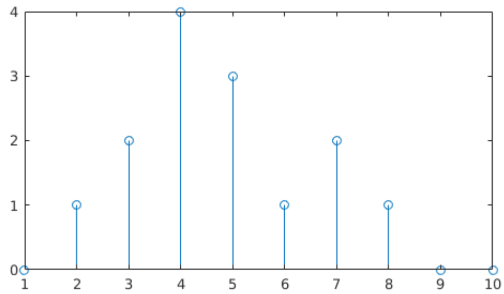
Opgavebeskrivelse

Til dette selvstudium skal vi arbejde med signalbehandling. Et diskret tidssignal er en vektor $\boldsymbol{x} \in \mathbb{R}^n$, der beskriver opførslen af et fænomen over tid. Det kan f.eks. være lyd (tale, musik, etc.), geofysiske målinger, sensor målinger, computterrelateret kommunikation (f.eks. fra en mobiltelefon) eller medicinske signaler (blodtryksmåling, kardiogrammer etc.). Et simpelt eksempel på et signal er

$$\boldsymbol{x} = \begin{bmatrix} 0 & 1 & 2 & 4 & 3 & 1 & 2 & 1 & 0 & 0 \end{bmatrix}^T.$$

Da $\boldsymbol{x} \in \mathbb{R}^{10}$, så siger vi, at $\boldsymbol{x}$ består af 10 samples. Nedenfor ses et plot af signalet:



Virkelige signaler er imidlertid meget mere komplicerede at arbejde med. Almindeligvis fremkommer diskrete tidssignaler ved, at man sampler et kontinuert signal med en bestemt samplingfrekvens, der angiver antallet af samples per sekund. Hvis man f.eks. anvender almindelig CD-kvalitet til indspilning af et musiksignal, så bruges der 44100 samples per sekund. Diskrete tidssignaler består derfor ofte af flere millioner samples.

På grund af signalernes størrelser, så er det væsentligt at komprimere dem, så de fylder mindst muligt. Dette skal selvfølgelig gøres ordentligt, så de komprimerede signaler ikke mister for meget information. Vi skal se på, hvordan man kan komprimere diskrete tidssignaler vha. den diskrete cosinus-transformation (DCT). Den DCT transformerer et signal til en sparse repræsentation (dvs. en vektor med mange 0'er), der nemt kan komprimeres. Den DCT anvendes i mange sammenhænge og ligger bl.a. til grund for MP3-komprimering. Selvstudiet er opdelt i tre mindre delopgaver:

- **Delopgave 1:** Her arbejder vi med sparse vektorer og ser vi på, hvordan Matlab kan komprimere sådanne vektorer, så de fylder mindst muligt.
- **Delopgave 2:** Her gennemgår vi forskellige egenskaber ved den DCT. Denne delopgave skal løses i hånden.
- **Delopgave 3:** Her anvender vi teknikkerne fra delopgavene 1 og 2 til at komprimere et virkeligt diskret tidssignal, der består af flere hundrede tusind samples.

Husk til delopgaverne 1 og 3 at tilføje kommentarer til jeres Matlab kode. Hvis I på et senere tidspunkt får brug for at kigge på koden igen, så er det vigtigt, at I har brugt fornuftige variabelnavne og kommentarer til at gøre koden læsbar.

Delopgave 1

Til denne delopgave skal vi se på, hvordan Matlab komprimerer sparse vektorer.

1. Lav et nyt Matlab-script. Indskriv og kør følgende kode

```
x = [1 zeros(1,20) 4 5 zeros(1,30)]';
stem(x)
```

   Vi kan se, at `x` er en sparse vektor, da mange af indgangene er 0.

2. For at få Matlab til at komprimere `x`, så skriver vi `x_sparse = sparse(x)` i scriptet. Hvis I nu skriver `x_sparse` i Matlabs "Command Window", så kan I se, at `x_sparse` kun indeholder information om de få indgange i `x`, der er forskellige fra nul.

3. Lad os nu se på, hvor meget hukommelse de to vektorer optager. Prøv at skriv `whos x` og `whos x_sparse` i Matlabs "Command Window". Hvilken af de to repræsentationer optager flest bytes?

4. Man skal selvfølgelig gerne kunne få den oprindelige vektor tilbage fra den sparse repræsentation. Udvid scriptet med `x_rec = full(x_sparse)`. I skulle nu gerne se, at `x_rec` blot rekonstruerer den oprindelige vektor `x`.

Delopgave 2

Til denne delopgave skal vi arbejde med den DCT. Opgaven skal løses i hånden, så I får ikke brug for Matlab. Den DCT er en $n \times n$ matrix C, hvis (i,j)'te indgang er givet ved

$$C_{i,j} = \sqrt{\frac{2}{n}} \cos\left(\frac{\pi}{4n}(2j-1)(2i-1)\right), \quad 1 \leq i, j \leq n.$$

Den DCT af $\boldsymbol{x} \in \mathbb{R}^n$ noteres som

$$\boldsymbol{y} = C\boldsymbol{x}.$$

For at belyse egenskaberne ved den DCT gennemgår vi en række mindre opgaver:

1. Vis, at

$$y_i = \sqrt{\frac{2}{n}} \sum_{j=1}^{n} x_j \cos\left(\frac{\pi}{4n}(2j-1)(2i-1)\right), \quad 1 \leq i \leq n.$$

2. Vis at C er symmetrisk.

3. Bestem C^T.

4. Man kan vise, at C er en ortogonal matrix (et bevis kan findes i [2]). Bestem C^{-1}. HINT: Brug resultaterne ovenfor og Sætning 6.9 i [1].

5. Man kalder ofte $\|\boldsymbol{x}\|$ for signalets energi. Vis at den DCT bevarer signalets energi i den forstand at $\|\boldsymbol{y}\| = \|\boldsymbol{x}\|$.

6. Vis at for $n = 2$, så er
$$C = \begin{bmatrix} \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) \\ \cos\left(\frac{3\pi}{8}\right) & -\cos\left(\frac{\pi}{8}\right) \end{bmatrix}.$$
Man kan vise, at $\cos^2(\pi/8) + \cos^2(3\pi/8) = 1$. Brug dette resultat til at bestemme C^2. Får I det forventede?

Delopgave 3

I denne delopgave skal vi komprimere et virkeligt diskret tidssignal. Det signal vi skal se på er et stykke tale, hvor en britisk mand reciterer en kort sætning. Den tilhørende lydfil 'speech.wav' er vedlagt som en del af materialet til dette selvstudium.

1. Lav et nyt Matlab script, hvor I indlæser og plotter signalet. Dette kan gøres ved:

```
[x,fs] = audioread('speech.wav');
plot(x)
```

   Her er `x` signalet og `fs` er samplingsfrekvensen.

2. Afspil signalet med `soundsc(x,fs)`.

3. Hvad er samplingfrekvensen? (antal samples per sekund)

4. Hvad er n? (længden af signalet) HINT: Brug `length(x)`.

5. Hvor mange sekunder varer signalet? (kan bestemmes ud fra samplingfrekvensen og n).

6. Vi kan se af plottet, at `x` ikke er en sparse vektor. Lad os nu undersøge om den DCT af `x` er sparse. Tilføj følgende kode til scriptet

```
y = dct(x,'type',4);
plot(y);
```

   Er `y` en sparse vektor?

7. Lad os undersøge, hvor mange 0'er `y` indeholder

```
numberOfZeros = numel(y) - nnz(y)
```

   Det viser sig altså, at `y` ikke indeholder nogen 0'er overhovedet — den har bare mange små værdier tæt på 0.

8. Vi udvælger nu de 87500 største værdier i `y` og sætter resten af indgangene lig nul:

```
[y_descend,index] = sort(abs(y),'descend');
coeff_needed = 87500;
y(index(coeff_needed:end)) = 0;
```

   Hvor mange 0'er indeholder `y` nu?

9. Lav en sparse repræsentation af `y` og kald den `y_sparse`. HINT: se delopgave 1.

10. Hvilken af de to vektorer `x` og `y_sparse` optager flest bytes? (Hvis I har lavet opgaven rigtigt, så vil I se, at `y_sparse` kun bruger halvt så mange bytes som `x`).

11. Vi ved fra delopgave 2, hvordan vi kan rekonstruere fra `y`. Dette gøres i Matlab ved følgende kode:

```
x_rec = idct(y,'type',4)
```

   Kan I høre forskel på `x` og `x_rec`?

References

[1] O. Geil. *Elementary Linear Algebra*. Pearson Education Limited, 2015.

[2] G. Strang. The discrete cosine transform. *SIAM Rev.*, 41(1):135–147, 1999.