Oppgåve 1: Vi leikar med lyd

Desse tre skripta finn de på på nettsida for intro-veka på ei eiga side for dei akutelle kode-snuttane.

PlayAndPlotSound.m

ReduceSampling.m

FilterFrequency.m

Køyr det første skriptet. Skriptet les inn musikk frå ei wav-fil som heiter *guitartune*. Dette er ei musikkfil som MATLAB har liggande frå før – men berre i Windows. De kan uansett godt byte ut denne med ein annan song de måtte ha liggande.

Skriptet les inn og spelar av musikken. I tillegg plottar det lydsignalet som musikken består av – både som funksjon av tida og som funksjon av frekvens. I det siste tilfellet plottar det både amplitude og fase. (Vi skal ikkje gå inn på detaljane her. De vil nok høyre meir om dette om de skal lære om signalbehandling seinare.)

Sjå litt på plotta og prøv å forstår kva dei viser. Kor mykje må du zoome inn på figur 1 for at det skal slutte å sjå «rotete» ut?

Kva skjer om du doblar eller halverar variabelen Fs i linje 7? Og kva om du gongar heile vektoren y med ein faktor? Prøv å gjere dette i skriptet –både med ein faktor som er større enn 1 og ein som er mindre enn 1.

Ta gjerne ein kjapp kikk på skriptet og sjå om de forstår kva som skjer. Noko av det er nokså avansert, noko av det er det ikkje. (Når det gjeld fft-funksjonen kan vi kort fortelle at denne tar signalet som funksjon av tid over til ein funksjon av frekvens i staden.)

Om du vil stoppe avspelinga av musikken, berre skriv clear i kommandovindauga. Dette fjernar også alle variablar frå minnet.

Vi skal no sjå om vi kan spare lagringsplass ved å redusere på informasjonen i lydfila. Kanskje inneheld lydfila mykje meir informasjon enn vi treng?

Alle først: Kor mange punkt har det diskrete lydsignalet? Eller sagt på ein annan måte: Kva verdi har Npkt i linje 14; Npkt = length(y) (ta bort semikolonet).

Kva om vi berre «brukar» annakvart av desse (set NewStep i linje 15 i ReduceSampling til 2). Kan de høyre at lyden blir dårlegare på noko vis? Kor mykje kan du redusere samplinga utan at det går ut over lydkvaliteten synest de? På kva måte blir lyden dårlegare? Svaret vil nok vere avhengig av kvaliteten på høgtalarane de brukar. Ser de forskjell når denne «reduserte» lyden blir plotta saman med den originale? (Her må de nok zoome igjen.)

Vi kan også redusere på informasjonen ved å kutte dei høgaste frekvensane. Dette skjer i det tredje skriptet, `FilterFrequency`. Veit de forresten kor høge tonar, gitt i kHz, ein reknar som høyrbare for eit menneske?

I linje 14 skal de sette verdien for `CutLimit_kHz`. Om denne er sett til 5, vil skriptet fjerne alle tonar med frekvens høgare enn 5 kilohertz. Igjen: Prøv å «tweeke» på denne for å finne ut kor lågt de kan gå ned før det går ut over kvaliteten på lyden.

Ser de nokon samanheng mellom desse to måtane å redusere informasjonen på?

Viss de skulle trenge meir å gjere på:

Ser de korleis de kan lage eit filter som kuttar ut dei *lågaste* frekvensane i staden for dei høgaste? (Det er ei ganske lita endring.)

Klarar de å *transponere* musikken?

Prøv å legge på støy (hint: `rand(1,N)`). Korleis høyrest *kvit støy* ut?

Korleis høyrest ei rein sinusbølge ut? Likninga for ei sinus-bølgje ser slik ut: `y=sin(2*pi*f*t)`, der `f` er frekvensen og `t` er ein vektor med dei aktuelle tidspunkta.

Oppgåve 2: Å knekke Cæsars chiffer.

Desse tre funksjonsfilene finn de på nettsida:

CaesarChipher.m

CharNumConvert.m

NumCharConvert.m

Opne dei og sjå litt på dokumentasjonen. Prøv å forstå kva dei gjer, og bruk dei til å kryptere eit par ord. Implementeringa er noko primitiv på den måten at ho berre tillét store bokstavar og ingen andre teikn – heller ikkje mellomrom. Bokstavane Æ, Ø og Å kan heller ikkje brukast her.

Del gruppa dykkar i to, og la kvar gruppe velje seg eit ord som de krypterar med Cæsars chiffer. (For ordens skuld: Bruk MATLAB-filene til å gjere krypteringa.) Gi det krypterte ordet til den andre undergruppa. Kven klarar å knekke den andre sin kode fortast?

