L3. Fourierova transformacija na vremenskom otvoru - STFT

Uvod

Na vježbama ćete se upoznati s dvije različite implementacije STFT-a te inverznog STFT-a u MATLAB-u. Primijenit ćete STFT analizu na umjetnim i stvarnim signalima i vizualizirati rezultate.

STFT

Fourierova transformacija na vremenskom otvoru (STFT) definirana je sljedećim izrazom:

$$X(\tau,\omega)=\int_{-\infty}^{+\infty}x(t)g^{*}(t-\tau)\,e^{-j\omega t}dt\,. \qquad (1)$$

Za vremenski otvor *g* izabire se funkcija konačne energije koja ima željena svojstva lokalizacije u vremenskoj i frekvencijskoj domeni. Jedan od čestih izbora je Gaussova funkcija, za koju je produkt efektivnih širina u obje domene minimalan.

Uobičajena vizualizacija svojstava otvora je 2D funkcija $g(t)\cdot|G(w)|$, gdje je $G(w)$ Fourierova transformacija od $g(t)$, a uobičajene mjere u obje domene su efektivne širine.

U MATLAB-u ćemo integral (1) računati numerički, Eulerovom aproksimacijom. Konačan signal trajanja *N* i funkcije razlaganja jednoliko otipkamo periodom *T*, pa za diskretne pomake $\tau = kT$ aproksimacija glasi:

$$X(kT,\omega)\approx T\sum_{n=0}^{N-1}x(nT)g^{*}(nT-kT)\,e^{-j\omega nT}\,. \qquad (2)$$

STFT možemo izračunavati na dva načina. U prvom, signal pomnožen s vremenskim otvorom predstavlja odsječak signala na kojeg se primjenjuje Fourierova transformacija:

$$X(kT,\omega)\approx T\sum_{n=0}^{N-1}\left[x(nT)g^{*}(nT-kT)\right]e^{-j\omega nT}\,. \qquad (3)$$

Uzorke rezultata možemo numerički izračunati kao

$$X(kT,mF)\approx T\cdot DFT\left[x(nT)g^{*}(nT-kT)\right]\,. \qquad (4)$$

Implementacija u MATLAB-u (bez množenja s *T*) izgleda ovako:

```
len = length(signal);          % dužina signala
lenw = length(window);         % dužina otvora  (< dužine signala)

signal = [zeros(1, lenw-1), signal, zeros(1, lenw-1)]; % proširenje signala nulama
coeffs = complex(zeros(lenw, len+lenw-1), zeros(lenw, len+lenw-1)); % za rezultat
window = conj(window);

neg_shift = floor((lenw-1)/2);  % korektivni faktor uslijed proširenja signala nulama
```

```
for i=1: len+lenw-1     % fft po svim vremenskim pomacima
   x = signal(i:i+lenw-1) .* window;          % umnožak odsječka signala i otvora
   coeffs(:, i) = (fft(ifftshift(x)) .* …     % koristi se fft funkcija uz …
       exp(-j*2*pi/lenw *([0:lenw-1]) * ((i-1)-neg_shift))).';   % …korekciju kašnjenja
end
```

Uočimo da se koristi funkcija *ifftshift* kako bi se trenutak t=0 postavio u sredinu otvora (tj. uzeo simetričan nekauzalan odsječak). Nadalje, *fft* se primjenjuje na odsječcima signala pomnoženog s otvorom. Kako je otvor manji od signala i pomiče se po njemu, za svaki je rezultat potrebno korigirati iznos kašnjenja. U frekvencijskoj domeni to odgovara množenju s odgovarajućom eksponencijalnom funkcijom.

U primjerima na vježbama koristit ćete pripremljenu funkciju *stft()*.

