
WELCOME

THI GIỮA KỲ
Môn học: Xử lý tín hiệu số

Giáo viên hướng dẫn: Ninh Khánh Duy

Họ và tên: Đỗ Nguyên Ánh

MSSV: 102200082

ĐỀ BÀI: Phân đoạn voiced vs unvoiced (1b)

1.Mô tả các hàm và việc tính toán được cài đặt:

a. signal_filter

Hàm này có công dụng giúp em lọc ra các mẫu dữ liệu trong các file .wav dựa theo filter là các nhãn trong file .lab

```
def signal_filter(file_name_lab, file_name_wav, filter):
    result = []
    file = open(file_name_lab, "r")
    list_temp = file.readlines()
    list_interval_by_filter = []
    for i in range(len(list_temp) - 2): # bỏ hai dòng cuối của file .lab (F0mean
        interval_by_filter = []
        if str(list_temp[i].split('\t')[2].strip()) == filter:

            start_time = float(list_temp[i].split('\t')[0])
            interval_by_filter.append(start_time)
            end_time = float(list_temp[i].split('\t')[1])
            interval_by_filter.append(end_time)

            list_interval_by_filter.append(interval_by_filter)
    # print(list_interval_by_filter)
    # [[1.88, 1.95], [2.16, 2.25], [2.6, 2.75], [3.34, 3.38], [3.45, 3.62], [3.8, 3
    Fs, x = wavfile.read(file_name_wav)
    t = len(x)/Fs
    # chuẩn hóa tín hiệu
    x = x/max(abs(x))
    # lấy mẫu tín hiệu theo mốc thời gian
    for i in range(len(list_interval_by_filter)):
        index_starts_sampling = floor((list_interval_by_filter[i][0]/t)*len(x))
        index_finished_sampling = floor((list_interval_by_filter[i][1]/t)*len(x))
        for j in range(index_starts_sampling,index_finished_sampling):
            result.append(x[j])
    return result
```

✓ 0.4s

b. binary_search

Tham khảo thuật toán trong:
+ CS425 Audio and Speech Processing
+ Phương pháp chia đôi

Giúp tìm được ngưỡng phân biệt voice và unvoice của từng file tín hiệu huấn luyện

```
def binary_search(f, g):
    T = 0.0
    T_min = max(min(f), min(g))
    T_max = min(max(f), max(g))
    T = 0.5*(T_min + T_max)
    elements_less_than_T_in_f = []
    elements_greater_than_T_in_g = []
    for i in range(len(f)):
        if f[i] < T:
            elements_less_than_T_in_f.append(f[i])
    for i in range(len(g)):
        if g[i] > T:
            elements_greater_than_T_in_g.append(g[i])
    nf = len(elements_less_than_T_in_f) # số phần tử nhỏ hơn T mà thuộc f
    ng = len(elements_greater_than_T_in_g) # số phần tử lớn hơn T mà thuộc g
    nf_old = -1; # số lượng phần tử nf cũ
    ng_old = -1; # số lượng phần tử ng cũ
    N_f = len(f)
    N_g = len(g)
    while not(nf ==nf_old and ng==ng_old):
        s1 = 0.0
        c1 = 0.0
        for i in range(len(f)):
            if f[i] > T:
                c1 = c1 + 1
                s1 = s1 + f[i]
        s1 = s1 - c1*T
        s2 = 0.0
        c2 = 0.0
        for i in range(len(g)):
            if g[i] < T:
```

c. Tìm ngưỡng chung

Tìm ngưỡng(threshold) của từng file tín hiệu

```
threshold_phone_F2 = binary_search(signal_filter("phone_F2.lab","phone_F2.wav","uv"),signal_filter("phone_F2.lab","phone_F2.wav","v"))
threshold_phone_M2 = binary_search(signal_filter("phone_M2.lab","phone_M2.wav","uv"),signal_filter("phone_M2.lab","phone_M2.wav","v"))
threshold_studio_F2 = binary_search(signal_filter("studio_F2.lab","studio_F2.wav","uv"),signal_filter("studio_F2.lab","studio_F2.wav","v"))
threshold_studio_M2 = binary_search(signal_filter("studio_M2.lab","studio_M2.wav","uv"),signal_filter("studio_M2.lab","studio_M2.wav","v"))
```