Lag eit skript som de kan bruke til å knekke koden for eit generelt Cæsar-kryptert ord. Dette kan gjerast på den måten at ein lagar ei for-løkke over alle moglege alfabet-forskyvingar. For kvar forskyving: Skriv det «dekrypterte» ordet til skjerm, og avgjer om det gir meining. Legg gjerne inn ein pause for kvar iterasjon (sjekk gjerne MATLAB-dokumentasjonen for pause – altså skriv >> help pause i kommandovindauga).

Oppgåve 3: Litt om RSA-kryptering.

RSA er ein veldig mykje brukt krypteringsmetode, oppkalla etter Ron Rivest, Adi Shamir og Leonard Adleman, som kom opp med metoden. Metoden, som baserar seg på *talteori*, er alt for komplisert til at vi skal gå i detalj på den her. Men det sentrale punktet er at det tar lang tid å faktorisere tal – altså skrive tal som produkt av lågare tal – om ein ikkje kjenner til eve.

To omgrep står sentralt: *Største felles divisor* og *primtal*. Et **primtal** er eit tal som ikkje kan faktoriserast – altså som ikkje er deleleg med andre tal enn 1 og seg sjølv. Eksempel på slike er 2, 3, 5, 7 og 11. Alle heiltal har sin bestemte sekvens av primtal når dei er fullstendig faktoriserte.

For eit talpar *a* og *b* er **største felles divisor** det største talet som er faktor i begge. For 15 og 10 er det 5, for eksempel, for 9 og 3 er det 3 og for 7 og 3 er det 1.

I dei to oppgåvene de får nedanfor vil de ha nytte av round og floor-funksjonane i MATLAB; sjekk dokumentasjonen (>> help round) for å finne ut kva dei gjer.

Oppgåve 3a: *Å finne primtal*

Denne kodesnutten kan brukast for å avgjere om n, som må vere tileigna frå før, er eit primtal:

```
divisor=2;
while abs(n/divisor-round(n/divisor))>1e-13 & divisor<=round(sqrt(n))
  divisor=divisor+1;
end
if divisor>sqrt(n)
  disp([num2str(n), ' er eit primtal'])
end
```

Her har vi innført ei while-løkke. De kjenner kanskje til slike frå før? Uansett: Dei kombinerer eigenskapar frå både for-løkker og if-satsar; det som står mellom while og end blir gjentatt så lenge den logiske påstanden til høgre for «while» er sann.

Forsøk å forstå logikken i kodesnutten. Ser de at vi kan avgjere om n er eit primtal eller ikkje på denne måten? Kanskje ser det unødvendig komplisert ut? Spør gjerne!

Lag eit skript som finn alle primtal opp til ei eller anna øvre grense. Kopiér gjerne inn kodesnutten over. Kor mange primtal under 100 finst det? Om vi går opp til 1000, kor mange finn vi då?

Oppgåve 3b: *Euklids algoritme*

Største felles divisor for to tal, *a* og *b*, kan bestemmast ved Euklids algoritme. Den går ut på følgande (når *a* er det største av dei to tala):

1) La *r* vere resten du får når du deler *a* på *b*.

2a) Dersom divisjonen går opp, altså r=0, er b største felles divisor.

2b) Dersom divisjonen ikkje går opp, lat b bli din nye a og r bli din nye b og bestem r på nytt – på same måte som i punkt 1.

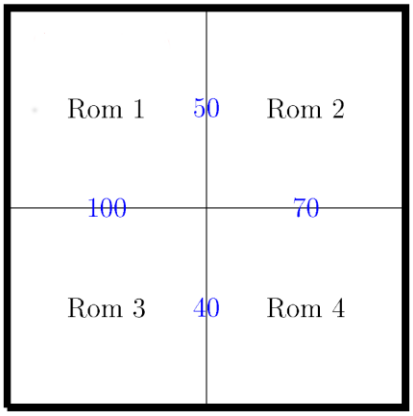
3) Gjenta punkt 2.

Hint: Punkt 1 kan for eksempel gjerast slik i MATLAB:

```
r=a-floor(a/b)*b;
```

Implementer denne metoden – anten som eit skript eller ei funksjonsfil. Test han på ein del tal-par. Kontroller gjerne at svara de får er dei same som MATLAB si eiga implementering av metoden; den heiter `gcd`.

Oppgåve 4: Temperaturfordelinga i fire rom



Skissa viser romma i eit lite hus med éin etasje. I kvart av romma er temperaturen T_1, T_2, T_3 og T_4. Vidare står der ein omn i kvart av romma. Desse leverar varme med effektane P_1, P_2, P_3 og P_4. T-ane er gitt i Celsius-grader og P-ane er gitt i Watt.

