Optimale detectie in aanwezigheid van ISI

We beschouwen het geval van lineaire digitale modulatie : het ontvangen signaal r(t) is gegeven door $r(t) = \sqrt{E_s}\sum_m a_m p(t - mT) + w(t)$. De corresponderende logaritmische kans-functie ln p(**r**|**a**) is dan $\ln p(\mathbf{r} \mid \mathbf{a}) = \frac{-1}{N_0}\left(E_s \sum_{m,n} a_m^* a_n g(mT - nT) - 2\sqrt{E_s}\sum_m \mathrm{Re}[a_m^* z_m] \right)$, waarbij z_m wordt bekomen door r(t) aan te leggen aan het matched filter (met impulsantwoord p*(-t)) en de uitgang van dit filter te bemonsteren op het tijdstip kT, terwijl $g(t) = \int p^*(-u)p(t-u)du$ de puls aan de uitgang van het matched filter voorstelt. We onderstellen dat $g(kT) \neq \delta_k$, zodat er ISI aanwezig is aan de uitgang van het matched filter.

Stel dat we op basis van de K+1 monsterwaarden $\mathbf{z}_K = (z_0, z_1, ..., z_K)^T$ een decisie maken met betrekking tot de K+1 symbolen $\mathbf{a}_K = (a_0, a_1, ..., a_K)^T$. De ML decisie minimaliseert de functie $J_{K+1}(\mathbf{a}_K)$, gegeven door $J_{K+1}(\mathbf{a}_K) = E_s \sum_{m=0}^{K}\sum_{n=0}^{K} a_m^* a_n g(mT - nT) - 2\sqrt{E_s}\sum_{m=0}^{K} \mathrm{Re}[a_m^* z_m]$. De functie $J_{K+1}(\tilde{\mathbf{a}}_K)$ stelt de 'kost' voor, van een 'testsequentie' $\tilde{\mathbf{a}}_K$, bij de aanvang van het symboolinterval met index K+1. Een exhaustieve zoekprocedure naar de ML decisie vereist een aantal berekeningen dat evenredig is met M^{K+1}, en dus exponentieel toeneemt met de lengte van de datasequentie.

Het aantal berekeningen kan echter sterk gereduceerd worden wanneer we aannemen aan dat g(kT) = 0 voor |k|>L, zodat enkel de L symbolen voor en de L symbolen na het nuttige symbool a_m bijdragen tot de ISI in z_m. Dit laat immers toe de functie $J_{K+1}(\mathbf{a}_K)$, recursief te berekenen op de volgende wijze : $J_{K+1}(\mathbf{a}_K) = J_K(\mathbf{a}_{K-1}) + \lambda(\mathbf{s}_K, a_K)$ voor K = 0, 1, ... , met $\lambda(\mathbf{s}_K, a_K) = E_s \mid a_K \mid^2 g(0) + 2\sqrt{E_s}\,\mathrm{Re}\left[a_K^* \left(-z_K + \sum_{k=1}^{L} a_{K-k} g(k) \right) \right]$. Hierbij is $\mathbf{s}_K = (a_{K-L}, a_{K-L+1}, ..., a_{K-1})^T$ de 'toestand' bij de aanvang van het symboolinterval met index K, en stellen we $\mathbf{a}_{-1} = \mathbf{s}_0$. Tevens hebben we gebruik gemaakt van de symmetrierelatie g(t) = g*(-t). De vector $\mathbf{s}_K$ kan M^L verschillende waarden aannemen, waarbij M het aantal constellatiepunten voorstelt. In plaats van de toestand voor te stellen door een vector van L datasymbolen, kan aan elke toestand een geheel getal uit de verzameling {0, 1, ..., M^L-1} geassocieerd worden. Het aantal mogelijke waarden van de toestand $\mathbf{s}_K$ stijgt exponentieel met de duur van g(t).

Aan elk van de mogelijke begintoestanden $\mathbf{s}_0$ wordt een kost $J_0(\mathbf{s}_0)$ gehecht. Is de begintoestand niet gekend, dan geven we elke begintoestand dezelfde kost (bijvoorbeeld $J_0(\mathbf{s}_0)$ = 0). Is de begintoestand wel gekend (dit is het geval wanneer de datasequentie wordt voorafgegaan door een gekende trainingsequentie), dan geven we aan deze begintoestand een kost gelijk aan nul, en aan de overige begintoestanden een kost die veel groter is dan nul.