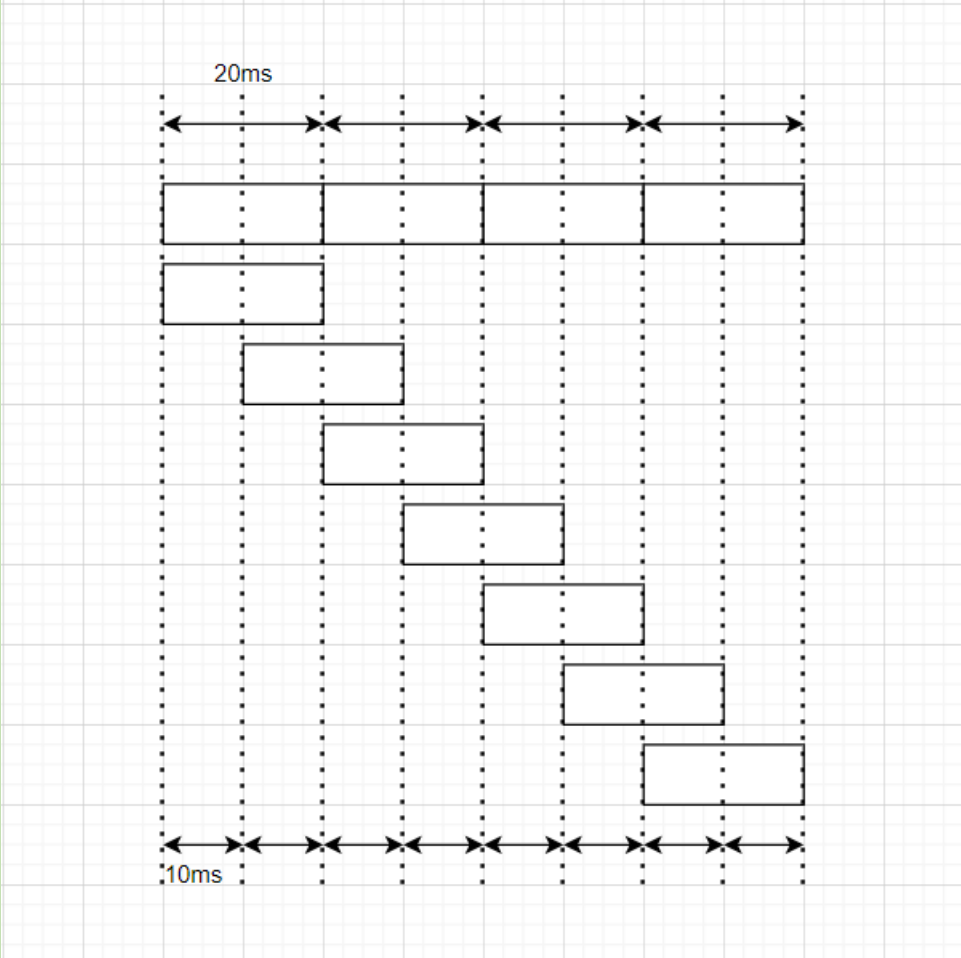
```
C:\Users\dongu\AppData\Local\Temp\ipykernel_20308\155001797.py:18: WavFileWarning: Chunk (non-data) not understood, skipping it.
  Fs, x = wavfile.read(file_name_wav)
```

