CAPITOLUL 6

REALIZAREA FILTRELOR DIGITALE

6.1 IMPLEMENTAREA FILTRELOR CU RĂSPUNS FINIT LA IMPULS (FIR)

Funcția de transfer a unui filtru cu răspuns finit la impuls de lungime N este de forma:

$$H(z)=\frac{Y(z)}{X(z)}=h_0+h_1z^{-1}+..+h_{N-1}z^{N-1} \qquad (6.1)$$

Ecuația cu diferențe finite este:

$$y(n)=\sum_{k=0}^{N-1}x(n-k)h_k \qquad (6.2)$$

unde $x(n)$ este intrarea filtrului la momentul n, $y(n)$ este ieşirea, h_k sunt coeficienții funcției pondere şi N este lungimea filtrului.

6.1.1 Forma transversală

Ecuația (6.2) poate fi implementată într-o structură transversală ca în figura următoare.

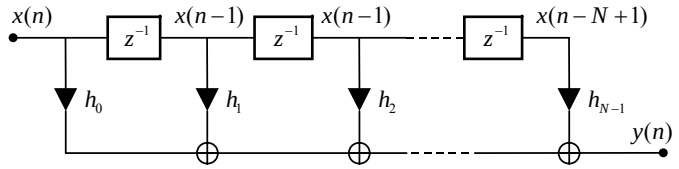

Figura 5.1. Forma transversală pentru FIR

Pentru calculul ieşirii $y(n)$ trebuie memorate stările filtrului şi coeficienții funcției pondere.

Stările filtrului sunt stocate într-o listă circulară care conține ultimele N eşantioane de intrare indexată modulo N.

Coeficienții filtrului sunt stocați într-un vector de lungime N.

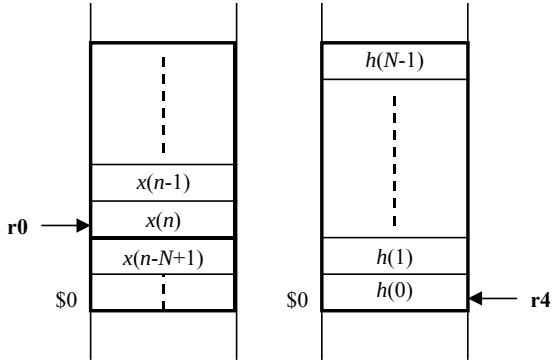

Figura 6.2. Organizarea memoriei pentru filtrul FIR

Se foloseşte o listă circulară de lungime N parcursă cu un index limitat modulo N. Eşantionul de intrare $x(n)$ este introdus în listă la indexul curent. Apoi este parcursă lista prin decrementarea indexului cu variabila k.

Calculul ieşirii $y(n)$ se face prin înmulțire cu acumulare într-o variabila **sum** inițializată cu 0 la fiecare iterație.

```
int main()
{
     Word16 x[DataBlockSize],y[DataBlockSize],delay[N],h[N];
     Word32 sum;
     short n,i;
     long unsigned k,index;

     ...

     fread(x,sizeof(Word16),DataBlockSize,fpx);

     for (n=0; n<DataBlockSize; n++)
     {
          delay[index%N]=x[n];
          sum=0;
          for (k=0; k<N; k++)
          {
               sum=L_mac(sum,h[k],delay[(index-k)%N]);
          }
          y[n]=round(sum);
          index++;
     }

     fwrite(y,sizeof(Word16),DataBlockSize,fpy);

     ...

}
```

Toți coeficienții sunt considerați în format fracționar (de modul subunitar) reprezentați în C în format Word16. Dacă în urma proiectării filtrului FIR apar și coeficienți de modul supraunitar, atunci toți coeficienții trebuie scalați prin împărțirea cu:

$$m = \max\left(\left|h_k\right|\right) \tag{6.3}$$

De asemenea, pentru a evita depășirea capacității registrelor, dacă eșantioanele intrării sunt în intervalul $(-1,1)$, atunci trebuie îndeplinită condiția:

$$s = \left|\sum_{k=0}^{N-1} h_k\right| \le 1 \tag{6.4}$$

Dacă nu este îndeplinită condiția (6.4) atunci fie intrarea $x(n)$, fie coeficienții h_k trebuie scalați cu factorul s. Dacă intrarea este împărțită la s, atunci gama dinamică a semnalului la intrare se micșorează, deci și raportul semnal-zgomot al structurii se micșorează. Pe de altă parte, dacă sunt scalați coeficienții, aceștia vor putea fi reprezentați pe mai puțini biți și este afectat răspunsul în frecvență al filtrului. Aceste implicații trebuie evaluate în funcție de aplicație și aleasă una din variante.

