Visualizações práticas do algoritmo SPIKE-Distance, para avaliar dissimilaridades entre Spike Trains

In [ ]:

```
import numpy as np
import random
import matplotlib.pyplot as plt
```

Gerador de Spikes customizado para vários casos de estudo.

In [ ]:

```
#########################################################################################
## Spike Count Generator for Desired spikes
def spike_count_generator(n=2, T=10, reduction_ratio=1, make_zero_st=False, make_ful
    spike_counts_per_st = np.zeros(n)

    if(make_complemetary_st == True):
        rand_count = np.random.randint(2, int(T/reduction_ratio))
        spike_counts_per_st[0] = rand_count
        spike_counts_per_st[1] = T - rand_count

        return spike_counts_per_st

    for i in range(n):
        if (make_zero_st == True):
            spike_counts_per_st[i] = 0
            make_zero_st = False
            continue
        if (make_full_st == True):
            spike_counts_per_st[i] = int(T/reduction_ratio)
            make_full_st = False
            continue
        spike_counts_per_st[i] = np.random.randint(2, int(T/reduction_ratio))

    return spike_counts_per_st

#########################################################################################
## Spike Times Generator for Desired spikes
def spike_times_generator(spike_count=None, n=2, T=10, duplicate=False, complementar
    desired_spike_train = []
    desired_spike_bin = np.zeros((n, T))
    spike_range = (0, T)
    if (duplicate == True):
        count = 0
        i = 0
        spike_time = random.sample(range(spike_range[0], spike_range[1]), int(spike_
        while (count < n):
            desired_spike_bin[count][spike_time] = 1
            desired_spike_train.append(np.sort(spike_time))
            count += 1
    else:
        if (complementary_spikes == True):
            rand_times = random.sample(range(spike_range[0], spike_range[1]), int(sp
            all_times = range(spike_range[0], spike_range[1])
            compl_times = np.array(list(set(all_times).symmetric_difference(set(rand
```

```
            desired_spike_bin[0][rand_times] = 1
            desired_spike_bin[1][compl_times] = 1
            desired_spike_train.append(np.sort(rand_times))
            desired_spike_train.append(np.sort(compl_times))

        elif (time_shift == True):
            spike_time = random.sample(range(spike_range[0], spike_range[1]), int(sp
            shifted_time = np.add(spike_time, shift_step)
            out_of_bounds = np.argwhere(shifted_time >= T)
            if (out_of_bounds.shape[0] > 0):
                shifted_time[out_of_bounds] = T-1

            desired_spike_bin[0][spike_time] = 1
            desired_spike_bin[1][shifted_time] = 1
            desired_spike_train.append(np.sort(spike_time))
            desired_spike_train.append(np.sort(shifted_time))
        else:
            for i in range(n):
                spike_time = random.sample(range(spike_range[0], spike_range[1]), in
                desired_spike_bin[i][spike_time] = 1
                desired_spike_train.append(np.sort(spike_time))


    return desired_spike_bin, desired_spike_train
```

Cálculo da Dissimilaridade e suas funções auxiliares:

In [ ]:

```
# Tempos do spike antecessor ao instante t
def tP(t, neuron_spike_times):
    res = neuron_spike_times[np.where(neuron_spike_times <= t)]
    if(res.shape[0] == 0):
        return 0
    else:
        return max(res)

# Tempos do spike sucessor ao instante t
def tF(t, neuron_spike_times):
    res = neuron_spike_times[np.where(neuron_spike_times > t)]
    if(res.shape[0] == 0):
        return 0
    else:
        return min(res)

# Intervalo entre spikes instantaneo
def x_isi(t, neuron_spike_times):
    return tF(t, neuron_spike_times)-tP(t, neuron_spike_times)

def media_x_isi(t, spike_times):

    return 0.5*(x_isi(t, spike_times[0])+x_isi(t, spike_times[1]))

# Diferencas absolutas entre sucessores e antecessores
def delta_tP(t, spike_times):
    return abs(tP(t, spike_times[0])-tP(t, spike_times[1]))

def delta_tF(t, spike_times):
    return abs(tF(t, spike_times[0])-tF(t, spike_times[1]))
```

```
# Intervalos para os spikes antecessores e sucessores
def xP(t, neuron_spike_times):
    return t - tP(t, neuron_spike_times)

def xF(t, neuron_spike_times):
    return tF(t, neuron_spike_times) - t

def media_xp(t, spike_times):
    return 0.5*(xP(t, spike_times[0]) + xP(t, spike_times[1]))

def media_xf(t, spike_times):
    return 0.5*(xF(t, spike_times[0]) + xF(t, spike_times[1]))

# Perfil de dissimiliaridade entre os spike trains - original
def So(t, spike_times):
    numerador = (delta_tP(t, spike_times)*media_xf(t, spike_times))+(delta_tF(t, spi
    denominador = media_x_isi(t, spike_times)**2
    return numerador/denominador

# Calcula a Dissimilaridade Normalizada [0, 1] dos perfis encontrados
def D(spike_times):
    count_zero = 0
    S_original = np.zeros(T)
    for t in range(T):
        S_original[t] = So(t, spike_times)
        if (S_original[t] == 0):
            count_zero += 1
    if (count_zero == T):
        S_original_norm = S_original
    else:
        S_original_norm = (S_original-min(S_original))/(max(S_original)-min(S_origin

    return S_original_norm

# Dissimilaridade Total Media: Como estamos em tempo discreto, é apenas um somatório
def D_total (d_profile):
    return np.mean(d_profile)

def print_spikes(spikes, diss, T):
    # plot
    fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(8, 10))

    ax1.set_ylabel("ST1 Spikes")
    ax1.step(range(T), spikes[0], linewidth=2.0)

    ax2.set_ylabel("ST2 Spikes")
    ax2.step(range(T), spikes[1], linewidth=2.0)

    ax3.set_ylabel("Dissimilaridade entre ST1 e ST2")
    ax3.plot(range(T), diss, linewidth=2.0)

    plt.show()
```

