(19) SUOMI - FINLAND
(FI)

PATENTTI- JA REKISTERIHALLITUS
PATENT- OCH REGISTERSTYRELSEN
FINNISH PATENT AND REGISTRATION OFFICE

(10) **FI 934063 A7**

(12) **JULKISEKSI TULLUT PATENTTIHAKEMUS**
PATENTANSÖKAN SOM BLIVIT OFFENTLIG
PATENT APPLICATION MADE AVAILABLE TO THE PUBLIC

(21) Patenttihakemus - Patentansökan - Patent application 934063

(51) Kansainvälinen patenttiluokitus - Internationell patentklassifikation - International patent classification (IPC^5)
G10L 9/14

(22) Tekemispäivä - Ingivningsdag - Filing date **19.01.1993**

(23) Saapumispäivä - Ankomstdag - Reception date **16.09.1993**

(41) Tullut julkiseksi - Blivit offentlig - Available to the public **16.09.1993**

(43) Julkaisupäivä - Publiceringsdag - Publication date **13.06.2019**

(86) Kansainvälinen hakemus - Internationell ansökan - International application **19.01.1993 PCT/SE1993/000024**

(32) (33) (31) Etuoikeus - Prioritet - Priority

27.01.1992 SE 9200217

(71) Hakija - Sökande - Applicant

1 • Telefonaktiebolaget L M Ericsson, 126 25 Stockholm, SVERIGE, (SE)

(72) Keksijä - Uppfinnare - Inventor

1 • Minde, Tor Björn, Gammelstad, SVERIGE, (SE)

(74) Asiamies - Ombud - Agent

Kolster Oy Ab, Salmisaarenaukio 1, 00180 Helsinki

(54) Keksinnön nimitys - Uppfinningens benämning - Title of the invention
Näytejonomuotoisen puhesignaalivektorin koodaustapa
Sätt att koda en samplad talsignalvektor

Näytejonomuotoisen puhesignaalivektorin koodaustapa

Tekniikan ala

Tämä keksintö liittyy näytejonomuotoisen puhesignaalivektorin koodaustapaan synteesin kautta tapahtuvan analyysimenettelyn avulla muodostamalla optimaalinen viritysvektori, joka muodostuu kiinteän koodikirjan koodivektorin ja pitkäaikaisennustajavektorin lineaarisesta yhdistelmästä.

Tekniikan taso

Tunnetaan pitkäaikaisennustajan, jota myöskin sanotaan "pitch predictoriksi" tai avoimeksi koodikirjaksi, määrääminen puhekoodausyksikössä niin sanotun suljetun silmukka-analyysin avulla (closed loop analysis) (W. Kleijn, D. Krasinski, R. Ketchum "Improved Speech Quality and Efficient Vector Quantization in SELP" (SELP:issä toteutettu parannettu puheenlaatu ja tehokas vektorikvantisointi), New York, 1988 ja P. Kabal, J. Moncet, C. Chu "Synthesis Filter Optimization and Coding: Application to CELP" (Syntetisointisuotimien optimointi ja koodaus: soveltaminen CELP:iin), IEE ICASSP-88, New York, 1988). Tämän voi toteuttaa esimerkiksi CELP-tyyppisessä (CELP = Code Excited Linear Predictive Coder, Koodiviritteinen lineaarinen ennustuskoodausyksikkö) koodausyksikössä. Tämäntyyppisessä analyysissa verrataan todellista puhesignaalivektoria arvioituun vektoriin, joka muodostetaan virittämällä synteesisuodin viritysvektorilla, joka käsittää aikaisemmin määrättyjen viritysvektoreiden näytteitä.

Pitkäaikaisennustajan määrääminen niin sanotun avoimen silmukka-analyysin (open loop analysis) avulla on myöskin tunnettu (R. Ramachandran, P. Kabal "Pitch Prediction Filters in Speech Coding" (Pitkäaikaisennustussuotimet puhekoodauksessa), IEEE Trans. ASSP Vol. 37, No. 4, huhtikuu 1989), jolloin koodattavaa puhesignaalivektoria

verrataan viivästettyihin puhesignaalivektoreihin puhesignaalin jaksollisuusominaisuuksien arviointia varten.

CELP-puhekoodausyksikön periaate perustuu LPC-synteesisuotimen (LPC = Linear Predictive Coding, Lineaarinen ennustuskoodaus) viritykseen pitkäaikaisennustajavektorin ja jonkintyyppisen kiinteän koodikirjan vektorin yhdistelmän avulla. Synteesisuotimen ulostulosignaalin tulee olla mahdollisimman yhdenmukainen koodattavan puhesignaalivektorin kanssa. Synteesisuotimen parametrit päivitetään jokaista uutta puhesignaalivektoria kohti, se on menettely perustuu kehyksiin. Tämä kehyksiin perustuva päivitys ei kuitenkaan aina ole riittävä pitkäaikaisennustajavektoria varten. Puhesignaalin muutosten seuraamiseksi erityisesti korkeilla perusäänitaajuuksilla (pitch) pitkäaikaisennustajavektori tulee päivittää useammin kuin kehyskohtaisesti. Tämän vuoksi se päivitetään usein alikehyskohtaisesti, jolloin alikehys voi olla esimerkiksi 1/4-kehys.

On osoittautunut että suljettu silmukka-analyysi tuottaa erittäin hyvän suorituskyvyn lyhyiden alikehysten tapauksissa, mutta että suorituskyky nopeasti huononee alikehysten pidentyessä.

Avoimen silmukka-analyysin suorituskyky ovat huonommat kuin suljetun silmukka-analyysin suorituskyky lyhyiden alikehysten tapauksissa, mutta sen suorituskyky ovat paremmat kuin suljetun silmukka-analyysin suorituskyky pitkien alikehysten tapauksissa. Suorituskyky pitkien alikehysten tapauksissa on verrattavissa suljetun silmukka-analyysin suorituskykyyn lyhyiden alikehysten tapauksissa, mutta ei ole aivan yhtä hyvä kuin tämä.

Syy siihen että halutaan mahdollisimman pitkät alikehykset, siitä huolimatta että lyhyet alikehykset parhaiten seuraisivat muutoksia, on se että lyhyet alikehykset merkitsevät tiheämpiä päivityksiä, mikä lisääntyneen kompleksiteetin lisäksi merkitsee suurempaa bittinopeutta koodatun signaalin siirrossa.

Tämä keksintö liittyy täten ongelmaan saavuttaa parempi suorituskyky pitempien alikehysten tapauksissa. Tämä ongelma käsittää koodausyksikön rakenteen ja analyysimenetelmän valinnat sellaisen suorituskyvyn saavuttamiseksi, joka on verrattavissa suljetun silmukka-analyysin suorituskykyyn lyhyiden alikehysten tapauksissa.

Eräs menetelmä suorituskyvyn parantamiseksi on täydellisen haun suorittaminen käsittäen kaikki pitkäaikaisennustajavektoreiden ja kiinteän koodikirjan vektoreiden yhdistelmät. Tämä tuottaisi sen yhdistelmän, joka parhaiten sopii jokaisen annetun alikehyksen puhesignaalivektoriin. Täten aikaansaatu kompleksiteetti olisi kuitenkin mahdoton toteuttaa nykytason digitaalisten signaaliprosessoreiden avulla.

Keksinnön selostus

Tämän keksinnön eräänä tarkoituksena on täten tarjota optimaalisempi näytejonomuotoisen puhesignaalivektorin koodaustapa pitempienkin alikehysten tapauksissa kompleksiteetin lisääntymättä olennaisesti.

Tämä tarkoitus toteutetaan keksinnössä

(a) muodostamalla pitkäaikaisennustajavektorin ensimmäinen arvio avoimen silmukka-analyysin avulla;

(b) muodostamalla pitkäaikaisennustajavektorin toinen arvio suljetun silmukka-analyysin avulla; ja

(c) yhdistämällä lineaarisesti täydellisen haun aikana jokainen ensimmäinen ja toinen arvio kiinteän koodikirjan kaikkiin koodivektoreihin sen viritysvektorin muodostamiseksi, joka tuottaa puhesignaalivektorin parhaan koodauksen.

Kuvioluettelo

Keksintö, keksinnön lisätarkoitukset sekä keksinnön avulla saavutetut edut ovat parhaiten ymmärrettävissä viitaten seuraavaan selostukseen ja liitteenä olevaan piirustukseen, jossa:

kuviossa 1 on esitetty suljetun silmukka-analyysin käsittävän tunnetun puhekoodausyksikön rakenne;

kuviossa 2 on esitetty suljetun silmukka-analyysin käsittävän toisen tunnetun puhekoodausyksikön rakenne;

kuviossa 3 on esitetty puhekoodausyksikön avoimen silmukka-analyysin tunnettu rakenne; ja

kuviossa 4 on esitetty puhekoodausyksikön tämän kirjoituksen mukainen rakenne keksinnön mukaisen menettelyn toteuttamiseksi.

Esitetty suoritusmuoto

Piirustuksen eri kuvioissa on kauttaaltaan käytetty samoja viitteitä vastaavissa elementeissä.