Ngưỡng chung của tập tín hiệu huấn luyện

```
threshold = (threshold_phone_F2 + threshold_phone_M2 + threshold_studio_F2 + threshold_studio_M2)/4
arrays_threshold = [threshold_phone_F2, threshold_phone_M2, threshold_studio_F2, threshold_studio_M2]
e = np.std(arrays_threshold)
threshold = threshold - e
print('threshold = ' + str(threshold))
```

```
threshold = -0.0499024473481833
```

d. Chia frame



Cài đặt hàm tạo Frame

```
def div_frame(Fs, x):
    list_result = []
    # lấy độ dài thời gian của mỗi khung tín hiệu là 0.02s
    length_frame = 0.02*Fs # số mẫu tín hiệu trong một frame
    overlap_frame_length = floor(length_frame/2) # overlap khung 0.01s
    number_frame = floor(len(x)/overlap_frame_length -1) # số khung frame
    for index_frame in range(1,number_frame+1):
        start_end = []
        start_index_frame = floor((index_frame-1)*overlap_frame_length+1)
        start_end.append(start_index_frame)
        end_index_frame = floor(length_frame + (index_frame-1)*overlap_frame_length)
        start_end.append(end_index_frame)
        list_result.append(start_end)
    return list_result
```

✓ 0.6s

e. STE và ZCR

STE: tổng bình phương của các giá trị dạng sóng trên một số lượng mẫu hữu hạn thuộc một khung

ZCR: số lần mà tín hiệu nằm trong khung băng qua trục 0 (xử lý theo khung)

Cài đặt STE (Short-Time Energy)

```
def STE(Fs,x):
    ste = np.zeros(len(div_frame(Fs, x)))
    for index_frame in range(len(div_frame(Fs, x))):
        ste[index_frame] =np.sum((x[div_frame(Fs, x)[index_frame][0]:(div_frame(Fs, x)[index_frame][1])])**2)
    return ste
```

✓ 0.8s

Cài đặt ZCR (Zero-Crossing Rate)

```
def sgn(x):
    if x >= 0:
        return 1
    else:
        return -1
# Số lần mỗi tín hiệu nằm trong 1 khung băng qua 0
def ZCR(Fs, x):
    zcr = np.zeros(len(div_frame(Fs, x)))
    for index_frame in range(len(div_frame(Fs, x))):
        count = 0
        frame = x[div_frame(Fs, x)[index_frame][0]:div_frame(Fs, x)[index_frame][1]]
        for i in range(div_frame(Fs, x)[index_frame][0],div_frame(Fs, x)[index_frame][1],1):
            if np.abs(sgn(x[i])-sgn(x[i-1])) == 0:
                count = count + 0
            elif np.abs(sgn(x[i])-sgn(x[i-1])) >= 0:
                count = count + 1
            zcr[index_frame] = count
    return zcr
```

✓ 0.1s

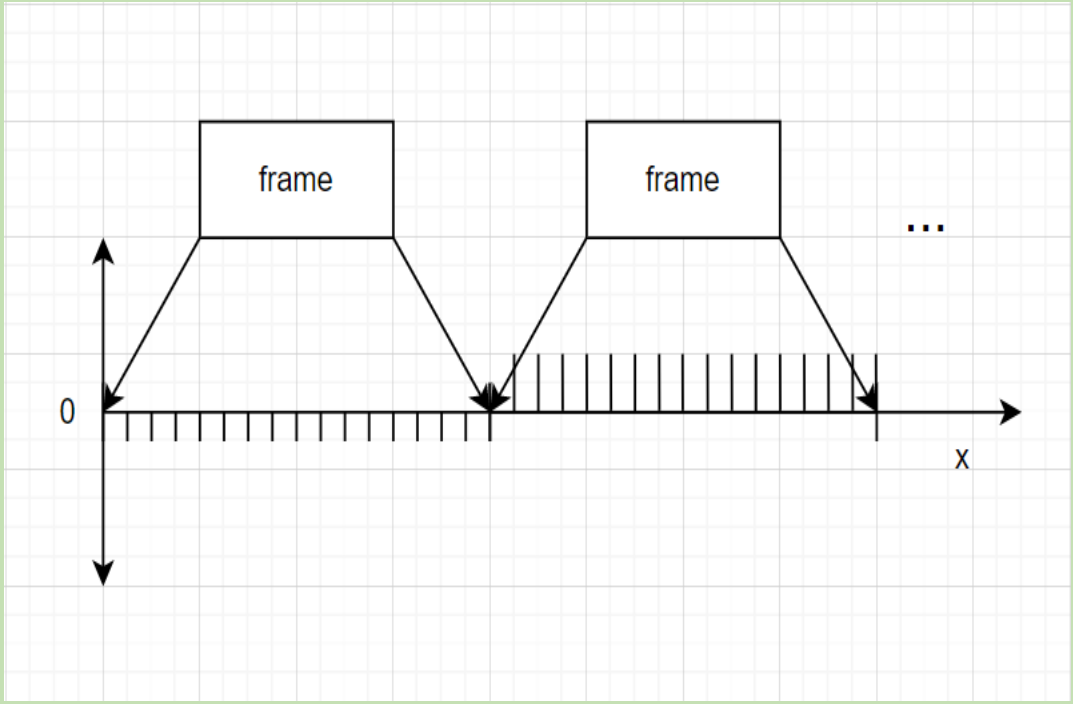
f. Chuyển về đồ thị dạng sóng (transfer)