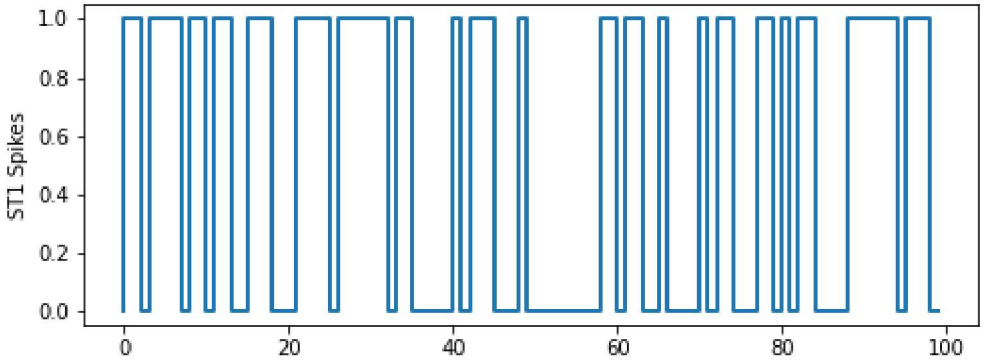
Caso 1: Dois spikes aleatorios

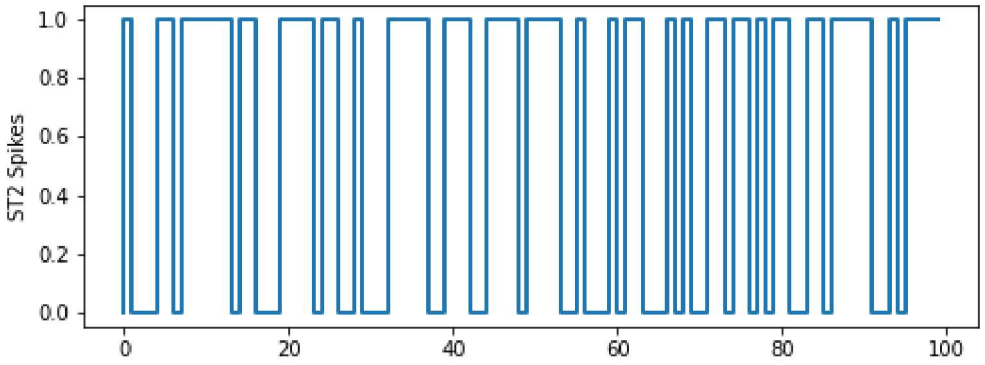
In [ ]:

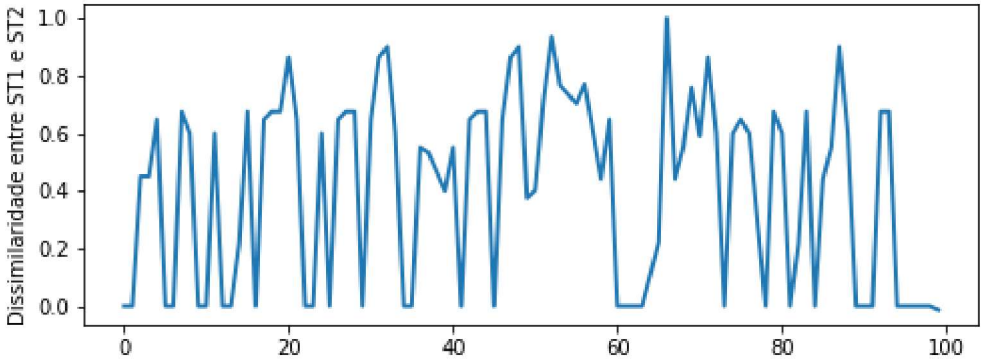
```
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
```

```
st_count = spike_count_generator(n, T)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T)

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T)))
```

```
[array([ 1,  2,  4,  5,  6,  7,  9, 10, 12, 13, 16, 17, 18, 22, 23, 24, 25,
       27, 28, 29, 30, 31, 32, 34, 35, 41, 43, 44, 45, 49, 59, 60, 62, 63,
       66, 71, 73, 74, 78, 79, 81, 83, 84, 89, 90, 91, 92, 93, 94, 96, 97,
       98]), array([ 1,  5,  6,  8,  9, 10, 11, 12, 13, 15, 16, 20, 21, 22, 23, 25,
26,
       29, 33, 34, 35, 36, 37, 40, 41, 42, 45, 46, 47, 48, 50, 51, 52, 53,
       56, 60, 62, 63, 67, 69, 72, 73, 75, 76, 78, 80, 81, 84, 85, 87, 88,
       89, 90, 91, 94, 96, 97, 98, 99])]
```








```
Dissimilaridade Total entre ST1 e ST2: 0.20361772506048825
```