6.2 IMPLEMENTAREA FILTRELOR CU RĂSPUNS INFINIT LA IMPULS (IIR)

Pentru un filtru IIR avem funcția de transfer de forma:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{6.5}$$

Coeficienții numărătorului b_k și ai numitorului a_k se determină printr-una din metodele de proiectare a filtrelor digitale cu răspuns infinit la impuls (folosind eventual mediul MATLAB) în funcție, bineînțeles, de condițiile de proiectare cerute.

Notând cu $x(n)$ secvența de la intrarea filtrului și cu $y(n)$ secvența de la ieșirea filtrului, rezultă ecuația cu diferențe finite:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k) \tag{6.6}$$

Pentru calculul ieșirii $y(n)$ există mai multe posibilități de implementare în funcție de structura aleasă. Performanțele implementării

(complexitate aritmetică memorie ocupată, transferuri paralele) diferă în funcție de structura aleasă.

De asemenea, în proiectare şi implementare trebuie evaluate efectele reprezentării semnalelor utilizând un număr finit de biți. În cazul reprezentării numerelor în formate cu virgulă fixă apare o depăşire dacă rezultatul unei operații aritmetice este de modul supraunitar. Acest lucru este posibil după operațiile de însumare şi de înmulțire cu o constantă de modul supraunitar.

Analiza posibilității depăşirilor comportă două aspecte:

- analiza funcției de transfer, $H(z)$ şi eventual scalarea acesteia astfel încât să se elimine sau să se reducă suficient de mult probabilitatea ca semnalul de ieşire să fie de modul supraunitar;
- analiza posibilității depăşirilor în toate nodurile rețelei, conducând (dacă este necesar) la o scalare a semnalului compensată în final, astfel încât să nu afecteze realizarea funcției de transfer impuse.

În filtrele digitale atât amplitudinea semnalului cât şi coeficienții sunt valori discrete sau cuantizate cu un număr finit de biți. Deşi astfel de sisteme sunt neliniare, efectele acestor cuantizări pot fi interpretate ca fiind surse de zgomot aditiv. Erorile de cuantizare ale coeficienților modifică funcția de transfer a filtrului şi se produc astfel variații ale zerourilor şi polilor, variații care schimbă nedorit răspunsul în frecvență al filtrelor. Erorile de cuantizare a semnalului pot fi privite ca surse de zgomot aditiv în diferite noduri ale rețelei.

Cuantizarea coeficienților are efecte diferite de cuantizarea semnalului din punct de vedere al zgomotului. Cuantizarea coeficienților produce erori în răspunsul în frecvență şi în amplitudine ale filtrului, fiind erori deterministe. Aceste erori pot fi minimizate folosind structuri de filtre mai puțin sensibile,

cu aceeaşi funcție de transfer şi alocând un număr suficient de biți pentru coeficienți.

Cuantizarea semnalului are efecte în diferite noduri ale filtrului, putând fi considerate ca surse de zgomot alb, aditiv, înrăutățind RSZ al ieşirii. Aceste erori pot fi minimizate realizând înmulțirea şi acumularea rezultatului în registre de lungime dublu-cuvânt.

O cale de mijloc pentru evitarea înrăutățirii raportului semnal-zgomot (RSZ) şi, pe de altă parte, evitarea depăşirii gamei dinamice (overflow) este scalarea. Nu este nevoie de scalare în fiecare nod al rețelei deoarece semnalele în unele noduri sunt doar întârzieri ale semnalelor din alte noduri. De asemenea, dacă rezultatul final al adunării semnalelor din mai multe noduri este subunitar, în cazul adunării în complement față de doi în final se obține rezultatul corect indiferent dacă în diferite noduri apar depăşiri la sumele parțiale. Trebuie astfel realizată o tehnică de scalare a semnalului la intrare pentru a evita depăşirile numai în nodurile unde acestea pot apare.