Hàm chuyển về đồ thị dạng sóng (liên tục)

```
# Hàm transfer có tác dụng chuyển các ste_norm hoặc zcr norm về giá trị của từng điểm
def transfer(Fs, x, feature):
    length_list_value_sub_frame_by_feature = len(feature) + 1
    list_value_sub_frame_by_feature = np.zeros(length_list_value_sub_frame_by_feature) # do các frame gối đầu lên nhau 1/2
    list_value_sub_frame_by_feature[0] = feature[0] # điểm đầu
    list_value_sub_frame_by_feature[length_list_value_sub_frame_by_feature-1] = feature[length_list_value_sub_frame_by_feature-2]
    for i in range(1,length_list_value_sub_frame_by_feature-1):
        list_value_sub_frame_by_feature[i] = (feature[i] + feature[i-1])/2  # feature[i]
    feature_wave = np.zeros(len(x))
    length_sub_frame = floor(len(x)/len(list_value_sub_frame_by_feature))
    start = 0
    end = start + length_sub_frame
    while end <= len(x):
        for i in range(len(list_value_sub_frame_by_feature)):
            feature_wave[start:end] = np.full((length_sub_frame,),list_value_sub_frame_by_feature[i])
            start = end
            end = end + length_sub_frame
    return feature_wave
```

✓ 0.4s

Giúp chuyển về dạng từng mẫu để dễ dàng vẽ hơn



g. Cài đặt ra quyết định

-Duyệt trong merger = ste –zcr

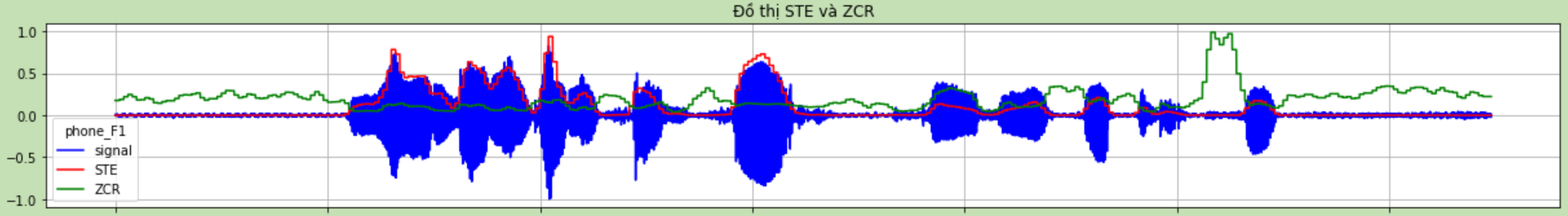
+Nếu ste tại điểm đó < 0.005 coi như silent