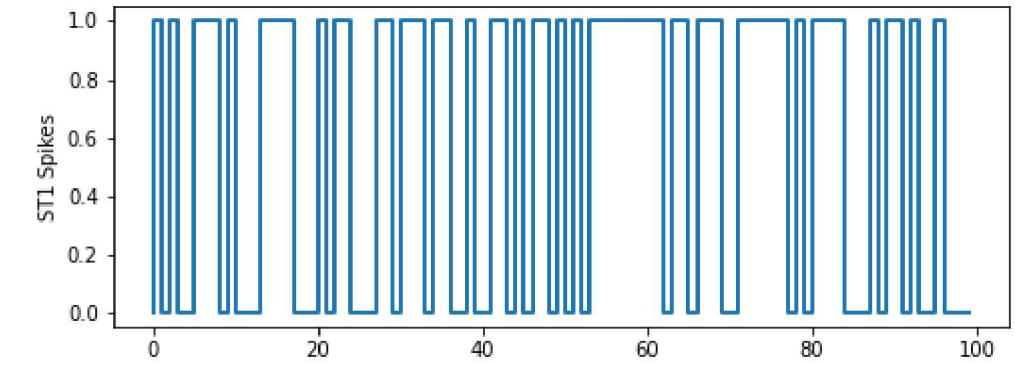
Caso 2: Spikes Identicos. Avaliando qual deve ser a dissimilaridade quando temos spikes iguais.

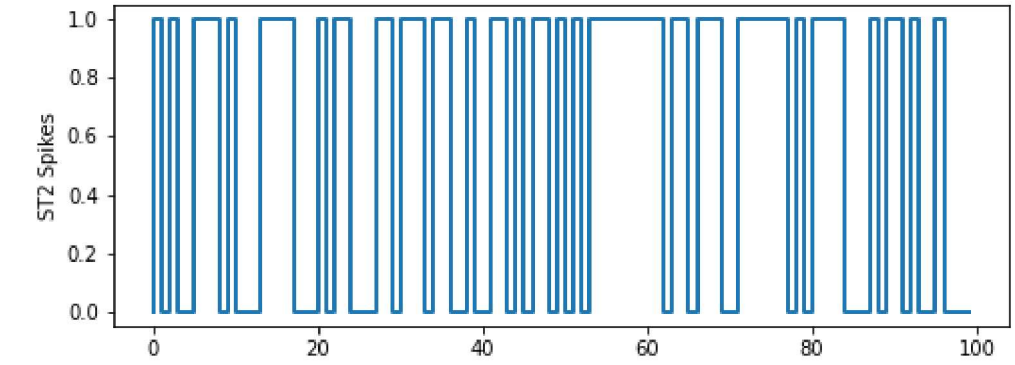
In [ ]:

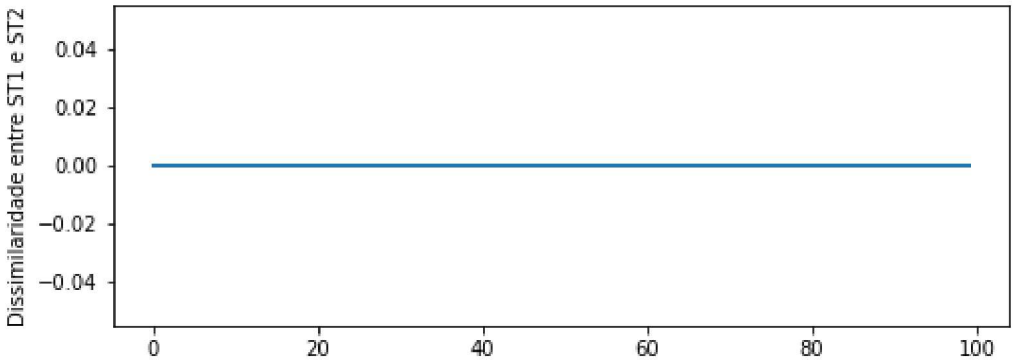
```
# Definição dos parâmetros iniciais:
```

```
n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T, duplicate=True)

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T)))
```

```
[array([ 1,  3,  6,  7,  8, 10, 14, 15, 16, 17, 21, 23, 24, 28, 29, 31, 32,
       33, 35, 36, 39, 42, 43, 45, 47, 48, 50, 52, 54, 55, 56, 57, 58, 59,
       60, 61, 62, 64, 65, 67, 68, 69, 72, 73, 74, 75, 76, 77, 79, 81, 82,
       83, 84, 88, 90, 91, 93, 96]), array([ 1,  3,  6,  7,  8, 10, 14, 15, 16, 17,
21, 23, 24, 28, 29, 31, 32,
       33, 35, 36, 39, 42, 43, 45, 47, 48, 50, 52, 54, 55, 56, 57, 58, 59,
       60, 61, 62, 64, 65, 67, 68, 69, 72, 73, 74, 75, 76, 77, 79, 81, 82,
       83, 84, 88, 90, 91, 93, 96])]
```








```
Dissimilaridade Total entre ST1 e ST2: 0.0
```

