Opgave A: Project Communicatietheorie

Jelle Baileul, jelle.bailleul@ugent.be

1 Inleiding

Dit is de opgave voor het project dat aansluit bij de cursus *Communicatietheorie.* Dit project wordt uitgevoerd in groepjes van 4 studenten (eventueel 3). Inschrijven in een groep gebeurt via Minerva, waar je ook de nodige bestanden terugvindt. Het groepsnummer bepaalt een aantal van de parameters die nodig zijn bij het oplossen van het project.

Het doel van dit project is om een afbeelding te versturen over een communicatiekanaal, waarbij je verschillende facetten van dit systeem moet implementeren, onderzoeken en bespreken. Voor dit alles gebruiken we Matlab dat beschikbaar is via Athena. Het is de bedoeling dat je de meegeleverde bestanden aanvult waar nodig. Bij het indienen van het project moet ook deze Matlab code ingediend worden, zodat die kan gecontroleerd worden op correctheid en plagiaat.

Bij dit project moet ook een verslag gemaakt worden waarin onderstaande vragen worden beantwoord en de resultaten geïllustreerd. Zorg ervoor dat dit verslag niet zomaar een opsomming van antwoorden op de vragen is en dat als er een figuur gevraagd wordt, deze duidelijk is! Let dus in het bijzonder op het volgende: assen benoemen, verschillende kleuren/markers, lineaire/logaritmische schaal, bereik van de x-as/y-as,.... Een elektronische versie (pdf) van het verslag moet samen met alle Matlab code **ingediend worden ten laatste op maandag 18 december 2017 voor 17u00** en dit via de dropbox van Minerva in een zipbestand met naam *groepXX.zip*.

2 Opmerkingen over het gebruik van Matlab

Matlab is een technische softwareomgeving en wordt gebruikt voor tal van wiskundige toepassingen. In dit project modelleren we er een communicatiekanaal in. Een uitgebreide beschrijving van de syntaxis, de verschillende functies en hoe te werken met Matlab is te vinden in de online documentatie[1]. Verder vind je op Minerva ook nog een document waarin de basis uitgelegd wordt aan de hand van voorbeelden. Met het oog op een vlotte implementatie van het project worden hier nog enkele zaken benadrukt.

- Een Matlab file bevat ofwel een script, een functie of een klasse. Een script voert een reeks van commando's uit en is voornamelijk handig om code die verschillende functies en klassen oproept in te bewaren om die zo te kunnen hergebruiken. Functies en klassen in Matlab zijn vrij gelijkaardig als in andere programmeertalen. In dit project krijg je een aantal files met klassen waarin enkele statische functies moeten aangevuld worden. Om deze functies te testen en om de verschillende simulaties uit te voeren is het heel handig om daarvoor verschillende scripts te schrijven.

- Matlab werkt intern voornamelijk met matrices en vectoren. Het vergt dan ook vaak veel minder tijd om iets te bereken als het geïmplementeerd is aan de hand van matrices en vectoren in vergelijking met een implementatie die lussen bevat.

- In Tabel 1 vind je enkele nuttige Matlab commando's die je kan gebruiken in het project. Wanneer er vermeld wordt dat je zelf iets moet implementeren mag je wel geen ingebouwde functies gebruiken die hetzelfde doen.

[1] https://nl.mathworks.com/help/matlab/

functienaam	*omschrijving*
a./b en a.*b	Matrices a en b puntsgewijs delen of vermenigvuldigen
integral	Numeriek berekenen van een integraal
reshape,kron,repmat	Functies om matrices te vervormen of uit te breiden
de2bi, bi2de	Zet integers om naar bits en omgekeerd
sort, find	Sorteer of zoek een vector in een matrix
rand,randn	Genereren van random data
plot,hist,bar,...	Functies om te plotten van data
mod	Bereken van de modulus

Tabel 1: Enkele nuttige Matlab commando's

- In dit project zul je ook werken met anonieme functies. Een anonieme functie is een functie die geassocieerd wordt met een variable van het type *function_handle.* Net zoals gewone functies accepteren deze functies input variabelen en geven ze één output variabele terug. Als we bijvoorbeeld een anonieme functie willen definiëren om het kwadraat van een getal te berekenen, schrijven we

  ```
  sqr = @(x) x.^2;
  ```

  Hierbij is de variable 'sqr' van het type *function_handle.* De input variabelen bevinden zich tussen de haakjes na de @ operator. Om dan het kwadraat van 5 te bereken, gebruik je volgende regel:

  ```
  sqr(5)
  ```

  wat uiteraard 25 oplevert. In Matlab hebben sommige functies, zoals *integral*, een *function_handle* als argument. Om bijvoorbeeld de integraal van x^2 te bereken van 0 tot 1 gebruik je volgend commando:

  ```
  q = integral(sqr,0,1);
  ```

  Verder hoef je niet altijd een variable aan te maken en kan je ook een *function_handle* gebruiken in de definitie van een andere *function_handle.* Zo kan je de integraal over x^4 van 0 tot 1, bereken op volgende twee manieren:

  ```
  q = integral(@(x) x.^4,0,1);
  ```

  ```
  q = integral(@(x) sqr(x).^2,0,1);
  ```

  Bemerk ten slotte dat we in de definitie van de verschillende anonieme functies telkens gebruik maken van '.^' (elementsgewijs) in plaats van '^'. Dit doen we omdat de *integral* functie van Matlab verwacht dat de meegegeven anonieme functie een vector heeft als input en een vector van dezelfde grootte als output.

3 De componenten van het digitaal communicatiesysteem

Het doel van het communicatiesysteem is het versturen van een afbeelding van de zender naar de ontvanger. De originele afbeelding is opgebouwd uit een matrix (899x600) van grijswaarden die voorgesteld zijn als een geheel getal tussen 0 en 255. Hierbij staat 0 voor zwart en 255 voor wit.

3.1 Zender

De zender heeft als taak de digitale informatie die aangeboden wordt door een bron om te zetten in een fysisch informatiesignaal dat kan verzonden worden over het kanaal. Volgende stappen worden daarvoor uitgevoerd:

- Eerst wordt de originele figuur beperkt in grootte door een kwantisatie stap met 8 reconstructieniveaus toe te passen. In subsectie 4.1 zullen we hiervoor verschillende kwantisators ontwerpen en met elkaar vergelijken.
- De gekwantiseerde data bevat meestal redundantie, dit is overtollige informatie. De te versturen sequentie van informatiebits kan dan ook worden ingekort of gecomprimeerd door een korte bitsequentie toe te kennen aan veel voorkomende macrosymbolen en een langere bitsequentie aan zeldzame macrosymbolen. Deze omzetting wordt broncodering genoemd. Dit jaar focust het project hier echter niet op.
- Als we informatie versturen over een kanaal, worden onvermijdelijk fouten geïntroduceerd. Om onze bitsequentie te beschermen tegen dergelijke fouten, voegen we op gecontroleerde wijze redundantie toe. Deze procedure wordt kanaalcodering genoemd. Hierbij wordt de bitsequentie opgesplitst in groepjes van k bits en aan elk groepje worden $(n-k)$ redundante bits toegevoegd. Op die manier verkrijgen we codewoorden van n bits. Dit leidt tot een nieuwe, langere bitsequentie. In subsectie 4.2 onderzoeken we de prestaties van een (14,10) code en gaan we na of we de foutprobabiliteit verder kunnen verlagen met behulp van retransmissieprotocols.
- Vervolgens (subsectie 4.3) wordt de bitsequentie klaargemaakt voor verzending. Daarvoor wordt hij opgedeeld in blokjes van m bits. Met elk van deze blokjes laat men een complex symbool corresponderen. De verzameling van mogelijke complexe symbolen is de constellatie. Deze bevat 2^m symbolen, waarvoor geldt dat (normeringsvoorwaarde)

$$\mathbb{E}[|a_k|^2] = 1 \tag{1}$$

  De sequentie van complexe symbolen $\{a_k\}$ wordt vervolgens op pulsen $p(t)$ geplaatst en over het kanaal verstuurd. We gebruiken een square-root raised cosine pulse in dit project. De (enkelzijdige) bandbreedte van zo een puls is $\frac{1+\alpha}{2T}$, waarbij T de symboolduur en α de roll-off factor voorstelt. De zender creëert dus het volgende signaal:

$$x(t) = \sqrt{E_s} \sum_{k=0}^{K-1} a_k p(t - kT) \tag{2}$$

  waarbij E_s de verzonden energie per symbool voorstelt. Later zullen we de verzonden energie per informatiebit E_b als referentie gebruiken. Er geldt dat

$$E_s = m\frac{k}{n}E_b \tag{3}$$

  In het vervolg van dit project zullen we altijd de genormeerde symbolen met een genormeerde puls versturen, zodat $E_s = 1$. Het complexe basisbandsignaal $x(t)$ wordt in de laatste stap van de zender op een draaggolf met frequentie f_0 geplaatst.

$$s(t) = \sqrt{2}\Re[x(t)e^{j2\pi f_0 t}] \tag{4}$$

3.2 Kanaal

Het resulterende (reële) signaal wordt verzonden over het kanaal, die gemodelleerd wordt door middel van een schalingsfactor h_{ch} en reële Gaussiaanse ruis die wit is in een voldoende groot frequentie-interval. Deze ruis $w(t)$ heeft een spectrale dichtheid gelijk aan $N_0/2$. Het ontvangen signaal $r(t)$ kan geschreven worden als volgt:

$$r(t) = h_{\text{ch}}s(t) + w(t) \tag{5}$$

3.3 Ontvanger

Aan de ontvanger wensen we het signaal $r(t)$ te bemonsteren. Daarom wordt het ontvangen signaal eerst gefilterd door een ideaal laagdoorlaatfilter[2]. Vervolgens wordt de uitgang van dit filter bemonsterd aan een debiet $\frac{N_s}{T} = \frac{1}{T_s}$, met $N_s \in \mathbb{N}$. De monsterwaarden aan de ontvanger kunnen we dan noteren als

$$r'_l = r\Big(l\frac{T}{N_s}\Big) = h_{\text{ch}} s(l\frac{T}{N_s}) + n_l \tag{6}$$

met

$$\mathbb{E}[n_{l_1} n_{l_2}] = \sigma^2 \delta_{l_1 - l_2} = \frac{N_0 N_s}{2T} \delta_{l_1 - l_2} \tag{7}$$

We wensen nu de verzonden informatie van de gebruiker te bepalen. Hiervoor voeren we de volgende stappen uit:

- Het bemonsterde signaal wordt gedemoduleerd met draaggolffrequentie f_0. In ons model veronderstellen we dat de demodulator niet perfect is en een extra rotatie θ introduceert. We bekomen

  $$r_l = \sqrt{2} r'_l e^{-j(2\pi f_0 l T_s + \theta)} \tag{8}$$

  Deze monsterwaarden worden vervolgens door een digitaal ontvangerfilter met impuls antwoord $\{p_l\}$ gestuurd dat gematched is aan de zenderpuls. De uitgang van dit filter wordt dan gegeven door

  $$y_n = T_s \sum_{l=-\infty}^{\infty} p_l r_{n-l} \tag{9}$$

  De resulterende sequentie $\{y_l\}$ bevat nog N_s monsterwaarden per symboolinterval. Na decimatie met een factor N_s verkrijgen we de sequentie $\{z_l\}$. De decisievariabele u_k wordt bekomen door z_k te schalen met factor $\frac{e^{j\hat{\theta}}}{\hat{h}_{\text{ch}}}$, i.e.,

  $$u_k = \frac{z_k}{\hat{h}_{\text{ch}}} e^{j\hat{\theta}} \tag{10}$$

  hierbij is $\hat{h}_{\text{ch}}$ de geschatte schalingsfactor van het kanaal en $\hat{\theta}$ de geschatte fase van de modulator. Ideaal zijn deze perfect gekend door de ontvanger. Indien de energie van de zenderpuls genormeerd is ($\int_{-\infty}^{+\infty} |P(f)|^2 df = 1$), geldt dat

  $$u_k = a_k + \tilde{w}_k \tag{11}$$

  waarbij $\tilde{w}_k$ complexwaardige witte Gaussiaanse ruis voorstelt met $\mathbb{E}[\tilde{w}_k (\tilde{w}_k)^*] = \frac{N_0}{|\hat{h}_{\text{ch}}|^2}$. Voor de overblijvende u_k zoeken we nu het dichtstbijzijnde constellatiepunt $\hat{a}_k$. Hieruit vinden we de m corresponderende bits terug via demapping.

- Vervolgens zal de decoder fouten in de ontvangen woorden corrigeren.
- Uiteindelijk zal de ontvanger de ontvangen bitsequentie decomprimeren en kan de afbeelding gereconstrueerd worden.

4 Opgave

4.1 Kwantisatie (*Quantization.m)*

Omdat de originele afbeelding ('ProjectA.mat') resulteert in behoorlijk wat geheugen zullen we deze grote datastroom beperken door de figuur te kwantiseren met 8 reconstructieniveaus (M=8) en dit met zo min

[2] Om te voldoen aan het bemonsteringstheorema van Shannon kiezen we een bandbreedte gelijk aan $\frac{N_s}{2T}$

mogelijk verlies van informatie. Een maat voor dit verlies is de GKD (gemiddeld kwadratische distorsie). Deze is gegeven door

$$GKD = \sigma_e^2 = \sum_{i=1}^{M} \int_{r_{i-1}}^{r_i} (q_i - u)^2 f_U(u)\, du. \tag{12}$$

met kwantisiatiedrempels $r_0 < r_1 < ... < r_M$ en reconstructieniveaus $q_1 < q_2 < ... < q_M$. Bij het ontwerp van de kwantisator gebruiken we de trapfunctie gebaseerd op het histogram van de samplewaarden als waarschijnlijkheidsdichtheidsfunctie (w.d.f) $f_U(u)$, i.e.,

$$f_U(u) = \begin{cases} N_U(\lfloor u \rfloor) & 0 \leq u \leq 255 \\ 0 & \text{elders} \end{cases} \tag{13}$$

waarbij $N_U(u)$ gelijk is aan de verhouding van het aantal keer dat de pixelwaarde u in de figuur voorkomt tot het totaal aantal pixels. Deze w.d.f $f_U(u)$ is, waar nodig, beschikbaar als anonieme functie in de variable *distr*.

- Maak een figuur van de distributie $f_U(u)$. Plot hierbij tevens een genormaliseerd histogram van de originele pixelwaarden om na te gaan of de gegeven anonieme functie inderdaad correct is. Doe dit in de functie *plot_ distribution().*

Beschouw nu eerst een uniforme kwantisator(*optimal_ linear_ quantizer()*) bepaald door symmetriepunt $x_0 = 127.5$ en stapgrootte Δ: de kwantisiatiedrempels zijn bijgevolg gegeven door $r_i = x_0 + \frac{(2i-M)\Delta}{2}$ voor $i = 1, 2, ..., M-1$ en de reconstructieniveaus door $q_i = x_0 + \left(i - \frac{M+1}{2}\right)\Delta$ voor $i = 1, 2, ..., M$. Zoals beschreven in de cursus, kunnen we bij een uniforme kwantisator de GKD opsplitsen in een granulaire bijdrage $\sigma^2_{\text{e,gr}}$ en een overlaadbijdrage $\sigma^2_{\text{e,ol}}$:

$$\sigma_e^2 = \sigma^2_{\text{e,gr}} + \sigma^2_{\text{e,ol}}, \tag{14}$$

met

$$\sigma^2_{\text{e,gr}} = \sum_{i=1}^{M} \int_{q_i - \frac{\Delta}{2}}^{q_i + \frac{\Delta}{2}} (q_i - u)^2 f_U(u)\, du \tag{15}$$

en

$$\sigma^2_{\text{e,ol}} = \int_{r_0}^{q_1 - \frac{\Delta}{2}} (q_1 - u)^2 f_U(u)\, du + \int_{q_M + \frac{\Delta}{2}}^{r_M} (q_M - u)^2 f_U(u)\, du. \tag{16}$$

- Maak een figuur die het verloop van σ_e^2, $\sigma^2_{\text{e,gr}}$ en $\sigma^2_{\text{e,ol}}$ weergeeft in functie van de stapgrootte Δ. Verklaar wat je ziet en bepaal, op basis van de figuur, de optimale stapgrootte Δ_{opt} die de totale GKD minimaliseert.
- Bepaal volgende karakteristieken van de kwantisator met optimale Δ_{opt}:
  - De GKD.
  - Signaal-kwantisatieruis verhouding (SQR) in dB. Deze is gedefinieerd als de verhouding van de variantie van de bronuitgang σ_U^2 en de GKD.
  - De waarschijnlijkheden horend bij de verschillende reconstructieniveaus en de bijhorende entropie H. Deze is gedefinieerd als $-\sum_{i=1}^{M} p_i \log_2(p_i)$ en is een maat voor de gemiddelde informatie-inhoud van een TG. Welk gevolg heeft deze waarde van de entropie voor optimale broncodering?
- Plot nu opnieuw de distributie $f_U(u)$ waarbij de bekomen kwantisatiedrempels en reconstructieniveaus duidelijk zijn aangegeven.
- Wat zou je verder kunnen doen om deze uniforme kwantisator te verbeteren zonder het uniforme karakter te verliezen en zonder M te verhogen.

Bij de optimale kwantisator (minimale GKD) liggen de kwantisatiedrempels en reconstructieniveaus vaak niet op vaste afstand van elkaar. Bijgevolg is deze optimale kwantisator niet uniform. Lloyd en Max hebben twee nodige voorwaarden geformuleerd opdat een kwantisator optimaal zou zijn. Een algoritme om een kwantisator te ontwerpen die voldoet aan de Lloyd-Max voorwaarden kan je vinden in Appendix A. Implementeer zelf dit algoritme in de functie *Lloyd_ max_ quantizer()*.

- Maak een figuur van de GKD in functie van het aantal iteratie en bespreek. Is de resulterende GKD kleiner dan de GKD bekomen met de uniforme kwantisator? Hoeveel dB winst maakt de Lloyd-Max kwantisator in termen van SQR ten opzichte van de uniforme kwantisator?
- Plot de entropie van het gekwantiseerde signaal in functie van het aantal iteraties. Verklaar wat je ziet. Voeg hiertoe ter referentie de entropie toe van een uniform verdeeld discrete toevalsgrootheid (TG) met een alfabet van M letters (dit is niet de entropie van de uniforme kwantisator!). Geef ook de waarschijnlijkheden horende bij de verschillende reconstructieniveaus.
- Plot nu opnieuw de distributie $f_U(u)$ waarbij de bekomen kwantisatiedrempels en reconstructieniveaus van de Lloyd-Max kwantisator duidelijk zijn aangegeven. Komt dit overeen met de verwachtingen?

Een suboptimale techniek om niet-uniforme kwantisators te ontworpen die vaak wordt toegepast in de praktijk is compansie. Hierbij passen we uniforme kwantisatie toe op een vervormde bronuitgang $g(u)$, waarna de inverse bewerking $g^{-1}(.)$ wordt uitgevoerd op de bekomen kwantisatiedrempels en reconstructieniveaus. In dit project gebruiken we een compansiekarakteristiek die ervoor zorgt dat de vervormde bronuitgang uniform verdeeld is in [-0.5,0.5]:

$$g(u) = \int_{-\infty}^{u} f_U(x)\,\mathrm{d}x - 0.5 \tag{17}$$

Implementeer deze kwantisator in *companding_ quantizer()* en maak hierbij gebruik van de functie *inverse()* die numeriek de inverse $g^{-1}(y)$ berekent voor een specifieke y.

- Maak een figuur van $g(u)$ in functie van u. Verklaar het verloop van deze functie.
- Geef dezelfde karakteristieken van de kwantisator als voor de uniforme kwantisator (GKD, SQR, entropie en de waarschijnlijkheden van de reconstructieniveaus).
- Plot nu opnieuw de distributie $f_U(u)$ waarbij de bekomen kwantisatiedrempels en reconstructieniveaus van de compansiekwantisator duidelijk zijn aangegeven. Zijn de prestaties van deze kwantisator beter of slechter dan deze van de uniforme kwantisator? Is dit altijd zo?

Implementeer tenslotte de functie *quantize()*, die de eigenlijke kwantisatie implementeert, en vergelijk visueel de oorspronkelijke figuur met de gekwantiseerde figuur voor de verschillende kwantisators. Wat bemerk je? Gebruik hiervoor de functie *show_figures()*.

4.2 Kanaalcodering (*Channel_ Coding.m)*

Omdat we verwachten dat sommige bits foutief ontvangen zullen worden, voegen we nu extra redundante bits toe om deze eventuele fouten te kunnen verbeteren aan de ontvanger. In dit deel van het project werk je best met een zelf gegeneerde random bitsequentie. In het bestand *Channel_ Coding.m* vind je de verschillende functies die je moet implementeren. Als praktische toepassing zullen we in deze sectie veronderstellen dat we een kanaal hebben dat gemodelleerd is door een BSC met p=0.05. Het criterium bij het ontwerp van de code en het retransmissieprotocol is dat de kans op een decodeerfout in een blok van 10 informatiebits kleiner moet zijn dan 0.001.

Eerst onderzoeken we de prestaties van de (14,10) lineaire blokcode met checkmatrix

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

- Bepaal voor deze code
  - de generatormatrix $\boldsymbol{G}$,
  - de minimale Hammingafstand,
  - het foutdetecterend en foutcorrigerend vermogen en
  - de syndroomtabel.

  Implementeer nu de encoder en de decoder van deze code in respectievelijk *Encode_ outer()* en *Decode_ outer()*. De encodering gebeurt aan de hand van de generatormatrix ($\boldsymbol{c} = \boldsymbol{bG}$) en de decodering via het syndroom ($\boldsymbol{s} = \boldsymbol{rH}^T$). In de decoder is het de bedoeling dat je zowel volledige decodering als zuivere foutdetectie implementeert. Hierbij stelt de outputvector 'bitsdec' de gedecodeerde bits na volledige decodering voor en is het $i$de element van vector 'error_det' gelijk aan 1 als er in het $i$de codewoord een fout is gedetecteerd bij zuivere foutdetectie en gelijk aan 0 in het andere geval (het bijhorende gedecodeerde informatiewoord kan dan gevonden worden in 'bitsdec').

- Bepaal in het geval van volledige decodering de kans op een decodeerfout, p_e. Geef zowel de exacte formule als een benadering voor $p \ll 1$. Voer tevens simulaties uit om de kans op een decodeerfout experimenteel te bepalen. Maak een figuur waarin de gesimuleerde resultaten en analytische uitdrukking met elkaar vergeleken worden als functie van p, waarbij p gaat van 0.001 tot 0.5. Gebruik hiervoor een dubbellogaritmische schaal.

- Is deze code krachtig genoeg om het ontwerpcriterium te halen bij $p = 0.05$? Vanaf welke p is dit criterium wel voldaan?

- Geef een analytische uitdrukking (exact + benadering) voor de kans op decodeerfalen, p_f, en de kans op een niet-detecteerbare fout, p_m, in het geval van zuivere foutdetectie. Hoe groot zijn beide kansen voor $p = 0.05$?

Om de kans op een decodeerfout te verminderen kan je in de praktijk gebruik maken van retransmissieprotocols, waarbij de decoder via een NACK bericht een retransmissie kan aanvragen. In een eerste retransmissieprotocol gebruiken we de code van hierboven. De decoder doet aan zuivere foutdetectie en genereert een NACK bericht als hij een fout detecteert, waarna een retransmissie wordt aangevraagd. We veronderstellen dat het aantal retransmissies beperkt is tot $T_{\max}$ en dat wanneer het maximaal aantal transmissies bereikt wordt, de decoder volledige decodering toepast.

- Schrijf de formule voor de kans dat er een decodeerfout optreedt na maximum $T_{\max}$ retransmissies, p_e^{ARQ}, als functie van p_e (volledige decodering), p_f en p_m (zuivere foutdetectie).

- Maak aan de hand van simulaties een figuur die de kans p_e^{ARQ1} weergeeft als functie van $T_{\max}$, waarbij $T_{\max}$ gaat van 0 tot en met 15 bij $p = 0.05$. Vergelijk met het analytische resultaat. Maak hierbij ook een figuur van het gemiddeld informatiedebiet in functie van $T_{\max}$ (het totaal aantal informatiebits gedeeld door het totaal aantal verzonden bits). Bespreek de bekomen figuren. Waardoor zijn de prestaties van dit retransmissieprotocol beperkt? Is het mogelijk om met deze methode te voldoen aan het designcriterium? Indien ja, hoeveel retransmissies moeten er minimaal toegelaten worden?

Tip: Bij de simulaties van de retransmissieprotocols is het niet aan te raden om dit codewoord per codewoord te doen zoals in de praktijk, maar is het efficiënter om transmissie per transmissie te simuleren. Eerst simuleer je een grote aantal informatiewoorden in de eerste transmissie. Aan de hand van de variable 'error_det' kan je dan nagaan welke codewoorden er een retransmissie nodig hebben en zo maak je een nieuwe bitstring die moet verzonden worden in de volgende transmissie.

In de praktijk wordt een retransmissieprotocol vaak geïmplementeerd aan de hand van 2 afzonderlijke codes: een inwendige code voor foutdetectie en een uitwendige code voor foutcorrectie. Hierbij wordt het informatiewoord eerst gecodeerd met de inwendige code en vervolgens wordt het verkregen codewoord nogmaals gecodeerd met de uitwendige code. Na transmissie over het kanaal wordt de ontvangen bitsequentie eerst volledig gedecodeerd door de uitwendige code, waarna de decoder van de inwendige code

zuivere foutdetectie uitvoert. Geeft deze laatste een NACK, dan wordt een retransmissie voor dit woord aangevraagd. Als uitwendige code zullen we bovenstaande (14,10) code gebruiken.

- Als inwendige code zullen we een CRC code (zie cursus sectie 9.6.4) gebruiken, aangezien deze typisch een zeer lage kans op een niet-detecteerbare fout heeft. Hierbij wordt is het codewoord de som van de informatieveelterm $b_{(n)}(x)$ en de restveelterm bij deling van $b_{(n)}(x)$ door een CRC-veelterm $g(x)$. Bij zuivere foutdetectie aan de ontvanger wordt dan nagegaan of de veeltermvoorstelling van het ontvangen woord een veelvoud is van $g(x)$. Toon aan dat elke CRC code een lineaire code is. Implementeer voor deze code de encoder en decoder voor zuivere foutdetectie in respectievelijk *Encode_ inner()* en *Decode_ inner()*.

Tip: Om in Matlab veeltermen met elkaar te vermenigvuldigen en te delen kan je gebruik maken van respectievelijk 'conv()' en 'deconv()'. Je mag er verder vanuit gaan dat in dit project de dimensie van de code (relatief) klein zal zijn. Hierdoor kan het efficiënt zijn om gebruik te maken van de codetabel voor codering en decodering.

- De eerste inwendige code die we gebruiken is de CRC-5 code met $g(x) = (x+1)(x^4+x+1) = x^5+x^4+x^2+1$. Bepaal voor deze code
  - de dimensie k,
  - het gegarandeerd burstfoutdetecterend vermogen,
  - minimale Hammingafstand, foutcorrigerend en foutdetecterend vermogen,
  - de kans op een niet-detecteerbare fout (in het bijzonder bij $p = 0.05$) en
  - Kan deze code het foutpatroon (1010101010) detecteren? Hoe kan je dit onmiddellijk zien aan de structuur van $g(x)$.
- Simuleer voor dit retransmissieprotocol de kans op een decodeerfout, p_e^{ARQ2a}, bij $p = 0.05$. Merk op dat, in tegenstelling tot het vorige retransmissieprotocol, voor de laatste retransmissie de inwendige code zuivere foutdetectie doet. Alle ontvangen woorden die geen codewoorden zijn, worden als foutief beschouwd. Maak een figuur van p_e^{ARQ2a} als functie van $T_{\max}$ (van 0 tot en met 6) met daarop de simulatieresultaten samen met de benadering (zie cursus) $p_e^{ARQ} \approx (p_{out:e})^{T_{\max}+1}$, waarbij $p_{out:e}$ de kans op een decodeerfout van de uitwendige code voorstelt. Geef ook de figuur van het informatiedebiet als functie van $T_{\max}$. Bespreek en verklaar beide figuren. Is het mogelijk om met dit retransmissieprotocol te voldoen aan het ontwerpcriterium? Zo ja, hoeveel retransmissies zijn er minimaal nodig?
- Een tweede CRC code die we onderzoeken, is een CRC-8 code met $g(x) = (x+1)(x^7+x^3+1) = x^8+x^7+x^4+x^3+x+1$. Bepaal voor deze code opnieuw
  - de dimensie k,
  - het gegarandeerd burstfoutdetecterend vermogen,
  - minimale Hammingafstand, foutcorrigerend en foutdetecterend vermogen,
  - de kans op een niet-detecteerbare fout (in het bijzonder bij $p = 0.05$) en
  -
  - Simuleer ook voor dit retransmissieprotocol de kans op een decodeerfout, p_e^{ARQ2b}, bij $p = 0.05$. Maak opnieuw een figuur van p_e^{ARQ2b} als functie van $T_{\max}$ (van 0 tot en met 6) met daarop de simulatieresultaten en de benadering. Toon ook de plot van het informatiedebiet als functie van $T_{\max}$. Wat is het verschil met de eerder besproken CRC code? Is het mogelijk om met dit retransmissieprotocol te voldoen aan het ontwerpcriterium? Zo ja, hoeveel retransmissies zijn er minimaal nodig?

4.3 Modulatie en detectie (*PHY.m*)

Ook dit deel van het project wordt best uitgevoerd met een random gegenereerde bitsequentie. In het bestand *PHY.m* zal je de functies vinden die je moet implementeren.

- Eerst gaan we de mapper implementeren. Deze dient volgende constellaties te ondersteunen: BPSK, 4QAM en 4PAM. Implementeer de functie *mapper()* die een bitsequentie omzet naar een symboolsequentie. Maak hiervoor gebruik van Gray-Mapping: hierbij worden elke twee groepjes van m bits die slechts 1 bit verschillen afgebeeld op twee complexe symbolen die op minimale Euclidische afstand van elkaar liggen in de constellatie. Implementeer ook de demapper (*demapper()*) die de inverse operatie uitvoert.

- Nu moeten de bekomen symbolen geplaatst worden op zenderpulsen. Maak gebruik van de square-root raised cosine pulse (gebruik de functie PHY.m.pulse()) met symboolinterval $T = 1\mu s$ en roll-off $\alpha = 1$. Plaats vervolgens het basisbandsignaal op een draaggolf met frequentie $f_0 = 3\,\text{MHZ}$ (zie (4)). In een praktisch systeem wordt het voorgaande in continue tijd uitgevoerd. In een Matlab-simulatie moet echter alles voorgesteld worden door middel van monsterwaarden. Daarom gaan we $s(t)$ bemonsteren aan $\frac{N_s}{T}$. Waaraan moet N_s voldoen opdat aan het bemonsteringstheorema van Nyquist/Shannon is voldaan? Kies voor de rest van het project de minimale N_s die hieraan voldoet. In theorie heeft een square root raised cosine pulse een oneindige duur, maar deze pulse sterft snel uit na enkele symboolperioden. Daarom kunnen we $p(t)$ beperken tot $2L_F$ symboolperioden: $p(t) \neq 0, |t| \leq L_f T$. Voor dit project beperken we p(t) tot 10 symboolperioden, of $10N_s + 1$ monsterwaarden ($L_F = 5$). Maak een figuur van de Fouriergetransformeerden van de originele puls en van de afgeknotte puls en vergelijk[3]. Denk goed na over de juiste schaal voor deze figuur (lineair/logaritmisch). Wat is het gevolg van het afknotten? Implementeer deze stappen in *modulatie*().

- Het banddoorlaatsignaal $s(t)$ wordt verstuurd over een kanaal (schalingsfactor h_{ch} en reële witte Gaussiaanse ruis met spectrale dichtheid $\frac{N_0}{2}$). De samples van het ontvangen signaal $\mathbf{r}$ worden gegeven door

$$r\Big(l\frac{T}{N_s}\Big) = h_{\text{ch}}\, s\Big(l\frac{T}{N_s}\Big) + n_l \tag{18}$$

  De variantie van de ruis n_l wordt gegeven door (7). Implementeer nu de functie *channel()* die de samples van het ontvangen signaal genereert. Stel h_{ch} in het volledige project gelijk aan 1.

- Vervolgens worden de monsterwaarden gedemoduleerd met $f_0 = 3\,\text{MHz}$ ($\theta = \frac{\pi}{16}$) en aan een digitaal filter, gematched aan de zenderdpuls, aangelegd. Voor een square-root raised cosine pulse zijn zender-en ontvanger puls/filter identiek. De uitgang van dit matched filter wordt gegeven door uitdrukking (9) maar waarbij de sommatie-index l loop van $-L_F N_s$ tot $L_F N_s$. Deze operatie kan gemakkelijk geïmplementeerd worden in Matlab met behulp van de functie *conv()*:

$$\boldsymbol{y} = T_s conv(\boldsymbol{r}, \boldsymbol{p}) \tag{19}$$

  We beperken p(t) opnieuw tot 10 symboolperioden, dus $L_F = 5$. Al deze stappen worden geïmplementeerd in de functie *demodulate*(). Verwacht je dat $\mathbf{y}$ een reëel of complex signaal is voor de BPSK constellatie? Verklaar.

- Maak een oogdiagram op basis van van het reële deel de ontvangen sequentie $\mathbf{y}$ (zonder ruis, voor de BPSK constellatie). Maak daarvoor gebruik van de standaard Matlab-functie *eyediagram()* en laat het overgangsverschijnsel aan het begin en einde van de sequentie weg[4]. Stel twee symboolperioden voor in het oogdiagram. Bespreek het verband tussen de inter-symbool interferentie en het tijdstip waarop de decimatie wordt uitgevoerd. Onderzoek hierbij ook de invloed van de roll-off factor (tussen 0 en 1). Maak nu ook een oogdiagram voor de 4-PAM constellatie met een roll-off factor gelijk aan 1. Wat is het verschil met BPSK? Voer nu de decimatie uit op de correcte tijdstippen.

[3] Zie ook: sectie A1.8.2 in extra document over 'simulatie van digitale communicatie'
[4] zie ook: sectie A1.9.1 in extra document over 'simulatie van digitale communicatie'

Implementeer de decimator(*downsample()*). Stel voor de rest van dit project de roll-off factor gelijk aan 1.

- Implementeer nu de functie *make_ decision_ variable*() die het gedecimeerde signaal schaalt met de factor $\hat{h}_{\text{ch}}$ en de fase compenseert met een rotatie over hoek van -$\hat{\theta}$. Veronderstel voorlopig dat het kanaal en demodulator perfect gekend is ($\hat{h}_{\text{ch}} = h_{\text{ch}}$ en $\hat{\theta} = \theta$). Maak nu een scatterdiagram (voor BPSK en 4QAM) van het signaal $\{u_k\}$ en onderzoek de invloed van E_b/N_0, maak hiervoor gebruik van de functie *scatter*().
- Op basis van het signaal u_k kan de ontvanger nu voor elke waarde het dichtstbijzijnde constellatiepunt zoeken. Dit noemt men harde decisie. Implementeer dit in *hard_ decisions()* en verifieer dat voor $N_0 = 0$ de juiste constellatiepunten worden teruggevonden.
- Bepaal nu de BER voor alle drie de constellaties aan de hand van simulaties voor verschillende warden van E_b/N_0 en maak een figuur[5]. Beschouw enkel het interval van E_b/N_0 waarvoor $10^{-1} > BER > 10^{-4}$ (ongeveer). Vergelijk het bekomen resultaat met de curves uit de cursus en waar mogelijk met de analytische uitdrukking.

Tot nu toe hebben we verondersteld dat de ontvanger gebruikt maakt van de correcte parameters (h_{ch}, θ) om de decisie uit te voeren. In de praktijk zijn deze parameters echter niet gekend en zullen die moeten geschat worden door de ontvanger wat kan leiden tot kanaalschattingsfouten. In dit project zullen we nagaan wat het gevolg is van de een amplitudeschattingsfout en faseschattingsfout.

- Eerst bestuderen we de invloed van een amplitudeschattinsfout bij zowel de BPSK als 4PAM constellatie. Maak hiervoor een scatterdiagram in afwezigheid van ruis voor beide constellaties bij $\epsilon = 0.1$ met $\epsilon = \frac{\hat{h}_{\text{ch}} - h_{\text{ch}}}{h_{\text{ch}}}$. Wat bemerk je? Maak nu opnieuw voor beide constellaties een figuur met daarop de BER-curves voor volgende waarden van $\epsilon = 0, 0.1, 0.2$. Verklaar het resultaat.
- Vervolgens gaan we na wat een het effect is van een fasefout als we de 4-QAM constellatie gebruiken. Maak ook hier een scatterdiagram in afwezigheid van ruis bij $\phi = \frac{\pi}{16}$ met $\phi = \hat{\theta} - \theta$. Wat bemerk je? Maak vervolgens opnieuw een figuur met de BER-curves voor de volgende waarden van $\phi = 0, \frac{\pi}{16}, \frac{\pi}{8}, \frac{\pi}{4}$. Verklaar het resultaat.

4.4 Volledig systeem

Kwantiseer de oorspronkelijke figuur met de Lloyd-Max kwantisator. Om de bekomen sequentie van reconstructieniveaus om te zetten naar bits kan je gebruik maken van een vaste-lengte codes waarbij elk reconstructieniveau overeenstemt met een sequentie van een gelijk aantal bits. Bepaal nu de E_b/N_0-waarde voor de BPSK constellatie waarvoor geldt dat BER≈ 0.05. Verstuur nu de afbeelding over een het kanaal op een draaggolf met frequentie $f_0 = 3\,\text{MHz}$ bij deze signaal-ruis voor volgende scenario's:

1. ongecodeerde transmissie.
2. gecodeerde transmissie met de lineaire blokcode zonder retransmissies.
3. retransmissieprotocol 1 met de lineaire blokcode en retransmissies.
4. retransmissieprotocol 2 met als inwendige code de CRC-5 code en de lineaire blokcode als uitwendige code.
5. retransmissieprotocol 2 met als inwendige code de CRC-8 code en de lineaire blokcode als uitwendige code.

Bepaal voor de 5 gevallen telkens het gemiddeld aantal bits dat verstuurd moet worden per pixel van de originele afbeelding en de gemiddelde kwadratische afwijking (GKA) tussen de gereconstrueerde en originele, niet-gekwantiseerde afbeelding. Vul daarvoor Tabel 3 aan. Bespreek tevens het visuele verschil tussen de ontvangen afbeeldingen van de verschillende scenario's

[5] zie ook: sectie 1.5 in extra document over 'simulatie van digitale communicatie'. Hierin worden twee methodes besproken om tot betrouwbare resultaten te komen.

scenario	1	2	3	4	5
gemiddeld aantal bits per pixelwaarde					
gemiddeld kwadratische afwijking (GKA)					

Tabel 2: Resultaten

Appendix A: Lloyd-Max algoritme

Voor elke optimale kwantisator zijn de Lloyd-Max voorwaarden voldaan. De voorwaarden zijn als volgt:

$$r_i = \frac{q_i + q_{i+1}}{2} \qquad i = 1, 2, ..., M-1$$
$$q_i = \frac{\int_{r_{i-1}}^{r_i} u f_U(u)\, du}{\int_{r_{i-1}}^{r_i} f_U(u)\, du} \qquad i = 1, 2, ..., M$$

waarbij r_0 en r_M gegeven zijn door het domein van $f_U(u)$. Uit dit stelsel van niet-lineaire voorwaarden is het echter niet eenvoudig om rechtstreeks de waardes voor r_i en q_i te bepalen. De optimale waarden kunnen echter wel benaderd worden door gebruik te maken van een iteratieve methode die om de beurt de grenzen en de niveaus opnieuw berekent. Dit proces gaat door tot wanneer de vermindering in distorsie voldoende klein is. Deze methode heet het Lloyd-Max algoritme en de stappen worden hieronder gegeven:

1. Initialiseer alle q_i (kies zelf), stel $l = 1$.
2. Update alle kwantisatiedrempels:$r_i = \frac{q_i+q_{i+1}}{2} \quad i = 1, 2, ..., M-1$
3. Update alle kwantisatieniveaus:$q_i = \frac{\int_{r_{i-1}}^{r_i} u f_U(u)\, du}{\int_{r_{i-1}}^{r_i} f_U(u)\, du} \quad i = 1, 2, ..., M$
4. Bepaal de distorsie: $\sigma^2_{\mathrm{e,l}} = \int_{r_0}^{r_M} (F_Q(u) - u)^2 f_U(u)\, du$
5. Indien $\frac{\sigma^2_{\mathrm{e,l\text{-}1}} - \sigma^2_{\mathrm{e,l}}}{\sigma^2_{\mathrm{e,l-1}}} < \epsilon$, stop. Zo niet, stel $l = l + 1$ en ga naar stap 2.

Hierbij is l de iteratie index en $\sigma^2_{\mathrm{e,l}}$ de GKD na de $l$de iteratie. De stopvoorwaarde wordt bepaal door ϵ. Kies deze voldoende klein.