U drugom načinu računanja STFT, signal moduliran eksponencijalnom funkcijom predstavlja novi signal na kojeg se primjenjuje konvolucija s preokrenutim vremenskim otvorom. Taj je pristup ekvivalentan filtriranju.

$$X(kT,\omega) \approx T\sum_{n=0}^{N-1}\left[x(nT)\, e^{-j\omega nT}\right]\cdot g^{*}\left[-\left(kT-nT\right)\right]. \qquad (5)$$

Implementacija u MATLAB-u (bez množenja s T) izgleda ovako:

```
len = length(signal);          % dužina signala
lenw = length(window);         % dužina otvora  (< dužine signala)

coeffs = complex(zeros(lenw, len+lenw-1), zeros(lenw, len+lenw-1)); % za rezultat

window = conj(window);
window = fliplr(window);    % preokretanje otvora u vremenu

for i=1:lenw    % filtriranje (konvolucija) po svim frekvencijama
   coeffs(i, :) = conv(signal .* exp(-j*2*pi*(i-1)/lenw * [0:len-1]), window);
end
```

U primjerima na vježbama koristit ćete pripremljenu funkciju *stft1()*.

Implementacija konvolucijom je superiorna kod kratkih otvora, dok je implementacija FFT-om superiorna kod velikih otvora čiji je broj uzoraka potencija broja 2.

Primjer korištenja

```
x=rand(1,128); % jednoliko distribuirani signal
% definiranje Gaussovog otvora
t=[-8:8]/4;
window=exp(-t.^2/2);
X=stft(x,window);     % STFT kao niz fft-ova
```

X1=stft1(x,window); % STFT kao slog filtara

% vizualizacija rezultata 3D prikaz
figure, colormap('copper'), surfl(abs(X)), shading('interp'), rotate3D on
ylabel('Frekvencija'), xlabel('sekunde')
% vizualizacija rezultata 2D prikaz
figure, h = imagesc(abs(X)); ylabel('Frekvencija'), xlabel('sekunde');
set(get(h, 'Parent'), 'YDir', 'normal'), colorbar

ISTFT

Izraz za inverznu Fourierovu transformaciju na vremenskom otvoru (ISTFT) je:

$$x(t) = \frac{1}{2\pi \langle h, g \rangle} \int_{\tau} \int_{\omega} X(\tau, \omega)\, h(t-\tau)\, e^{j\omega t} d\omega\, d\tau\,. \qquad (6)$$

U slučaju da je rekonstrukcijska funkcija $h = g$, onda se izraz pojednostavljuje:

$$x(t) = \frac{1}{2\pi \|g\|^2} \int_{\tau} \int_{\omega} X(\tau, \omega)\, g(t-\tau)\, e^{j\omega t} d\omega\, d\tau\,. \qquad (7)$$

Implementacija u MATLAB-u je opet zasnovana na otipkanim signalima i slikama te Eulerovoj aproksimaciji integrala.

U primjerima na vježbama koristit ćete se pripremljenom funkcijom *istft()*.

Možete se uvjeriti da je potpuna rekonstrukcija moguća i kada je funkcija $h \neq g$, čak i kada se radi o vrlo različitim otvorima! Jedini slučaj kada je rekonstrukcija nemoguća jest kada je skalarni produkt u nazivniku (6), odnosno norma (7) jednaka nuli.

Dodatna uputa: za otipkane (diskretne) *h* i *g* skalarni produkt je *h* * *g*'. Kako je ISTFT implementiran korištenjem funkcije *ifft()* nije potrebno dijeljenje s 2*pi.

Pogledajte programski kôd funkcije!

Dva su načina korištenja funkcije ISTFT:

x_rekonstruirani = istft(X, window); % ako je rekonstrukcijski otvor isti
x_rekonstruirani = istft(X, reconstruction_window, analysis_window); % ako su različiti

U drugom slučaju se analizirajući otvor koristi isključivo za računanje <*h*, *g*>, odnosno konstante u nazivniku. Uz nepoznavanje analizirajućeg otvora rekonstrukcija je ipak moguća, ali će rezultat biti potrebno skalirati (pronaći pravi iznos konstante u nazivniku).