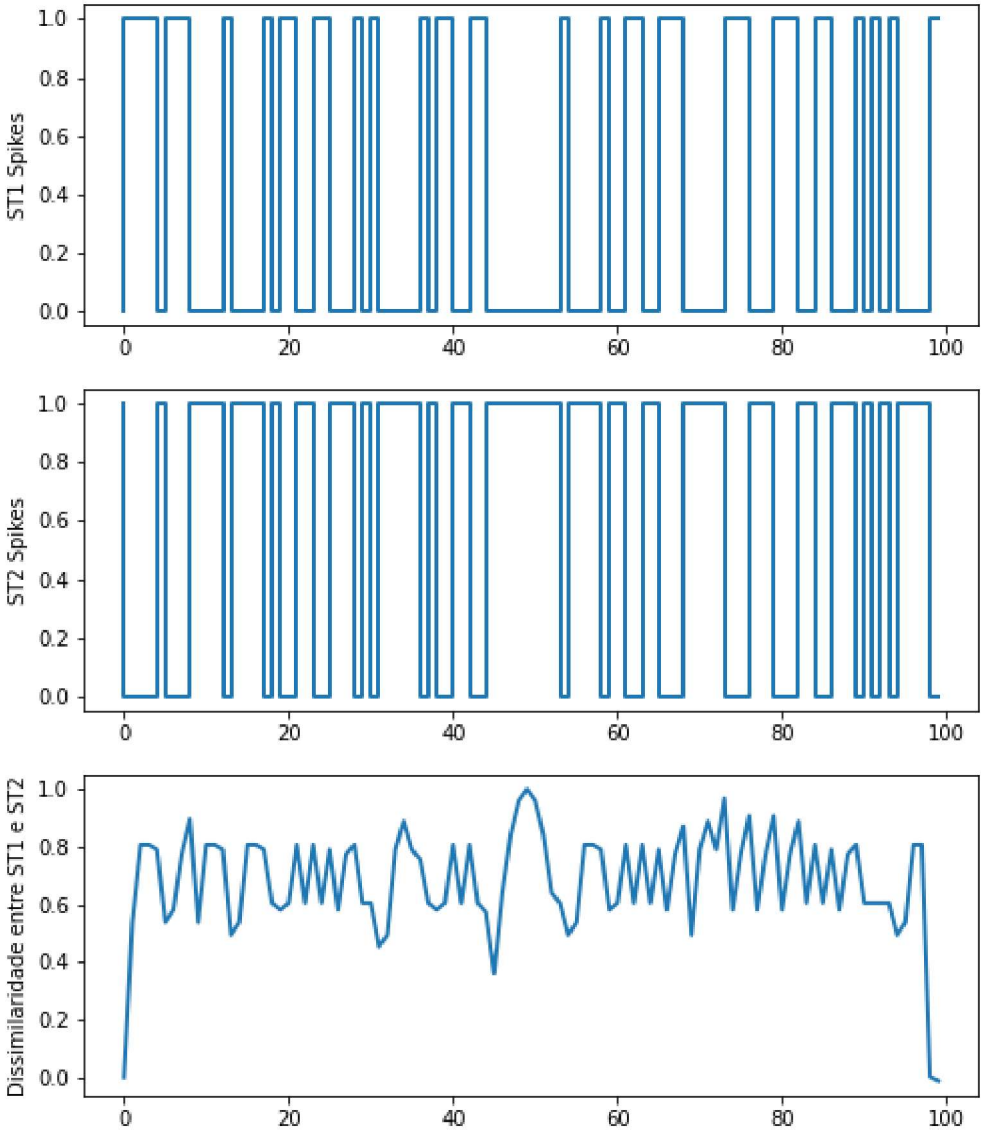
Caso 3.0: Spikes Complementares Aleatórios. Nesse caso eles são 100% diferentes em

tempos de disparo. O perfil de dissimilaridade assume valores negativos.

In [ ]:

```
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T, complementary_sp

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T))
```

```
[array([ 1,  2,  3,  4,  6,  7,  8, 13, 18, 20, 21, 24, 25, 29, 31, 37, 39,
       40, 43, 44, 54, 59, 62, 63, 66, 67, 68, 74, 75, 76, 80, 81, 82, 85,
       86, 90, 92, 94, 99]), array([ 0,  5,  9, 10, 11, 12, 14, 15, 16, 17, 19, 22,
23, 26, 27, 28, 30,
       32, 33, 34, 35, 36, 38, 41, 42, 45, 46, 47, 48, 49, 50, 51, 52, 53,
       55, 56, 57, 58, 60, 61, 64, 65, 69, 70, 71, 72, 73, 77, 78, 79, 83,
       84, 87, 88, 89, 91, 93, 95, 96, 97, 98])]
```



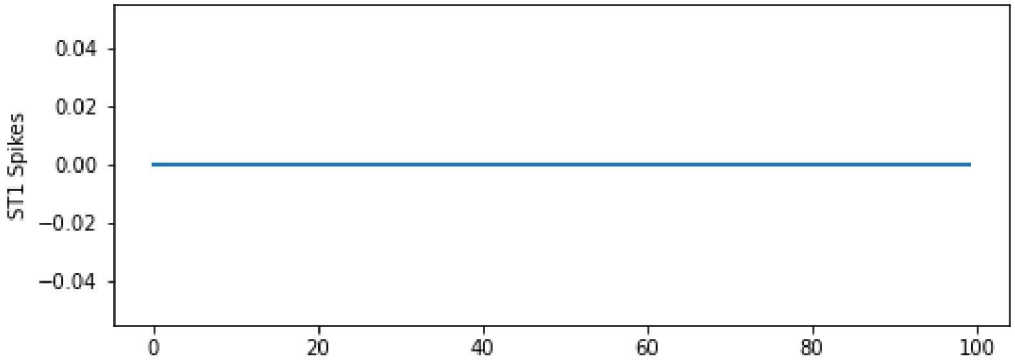

```
Dissimilaridade Total entre ST1 e ST2: 0.341369369558626
```