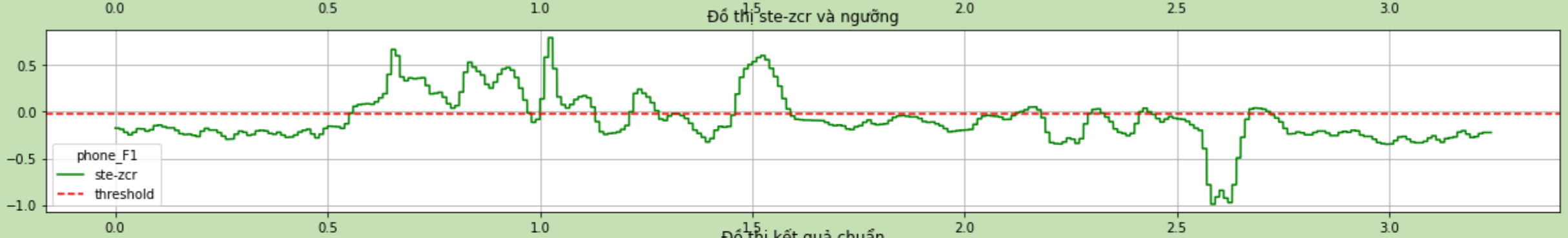
+Nếu merger tại điểm đó < threshold thì đánh dấu là unvoiced. Ngược lại thì đánh dấu là voiced

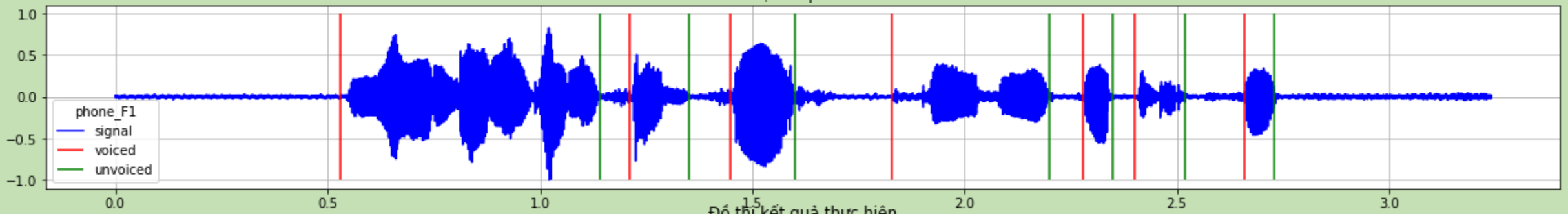
```
# Ra quyết định voiced và unvoiced
check = []
transfer_ste_norm = transfer(Fs, x, ste_norm)
merger = np.array(transfer(Fs, x, ste_norm)) - np.array(transfer(Fs, x, zcr_norm))
for i in range(len(merger)):
    if transfer_ste_norm[i] < 0.005:
        check.append(0) # uv(sil)
    elif merger[i] < threshold:
        check.append(0) # unvoiced
    else:
        check.append(1) # voiced

flag_voiced = []
flag_unvoiced = []
for i in range(1,len(check)):
    if check[i] == check[i-1]:
        continue
    if check[i] != check[i-1]:
        if check[i] == 1:
            flag_voiced.append(i)
        else:
            flag_unvoiced.append(i)
# lưu theo thời gian để vẽ
time_flag_voiced = np.array(flag_voiced)/Fs
time_flag_unvoiced = np.array(flag_unvoiced)/Fs
```

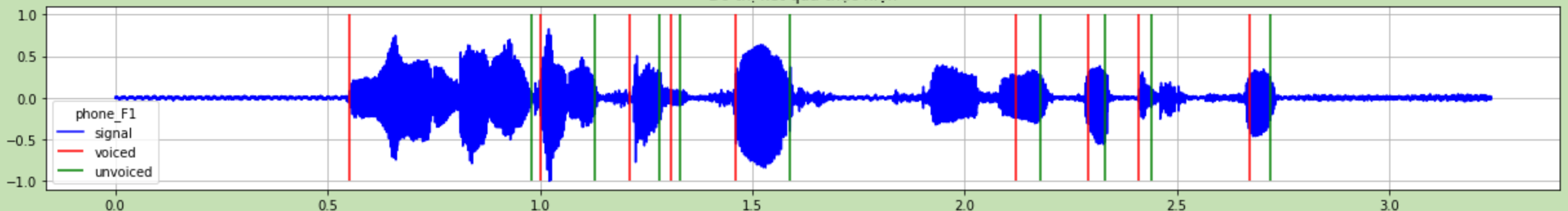
2. Kết quả trực quan:

phone_F1

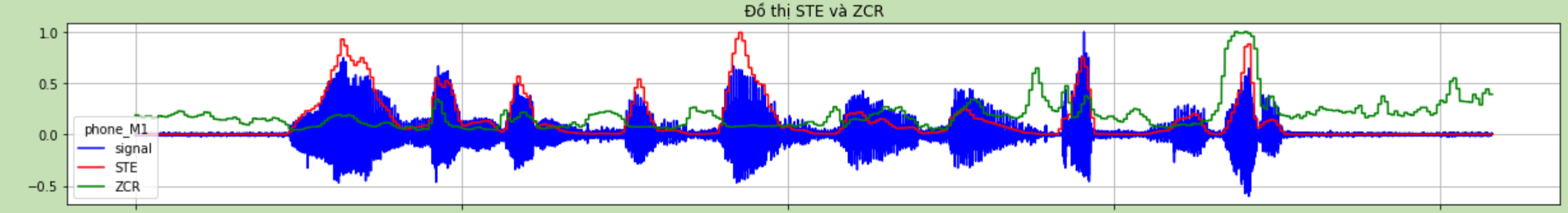




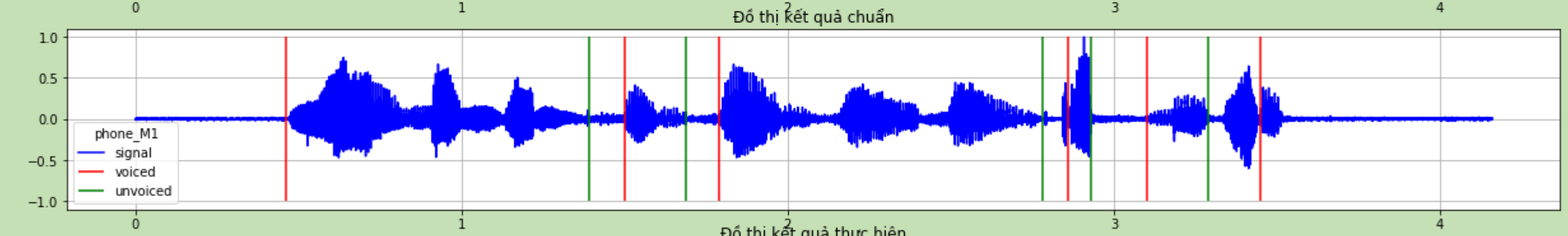


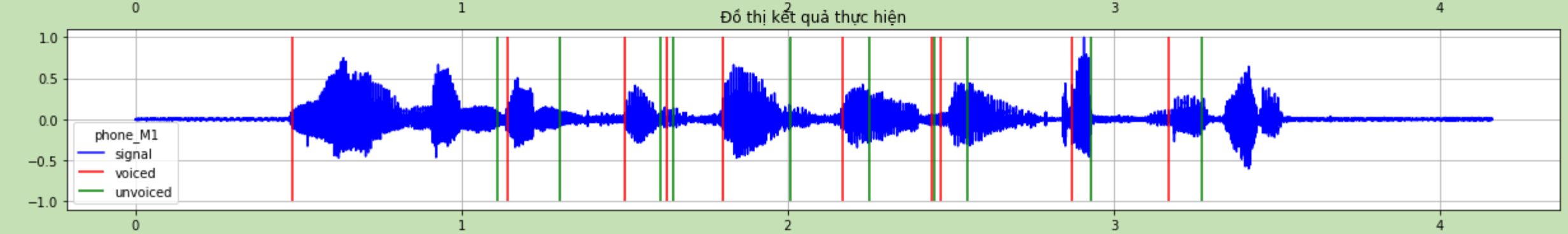


phone_M1

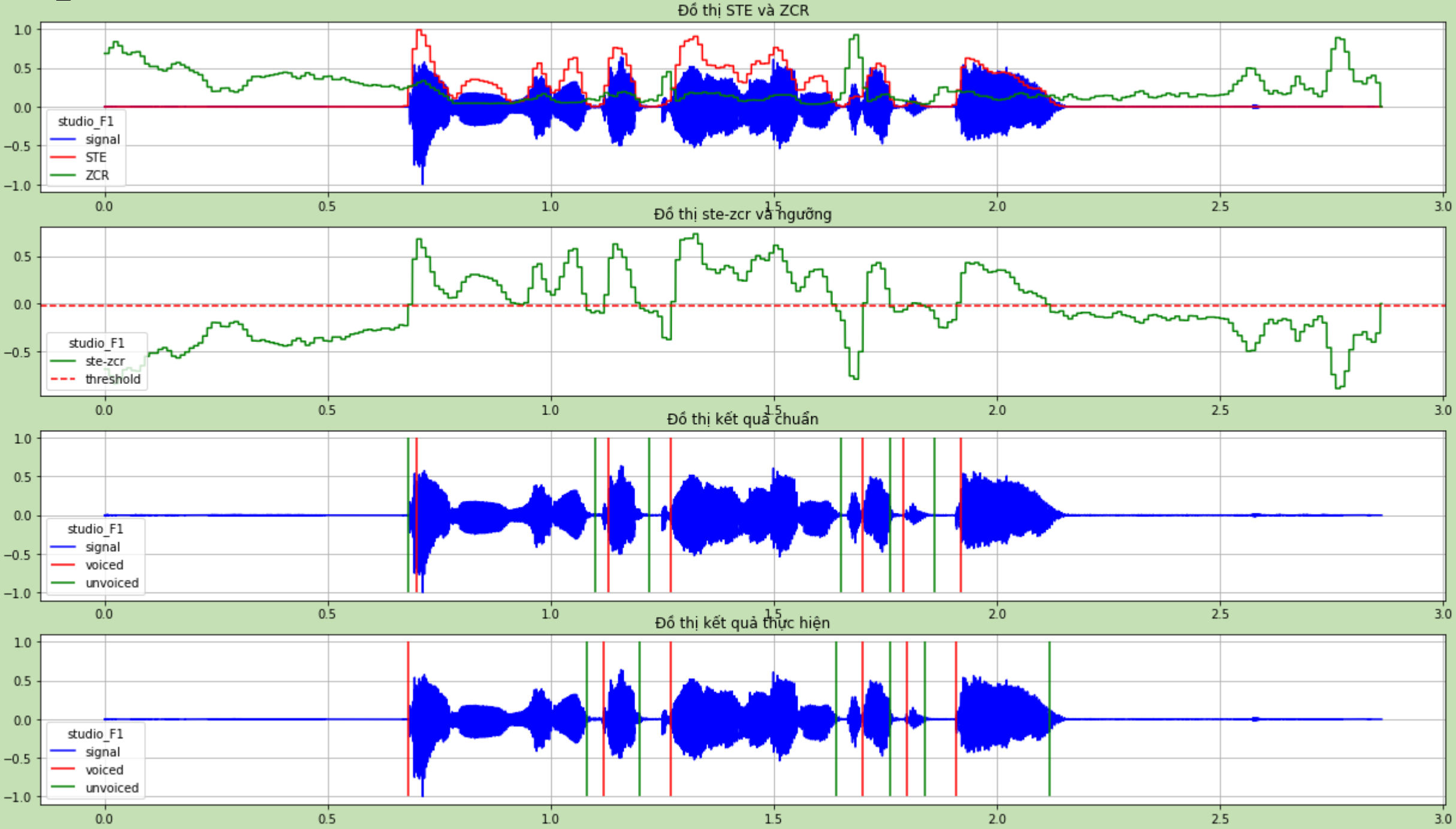

Đồ thị STE và ZCR
phone_M1
signal
STE
ZCR


Đồ thị ste-zcr và ngưỡng
phone_M1
ste-zcr
threshold

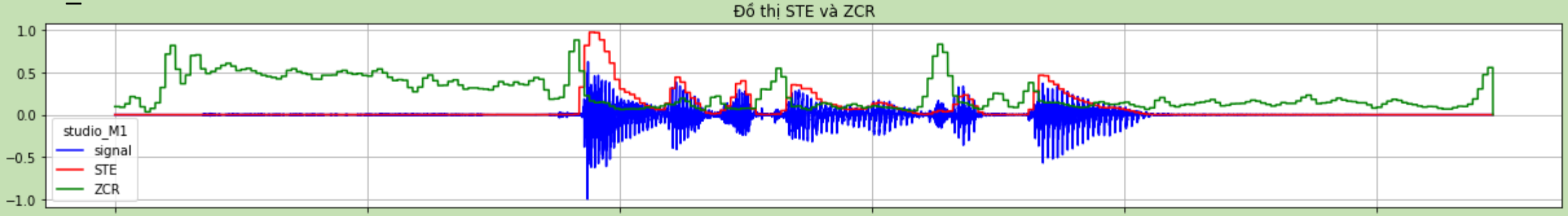