Kuviossa 1 on esitetty suljetun silmukka-analyysin käsittävän tunnetun puhekoodausyksikön rakenne. Koodausyksikkö käsittää katkoviivalla piirretyn pystysuoran keskiviivan vasemmalla puolella olevan synteesiosan. Tämä synteesiosa käsittää olennaisesti kolme osaa, se on adaptiivisen koodikirjan 10, kiinteän koodikirjan 12 ja LPC-synteesisuotimen 16. Adaptiivisesta koodikirjasta 10 valittu vektori kerrotaan vahvistuskertoimella g_I signaalin p(n) muodostamiseksi. Samalla tavalla kiinteästä koodikirjasta valittu vektori kerrotaan vahvistuskertoimella g_J signaalin f(n) muodostamiseksi. Signaalit p(n) ja f(n) lasketaan yhteen yhteenlaskijassa 14 viritysvektorin ex(n) muodostamiseksi, joka virittää synteesisuotimen 16 arvioidun puhesignaalivektorin s(n) muodostamiseksi.

Arvioitu vektori vähennetään todellisesta puhesignaalivektorista s(n) kuvion 1 oikeassa osassa eli analyysiosassa sijaitsevassa yhteenlaskijassa 20 virhesignaalin e(n) muodostamiseksi. Tämä virhesignaali johdetaan painotussuotimen 22 kautta painotetun virhesignaalin $e_w(n)$ muodostamiseksi. Tämän painotetun virhevektorin komponentit neliöidään ja lasketaan yhteen yksikössä 24 painotetun virhevektorin energiamäärän mitan muodostamiseksi.

Tarkoituksena on nyt minimoida tämä energia, se on valita se adaptiivisen koodikirjan 10 vektorin ja sen vahvistuksen g_I sekä kiinteän koodikirjan 12 sen vektorin ja sen vahvistuksen g_J yhdistelmä, joka antaa pienimmän energia-arvon, se on joka suotimessa 16 tapahtuvan suodatuksen jälkeen mahdollisimman yhdenmukainen puhesignaalivektorin s(n) kanssa. Tämä optimointi jaetaan kahteen vaiheeseen. Ensimmäisessä vaiheessa oletetaan että f(n) = 0 ja määrätään adaptiivisen koodikirjan 10 paras vektori sekä g_I. Näiden parametrien määrittelemisen jälkeen määrätään se vektori ja se vahvistuskerroin g_J, jotka yhdessä äsken valittujen parametrien kanssa minimoivat energian (menetelmää kutsutaan joskus "one at a time"-menetelmäksi).

Adaptiivisen koodikirjan 10 paras indeksi I ja vahvistus g_I lasketaan seuraavien kaavojen avulla:

$ex(n) = p(n)$	Viritysvektori (f(n) = 0)
$p(n) = g_i \cdot a_i(n)$	Skaalattu adaptiivinen koodikirjavektori
$\hat{s}(n) = h(n) * p(n)$	Synteettinen luku (* = konvoluutio)
$e(n) = s(n) - \hat{s}(n)$	Virhevektori
$e_w(n) = w(n)*(s(n)-\hat{s}(n))$	Painotettu virhe
$E = \Sigma[e_w(n)]^2$ n=0..N-1	Neliöity painotettu virhe
N = 40 (esimerkiksi)	Vektorin pituus
$s_w(n) = w(n)*s(n)$	Painotettu luku
$h_w(n) = w(n)*h(n)$	Synteesisuotimen painotettu impulssivaste
$\min E_i = \min \sum_{n=0}^{N-1} [e_{wi}(n)]^2$	Optimaalisen indeksin haku adaptiivisesta koodikirjasta
$\frac{\delta E_i}{\delta g_i} = 0 \Rightarrow g_i = \frac{\sum_{n=0}^{N-1} s_w(n) \cdot a_i(n) * h_w(n)}{\sum_{n=0}^{N-1} [\hat{s}_{wi}(n)]^2}$	Indeksin i vahvistus

Suotimen 16 suodinparametrit päivitetään jokaista puhesignaalikehystä kohti analysoimalla puhesignaalikehys LPC-analysaattorissa 18. Päivitys on merkitty analysaattorin 18 ja suotimen 16 välisellä katkoviivayhteydellä. Samoin on merkitty katkoviiva yksikön 24 ja viiveosan 26 väliin. Tämä yhteys symbolisoi adaptiivisen koodikirjan 10 päivitys lopuksi valitulla viritysvektorilla ex(n).

Kuviossa 2 on esitetty suljetun silmukka-analyysin käsittävän toisen tunnetun puhekoodausyksikön rakenne. Kuvion 2 oikeanpuoleinen analyysiosa on identtinen kuvion 1 analyysiosan kanssa. Sen sijaan synteesiosa eroaa kuvion 1 synteesiosasta siten että adaptiivinen koodikirja 10 ja vahvistuselementti g_I on korvattu takaisinkytkentäsilmukalla, joka käsittää viiveosan 28 ja vahvistuselementin g_L sisältävän suotimen. Koska adaptiivisen koodikirjan vektorit sijaitsevat näytteenottovälin välein, se on peräkkäiset vektorit eroavat toisistaan vain ensimmäisen ja viimeisen komponentin osalta, on osoitettavissa että kuvion 2 suodinrakenne vastaa kuvion 1 adaptiivisen koodikirjan suodinrakennetta kunhan viive L ei alita vektorinpituutta N.

Viiveen (lag) L alittaessa vektorinpituuden N saadaan kuvion 1 adaptiiviselle koodikirjalle:

$v_p(n)$ n=-Maxlag...-1 Pitkäaikaismuisti (adaptiivinen koodikirja)

$$v(n) = \begin{cases} v_p(n) & n=0...L-1 \\ 0 & n=L...N-1 \end{cases}$$ Vektorin muodostaminen

$v(n) = v(n-L)$ n=L...N-1 Syklinen toisto

se on pituuden N omaava adaptiivinen koodikirjakvektori muodostetaan toistamalla syklisesti komponentit 0...L-1. Lisäksi pätee:

$$p(n) = g_I \cdot v(n) \qquad n=0...N-1$$
$$ex(n) = p(n) + f(n) \quad n=0...N-1$$

jossa viritysvektori ex(n) muodostetaan yhdistelemällä lineaariseti adaptiivinen koodikirjavektori ja kiinteä koodikirjavektori.

Viiveen (lag) L alittaessa vektorinpituuden N pätee kuvion 2 suodinrakenteelle:

$$v(n) = g_L \cdot v(n-L) + f(n) \qquad n=0...L-1$$
$$v(n) = g_L^2 \cdot v(n-2L) + g_L \cdot f(n-L) + f(n) \qquad n=L...N-1$$
$$ex(n) = v(n)$$

se on viritysvektori ex(n) muodostetaan suodattamalla kiinteä koodikirjavektori vektorirakenteen g_L, 28 läpi.

Sekä kuvion 1 että kuvion 2 rakenne perustuvat todellisen signaalivektorin s(n) vertailuun arvioituun signaalivektoriin $\bar{s}(n)$ ja painotetun neliöidyn virheen minimointiin pitkäaikaisennustajavektoria määrättäessä.

Toinen tapa pitkäaikaisennustajavektorin arvioimiseksi on todellisen puhesignaalivektorin s(n) vertailu sen (avoin silmukka-analyysi) viivästettyihin versioihin mahdollisen jaksollisuuden havaitsemiseksi, mitä alla sanotaan viivästämiseksi (pitch lag). Kuviossa 3 on esitetty tällaisen rakenteen analyysiosan esimerkki. Puhesignaali s(n) painotetaan suotimessa 22 ja suotimen 22 ulostulosignaali $s_w(n)$ johdetaan osin suoraan, osin viivesuotimen 30 ja vahvistuskertoimen g_1 sisältävän viivesilmukan kautta yhteenlaskijaan 32, joka muodostaa painotetun signaalin ja viivästetyn signaalin erotuksen. Erotussignaali $e_w(n)$ johdetaan sen jälkeen yksikköön 24, joka neliöi ja laskee yhteen komponentit.

Optimaalinen viive L ja vahvistus g_L lasketaan seuraavasti:

$e_w(n) = s_w(n) - g_1 \cdot s_w(n-1)$ Painotettu virhevektori

$E = \Sigma[e_w(n)]^2 \quad n=0...N-1$ Neliöity painotettu virhe

$$\min E_1 = \min \sum_{n=0}^{N-1} [s_w(n)-g_1 \cdot s_w(n-1)]^2$$ Optimaalisen viiveen 1 haku

$$\frac{\delta E_1}{\delta g_1} = 0 \Rightarrow g_1 = \frac{\sum_{n=0}^{N-1} s_w(n) \cdot s_w(n-1)}{\sum_{n=0}^{N-1} [s_w(n-1)]^2}$$ Viivettä 1 vastaava vahvistus

Kuvion 2 mukaisen suodinrakenteen suljettu silmukka-analyysi eroaa kuvatusta kuvion 1 adaptiivisen koodikirjan suljetusta silmukka-analyysista viiveen L alittaessa vektorinpituuden N.

Adaptiivisen koodikirjan vahvistuskerroin saatiin ratkaisemalla ensimmäisen asteen yhtälö. Suodinrakenteen vahvistuskerroin saatiin ratkaisemalla korkeampien asteiden yhtälöitä (P. Kabal, J. Moncet, C. Chu "Synthesis Filter Optimization and Coding: Application to CELP", IEE ICASSP-88, New York, 1988).