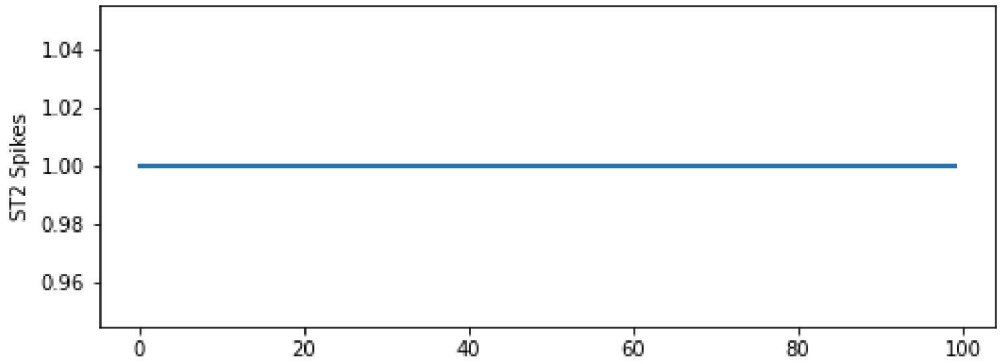
Caso 3.1: Spikes Complementares, sendo um completo de 1s e outro de 0s.

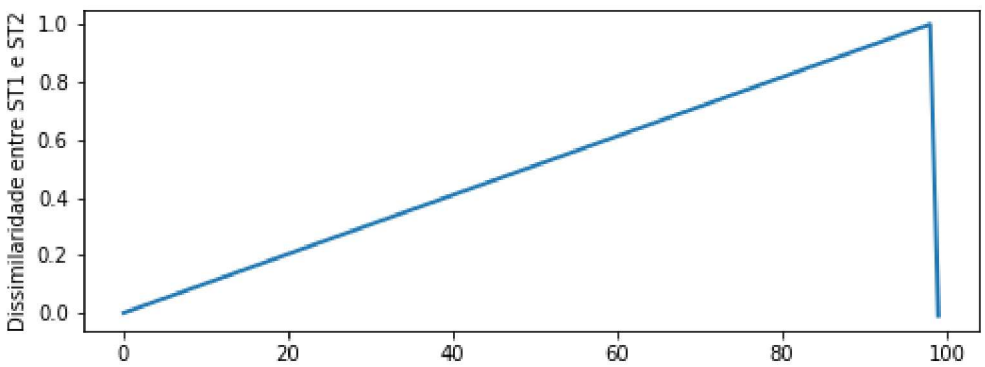
In [ ]:

```
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T, make_full_st=True, make_zero_st=True)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T)

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T)))
```

```
[array([], dtype=float64), array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 1
2, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])]
```







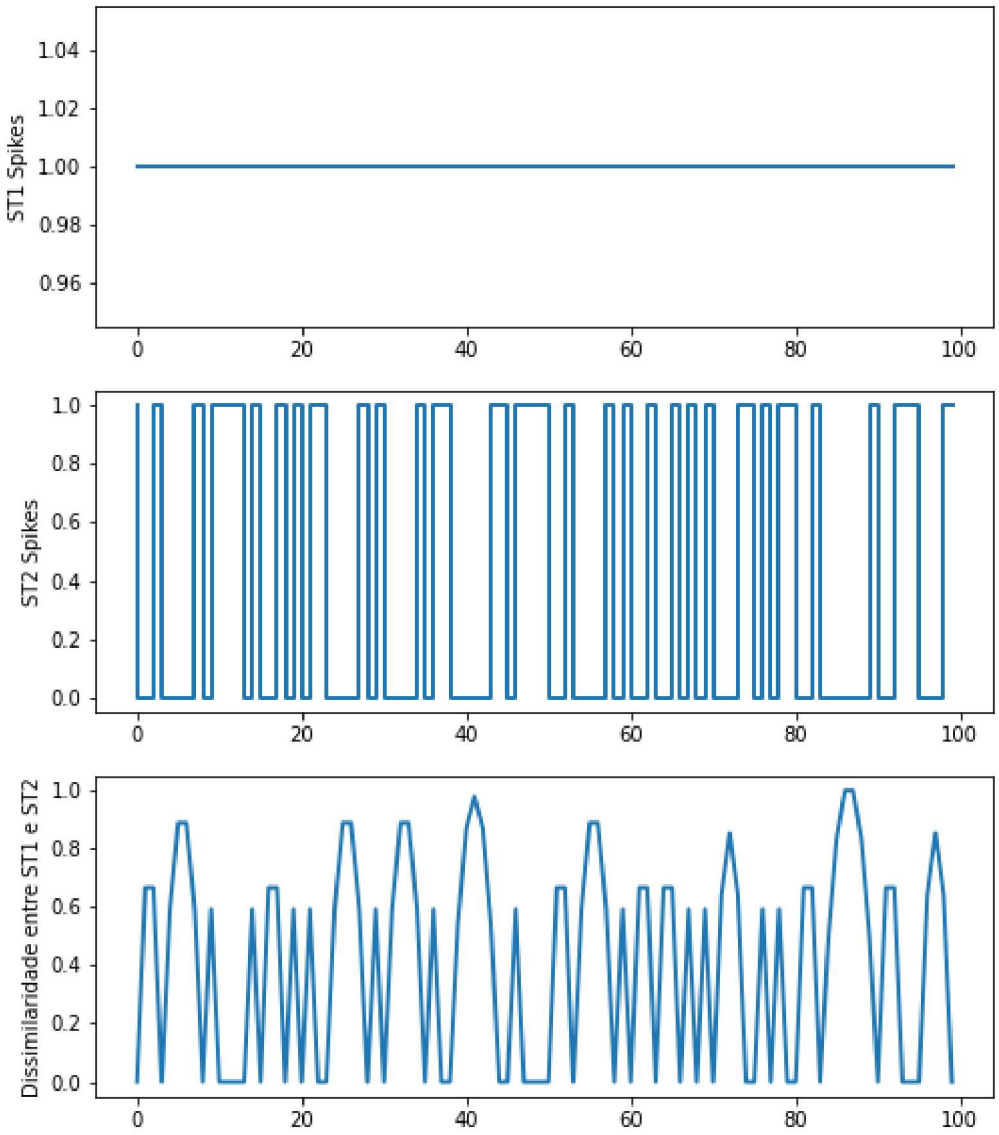

```
Dissimilaridade Total entre ST1 e ST2: 0.2474489795918367
```