Đồ thị kết quả chuẩn
phone_M1
signal
voiced
unvoiced


Đồ thị kết quả thực hiện
phone_M1
signal
voiced
unvoiced

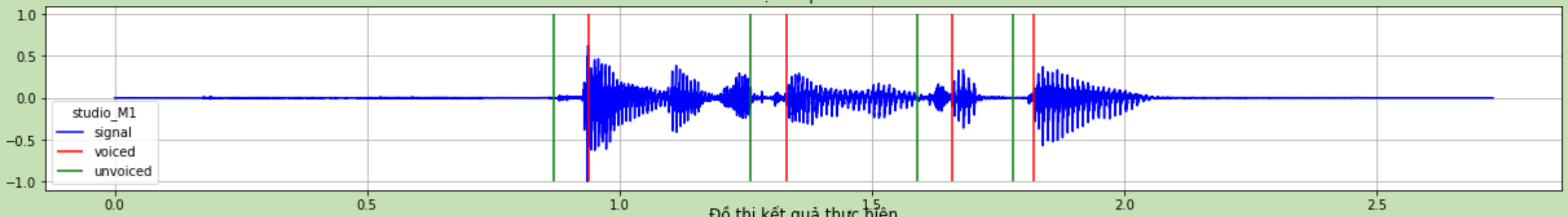
studio_F1

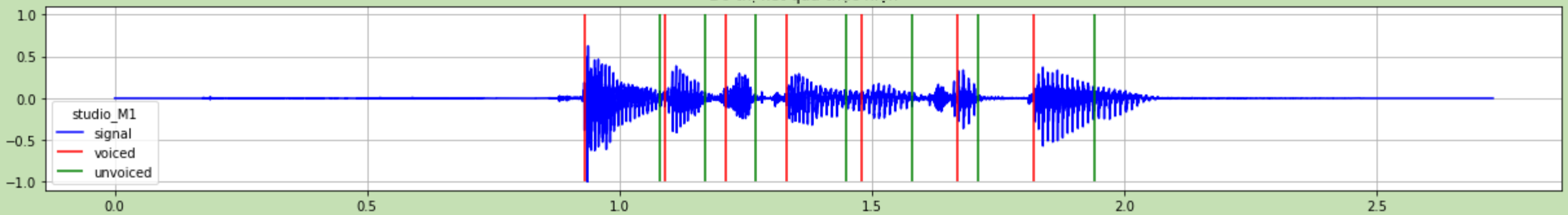

Đồ thị STE và ZCR
studio_F1
signal
STE
ZCR
Đồ thị ste-zcr và ngưỡng
studio_F1
ste-zcr
threshold
Đồ thị kết quả chuẩn
studio_F1
signal
voiced
unvoiced
Đồ thị kết quả thực hiện
studio_F1
signal
voiced
unvoiced

studio_M1


Đồ thị STE và ZCR
studio_M1
signal
STE
ZCR


Đồ thị ste-zcr và ngưỡng
studio_M1
ste-zcr
threshold


Đồ thị kết quả chuẩn
studio_M1
signal
voiced
unvoiced


Đồ thị kết quả thực hiện
studio_M1
signal
voiced
unvoiced


THANK YOU