Välin N/2<L<N viiveelle ehdon f(n)=0 ollessa voimassa pätee yhtälö:

$$ex(n) = \begin{cases} g_L v(n-L) & n=0...L-1 \\ g_L^2 v(n-2L) & n=L...N-1 \end{cases}$$

kuvion 2 viritykselle ex(n). Tämä viritys suodatetaan sen jälkeen synteesisuotimen 16 läpi, joka tuottaa seuraaviin komponentteihin jaetun synteettisen signaalin:

$$\hat{s}(n) = \hat{s}_L(n) = g_L \cdot h(n) * v(n-L) \quad n=0...L-1$$
$$\hat{s}(n) = \hat{s}_L(n) + s_{2L}(n) \quad n=L...N-1$$
$$\hat{s}_{2L}(n) = g_L^2 \cdot h(n) * v(n-2L) \quad n=L...N-1$$

Neliöity painotettu virhe on kirjoitettavissa seuraavasti:

$$E_L = \sum_{n=0}^{N-1} [e_{wL}(n)]^2$$

Tässä $e_{wL}(n)$ on määritelty seuraavasti:

$e_{wL}(n) = [s_w(n) - \hat{s}_w(n)]$	Painotettu virhevektori
$s_w(n) = w(n)*s(n)$	Painotettu luku
$\hat{s}_w(n) = h_w(n)*\hat{s}(n)$	Painotettu synteettinen signaali
$h_w(n) = w(n)*h(n)$	Synteesisuotimen painotettu impulssivaste

Optimaalinen viive (lag) saadaan seuraavasti:

$$\min E_L = \min \sum_{n=0}^{N-1} [e_{wL}(n)]^2$$

Neliöity painotettu virhe on nyt kehitettävissä seuraavasti:

$$E_L = \sum_{n=0}^{N-1} |s_w(n)|^2 - 2g_L \sum_{n=0}^{N-1} s_w(n)\hat{s}_{wL}(n)$$

$$+ g_L^2 \sum_{n=0}^{N-1} |\hat{s}_{wL}|^2 - 2g_L^2 \sum_{n=L}^{N-1} s_w(n)\hat{s}_{w2L}(n)$$

$$+ 2g_L^3 \sum_{n=L}^{N-1} \hat{s}_{wL}(n)\hat{s}_{w2L}(n) + g_L^4 \sum_{n=L}^{N-1} |\hat{s}_{w2L}(n)|^2$$

Ehto

$$\frac{\delta E_L}{\delta g_L} = 0$$

johtaa kolmannen asteen yhtälöön vahvistukselle g_L.

Tämän hakustrategian kompleksiteetin pienentämiseksi voidaan käyttää menetelmää (P. Kabal, J. Moncet, C. Chu "Synthesis Filter Optimization and Coding: Application to CELP", IEE ICASSP-88, New York, 1988), jossa suljettu silmukka-analyysi käsittää kvantisointia. Tässä menetelmässä kvantisoituja vahvistuskertoimia käytetään neliövirheen arviointiin. Mentelmä voidaan jokaista hakuviivettä kohti

kiteyttää seuraavasti: Ensin lasketaan neliövirheen kaikki summakomponentit. Sen jälkeen kokeillaan kaikki g_L:n kvantisointiarvot E_L:n yhtälössä. Lopuksi valitaan se g_L:n arvo, joka tuottaa pienimmän neliövirheen. Käytettäessä pientä kvantisointiarvojen määrää, tyyppiesimerkissä 8 - 16 arvoa vastaten 3 - 4 bitin kvantisointia, tämä menetelmä johtaa huomattavasti pienempään kompleksiteettiin kuin yritettäessä ratkaista yhtälöt suljetussa muodossa.

Kuvion 3 analyysirakenteen kanssa käytettävänä synteesiosana voidaan keksinnön selostetussa suoritusmuodossa käyttää kuvion 2 mukaisen rakenteen vasenta osaa eli synteesiosaa. Tässä keksinnössä tämä on hyödynnetty kuvion 4 rakenteen aikaansaamiseksi.

Kuvion 4 vasen osa eli synteesiosa on identtinen kuvion 2 synteesiosan kanssa. Kuvion 4 oikeassa osassa eli analyysiosassa kuvion 2 oikea osa on yhdistetty kuvion 3 rakenteeseen.

Keksinnön mukaisessa menettelyssä määrätään ensin pitkäaikaisennustajavektorin arvio osin suljetulla silmukka-analyysilla, osin avoimella silmukka-analyysilla. Koska nämä kaksi arviota eivät ole suoraan verrattavissa toisiinsa (toinen arvio vertaa todellista signaalia arvioituun signaaliin, toisen arvion verratessa todellista signaalia sen viivästettyyn versioon). Koodausparametrien lopullista määräämistä varten suoritetaan tämän takia kiinteän koodikirjan 12 täydellinen haku jokaista mainittua arviota kohti. Näiden hakujen tulokset ovat nyt suoraan verrattavissa toisiinsa, koska todellista puhesignaalia kummassakin tapauksessa on verrattu arvioituun signaaliin. Koodauksen perustaksi tulee nyt se arvio, joka antoi parhaan tuloksen, se on pienimmän neliövirheen.

Kuvioon 4 on piirretty kaksi kaavamaista vaihtokytkintä 34 ja 36 tämän menettelyn ilmaisemiseksi.

Ensimmäisessä määräämisvaiheessa avataan vaihtokytkin 36 "maayhteyden" (nollasignaalin) aikaansaamiseksi,

siten että ainoastaan todellinen puhesignaali s(n) pääsee painotussuotimeen 22. Samanaikaisesti suljetaan vaihtokytkin 34 siten että avoin silmukka-analyysi on suoritettavissa. Avoimen silmukka-analyysin jälkeen avataan vaihtokytkin 34 "maayhteyden" aikaansaamiseksi ja suljetaan vaihtokytkin 36, siten että suljettu silmukka-analyysi on suoritettavissa samalla tavalla kuin kuvion 2 rakenteessa.

Lopuksi suoritetaan kiinteän koodikirjan 12 haku jokaista saatua arviota kohti (asetetaan suotimen 28 ja vahvistuskertoimen g_L avulla). Se kiinteän koodikirjan vektorin, vahvistuskertoimen g_j ja pitkäaikaisennustajavektorin arvion yhdistelmä, joka tuotti parhaan tuloksen määrää koodausparametrit.

Edellisestä ilmenee että kohtuullisesti lisätty kompleksiteetti (pitkäaikaisennustajavektorin kaksinkertainen määrääminen ja kiinteän koodikirjan kaksinkertainen haku) mahdollistaa avoimen ja suljetun silmukka-analyysin parhaiden ominaisuuksien hyödyntämisen suorituskyvyn parantamiseksi pitkien alikehysten tapauksissa.

Pitkäaikaisennustajavektorin suorituskyvyn parantamiseksi edelleen voidaan käyttää korkeamman asteen pitkäaikaisennustajaa (R. Ramachandran, P. Kabal "Pitch Prediction Filters in Speech Coding", IEEE Trans. ASSP Vol. 37, No. 4, huhtikuu 1989; P. Kabal, J. Moncet, C. Chu "Synthesis Filter Optimization and Coding: Application to CELP", IEE ICASSP-88, New York, 1988) tai suuren erotuskyvyn pitkäaikaisennustajaa (P. Kroon, B. Atal "On the Use of Pitch Predictors with High Temporal Resolution" (Suuren aikaerotuskyvyn pitkäaikaisennustajien käytöstä), IEEE trans. SP. Vol. 39, No. 3, maaliskuu 1991).

Asteen p pitkäaikaisennustajan yleinen muoto voidaan esittää seuraavasti:

$$P(z) = 1 - \sum_{k=0}^{p-1} g(k)\, z^{-(M+k)}$$

jossa M on viive ja g(k):n arvot ovat ennustuskertoimet.

Suuren erotuskyvyn ennustajan tapauksessa viive voi saada suuremman erotuskyvyn arvoja, se on muita kuin kokonaislukuarvoja. Alipäästösuotimesta johdetuilla interpolaatiosuotimilla $p_l(k)$ (monivaihesuotimilla) saadaan:

$$p_l(k) = h(k \cdot D - l) \qquad l=0...D-1,\ k=0...q-1$$

jossa

l = erottelun eri osia vastaavien interpolaatiosuotimien numerot,

D = erotuskyvyn aste, se on $D \cdot f_s$ antaa interpolaatiosuotimen näytteenottotaajuuden,

q = interpolaatiosuotimen suodinkertoimien määrä.

Näiden suotimien avulla saadaan tehokas kokonaisluvuista poikkeava viive M + l/D. Pitkäakaisennustajan muoto on täten seuraava:

$$P(z) = 1 - g \sum_{k=0}^{q-1} p_l(k)\, z^{-(M-I+k)}$$

jossa g on alipäästösuotimen suodinkerroin ja I on alipäästösuotimen viive. Tätä pitkäaikaisennustajaa varten kanavalla lähetetään kvantisoitu g ja kokonaisluvuista poikkeava viive M + l/D.

Tämä keksintö merkitsee kahden pitkäaikaisennustajavektorin arvion muodostamista, toinen avoimella silmukka-analyysilla ja toinen suljetulla silmukka-analyysilla. Tämän vuoksi kompleksiteetin vähentäminen näissä arviomäärityksisssä olisi toivottavaa. Koska suljettu silmukka-analyysi on avointa silmukka-analyysia kompleksisempi, esitetty keksinnön suoritusmuoto perustuu siihen että avoimen silmukka-analyysin arviota voi käyttää suljettua silmukka-analyysia varten. Suljetussa silmukka-analyysissa esitetyn menetelmän mukainen haku suoritetaan vain avoimessa silmukka-analyysissa saadun viiveen L lähialueella,

tai sen monikertojen tai alimonikertojen lähialueilla. Täten kompleksiteettia voidaan vähentää, koska suljetussa silmukka-analyysissa ei suoriteta täydellistä hakua.

Keksinnön lisäseikkoja ilmenee liitteestä, joka sisältää keksinnön mukaisen menetelmän simuloivan PASCAL-ohjelman.

Ammattimies ymmärtää että keksinnön eri muutokset ja muunnokset ovat mahdollisia poikkeamatta keksinnön puitteista, jotka määräytyvät liitteenä olevista patenttivaatimuksista. On esimerkiksi myös mahdollista yhdistää kuvion 4 oikea osa eli analyysiosa kuvion 1 vasempaan osaan eli synteesiosaan. Tällaisessa suoritusmuodossa pitkäaikaisennustajavektorin molemmat arviot tallennetaan vuorotellen adaptiiviseen koodikirjaan kiinteän koodikirjan käsittävän haun aikana. Kiinteän koodikirjan haun päätyttyä jokaisen arvion osalta tallennetaan lopuksi parhaan koodauksen tuottava yhdistelmävektori adaptiiviseen koodikirjaan.

APPENDIX

```
{ DEFINITIONS }
{ Program definition }

program Transmitter(input,output) ;
{ -- }
{ Constant definitions }

const

  trunclength     = 20;             { length for synthesis filters }
  number_of_frames = 2000;
{ -- }
{ Type definitions }

type

  SF_Type          = ARRAY[0..79] of real ;     { Subframes      }
  CF_Type          = ARRAY[0..10] of real ;     { Filter coeffs  }
  FS_Type          = ARRAY[0..10] of real ;     { Filter states  }
  Win_Type         = ARRAY[0..379] of real;     { Input frames   }
  hist_type        = ARRAY[-160..-1] of real;   { ltp memory     }
  histSF_type      = ARRAY[-160..79] of real;   { ltp memory+sub }
  delay_type       = ARRAY[20..147] of real;    { error vectors  }
  out_type         = ARRAY[1..26] OF integer;   { output frames  }
{ -- }
{ Variable definitions }

{ General variables }

var
  i, k                          : integer ;

{ --- }
{ Segmentation variables }

  frame_nr, subframe_nr      : integer ;     { frame counters     }
  SpeechInbuf                : win_type;     { speech input frame }
```

```
    CodeOutbuf                  : out_type;      { code output frame  }
  { --- }
  { Filter Memorys }

    FS_zero_state               : FS_type;    { zeroed filter state    }
    FS_analys                   : FS_type;    { Analysis filter state  }
    FS_temp                     : FS_type;    { Temporary filter state }
    FS_Wsyntes                  : FS_type;    { synthesis filter state }
    FS_ringing                  : FS_type;    { saved filter state     }
  { --- }
  { Signal Subframes }

    Zero_subframe           : SF_type;      { zeroed subframe          }
    Original_Speech         : SF_type;      { Input speech             }
    Original_WSpeech        : SF_type;      { Input weighted speech    }
    Original_Residue        : SF_type;      { After LPC analys filter  }
    Weighted_excitation     : SF_type;      { Weighted synthesis excit }
    Weighted_speech1        : SF_type;      { After weighted synthes   }
    Weighted_speech2        : SF_type;      { After weighted synthes   }
    Ringing                 : SF_type;      { filter ringing           }
    Prediction1             : SF_type;      { pitch prediction model   }
    Prediction2             : SF_type;      { pitch prediction mode2   }
    Prediction              : SF_type;      { prediction from LTP      }
    Prediction_Syntes       : SF_type;      { Weighted synth from LTP  }
    Excitation1             : SF_type;      { excitation model         }
    Excitation2             : SF_type;      { excitation mode2         }
    Excitation              : SF_type;      { Exc from LTP and CB      }
    Weighted_Speech         : histSF_type;  { weighted synthes memory  }
  { --- }
  { Short term prediction varaibles }

    A_Coeff             : CF_type;    { A coef of synth filter }
    A_Coeffnew          : CF_type;    { A coef of new synth filter }
    A_Coeffold          : CF_type;    { A coef of old synth filter }
    A_W_Coeff           : CF_type;    { A coef of weigth synt  }
    H_W_syntes          : SF_type;    { Trunc impulse response }
  { --- }
  { LTP and Codebook decision variables }
```

```
    power                : real ;      { Power of tested vector      }
    corr                 : real ;      { Corr vector vs signal       }
    best_power1          : real ;      { Power of best vector so far }
    best_corr1           : real ;      { Corr  of best vector so far }
    best_power2          : real ;      { Power of best vector so far }
    best_corr2           : real ;      { Corr  of best vector so far }

    in_power             : real ;      { Power of signal             }
    best_error1          : real ;      { total error model           }
    best_error2          : real ;      { total error mode2           }
    mode                 : integer;    { mode decision               }
  { --- }
  { LTP variables }

    delay                : integer ;   { Delay of this vector        }
    upper                : integer ;   { Highest delay of subframe   }
    lower                : integer ;   { Lowest delay of subframe    }

    PP_gain1             : real ;      { gain of this vect model     }
    PP_gain2             : real ;      { gain of this vect mode2     }
    PP_delay             : integer ;   { Best delay in total search  }
    PP_gain_code         : integer ;   { Coded gain of best vector   }
    PP_best_error        : real ;      { Best error criterion search }

    gain                 : real ;      { gain of this vect           }
    gain_code            : integer ;   { Coded gain of this vector   }

    PP_gain_code1        : integer ;   { Coded gain model            }
    PP_gain_code2        : integer ;   { Coded gain mode2            }
    PP_delay1            : integer ;   { best delay model            }
    PP_delay2            : integer ;   { best delay mode2            }

    PP_history           : hist_type;  { LTP memory                  }
    PP_Overlap           : SF_type;    { ltp synthesis repetition    }
    Openpower            : delay_type; { vector of power             }
    Opencorrelation      : delay_type; { vector of correlations      }
  { --- }
  { Codebook variables }
```

```
    CB_gain_code        : integer; { Gain c for best vector          }
    CB_index            : integer; { Index for best vector           }
    CB_gain1            : real;    { Gain for best vector model      }
    CB_gain_code1       : integer; { Gain code for best vector model1}
    CB_index1           : integer; { Index for best vector model1    }
    CB_gain2            : real;    { Gain for best vector mode2      }
    CB_gain_code2       : integer; { Gain code for best vector mode2}
    CB_index2           : integer; { Index for best vector mode2     }
{ --- }
{ -- }
{ Table definitions }

{ Tables for the LTP }

{ Convert PP_gain_code4 to gain }

  TB_PP_gain : ARRAY[0..15] OF real;
    { Initialized by program }
{ ---- }
{ Convert Gain to PP_gain_code4 }

  TB_PP_gain_border  : ARRAY[0..15] of real ;
    { Initialized by program }
{ ---- }
{ --- }
{ -- }
{ Procedure definitions }


{ LPC analysis }

{ Initializations }

procedure Initializations;
extern;
{ ---- }
{ Getframe }

procedure getframe(var inbuf : win_type);
```

```
extern;
{ ---- }
{ Putframe }

procedure putframe(outbuf : out_type);
extern;
{ ---- }
{ LPCAnalysis }

procedure LPCAnalysis(Inbuf: win_type; var A_coeff : CF_type;
                      var CodeOutbuf : out_type );
extern;
{ ---- }
{ AnalysisFilter }

procedure AnalysisFilter(var Inp: SF_type; var A_coeff : CF_type;
         var Outp : SF_type; var FS_temp : FS_type);
var
  k,m             : integer;
  signal          : real;

begin

  for k:= 0 to 79 do begin
    signal:= Inp[k];
    FS_temp[0] := Inp[k];
    for m := 10 downto 1 do begin
      signal := signal + A_Coeff[m] * FS_temp[m];
      FS_temp[m] := FS_temp[m-1];
    end ;
    Outp[k] := signal;
  end ;
end ;
{ ---- }
{ SynthesisFilter }

procedure SynthesisFilter(var Inp: SF_type; var a_coeff : CF_type;
         var Outp : SF_type; var FS_temp : FS_type);
```

```
var
  k,m              : integer;
  signal           : real;

begin

  for k:= 0 to 79 do begin
    signal := Inp[k];
    for m := 10 downto 1 do begin
      signal := signal - A_Coeff[m] * FS_temp[m];
      FS_temp[m] := FS_temp[m-1];
    end ;
    Outp[k] := signal;
    FS_temp[1] := signal;
  end ;
end ;
{ ---- }
{ LPCCalculations }

procedure LPCCalculations(sub : integer; A_coeffn,
                          A_coeffo : CF_type;
                          var A_coeff, A_W_coeff : CF_type;
                          var H_syntes : SF_type); extern;
{ ---- }
{ --- }
{ LTP analysis }

{ PowerCalc }

procedure PowerCalc(var Speech : SF_type; var power : real);

var
  i                : integer;

begin
  power :=0;
  for i:=0 to 79 do begin
    power:=power+SQR(Speech[i]);
  end ;
```

```
end ;
{ ---- }
{ CalcPower }

procedure CalcPower(var Speech : histSF_type; delay : integer;