De opeenvolging van de mogelijke toestanden in de tijd geeft aanleiding tot een trellis, waarvan de takken ('branches') corresponderen met de mogelijke toestandsovergangen ('transities'). De toestand $\mathbf{s}_{k+1}$ wordt volledig bepaald door de toestand $\mathbf{s}_k$ en het symbool a_k. Uit elke toestand vertrekken dus M takken, en in elke toestand komen M takken toe. Met elke transitie correspondeert een kost $\lambda(\mathbf{s}_k, a_k)$, die bepaalt wordt door de toestand waaruit de beschouwde tak vertrekt en door het symbool a_k dat de transitie veroorzaakt. Het bepalen van

de ML decisie is dus equivalent met het vinden van het pad van een begintoestand $\mathbf{s}_0$ tot een eindtoestand $\mathbf{s}_{K+1}$ waarvan de totale kost (dit is de kost van de begintoestand plus de som van de kosten van de takken van het pad) minimaal is.

De ML decisie wordt bekomen aan de hand van het 'Viterbi-algoritme' (dat een toepassing is van 'dynamisch programmeren'). Het Viterbi-algoritme maakt gebruik van het feit dat het optimale pad van een begintoestand tot een eindtoestand ook optimaal is over elk deeltraject dat tussenliggende toestanden verbindt. De eerste stap van het Viterbi-algoritme gaat als volgt. We beschouwen alle M takken die in een bepaalde toestand $\mathbf{s}_1$ (op tijdstip 1) toekomen. We behouden enkel de tak waarvoor de som van de kost van de tak en van de overeenkomstige begintoestand minimaal is, en beschouwen deze som als de kost van de toestand $\mathbf{s}_1$: dit is de minimale kost over alle paden (bestaande uit één tak) die leiden van een begintoestand tot toestand $\mathbf{s}_1$. Deze procedure wordt toegepast op elke toestand $\mathbf{s}_1$ op tijdstip 1. In de tweede stap van het Viterbi-algoritme beschouwen we alle M takken die toekomen in een bepaalde toestand $\mathbf{s}_2$ op tijdstip 2; we behouden enkel de tak waarvoor de som van de kost van de tak en van de overeenkomstige toestand op tijdstip 1 minimaal is, en beschouwen deze som als de kost van de toestand $\mathbf{s}_2$: dit is de minimale kost over alle paden (bestaande uit twee takken) die leiden van een begintoestand tot toestand $\mathbf{s}_2$; het corresponderende pad met minimale kost wordt bekomen door vanuit $\mathbf{s}_2$ terug te gaan 'in de tijd', en bestaat uit de enige in $\mathbf{s}_2$ toekomende tak die werd behouden, voorafgegaan door de enige overblijvende tak die toekomt in de toestand $\mathbf{s}_1$ (op tijdstip 1) waaruit de in $\mathbf{s}_2$ toekomende tak vertrekt. Deze procedure wordt toegepast op elke toestand $\mathbf{s}_2$ op tijdstip 2. Op dezelfde wijze worden de volgende stappen van het Viterbi-algoritme uitgevoerd, totdat uiteindelijk de kost in elk van de eindtoestanden (op tijdstip K+1) is bekomen. Tenslotte bepalen we de eindtoestand met minimale kost, alsook, door vanuit deze eindtoestand terug te gaan in de tijd, het unieke pad (bestaande uit K+1 takken) dat van een begintoestand tot deze eindtoestand met minimale kost leidt. De K+1 datasymbolen die aanleiding geven tot dit pad vormen de ML decisie. Het aantal berekeningen vereist voor het Viterbi-algoritme is evenredig met $M^L(K+1)$; dit aantal is evenredig met de lengte van de datasequentie, en stijgt exponentieel met de duur van de puls g(t). Wanneer de puls g(t) slechts traag uitsterft, is de rekencomplexiteit van het Viterbi-algoritme te groot voor een praktische implementatie; in dit geval zal men zijn toevlucht nemen tot de minder complexe (maar t.o.v. het Viterbi-algoritme suboptimale) egalisatiefilters die de ISI elimineren (of sterk reduceren) ten koste van een toename van het ruisniveau aan de ingang van de decisie-eenheid.

