Ivan Krajnović
Vladimir Mikašinović
RAČ

Projekt iz kolegija Napredne metode digitalne obrade signala:

Brza wavelet transformacija korištenjem ljestvičaste realizacije

0. Uvod

Otkako su računala postala sveprisutna, a DSP algoritmi svakodnevno primjenjivani narasla je potreba za učinkovitim realizacijama algoritama obrade signala. A kako su waveleti jedna od složenijih i zahtjevnijih transformacija za očekivati je da se ni mogućnosti za mala povećanja u brzini izvođenja neće ignorirati.

U ovom seminaru ćemo prezentirati realizaciju brzih wavelet slogova koristeći ljestvičastu realizaciju čime se može dobiti dodatnih max. 50% ubrzanja (za duge filtre) u odnosu na polifaznu realizaciju.

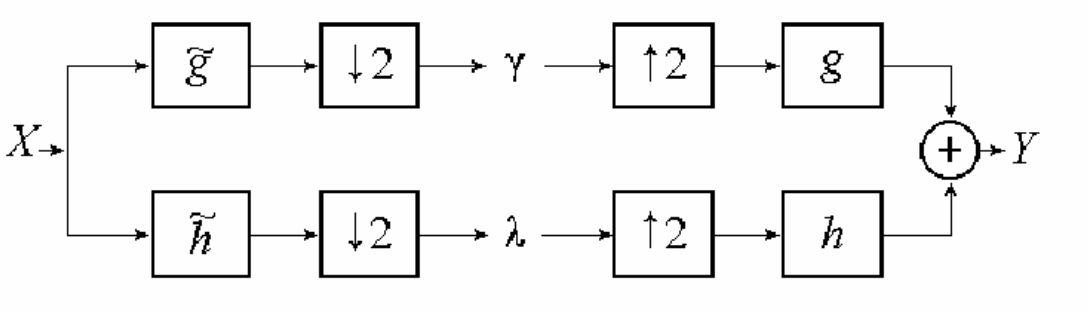
Dok na papiru i u kontekstu algoritama 50% možda i ne zvuči puno - itekako se isplati, a pogotovo kod embedded uređaja koji još dugo vremena neće imati procesorske moći i memorije za razbacivanje i u realtime primjenama gdje je brzina direktno vezana uz cijenu i mogućnosti sustava.

U uvodnom dijelu seminara ćemo se nakratko podsjetiti polifazne dekompozicije, u središnjem ćemo objasniti pojedinosti vezane za ljestvičastu realizaciju, a na kraju seminara slijedi par primjera i objašnjenja te MATLAB kod kojim su primjeri generirani.

1. Polifazna reprezentacija

Kod uobičajenog crtanja dekompozicijskog dijela wavelet filtarskih slogova nakon filtara slijede decimatori za faktor 2. Iako je takva reprezentacija zgodna i razumljiva očito je kako nije najbolje rješenje budući da pola podataka koje smo izračunali odmah odbacujemo.

Slična konstatacija vrijedi i za rekonstrukcijsku stranu gdje u ulazni signal nepotrebno ubacujemo nule i povećavamo broj računskih operacija.



Slika 1: "obična" realizacija wavelet filtarskog sloga

U stvarnosti se wavelet slogovi ne realiziraju na ovaj način nego se na njih primjenjuje (barem) polifazna dekompozicija koja broj operacija prepolovljuje i dovodi ga u očekivane okvire.

Polifaznu dekompoziciju filtra izvodimo tako da razdvojimo parne i neparne uzorke signala i impulsnog odziva filtra i dopustimo djelovanje samo parnih na parne i neparnih na neparne. Zašto je tome upravo tako - objašnjeno je u nastavku.

Počinjemo sa rastavljanjem ulaznog signala na parne i neparne uzorke:

$$X(z) = X_0(z^2) + z^{-1}X_1(z^2)$$

a isto postupimo i sa impulsnim odzivima dekompozicijskih filtara H_0 i H_1:

$$H_0(z) = H_{00}(z^2) + z^{-1}H_{01}(z^2)$$
$$H_1(z) = H_{10}(z^2) + z^{-1}H_{11}(z^2)$$

Pokušamo li izračunati rezultat filtracije takvim slogom:

$$\begin{aligned} H_0(z)X(z) &= H_{00}(z^2)X_0(z^2) + z^{-1}H_{00}(z^2)X_1(z^2) + \\ &\quad z^{-1}H_{01}(z^2)X_0(z^2) + z^{-2}H_{01}(z^2)X_1(z^2) \\ H_1(z)X(z) &= H_{10}(z^2)X_0(z^2) + z^{-1}H_{10}(z^2)X_1(z^2) + \\ &\quad z^{-1}H_{11}(z^2)X_0(z^2) + z^{-2}H_{11}(z^2)X_1(z^2) \end{aligned}$$

i podsjetimo li se da nakon filtracije dolaze decimatori koji odbacuju neparne uzorke - dolazimo do zaključka da **neparne članove ne moramo ni računati (članovi sa z^{-1})**.

Izbacimo li onda te članove i jednadžbe za $H_0(z)X(z)$ i $H_1(z)X(z)$ prikažemo u matričnom obliku dobijamo:

$$\begin{bmatrix} H_0(z)X(z) \\ H_1(z)X(z) \end{bmatrix} = H_P(z)X(z)$$

gdje su $H_P(z)$ i $X(z)$:

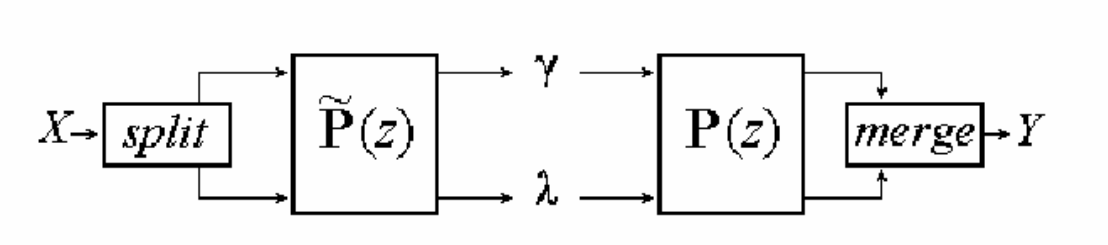
$$H_P(z) = \begin{bmatrix} H_{00}(z) & H_{01}(z) \\ H_{10}(z) & H_{11}(z) \end{bmatrix}$$
$$X(z) = \begin{bmatrix} X_0(z) \\ X_1(z) \end{bmatrix}$$

što je uobičajen način predstavljanja pomoću polifaznih matrica.

Polifazna matrica rekonstrukcijske strane je tada:

$$F_P(z) = \begin{bmatrix} F_{01}(z) & F_{11}(z) \\ F_{00}(z) & F_{10}(z) \end{bmatrix}$$

Cijeli wavelet slog crtamo ovako - "skrivajući" polifaznu realizaciju u jedan blok sa 2 ulaza i 2 izlaza:



Slika 2: Polifazna realizacija wavelet filtarskog sloga

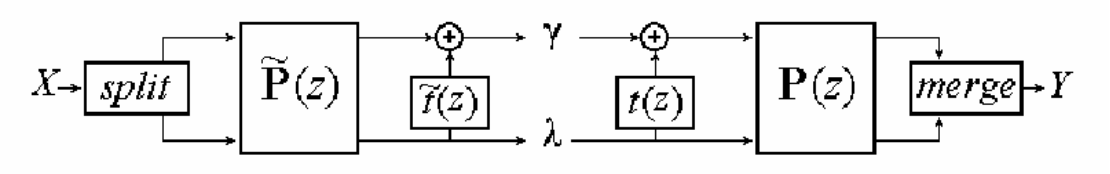
Ukupna ušteda operacija ovakvim načinom računanja je:

1/2 (kraći filtri) * 1/2 (kraći signal) * 2 (računaju se dvije filtracije) = 1/2 = **50%**

Da se može i bolje od ovoga vidjet ćemo u nastavku seminara.

2. Podizanje (lifting)

Pojam podizanja (iako malo nespretan) se u kontekstu wavelet transformacija koristi za realizacije u kojima se putem filtara postavljenih kao ljestve svojstva neparne (ili parne) grane "podižu" prema željenim koristeći signal iz druge grane.



Slika 3: Polifazni filter s jednim stupnjem podizanja

Jednostavan primjer podizanja bi bila linearna interpolacija parnih uzoraka na temelju neparnih. Filtar koji bi bio u "ljestvama" bi bio jednostavan usrednjivač na temelju 2 uzorka.

Ovakva realizacija vodi na kaskadiranje pojedinih "prečki" ljestava, a u matričnom zapisu na umnoške alternirajućih koraka podizanja:

$$H_{Pnew} = \prod \left\{ \begin{bmatrix} 1 & 0 \\ T_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -S(z) \\ 0 & 1 \end{bmatrix} \right\} H_P(z)$$

Dodavanjem koraka podizanja možemo dobiti filtar željenih svojstava, no pokazuje se da je moguć i obrnuti smjer - dizajnira se filtarski slog i faktorizira se u korake.

Dodavanjem jednog koraka podizanja dobijamo:

$$H_{0new}(z) = H_0 - H_1 S(z^2)$$
$$H_0(z) = H_{0new}(z) + H_1 S(z^2)$$

Iz čega se jasno vidi da inverzni filtarski slog možemo lako dobiti. Podizanje uvijek djeluje samo na jednu granu tako da ćemo inverz dobiti jednostavnom primjenom koraka podizanja unazad i promjenom predznaka na zbrajalima.

Treba naglasiti da navedena mogućnost vrijedi bez obzira na "ljepotu" funkcija S i T (linearnost, vremensku nezavisnost...)

Neke prednosti ljestvičaste realizacije:
1. Ubrzanje do 50% (za dugačke filtre)
2. In-place računanje štedi memoriju
3. Ljestvičasta struktura jamči mogućnost rekonstrukcije i to za bilo kakve funkcije podizanja S i T.

3. Euklidov algoritam

Da je faktorizacija u korake podizanja moguća za svaki filtar pokazali su Daubechies i Sweldens. Za određivanje faktorizacije predložili su Euklidov algoritam dijeljenja (nalaženja najvećeg zajedničkog nazivnika) primjenjen na Laurentove polinome koji predstavljaju impulsne odzive filtara.

Euklidov algoritam za Laurentove polinome:

$a_0(z) = a(z)$

$b_0(z) = b(z)$

počevši od i=0 ponavljaj slijedeće korake dok b_i ne postane 0:

$a_{i+1}(z) = b_i(z)$

$b_{i+1}(z) = a_i(z) \% b_i(z)$

gdje % predstavlja operaciju modulo.

Rezultat Euklidovog algoritma prikazan matrično jest:

$$\begin{bmatrix} a(z) \\ b(z) \end{bmatrix} = \prod \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_n(z) \\ 0 \end{bmatrix}$$

Treba napomenuti da rezultat dijeljenja Laurentovih polinoma nije jednoznačno određen već samo do faktora z^P - postoji mnogo različitih faktorizacija što daje veliku slobodu izbora.

4. Faktorizacija polifaznih matrica

Uzmemo li našu polifaznu matricu i na njoj primijenimo jedan korak podizanja dobit ćemo:

$$P_{new}(z) = P(z) \begin{bmatrix} 1 & 0 \\ S(z) & 1 \end{bmatrix} = \begin{bmatrix} H_{00}(z) + S(z)H_{01}(z) & H_{01}(z) \\ H_{10}(z) + S(z)H_{11}(z) & H_{11}(z) \end{bmatrix}$$

Primjetimo da izrazi u prvom stupcu desne matrice izgledaju kao poznati izrazi za dijeljenje s ostatkom (A=B*Q+R) te ćemo primjenom Euklidovog postupka na prvi stupac moći obrnuti postupak i iz već gotove polifazne matrice dobijati jedan po jedan korak podizanja.

Postupak završava kada je matrica potpuno faktorizirana i tada se obično dodaje i korak skaliranja oblika:

$$\begin{bmatrix} K_A & 0 \\ 0 & K_B \end{bmatrix}$$

gdje su K_A i K_B recipročne konstante.

5. Primjeri i komentari

haar-ov wavelet:

$$H_P = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

```
Koraci podizanja su:

       | 0.7071       0    |
       |                   |
 ans = |                   |
       |                   |
       |    0       1.414  |

       |   1        0  |
       |               |
 ans = |               |
       |               |
       | - 0.5      1  |

       | 1      1  |
       |           |
 ans = |           |
       |           |
       | 0      1  |
```

što odgovara Matlabovoj faktorizaciji (Matlab drugačije slaže filtere u polifaznu matricu pa su koraci transponirani):

```
  | 1     0  |   | 1     - 0.5  |   | 0.7071       0    |
  |          |   |              |   |                   |
  |          |   |              |   |                   |
  |          |   |              |   |                   |
  | 1     1  |   | 0       1    |   |    0       1.414  |
```

db4 wavelet:

```
Koraci podizanja su:

       | 0.1224*z^(-1)            0        |
       |                                   |
 ans = |                                   |
       |                                   |
       |       0           8.172*z^(+1)    |

       | 1     - 3.072*z^(-1)  |
       |                       |
 ans = |                       |
       |                       |
       | 0           1         |

       |               1                  0  |
       |                                     |
 ans = |                                     |
       |                                     |
       | + 0.0006873*z^(+1) + 0.05762     1  |

       | 1     - 16.26 + 5.199*z^(-1)  |
       |                               |
 ans = |                               |
       |                               |
       | 0               1             |
```

```
       |           1                0  |
       |                               |
ans =  |                               |
       |                               |
       | - 0.116*z^(+1) - 0.292     1  |

       | 1     3.103  |
       |              |
ans =  |              |
       |              |
       | 0       1    |
```

što je također identično Matlabovoj faktorizaciji.

Primjere za filtere višeg reda nisam priložio jer bi zauzimali previše prostora, a sve ih je moguće dobiti korištenjem priloženog programa ako ga se uputi na "pravi put" kod odabira rezultata dijeljenja - npr. gledajući već gotovu Matlabovu faktorizaciju ili metodom pokušaja i promašaja.

Nažalost taj put (pogotovo za filtre višeg reda) nije očit i ne postoji neka pametna metoda kojom bi ga odredili bez da razmatramo sve mogućnosti. (Možda i postoji, ali nisam vidio da se u literaturi spominje)

Matlab, koliko sam uspio primijetiti, isprobava sve faktorizacije no to dosta traje (22 sekunde za db8 na mom P4), a sam algoritam je relativno velik i razasut kroz nekoliko prilično kompliciranih funkcija.

S obzirom da nije imalo smisla direktno koristiti Matlabove funkcije za faktorizaciju odlučio sam se ipak napisati ovakvu pojednostavljenu verziju koja demonstrira osnovni algoritam ali zahtijeva ljudsku pomoć kod odabira rezultata dijeljenja.