                    var Powerout : delay_type);

var
  k                 : integer;
  power             : real;

begin
  power :=0;
  for k:=0 to 79 do begin
    power := power + SQR(Speech[k-delay]);
  end ;
  Powerout[delay]:= power;
end ;
{ ---- }
{ CalcCorr  }

procedure CalcCorr(var Speech : histSF_type; delay : integer;
                   var Corrout : delay_type);


var
  corr              : real;

begin
  corr := 0;
  for k:=0 to 79 do begin
    corr := corr + Speech[k] * Speech[k-delay];
  end ;
  Corrout[delay] := corr;
end ;
{ ---- }
{ CalcGain }

procedure CalcGain(var power: real; var corr: real; var gain: real;
                                var gain_code : integer);
```

```
 begin
   if power = 0 then begin
     gain:=0;
   end else begin
     gain := corr/power;
   end ;

   gain_code:=0;
   while (gain > TB_PP_gain_border[gain_code])
                 and (gain_code<15) do begin
     gain_code := gain_code+1;
   end ;
   gain := TB_PP_gain[gain_code];
 end;
 { ---- }
 { Decision     }

 procedure Decision(var in_power, power, corr, gain : real;
                    delay : integer;
                    var best_error, best_power, best_corr : real;
                    var best_delay : integer);

 begin
   if (in_power+SQR(gain)*power-2*gain*corr < best_error) then begin
     best_delay := delay;
     best_error := in_power+SQR(gain)*power-2*gain*corr;
     best_corr := corr;
     best_power := power;
   end ;
 end ;
 { ---- }
 { GetPrediction }

 procedure GetPrediction(var delay : integer; var gain : real;
         var Hist : hist_type; var Pred : SF_type);

 var
   i,j              : integer;
   sum              : real;
```

```
begin
  for i:=0 to 79 do begin
    if (i-delay) < 0 then
      Pred[i] := gain * Hist[i-delay]
    else
      Pred[i] := gain * Pred[i-delay];
  end ;
end ;

{ ---- }
{ CalcSyntes }

procedure CalcSyntes(delay : integer; var Hist : hist_type;
                     var H_syntes : SF_type; var Pred, Overlap :
SF_type);

var
  k,i              : integer;
  sum              : real;

begin
  for k:=0 to Min(delay-1,79) do begin
    sum:=0;
    for i:=0 to Min(k,trunclength-1) do begin
      sum := sum + H_syntes[i] * Hist[k-i-delay];
    end ;
    Pred[k] := sum;
  end ;
  for k:=delay to 79 do begin
    Pred[k] := Pred[k-delay];
  end ;

  for k:=delay to Min(79,2*delay-1) do begin
    sum:=0;
    for i:=k-delay+1 to trunclength-1 do begin
      sum := sum + H_syntes[i] * Hist[k-i-delay];
    end ;
    Overlap[k]:= sum;
  end ;
```

```
    for k:=2*delay to 79 do begin
      Overlap[k]:= Overlap[k-delay];
    end ;
  end ;
  { ---- }
  { CalcPowerCorrAndDecision1 }

  procedure CalcPowerCorrAndDecision1(delay : integer; var Speech,
                Pred, Overlap : SF_type; var in_power : real;
                var best_error, best_gain : real;
                var best_gain_code, best_delay : integer);

  var
    k,j             : integer;
    virt            : integer;
    gcode1          : integer;
    gcode2          : integer;
    gainc           : integer;
    gain            : real;
    gain2           : real;
    gain3           : real;
    gain4           : real;
    gain5           : real;
    gain6           : real;
    gain7           : real;
    gain8           : real;
    error           : real;
    corr            : ARRAY[1..4] of real;
    power           : ARRAY[1..4] of real;
    corro           : ARRAY[2..4] of real;
    Powero          : ARRAY[2..4] of real;
    ccorr           : ARRAY[2..4] of real;
    Zero3           : ARRAY[2..4] of real := (0.0, 0.0, 0.0);
    Zero4           : ARRAY[1..4] of real := (0.0, 0.0, 0.0, 0.0);

  begin
    corr   := Zero4;
    power  := Zero4;
    corro  := Zero3;
```

```
Powero := Zero3;
ccorr  := Zero3;

virt:= 79 DIV delay;

corr[1]:= 0;
for k:=0 to Min(delay-1,79) do
  corr[1]:= corr[1] + Speech[k]*Pred[k];
power[1]:= 0;
for k:=0 to Min(delay-1,79) do
  power[1]:= power[1] + SQR(Pred[k]);

for j:= 1 to virt do begin
  corro[j+1]:= 0;
  for k:=j*delay to Min((j+1)*delay-1,79) do
    corro[j+1]:= corro[j+1] + Speech[k]*Overlap[k];
  powero[j+1]:= 0;
  for k:=j*delay to Min((j+1)*delay-1,79) do
    powero[j+1]:= powero[j+1] + SQR(Overlap[k]);
  corr[j+1]:= 0;
  for k:=j*delay to Min((j+1)*delay-1,79) do
    corr[j+1]:= corr[j+1] + Speech[k]*Pred[k];
  power[j+1]:= 0;
  for k:=j*delay to Min((j+1)*delay-1,79) do
    power[j+1]:= power[j+1] + SQR(Pred[k]);
  ccorr[j+1]:= 0;
  for k:=j*delay to Min((j+1)*delay-1,79) do
    ccorr[j+1]:= ccorr[j+1] + Pred[k]*Overlap[k];
end;

gcode1:= 0;
gcode2:= 15;

for gainc:= gcode1 to gcode2 do begin
  gain := TB_PP_gain[gainc];
  gain2:= SQR(gain);
  gain3:= gain*gain2;
  gain4:= SQR(gain2);
  gain5:= gain*gain4;
```

```
        gain6:= SQR(gain3);
        gain7:= gain*gain6;
        gain8:= SQR(gain4);
        error:= in_power - 2*gain*(corr[1] + corro[2])
              + gain2*(power[1] + powero[2] - 2*corr[2] - 2*corro[3])
              + gain3*(2*ccorr[2] - 2*corr[3] - 2*corro[4])
              + gain4*(power[2] + powero[3] - 2*corr[4])
              + 2*gain5*ccorr[3] + gain6*(power[3] + powero[4])
              + 2*gain7*ccorr[4] + gain8*power[4];
        if error < best_error then begin
          best_gain_code:= gainc;
          best_error:= error;
          best_delay:= delay;
        end;
      end;
      best_gain := TB_PP_gain[best_gain_code];
    end;
    { ---- }
    { CalcPowerCorrAndDecision2 }

    procedure CalcPowerCorrAndDecision2(delay : integer; var Speech,
                  Pred, Overlap : SF_type; var in_power : real;
                  var best_error, best_gain : real;
                  var best_gain_code, best_delay : integer);

    var
      k,i               : integer;
      gain_code         : integer;
      gain              : real;
      error             : real;
      corr1             : real;
      power1            : real;

    begin

      corr1:= 0;
      for k:=0 to 79 do
        corr1:= corr1 + Speech[k]*Pred[k];
      power1:= 0;
```

```
    for k:=0 to 79 do
      power1:= power1 + SQR(Pred[k]);

    if power1 = 0 then begin
      gain:=0;
    end else begin
      gain := corr1/power1;
    end ;
    gain_code:=0;
    while (gain > TB_PP_gain_border[gain_code])
      and (gain_code<15) do begin
      gain_code := gain_code+1;
    end ;
    gain := TB_PP_gain[gain_code];
    error:= in_power -2*gain*corr1 +SQR(gain)*power1;
    if error < best_error then begin
      best_gain:= gain;
      best_gain_code:= gain_code;
      best_error:= error;
      best_delay:= delay;
    end;
  end ;
  { ---- }
  { PredictionRecursion }

  procedure PredictionRecursion(delay : integer; var Hist : hist_type;
                var H_syntes : SF_type; var Pred, Overlap : SF_type);

  var
    k                : integer;

  begin
    for k:=Min(79,delay-1) downto trunclength do begin
      Pred[k]:= Pred[k-1];
    end ;
    for k:=trunclength-1 downto 1 do begin
      Pred[k] := Pred[k-1] + H_syntes[k] * Hist[-delay];
    end ;
    Pred[0] := H_syntes[0] * Hist[-delay];
```

```
    for k:=delay to 79 do begin
      Pred[k] := Pred[k-delay];
    end ;

    if 2*delay-1 < 80 then
      Overlap[2*delay-1]:= 0;
    for k:=Min(79,2*delay-2) downto delay do begin
      Overlap[k]:= Overlap[k-1];
    end ;
    for k:=2*delay to 79 do begin
      Overlap[k]:= Overlap[k-delay];
    end ;
  end ;
  { ---- }
  { --- }
  { Innovation analysis }

  { InnovationAnalysis }

  procedure InnovationAnalysis(speech : SF_type; A_coeff : CF_type;
                   H_syntes: SF_type; PP_delay: integer; PP_gain: real;
                   var index, gain_code : integer; var gain : real);
  extern;
  { ---- }
  { GetExcitation }

  procedure GetExcitation(index : integer; gain : real;
              var Excit : SF_type);
  extern;
  { ---- }
  { LTPSynthesis }

  procedure LTPSynthesis(delay : integer; a_gain : real;
                   var Excitin : SF_type; var Excitout : SF_type);

  var
    i               : integer;

  begin
```

```
    for i:=0 to 79 do begin
      if (i-delay) >= 0 then
        Excitout[i]:= Excitin[i] + a_gain*Excitout[i-delay]
      else Excitout[i]:= Excitin[i];
    end ;
  end ;
  { ---- }
  { --- }

  { -- }
  { - }
  { MAIN PROGRAM }

  { Begin }

  begin
  { -- }
    { Initialization }

    { Init Coding parameters }

    Initializations;
    { --- }
    { Zero history }

    for i:=-160 to -1 do begin
      PP_history[i]    := 0;
      Weighted_speech[i]    := 0;
    end ;
    { --- }
    { Zero filter states }

    for i:=0 to 10 do begin
      FS_zero_state[i]          := 0;
      FS_analys[i]              := 0;
      FS_temp[i]                := 0;
      FS_Wsyntes[i]             := 0;
      FS_ringing[i]             := 0;
    end ;
```

```
{ --- }
{ Init other variables }

for i:=0 to 79 do