Caso 4.1: (Teste com spikes extremos): Um Spike de 1s e outro aleatório. Aqui o perfil contém valores negativos.

In [ ]:

```
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T, make_full_st=True)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T)

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T)))
```

```
[array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]), array([ 0,  3,
8, 10, 11, 12, 13, 15, 18, 20, 22, 23, 28, 30, 35, 37, 38,
       44, 45, 47, 48, 49, 50, 53, 58, 60, 63, 66, 68, 70, 74, 75, 77, 79,
       80, 83, 90, 93, 94, 95, 99])]
```



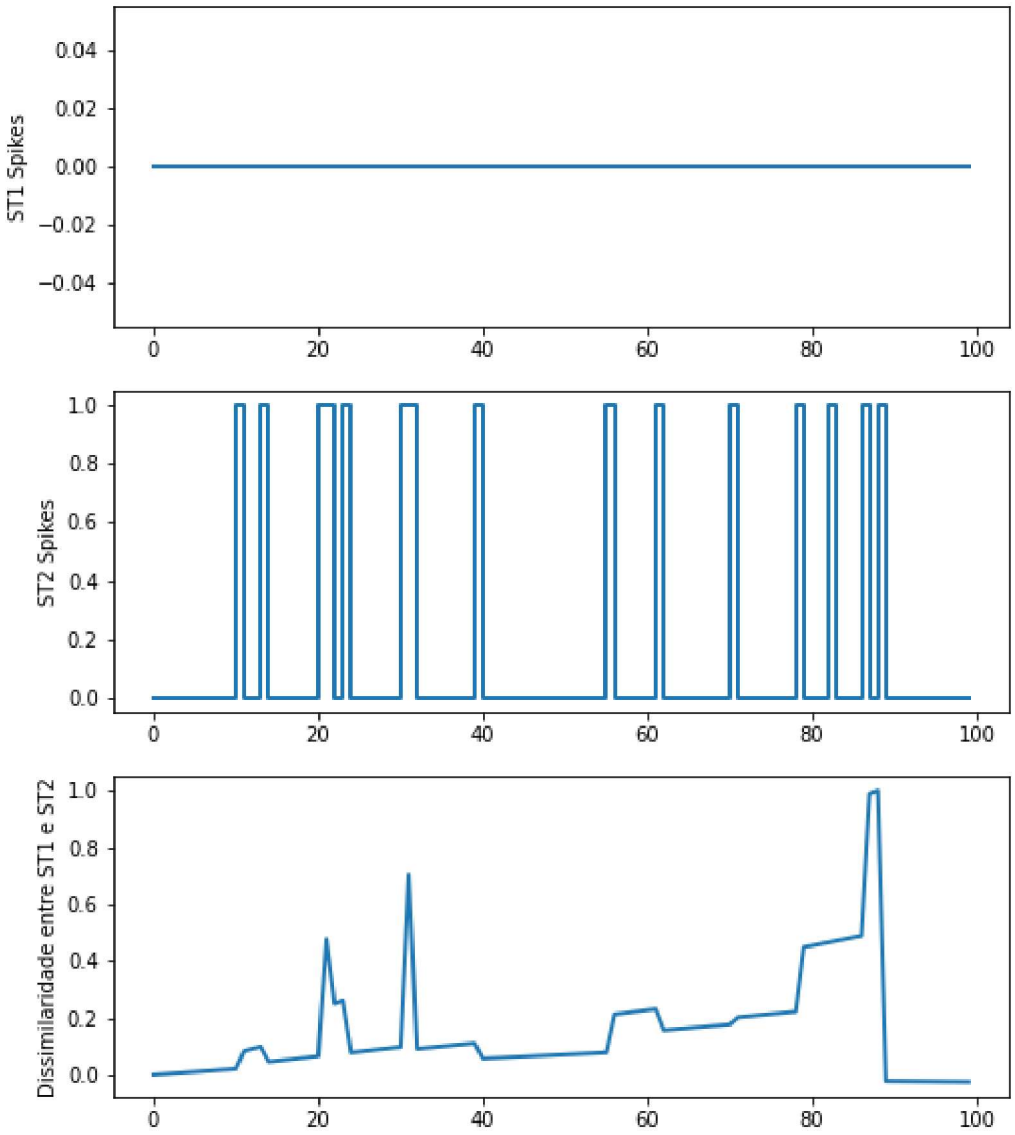

```
Dissimilaridade Total entre ST1 e ST2: 0.20519576719576718
```

Caso 4.2 (Teste com spikes extremos): Um Spike de 0s e outro aleatório.