Fig. 1 illustreert het Viterbi-algoritme in het geval van basisbandmodulatie met 2-PAM, zodat alle grootheden reëel zijn. We onderstellen g(0) = 1, g(T) = -0.6, g(2T) = 0.3, en g(mT) = 0 voor |m|>2. Voor elke k zijn dus2^2 = 4 toestanden $\mathbf{s}_k$, die we voorstellen door de getallen 0 (a_{k-2} = 1, a_{k-1} = 1), 1 (a_{k-2} = 1, a_{k-1} = -1), 2 (a_{k-2} = -1, a_{k-1} = 1) en 3 (a_{k-2} = -1, a_{k-1} = -1). Bij de berekening van de kost $\lambda(\mathbf{s}_k, a_k)$ hebben we voor de eenvoud E_s = 1 ondersteld, en de term in $|a_k|^2$ buiten beschouwing gelaten omdat $|a_k|^2$ = 1. De monsterwaarden z_k werden bepaald in afwezigheid van ruis, om te illustreren dat het Viterbi-algoritme dan de correcte datasymbolen oplevert. De datasymbolen en monsterwaarden zijn gegeven in onderstaande tabel. De begintoestand $\mathbf{s}_0$ is 3 (m.a.w., a_{-2} = a_{-1} = -1), en is gekend ondersteld; de correcte begintoestand heeft een kost gelijk aan nul, de kost van de overige begintoestanden is gelijk aan 20. De streeplijnen geven de paden aan die met minimale kost vanuit een begintoestand de beschouwde toestand bereiken; bij elke toestand is deze minimale kost weergegeven. De dikke volle lijn is het optimale pad doorheen de trellis, dat correspondeert met de ML decisie. Merk op dat de toestanden gelegen op het optimale pad volledig in overeenstemming zijn met de uitgezonden datasequentie, zodat de ML decisie correct is in afwezigheid van ruis.

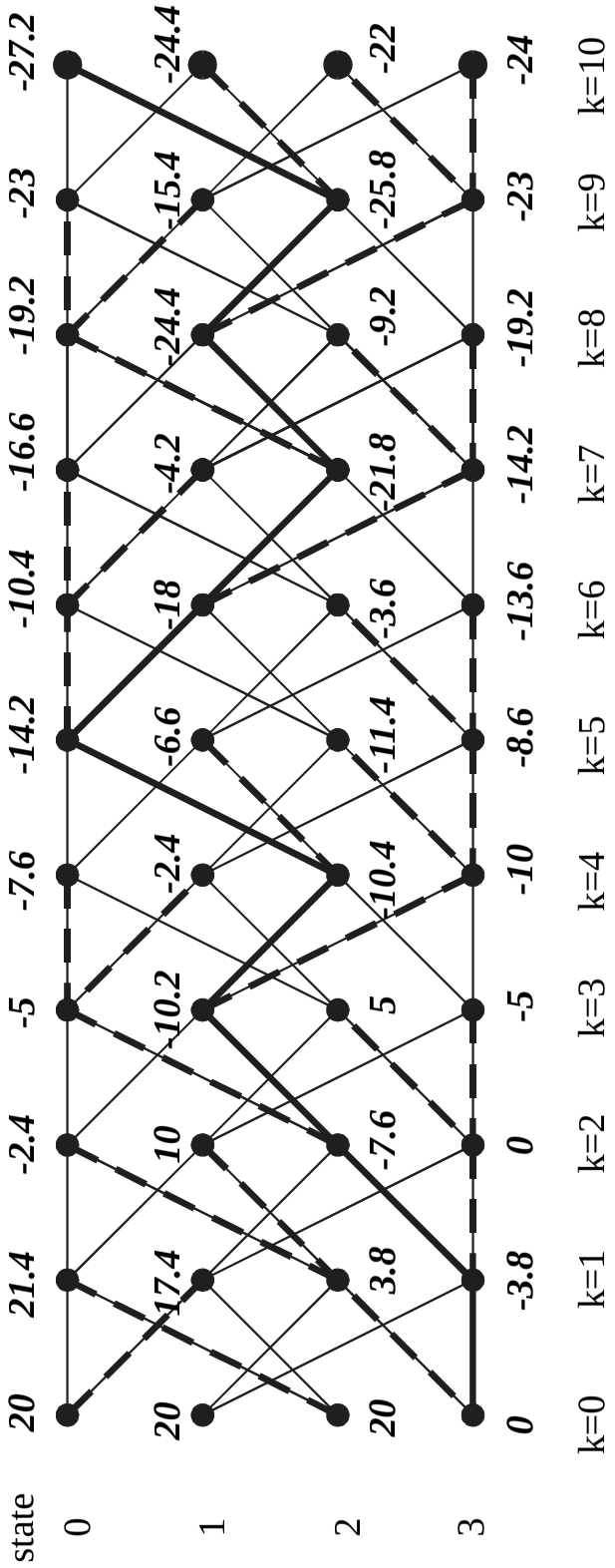

Fig. 1 : Illustratie Viterbi-algoritme (s_k = 0, 1, 2, 3 ⇔ (a_{k-2}, a_{k-1}) = (1,1), (1,-1), (-1,1), (-1,-1))

k	0	1	2	3	4	5	6	7	8	9
a_k	-1	1	-1	1	1	-1	1	-1	1	1
z_k	-1.6	2.2	-2.2	1	1	-2.2	2.8	-2.2	1.6	-0.2