Se presupune că semnalul de intrare este în modul subunitar:

$$|x(n)| \le 1 \tag{6.7}$$

Fie $H_m(z)$ funcția de transfer de la intrare la nodul *m*. Pentru a nu avea depăşire în nodul *m* trebuie îndeplinită condiția:

$$|y_m(n)| \le 1 \tag{6.8}$$

Avem:

$$|y_m(n)| = \left|\sum_{k=0}^{\infty} h_m(k)x(n-k)\right| \le \sum_{k=0}^{\infty} |h_m(k)||x(n-k)| \le \sum_{k=0}^{\infty} |h_m(k)| = k_m \tag{6.9}$$

Nodurile în care pot apare depăşiri sunt în partea recursivă a structurii şi, bineînțeles, la ieşirea filtrului. Scalarea este necesară dacă cel puțin unul din coeficienții k_m este supraunitar. Coeficientul cu care se face scalarea la intrarea structurii va fi:

$$s \le \min\left(\frac{1}{k_1}, \frac{1}{k_2}, \ldots, \frac{1}{k_m}\right) \tag{6.10}$$

În cazul în care intrarea este zero sau o constantă, există posibilitatea ca ieşirea filtrului, în loc de o apropiere asimptotică de o valoarea constantă, să oscileze cu amplitudini relativ mici. Aceste oscilaţii sunt numite cicluri limită. Aceştia pot fi reduşi la amplitudini acceptabile crescând numărul de biţi de reprezentare cu virgulă fixă a semnalului.

Oscilațiile în depăşiri sunt produse de caracteristica circulară a sumatorului în complement față de 1 sau față de 2. Aceste oscilații pot produce depăşiri ale gamei dinamice a filtrului IIR. Soluția acestei probleme este folosirea aritmeticii cu saturație la adunare ceea ce limitează rezultatul adunării în modul subunitar.

6.2.1 Forma directă 1

Scriind relația intrare-ieşire sub forma:

$$W(z) = X(z) \cdot \sum_{k=0}^{M} b_k z^{-k} \tag{6.11}$$

$$Y(z) = \frac{W(z)}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{6.12}$$

Rezultă ecuațiile cu diferențe finite:

$$w(n) = \sum_{k=0}^{M} b_k z^{-k} \tag{6.13}$$

$$y(n) = w(n) - \sum_{k=1}^{N} a_k y(n-k) \tag{6.14}$$

Astfel, pentru un filtru de ordin N rezultă schema de mai jos:

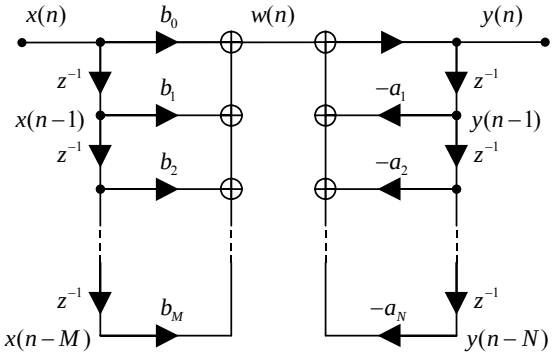

Figura 5.3. Filtru IIR forma directă 1

Evident, pentru calculul lui $w(n)$ se foloseşte programul pentru implementarea unui filtru FIR.

Pentru secțiunea recursivă se va implementa o a doua linie de întârziere pentru stocarea ieşirilor anterioare ale filtrului.

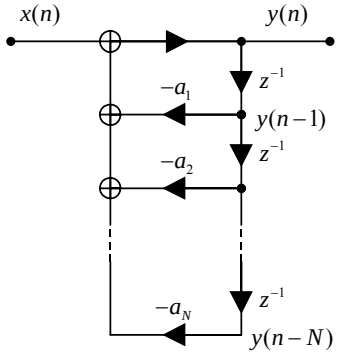

Figura 5.7. Filtru IIR de ordin N numai cu poli

```
int main()
{
Word16 x[DataBlockSize],y[DataBlockSize];
Word16 delay_b[M+1],delay_a[N+1],b[M+1],a[N];
Word32 sum;
short n,i;
long unsigned k,ix,iy;

...

fread(x,sizeof(Word16),DataBlockSize,fpx);

for (n=0; n<DataBlockSize; n++)
{
     delay_b[ix%(M+1)]=x[n];
     sum=0;
     for (k=0; k<=M; k++)
     {
          sum=L_mac(sum,b[k],delay_b[(ix-k)%(M+1)]);
     }
     for (k=1; k<=N; k++)
     {
          sum=L_msu(sum,a[k],delay_a[(iy-k)%(N+1)]);
     }
     y[n]=round(sum);
     delay_b[iy%(N+1)]=y[n];

     ix++;
     iy++;
}

fwrite(y,sizeof(Word16),DataBlockSize,fpy);

...

}
```


Să considerăm acum implementarea unui filtru de ordin 2 numai cu poli, cu funcția de transfer de forma:

$$H(z)=\frac{1}{1+a_1z^{-1}+a_2z^{-2}} \tag{6.15}$$

Ieșirea va fi calculată cu formula:

$$y(n)=x(n)-a_1\cdot y(n-1)-a_2\cdot y(n-2) \tag{6.16}$$

Pentru un filtru de ordin 2 nu este necesară implementarea cu liste circulare, stările filtrului putând fi stocate în două variabile de stare. Coeficienții a_1 și a_2 se consideră de modul subunitar.

Figura 5.4. Organizarea memoriei pentru un filtru de ordin 2 numai cu poli

```
Word16 x[DataBlockSize],y[DataBlockSize];
Word16 y1,y2,a[]={WORD16(a1), WORD16(a2)};
Word32 sum;

...

fread(x,sizeof(Word16),DataBlockSize,fpx);

for (n=0; n<DataBlockSize; n++)
{
     sum=L_deposit_h(x[n]);
     sum=L_msu(sum,a[0],y1);
     y[n]=msu_r(sum,a[1],y2);
     y2=y1;
     y1=y[n];
}

fwrite(y,sizeof(Word16),DataBlockSize,fpy);

...
```

În exemplul anterior am considerat coeficienții numitorului de modul subunitar. Fie un filtru de ordin 2 stabil. Polii funcției de transfer sunt în cercul de rază unitate, deci de modul subunitar.

Avem:

$$H(z)=\frac{1}{1+a_1z^{-1}+a_2z^{-2}}=\frac{1}{(1-p_1z^{-1})(1-p_2z^{-2})} \tag{6.17}$$

Folosind relațiile între coeficienții şi rădăcinile polinomului de la numitor obținem:

$$\begin{aligned}|a_1|&=|p_1+p_2|\le|p_1|+|p_2|\le 2\\ |a_2|&=|p_1||p_2|\le 1\end{aligned} \tag{6.18}$$

Aceasta înseamnă că, în cazul unui filtru de ordin 2 coeficienții numitorului sunt de modul mai mic ca 2. Pentru a-i putea reprezenta în virgulă fixă, în complement față de 2, este necesară scalarea coeficienților cu 2. Structura echivalentă este prezentată mai jos:

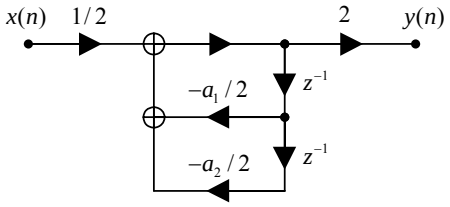

Figura 5.5. Filtru IIR de ordin 2 numai cu poli cu coeficienții scalați

Ecuația cu diferențe finite devine:

$$y(n)=2\left[\frac{x(n)}{2}-\frac{a_1}{2}y(n-1)-\frac{a_2}{2}y(n-2)\right] \tag{6.19}$$

În exemplul următor este implementat un filtru IIR de ordin 2, numai cu poli, cu coeficienții scalați cu 2. Organizarea memoriei:

Figura 5.6. Organizarea memoriei pentru un filtru de ordin 2 numai cu poli cu coeficienții scalați

```
for (n=0; n<DataBlockSize; n++)
{
     sum = L_deposit_h(shr(x[n],1));
     sum = L_msu(sum,a[0],y1);
     y[n]= shl(msu_r(sum,a[1],y2),1);
     y2 = y1;
     y1 = y[n];
}
```