In [ ]:

```python
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T, make_zero_st=True)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T)

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T)))
```

```
[array([], dtype=float64), array([11, 14, 21, 22, 24, 31, 32, 40, 56, 62, 71, 79, 8
3, 87, 89])]
```



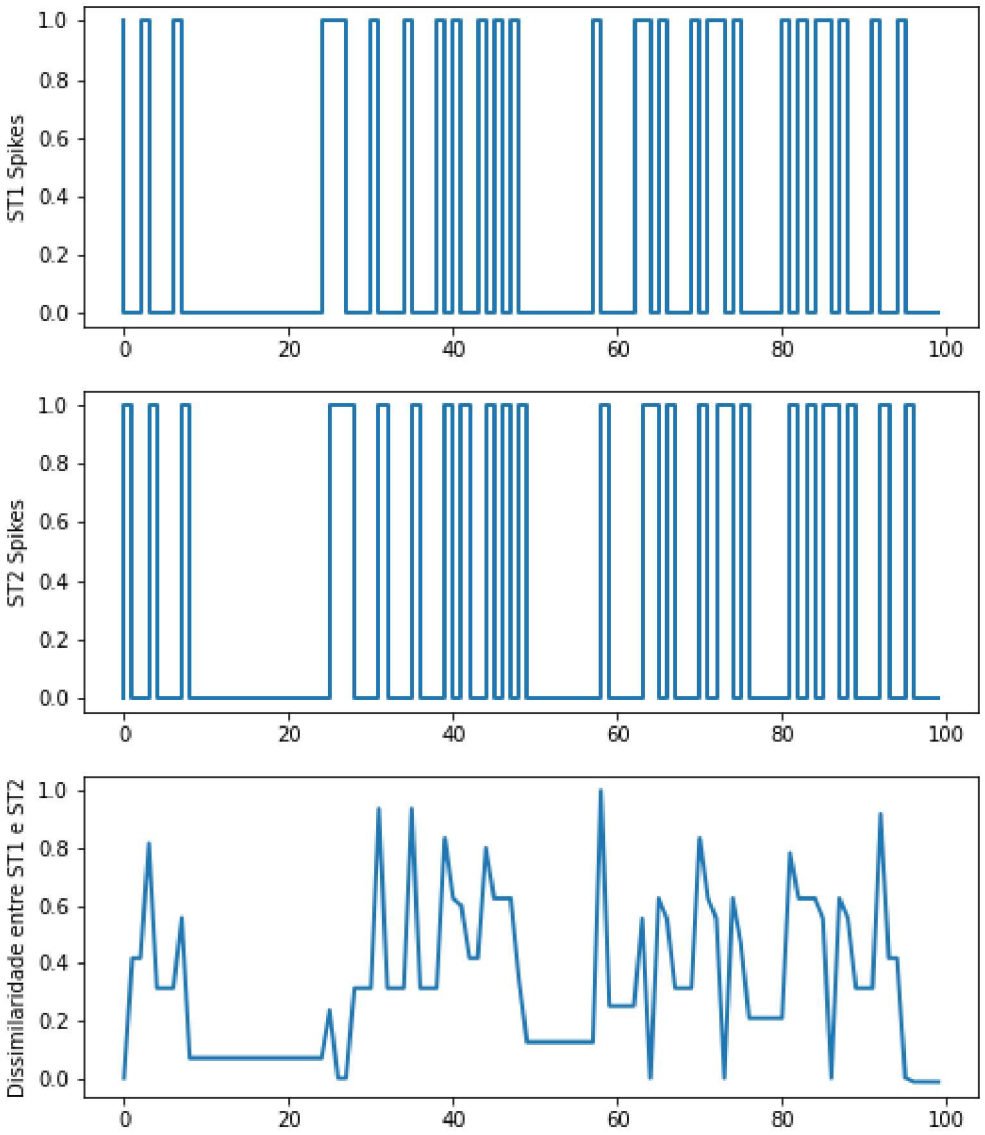

```
Dissimilaridade Total entre ST1 e ST2: 0.07481613891726252
```

Caso 5.1: Dois spikes com desvio de 1 unidade de tempo, para todos os instantes. Aqui, ST1 atrasado em 1 unidade de ST2.