  PP_Overlap[i]:=0;
for i:=0 to 79 do begin
  H_W_syntes[i]:=0;
  Zero_subframe[i]:=0;
end;
{ --- }
{ -- }
{ For frame_nr:= 1 to number_of_frames do begin }

for frame_nr:= 1 to number_of_frames do begin
{ -- }
  { LPC analysis }

  getframe(SpeechInbuf);
  A_coeffold:= A_coeffnew;
  LPCAnalysis(SpeechInbuf,A_Coeffnew,CodeOutbuf);
  { -- }
  { For subframe_nr:=1 to 4 do begin }

  for subframe_nr:=1 to 4 do begin
  { -- }
    { Subframe pre processing }

    { Get subframe samples }

    for i:=0 to 79 do begin
      Original_speech[i]:= SpeechInbuf[i+(subframe_nr-1)*80];
    end ;
    { --- }
    { LPC calculations }

    LPCCalculations(subframe_nr, A_coeffnew, A_coeffold, A_coeff,
                    A_W_coeff, H_W_syntes);
    { --- }
    { Weighting filtering }
```

```
AnalysisFilter(Original_Speech,A_coeff,
  Original_Residue,FS_analys);
SynthesisFilter(Original_residue,A_W_coeff,
  Original_Wspeech,FS_Wsyntes);
{ --- }
{ -- }
{ Mode 1 }

{ Open loop LTP search }

{ LTP preprocessing }

{ Initialize weighted speech }

for i:=0 to 79 do begin
  Weighted_speech[i]:= Original_Wspeech[i];
end ;
{ ----- }
{ Calculate power of weighted_speech to in_power }

PowerCalc(Original_Wspeech,in_power);
{ ----- }
{ Get limits lower and upper }

lower := 20;
upper := 147;
{ ----- }
{ ---- }
{ Openloop search of integer delays }

{ Calc power and corr for first delay }

delay := lower;
CalcCorr(Weighted_speech,delay,Opencorrelation);
CalcPower(Weighted_speech,delay,Openpower);
{ ----- }
{ Init best delay }

PP_delay := lower;
```

```
best_corr1 := Opencorrelation[PP_delay];
best_power1 := Openpower[PP_delay];
CalcGain(best_power1,best_corr1,PP_gain1,PP_gain_code1);
PP_best_error := In_power+SQR(PP_gain1)*best_power1
        -2*PP_gain1*best_corr1;
{ ----- }
{ For delay := lower+1 to upper do begin }

for delay := lower+1 to upper do begin
{ ----- }
  { Calculate power  }

  CalcPower(Weighted_speech,delay,Openpower);
  { ----- }
  { Calculate corr  }

  CalcCorr(Weighted_speech,delay,Opencorrelation);
  { ----- }
  { Calculate gain }

  power:= Openpower[delay];
  corr:= Opencorrelation[delay];
  CalcGain(power,corr,gain,gain_code);
  { ----- }
  { Decide if best vector so far }

  Decision(in_power,power,corr,gain,delay,
            PP_best_error,best_power1,best_corr1,PP_delay);
  { ----- }
{ End }

end ;
{ ----- }
{ ---- }
{ LTP postprocessing }

{ Calculate gain }

CalcGain(best_power1, best_corr1, PP_gain1, PP_gain_code);
```

```
{ ----- }
{ Get prediction according to delay and gain }

PP_delay1:= PP_delay;
PP_gain_code1:= PP_gain_code;

GetPrediction(PP_delay1, PP_gain1, PP_history, Prediction1);
{ ----- }
{ Synthesize prediction and remove memory ringing }

FS_temp:= FS_ringing;
SynthesisFilter(Prediction1,A_W_coeff,
         Prediction_syntes,FS_temp);
{ ----- }
{ Residual after LTP and STP }

for i:=0 to 79 do begin
  Weighted_Speech1[i] := Weighted_Speech[i]
                       - Prediction_syntes[i];
end ;
{ ----- }
{ Update Weighted_speech  }

for i:= -160 to -1 do begin
  Weighted_speech[i]:= Weighted_speech[i+80];
end ;
{ ----- }
{ ---- }
{ --- }
{ Excitation coding }

{ Innovation Analysis }

InnovationAnalysis(Weighted_speech1, A_W_coeff, H_W_syntes,
            PP_delay1, PP_gain1,
            CB_index1,CB_gain_code1,CB_gain1);
{ ---- }
{ Get Excitation }
```

```
GetExcitation(CB_index1, CB_gain1, Excitation1);
{ ---- }
{ Synthesize excitation }

LTPSynthesis(PP_delay1, PP_gain1, Excitation1, Excitation1);
FS_temp:= FS_zero_state;
SynthesisFilter(Excitation1,A_W_coeff,
         Weighted_excitation,FS_temp);
for k:= 0 to 79 do begin
  Weighted_speech1[k] := Weighted_speech1[k]
                       - Weighted_excitation[k];
end ;
{ ---- }
{ Calculate error }

PowerCalc(Weighted_speech1, Best_error1);
{ ---- }
{ --- }
{ -- }
{ Mode 2 }

{ Closed loop LTP search }

{ LTP preprocessing }

{ Remove ringing }

FS_temp:= FS_ringing;
SynthesisFilter(Zero_subframe,A_W_coeff,Ringing,FS_temp);
for k:= 0 to 79 do begin
  Original_Wspeech[k] := Original_Wspeech[k] - Ringing[k];
  Weighted_speech[k] := Original_Wspeech[k];
end ;
{ ----- }
{ Calculate power of weighted_speech to IN_power }

PowerCalc(Original_Wspeech,in_power);
{ ----- }
{ Get limits lower and upper }
```

```
        lower := 20;
        upper := 147;
        { ----- }
        { ---- }
        { Exhaustive search of integer delays }

        { Calc prediction for first delay }

        delay := lower;
        CalcSyntes(delay, PP_history, H_W_syntes, Prediction,
                   PP_overlap);
        { ----- }
{       Init decision }

        PP_delay       := delay;
        PP_gain_code   := 0;
        PP_best_error  := in_power;
{ ----- }
        { Calc power and corr decide gain }

        if delay <= 79 then begin

          CalcPowerCorrAndDecision1(delay, Original_Wspeech,
                   Prediction, PP_overlap, in_power, PP_best_error,
                   PP_gain2, PP_gain_code, PP_delay);
        end else begin

          CalcPowerCorrAndDecision2(delay, Original_Wspeech,
                   Prediction, PP_overlap, in_power, PP_best_error,
                   PP_gain2, PP_gain_code, PP_delay);
        end ;
        { ----- }
        { For delay := lower+1 to upper do begin }

        for delay := lower+1 to upper do begin
        { ----- }
          { Prediction recursion }

          PredictionRecursion(delay, PP_history, H_W_syntes,
```

```
                Prediction, PP_overlap);
    { ----- }
    { Calc power and corr decide gain }

    if delay <= 79 then begin

      CalcPowerCorrAndDecision1(delay, Original_Wspeech,
              Prediction, PP_overlap, in_power, PP_best_error,
              PP_gain2, PP_gain_code, PP_delay);
    end else begin

      CalcPowerCorrAndDecision2(delay, Original_Wspeech,
              Prediction, PP_overlap, in_power, PP_best_error,
              PP_gain2, PP_gain_code, PP_delay);
    end ;
    { ----- }
  { End }

  end ;
  { ----- }
  { ---- }
  { LTP postprocessing }

  { Get prediction according to PP_delay and gain }

  PP_delay2:= PP_delay;
  PP_gain_code2:= PP_gain_code;

  GetPrediction(PP_delay2, PP_gain2, PP_history, Prediction2);
  { ----- }
  { Synthesize prediction to prediction_syntes }

  FS_temp:= FS_zero_state;
  SynthesisFilter(Prediction2,A_W_coeff,
           Prediction_syntes,FS_temp);
  { ----- }
  { Residual after LTP and STP }

  for i:=0 to 79 do begin
```

```
    Weighted_Speech2[i]:= Weighted_Speech[i]
                        - Prediction_syntes[i];
  end ;
  { ----- }
  { ---- }
  { --- }
  { Excitation coding }

  { Innovation Analysis }

  InnovationAnalysis(Weighted_speech2,A_W_Coeff, H_W_syntes,
                     PP_delay2, PP_gain2,
                     CB_index2,CB_gain_code2,CB_gain2);
  { ---- }
  { Get Excitation }

  GetExcitation(CB_index2, CB_gain2, Excitation2);
  { ---- }
  { Synthesize excitation }

  LTPSynthesis(PP_delay2, PP_gain2, Excitation2, Excitation2);
  FS_temp:= FS_zero_state;
  SynthesisFilter(Excitation2,A_W_coeff,
           Weighted_excitation,FS_temp);
  for k:= 0 to 79 do begin
    Weighted_speech2[k] := Weighted_speech2[k]
                        - Weighted_excitation[k];
  end ;
  { ---- }
  { Calculate error }

  PowerCalc(Weighted_speech2, Best_error2);
  { ---- }
  { --- }
  { -- }
  { Subframe post processing }

  { Mode Selection }
```

```
 if best_error1 < best_error2 then begin

   mode:= 1;
   Prediction:= Prediction1;
   Excitation:= Excitation1;
   PP_delay:= PP_delay1;
   PP_gain_code:= PP_gain_code1;
   CB_index:= CB_index1;
   CB_gain_code:= CB_gain_code1;

 end else begin

   mode:= -1;
   Prediction:= Prediction2;
   Excitation:= Excitation2;
   PP_delay:= PP_delay2;
   PP_gain_code:= PP_gain_code2;
   CB_index:= CB_index2;
   CB_gain_code:= CB_gain_code2;
end;
{ --- }
{ Output parameters }

CodeOutbuf[10+(subframe_nr-1)*4+1]:= PP_delay;
CodeOutbuf[10+(subframe_nr-1)*4+2]:= PP_gain_code;
CodeOutbuf[10+(subframe_nr-1)*4+3]:= CB_index;
CodeOutbuf[10+(subframe_nr-1)*4+4]:= CB_gain_code;
{ --- }
{ Get excitation }

for i:=0 TO 79 do begin
  Excitation[i] := Excitation[i] + Prediction[i];
end ;
{ --- }
{ Update PP_history with Excitation }

for i:= -160 to -81 do begin
  PP_history[i] := PP_history[i+80];
end ;
```

```
            for i:= -80 to -1 do begin
              PP_history[i] := Excitation[i+80];
            end ;
            { --- }
            { Synthesize ringing }

            SynthesisFilter(Excitation,A_W_coeff,
                     Weighted_excitation,FS_ringing);
            { --- }
            { -- }
          { End this subframe }

          end ;
          putframe(CodeOutbuf);
          { -- }
        { End this frame }

        end ;
        { -- }
      { End Program }

      end .
      { -- }
      { - }
```

Patenttivaatimukset:

1. Näytejonomuotoisen puhesignaalivektorin (s(n)) koodaustapa synteesin kautta tapahtuvan analyysimenettelyn avulla muodostamalla optimaalinen viritysvektori, joka muodostuu kiinteän koodikirjan koodivektorin ja pitkäaikaisennustajavektorin lineaarisesta yhdistelmästä, t u n n e t t u siitä, että

(a) pitkäaikaisennustajavektorin ensimmäinen arvio muodostetaan avoimessa silmukka-analyysissa (22, 24, 30, 32, 34, 36);

(b) pitkäaikaisennustajavektorin toinen arvio muodostetaan suljetussa silmukka-analyysissa (g_L, 14, 16, 20, 22, 24, 28, 34, 36); ja

(c) yhdistetään lineaarisesti (g_J, g_L, 14, 16, 20, 22, 24, 28, 36) täydellisen haun aikana jokainen ensimmäinen ja toinen arvio kiinteän koodikirjan (12) kaikkiin koodivektoreihin sen viritysvektorin muodostamiseksi, joka tuottaa puhesignaalivektorin (s(n)) parhaan koodauksen.

2. Patenttivaatimuksen 1 mukainen tapa, t u n n e t t u siitä, että vaiheen (c) pitkäaikaisennustajavektorin ensimmäinen ja toinen arvio muodostetaan yhdessä ja samassa suotimessa (28, g_L).

3. Patenttivaatimuksen 1 mukainen tapa, t u n n e t t u siitä, että vaiheen (c) pitkäaikaisennustajavektorin ensimmäinen ja toinen arvio tallennetaan yhteen ja samaan adaptiiviseen kodikirjaan (10) ja haetaan siitä.

4. Jonkin edeltävän patenttivaatimuksen mukainen tapa, t u n n e t t u siitä, että pitkäaikaisennustajavektorin ensimmäisen ja toisen arvion muodostaa suuren erotuskyvyn ennustaja.

5. Jonkin edeltävän patenttivaatimuksen mukainen tapa, t u n n e t t u siitä, että pitkäaikaisennustajavektorin ensimmäisen ja toisen arvion muodostaa ennustaja, jonka aste $p > 1$.

6. Jonkin patenttivaatimuksen 2, 4 tai 5 mukainen tapa, t u n n e t t u siitä, että sekä ensimmäiseen että toiseen arvioon kohdistetaan vahvistuskerroin (g_L), joka valitaan kvantisoitujen vahvistuskertoimien joukosta.

7. Jonkin edeltävän patenttivaatimuksen mukainen tapa, t u n n e t t u siitä, että sekä ensimmäinen että toinen arvio edustaa tunnusomaista viivettä (L) ja että toisen arvion viive haetaan ensimmäisen viiveen tai sen monikertojen tai alimonikertojen lähialueilta.

P A T E N T K R A V

1. Sätt att koda en samplad talsignalvektor (s(n)) i ett analys-genom-syntesförfarande genom bildande av en optimal excitationsvektor bestående av en linjär kombination av en kodvektor ur en fast kodbok (12) och en långtidsprediktorvektor, k ä n n e t e c k n a t av

(a) att ett första estimat på långtidsprediktorvektorn bildas i en öppen slinganalys (22, 24, 30, 32, 34, 36);

(b) att ett andra estimat på långtidsprediktorvektorn bildas i en sluten slinganalys (g_L, 14, 16, 20, 22, 24, 28, 34, 36); och

(c) att det första och det andra estimatet vart och ett i en uttömmande sökning linjärt kombineras (g_J, g_L, 14, 16, 20, 22, 24 28, 36) med alla kodvektorerna i den fasta kodboken (12) för bildande av den excitationsvektor som ger den bästa kodningen av talsignalvektorn (s(n)).

2. Sätt enligt krav 1, k ä n n e t e c k n a t av att det första och det andra estimatet av långtidsprediktorvektorn i steg (c) bildas i ett och samma filter (28, g_L).

3. Sätt enligt krav 1, k ä n n e t e c k n a t av att det första och det andra estimatet av långtidsprediktorvektorn i steg (c) lagras i och hämtas ur en och samma adaptiva kodbok (10).

4. Sätt enligt något av föregående krav, k ä n n e t e c k n a t av att det första och det andra estimatet av långtidsprediktorvektorn bildas av en högupplösande prediktor.

5. Sätt enligt något av föregående krav, k ä n n e t e c k n a t av att det första och det andra estimatet av långtidsprediktorvektorn bildas av en prediktor med ett ordningstal p>1.

6. Sätt enligt något av kraven 2, 4-5, k ä n n e t e c k n a t av att det första och andra estimatet påläggs vardera en förstärkningsfaktor (g_L), vilka väljs ur en uppsättning kvantiserade förstärkningsfaktorer.

7. Sätt enligt något av föregående krav, k ä n n e t e c k n a t av att det första och andra estimatet representerar vardera en karakteristisk fördröjning (L) och att det andra estimatets fördröjning söks i områden kring det första estimatets fördröjning samt multipler eller submultipler av denna.

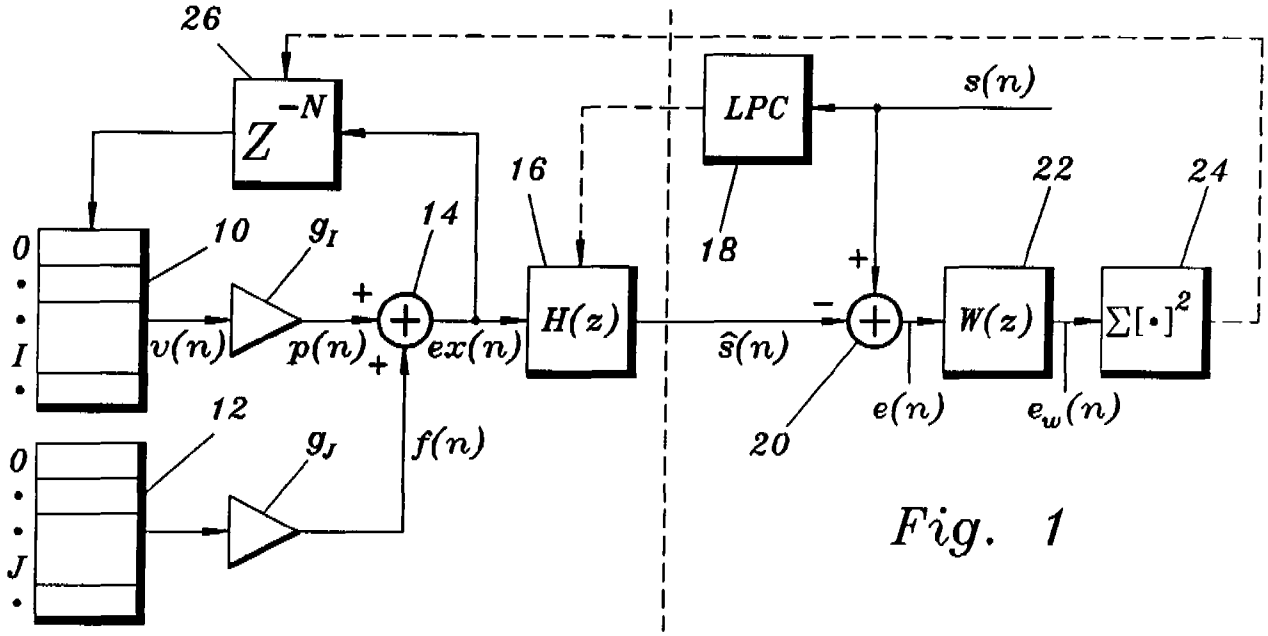

Fig. 1

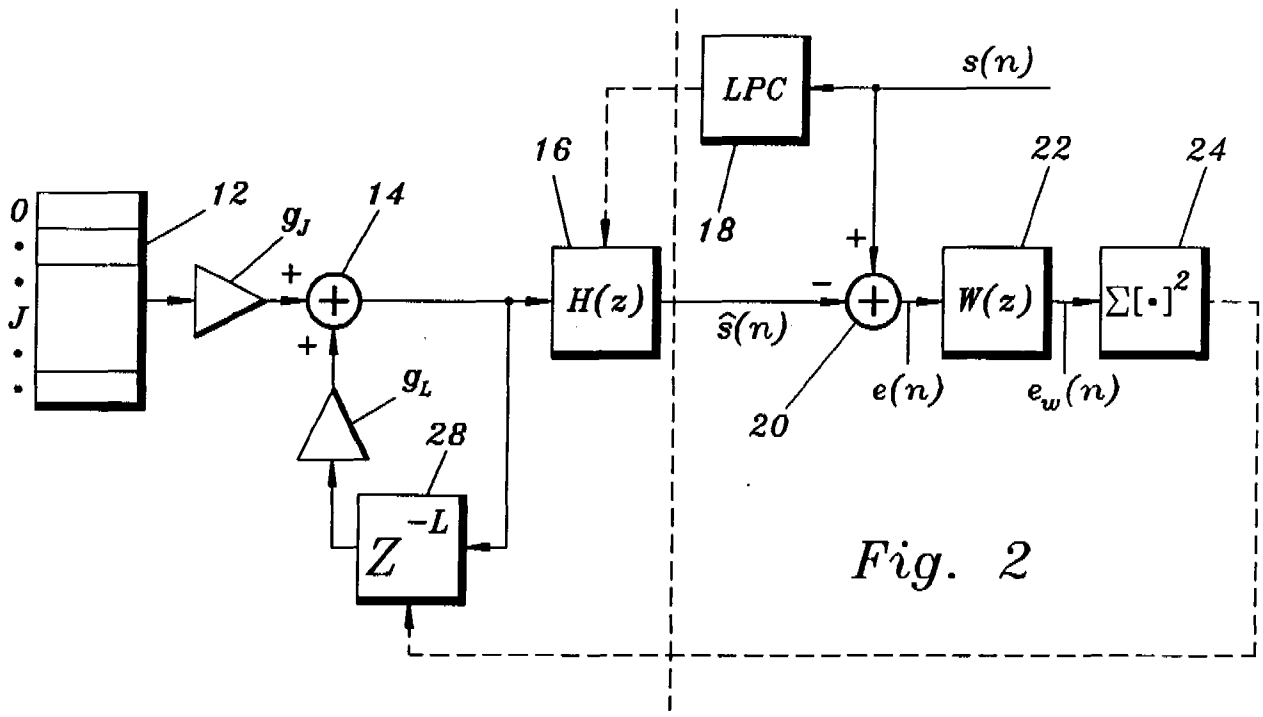

Fig. 2

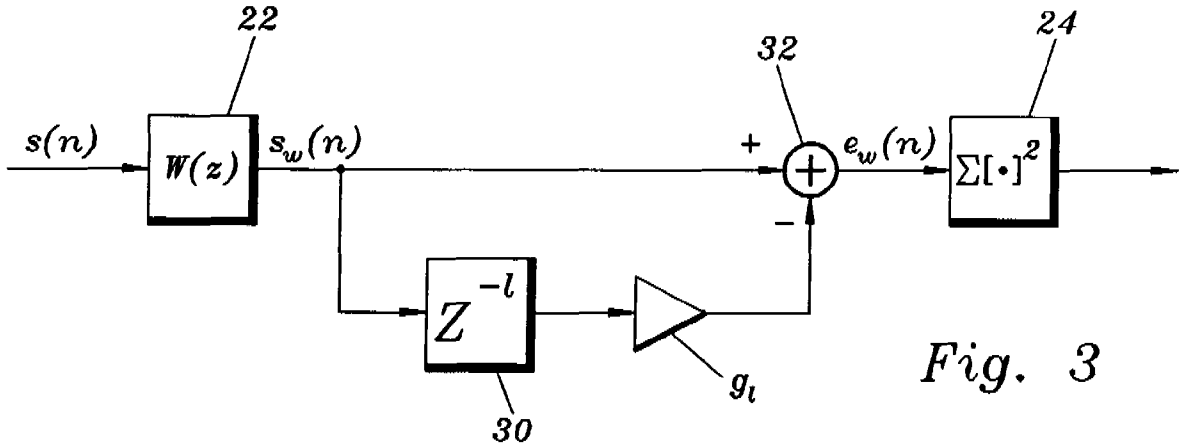

Fig. 3

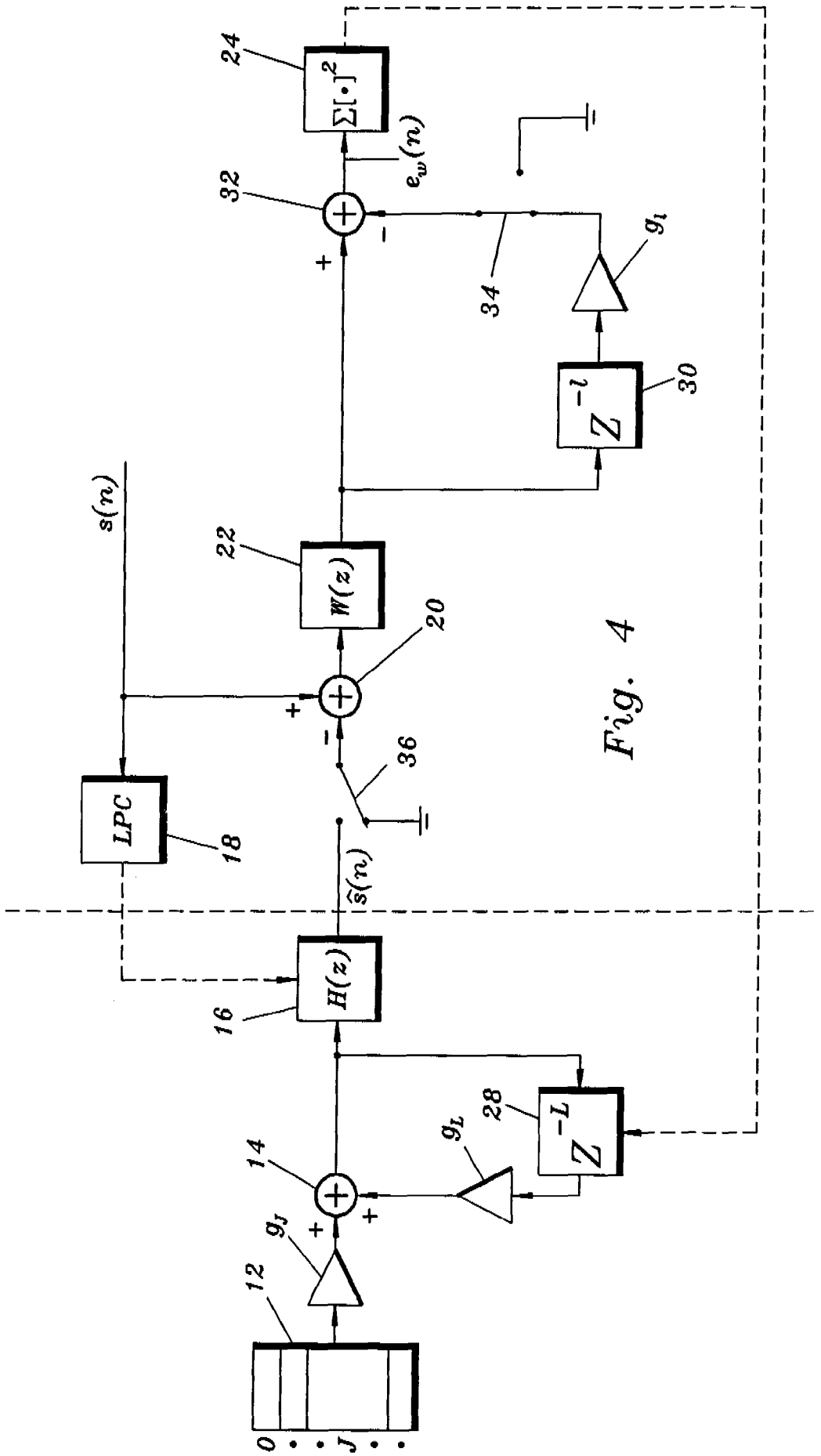
24
$\Sigma[\cdot]^2$
$e_w(n)$
32
+
−
34
g_1
Z^{-l}
30
s(n)
22
W(z)
20
+
−
36
LPC
18
$\hat{s}(n)$
H(z)
16
14
+
+
g_L
28
Z^{-L}
g_J
12
0
J

Fig. 4