6.2.2 Forma directă 2

Relația intrare-ieşire se scrie:

$$W(z)=\frac{X(z)}{1+\sum_{k=1}^{N}a_kz^{-k}} \tag{6.20}$$

$$Y(z)=W(z)\sum_{k=0}^{M}b_kz^{-k} \tag{6.21}$$

De unde rezultă ecuațiile cu diferențe finite:

$$w(n)=x(n)-\sum_{k=0}^{N}a_kw(n-k) \tag{6.22}$$

$$y(n)=\sum_{k=0}^{M}b_kw(n-k) \tag{6.23}$$

Aceste relații conduc la structura din figura următoare (cunoscută şi cu denumirea de forma canonică):

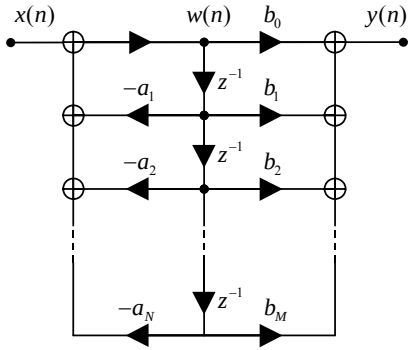

Figura 5.9. Filtru IIR forma directă 2

Se observă pentru această structură o reducere a numărului celulelor de întârziere față de forma directă 1, deci o reducere a memoriei necesare.

Să considerăm pentru început un filtru de ordin 2 cu $b_0 = 1$. Funcția de transfer poate fi scrisă:

$$H(z) = \frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{6.24}$$

Filtrul implementat în forma directă 2 are structura de mai jos şi se mai numeşte şi filtru biquad:

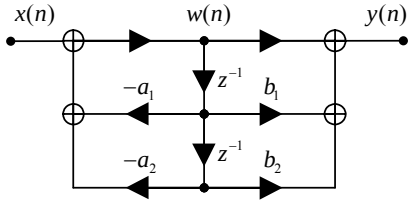

Figura 5.10. Filtru IIR biquad


```
Word16 x[DataBlockSize],y[DataBlockSize];
Word16 w1,w2;
Word16 a[]={WORD16(a1), WORD16(a2)};
Word16 b[]={WORD16(b1), WORD16(b2)};

Word32 sum;

...

fread(x,sizeof(Word16),DataBlockSize,fpx);

for (n=0; n<DataBlockSize; n++)
{
     sum = L_deposit_h(x[n]);
     sum = L_msu(sum,a[0],w1);
     sum = L_msu(sum,a[1],w2);
     w = round(sum);
     sum = L_mac(sum,b[0],w1);
     y[n]= mac_r(sum,b[1],w2);
     w2=w1;
     w1=w;
}

fwrite(y,sizeof(Word16),DataBlockSize,fpy);

...
```

Trebuie menționat că în cazul unei funcții de transfer cu $b_0 \neq 1$ se poate face o normare a coeficienților numărătorului cu b_0, normare care poate fi compensată la ieşire prin înmulțirea cu o constantă corespunzătoare.

Ca şi în cazul formei directe 1, pentru coeficienți mai mici ca 2 în modul poate fi implementat un program cu scalare a coeficienților cu 2 pornind de la ecuația cu diferențe finite:

$$w(n) = 2\left[\frac{x(n)}{2} - \frac{a_1}{2} w(n-1) - \frac{a_2}{2} w(n-2)\right] \tag{6.25}$$

$$y(n) = 2\left[\frac{w(n)}{2} + \frac{b_1}{2} w(n-1) + \frac{b_2}{2} w(n-2)\right] \tag{6.26}$$

şi având structura:

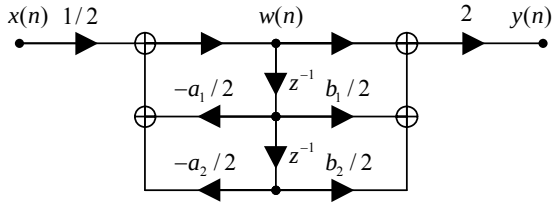

Figura 5.12. Filtru biquad cu coeficienții scalați

Coeficienții sunt stocați in vectorii **a** şi **b** împărțiți la 2. Valorile intermediare $w(n)$ trebuie stocate în memorie nescalate, adică rezultatul sumatorului de la intrare trebuie înmulțit cu 2 înainte de a fi stocat în celula de memorie.

```
for (n=0; n<DataBlockSize; n++)
{
     sum = L_deposit_h(shr(x[n],1));
     sum = L_msu(sum,a[0],w1);
     sum = L_msu(sum,a[1],w2);
     w = round(shl(sum,1));
     sum = L_mac(sum,b[0],w1);
     y[n]= shl(mac_r(sum,b[1],w2),1);
     w2=w1;
     w1=w;
}
```

6.2.3 Realizarea în cascadă

Se realizează factorizarea funcției de transfer în funcții de ordin 2 cu coeficienți reali implementate cu structuri biquad:

$$H(z)=\prod_{k=1}^{p}\frac{1+b_{k,1}z^{-1}+b_{k,2}z^{-2}}{1+a_{k,1}z^{-1}+a_{k,2}z^{-2}}=\prod_{k=1}^{p}H_k(z) \qquad (6.27)$$

Coeficienții secțiunilor de ordin 2 au garantat modulul mai mic decât 2 dacă filtrul este stabil şi de fază minimă.

Forma cascadă pentru două structuri biquad este prezentată mai jos:

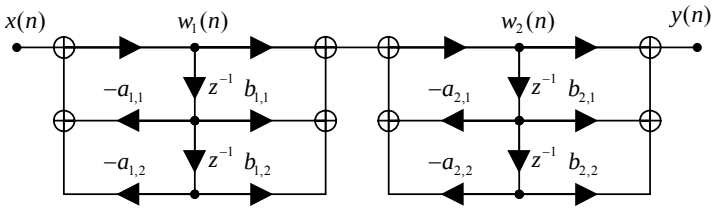

Figura 5.14. Forma cascadă cu structuri biquad

```
for (n=0; n<DataBlockSize; n++)
{
  sum = L_deposit_h(x[n]);

  for (k=0; k<NrSect; k++)
  {
     sum = L_msu(sum,a[k][0],w1[k]);
     sum = L_msu(sum,a[k][1],w2[k]);
     w = round(sum);
     sum = L_mac(sum,b[k][0],w1[k]);
     sum = L_mac(sum,b[k][1],w2[k]);
     w2[k] = w1[k];
     w1[k] = w;
  }
  y[n] = round(sum);
}
```

6.2.4 Realizarea în paralel

În acest caz se pornește de la descompunerea în fracții simple a funcției $H(z)$. Funcția de transfer poate fi scrisă sub forma:

$$H(z)=C+\sum_{p=1}^{P}\frac{b_{0,p}+b_{1,p}z^{-1}}{1+a_{1,p}z^{-1}+a_{2,p}z^{-2}}=C+\sum_{p=1}^{P}H_p(z) \qquad (6.28)$$

Se obține schema din figura 5.15.

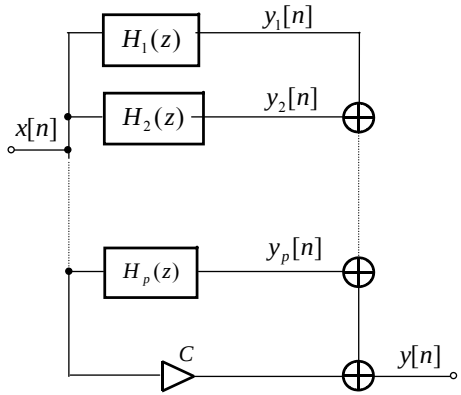

Figura 5.14. Realizarea în paralel

```
for (n=0; n<DataBlockSize; n++)
{
  y[n] = 0;

  for (p=0; p<NrSect; p++)
  {
     sum = L_deposit_h(x[n]);
     sum = L_msu(sum,a[p][0],w1[k]);
     w = msu_r(sum,a[p][1],w2[k]);

     sum = L_mult(b[p][0],w);
     sum = L_mac(sum,b[p][1],w1[k]);
     y[n] = add(round(sum),y[n]);
     w2[k] = w1[k];
     w1[k] = w;
  }
}
```