In [ ]:

```
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T, time_shift=True,

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T))
```

```
[array([ 0,  3,  7, 25, 26, 27, 31, 35, 39, 41, 44, 46, 48, 58, 63, 64, 66,
       70, 72, 73, 75, 81, 83, 85, 86, 88, 92, 95]), array([ 1,  4,  8, 26, 27, 28,
32, 36, 40, 42, 45, 47, 49, 59, 64, 65, 67,
       71, 73, 74, 76, 82, 84, 86, 87, 89, 93, 96])]
```




```
Dissimilaridade Total entre ST1 e ST2: 0.1606981003109371
```

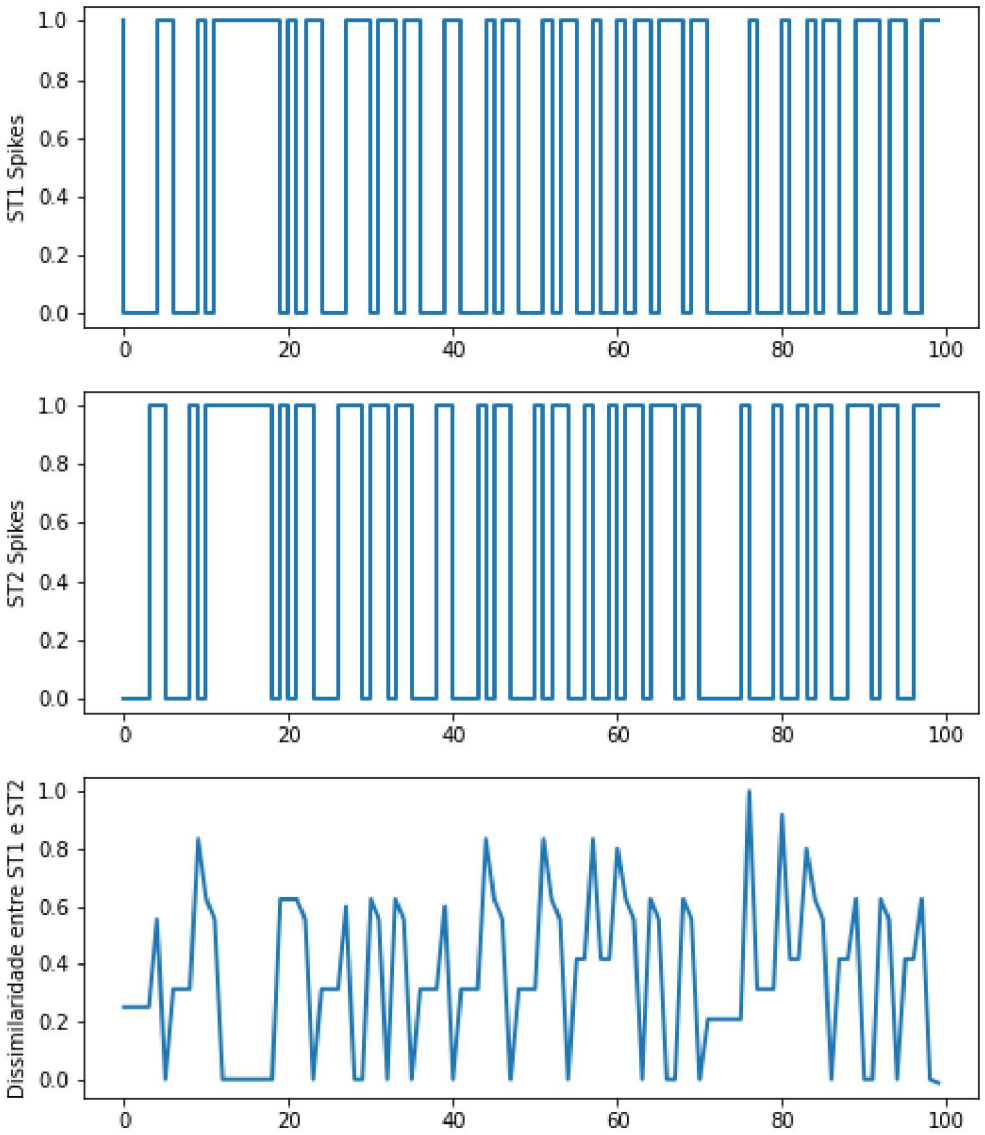
Caso 5.2: Dois spikes com desvio de 1 unidade de tempo, para todos os instantes. Aqui, ST1 adiantado em 1 unidade de ST2.

In [ ]:

```
# Definição dos parâmetros iniciais:

n = 2  # Quantidade de spike trains a comparar
T = 100 # Tempo total de simulação

# Geração de 2 Spike Trains
st_count = spike_count_generator(n, T)
ST_bin, STs = spike_times_generator(spike_count=st_count, n=n, T=T, time_shift=True,

print(STs)

S_original_norm = Dissimilarity(STs)
print_spikes(ST_bin, S_original_norm, T)
print("Dissimilaridade Total entre ST1 e ST2: {}".format(sum(S_original_norm)/(2*T))
```

```
[array([ 0,  5,  6, 10, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 24, 28, 29,
       30, 32, 33, 35, 36, 40, 41, 45, 47, 48, 52, 54, 55, 58, 61, 63, 64,
       66, 67, 68, 70, 71, 77, 81, 84, 86, 87, 90, 91, 92, 94, 95, 98, 99]), array
([-1,  4,  5,  9, 11, 12, 13, 14, 15, 16, 17, 18, 20, 22, 23, 27, 28,
```

```
       29, 31, 32, 34, 35, 39, 40, 44, 46, 47, 51, 53, 54, 57, 60, 62, 63,
       65, 66, 67, 69, 70, 76, 80, 83, 85, 86, 89, 90, 91, 93, 94, 97, 98])]
```




```
Dissimilaridade Total entre ST1 e ST2: 0.179569729534474
```

In [ ]: