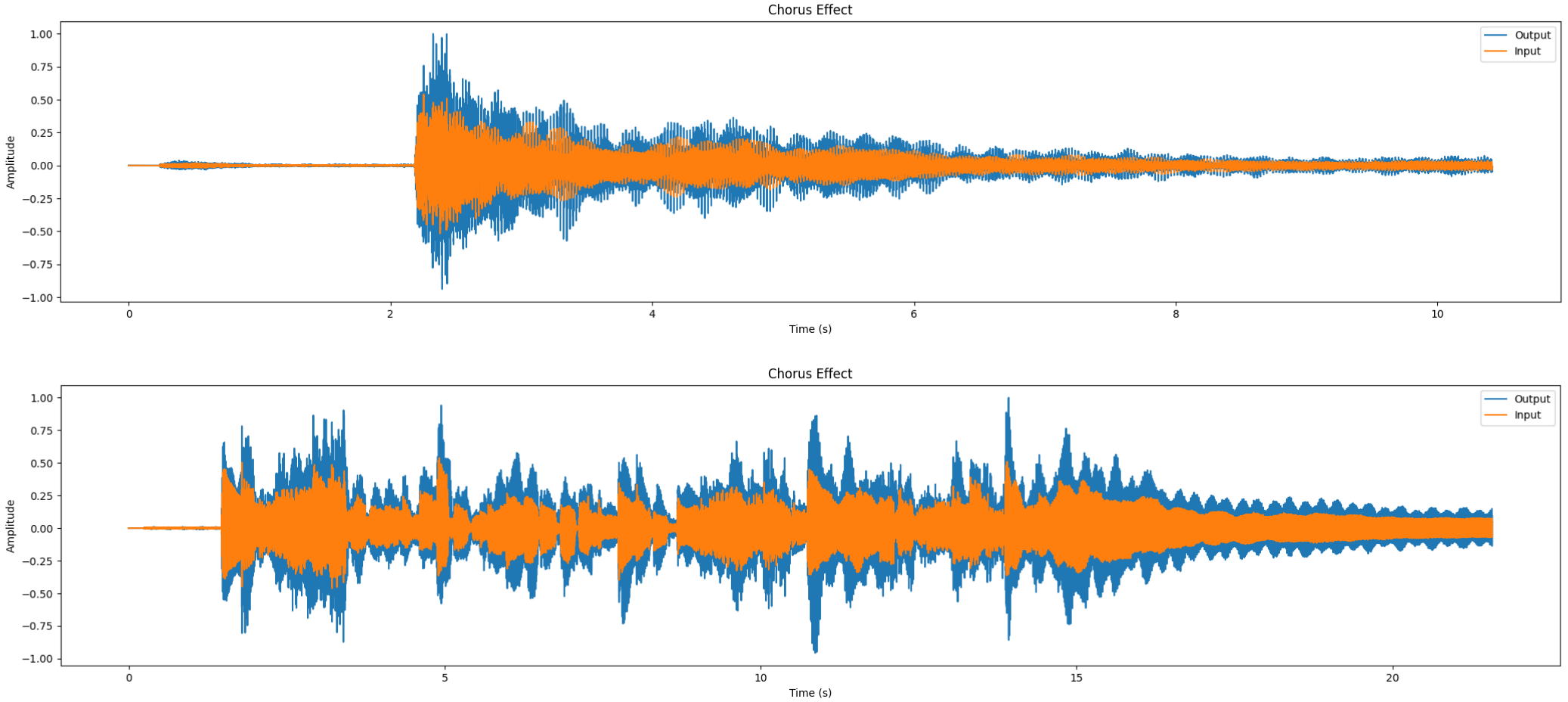
- Simulira zvuk više glasova ili instrumenata koji sviraju istu ulogu
- Efekt kašnjenja s moduliranim vremenom kašnjenja
  - LFO (Low-frequency Oscilator) – stvara raštimirani efekt koji ostavlja dojam da više izvora svira zajedno
  - kontinuirana izmjena između izvornog i modificiranog signala

- Učinak raštimiranja – različite razine dubine modulacije
- Brzina LFO-a - različite brzine modulacije

Chorus


Chorus Effect
Output
Input
Amplitude
Time (s)
Chorus Effect
Output
Input
Amplitude
Time (s)

Chorus


Chorus Effect
Output
Input
Amplitude
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
Time (s)
Chorus Effect
Output
Input
Amplitude
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
Time (s)

Chorus

```
def chorus(input_signal, fs, depth, rate, delay_time):
        # Stvaranje izlaznog signala kao niza nuli
        output_signal = np.zeros_like(input_signal)

        # Izračun broja zakašjelih uzoraka preko vremena kašnjenja
        delay_samples = int(fs * delay_time)
        # Priprema LFO-a (modulacija signala kroz vrijeme)
        lfo_period = int(fs / rate)
        lfo = depth * np.sin(2 * np.pi * np.arange(len(input_signal)) / lfo_period)
```

Chorus

```
# Primjena chorus efekta korištenjem LFO-a i kašnjenja
for i in range(len(input_signal)):
        # Suma ulaznog sa zakašnjelim i moduliranim signalom za trenutni uzorak
        if i >= delay_samples:
                output_signal[i] = input_signal[i] + lfo[i] * input_signal[i -
                    delay_samples] + (1 - depth) * output_signal[i - delay_samples]
        else:
                output_signal[i] = input_signal[i]

# Normalizacija izlaznog signala kako bi se spriječilo odrezivanje
output_signal /= np.max(np.abs(output_signal))

return output_signal
```