6. MATLAB kod

```
clear all;

% Odabrani wavelet
odabir='db2';

% Uzmi Laurentove polinome koji odgovaraju odabranom waveletu
% Prvo ide rekonstrukcijski par, a onda analiticki
 [F0,F1,H0,H1]=wave2lp(odabir);

% Ispisi Matlabovo rjesenje za faktorizaciju
[MF,PM]=ppmfact(H0,H1);
disp(PM);
displmf(MF{1});
disp('-----');

% Uzmi polinome H00, H01, H10, H11
H00=even(H0);
H01=odd(H0);
H10=even(H1);
H11=odd(H1);

% Slozi polifaznu matricu
M=laurmat({H00,H01;H10,H11})

% Euklidski algoritam
k=0;
MA=laurmat({1,0;0,1});

% Cell matrix za pospremanje rezultata faktorizacije
ki=1;
koraci={};

while 1
    % primal ili dual faktorizacija ?
    if k==1
        A=H10;
        B=H11;
    else
        A=H01;
        B=H00;
    end

    % Ispisi i podijeli polinome
    A
    B
    DM=euclidediv(A,B);

    % Ispisi sve kombinacije Q i R
    for i=1:size(DM,1)
        disp(sprintf('Kombinacija br. %d :',i));
        disp('Q:');
        disp(DM{i,1});
        disp('R:');
        disp(DM{i,2});
    end

    % I daj korisniku neka odabere jednu
    j=input('Odaberi kombinaciju :');
    Q=DM{j,1};
    R=DM{j,2};

    % Izracunaj novu polifaznu matricu i korak podizanja
    if k==1
        korak=laurmat({1 0;Q 1})
        H00=H00-Q*H01;
        H10=H10-Q*H11;
    else
        korak=laurmat({1 Q;0 1})
        H01=H01-Q*H00;
        H11=H11-Q*H10;
    end
```

```
    % Ispisi novu matricu i zabiljezi korak
    nova=laurmat({H00 H01;H10 H11})
    koraci{ki}=korak;
    ki=ki+1;

    % Uvjeti zaustavljanja i zavrsni koraci skaliranja:
    % Ako je matrica gornje trokutasta - dodaj korak skaliranja i zavrsi
    if nova{2,1}==0
        n=nova;
        n{1,1}=1;
        n{2,2}=1;
        n{1,2}=n{1,2}/nova{1,1};
        koraci{ki}=n;
        ki=ki+1;

        n=nova;
        n{1,2}=0;
        koraci{ki}=n;
        break;
    end
    % Ako je matrica donje trokutasta - dodaj korak skaliranja i zavrsi
    if nova{1,2}==0
        n=nova;
        n{1,1}=1;
        n{2,2}=1;
        n{2,1}=n{2,1}/nova{2,2};
        koraci{ki}=n;
        ki=ki+1;

        n=nova;
        n{2,1}=0;
        koraci{ki}=n;
        break;
    end

    % Promijeni tip faktorizacije
    k=1-k;
end

MA=laurmat({1 0;0 1});

disp('Koraci podizanja su:');
for j=ki:-1:1
    koraci{j}
    MA=MA*koraci{j};
end

disp('Umnozak svih koraka je:');
MA

return
```

Literatura:

1. Factoring Wavelet Transforms into Lifting Steps,
   I. Daubechies, W. Sweldens
   J. Fourier Anal. Appl., Vol. 4, Nr. 3, pp. 247-269, 1998.
   http://cm.bell-labs.com/who/wim/papers/factor/factor.pdf

2. The fast lifting wavelet transform tutorial
   C. Valens, 1999.-2004.
   http://perso.wanadoo.fr/polyvalens/clemens/download/tflwt_26022004.pdf

3. Predavanja iz Naprednih metoda digitalne obrade signala
   D. Seršić, Fakultet elektrotehnike i računarstva Zagreb, 2004./2005.
   http://nmdos.zesoi.fer.hr/predavanja.html

4. Matlab 7.0 dokumentacija, posebno wavelet funkcije vezane uz lifting i objekte laurpoly i laurmat.
   http://www.mathworks.com