Ein dag er utetemperaturen T_{ute}=5°C. Omnen i rom 1 er den einaste som står på – og den står på 800 Watt (P_1=800, P_2=P_3=P_4=0). Kor mykje varme som vandrer gjennom kvar vegg er bestemt av temperaturforskjellen på kvar side av veggen og kor godt veggen er isolert. Frå rom 1 til rom 2, for eksempel, går det varmen Q_{12}=k_{12} (T_1-T_2), frå rom 1 og ut går varmen k_{ut}(T_1-T_{ute}). Vi tenkjer oss at k_{ut}=20 er felles for alle romma. Dei andre k-verdiane er (målt i Watt per Celcius-grader):

k_{12}=50

k_{13}=100

k_{24}=70

k_{34}=40

Rom 1 og 4 er ikkje i kontakt – heller ikkje rom 2 og 3.

Etter ei viss tid vil temperaturen i kvart rom ha stabilisert seg. Då må det vandre like mykje inn i som ut av kvart rom. For rom 1, for eksempel, får vi

$$k_{ute}(T_1-T_{ute}) + k_{12}(T_1-T_2) + k_{13}(T_1-T_3) = P_1 .$$

For rom 2 får vi på same måte at

$$k_{ute}(T_2-T_{ute}) + k_{12}(T_2-T_1) + k_{24}(T_2-T_4) = P_2 .$$

og så vidare.

Om vi set opp likningane for dei to siste romma og set inn tal, får vi følgande likningssystem:

$$\begin{aligned}
170\,T_1 - 50\,T_2 - 100\,T_3 &= 900\\
-50\,T_1 + 140\,T_2 - 70\,T_4 &= 100\\
-100\,T_1 + 160\,T_3 - 40\,T_4 &= 100\\
-70\,T_2 - 40\,T_3 + 130\,T_4 &= 100
\end{aligned}$$

Prøv gjerne sjølv å komme fram til dette sjølve – i alle fall eit par av likningane.

Løys likningssystemet ved å sette opp ei matrise i MATLAB og så bruke `rref`-funksjonen på denne.

Utan å rekne på noko som helst: Kva ville svaret ha vore om *alle* omnane var skrudd av? (De kan godt sjekke at svaret de får i MATLAB stemmer med intuisjonen.)

No tenkjer vi oss at vi i staden for å finne ut kva temperaturane blir, vil bruke omnane til å bestemme kva temperaturfordeling vi skal ha i romma.

Om vi krever at vi skal ha T_1=18, T_2=T_3=17 og T_4=16, kva må effekten på omnane, altså P-ane, vere? (Oppgåva er ikkje så vanskeleg, altså.)

Til sist tenkjer vi oss at vi utetemperaturen varierar – slik det gjerne er i praksis.

Fila `TempFunk.m` er ei funksjonsfil som gir ute-temperaturen gjennom 36 timar. De finn ho på nettsida.

Bruk mellom annan denne til å plotte korleis vi må regulere dei fire omnane gjennom dette døgnet for å halde på den same temperaturfordelinga. Altså: Bestem dei fire effektane P_1, P_2, P_3 og P_4 som funksjonar av tid slik at T-ane blir verande uendra. Det kan vere greit å vite at funksjonsfila for ute-temperaturen som funksjon av tid kan ta vektorar som argument, ikkje berre enkelt-tal.

Oppgåve 5: Vi leikar med bilde

Vi kan sjå på eit digitalt fargebilde som tre matriser – ei for raudt, ei for grønt og ei for blått. Kva av desse matrisene svarar til eit rutenett med *m* rekker og *n* søyler – til saman *m* gonger *n* ruter – eller *pixlar*. Dette er oppløysinga på bildet. Fargen i kvar av desse rutene er gitt ved tre verdiar – ein tilsvarande kvar av dei tre grunnfargane – altså kor mykje raudt, kor mykje grønt og kor mykje blått fargen der skal bestå av.

Vel dykk eit eller anna blide de har liggande og last det inn i MATLAB (bildet treng ikkje vere i jpg-fomat):

```
V=imread('Bildenamn.jpg');
```

Ta ein kikk på det:

```
figure(1)

image(V)
```

Kva format har «matrisa», kor stor oppløysing har bildet?

Hint: `[a b c]=size(V);`

For kvar *m* og *n* verdi har vi altså tre fargeverdiar. Kvart av desse tala er eit heiltal mellom 0 og 255.

For å bli kjend med slike fargekodar, altså RGB-kodar: Finn dykk ein eller annan palett på nettet og undersøk følgande:
Korleis får vi svart og kvitt?
Kva farge er [200 250 150]?
Korleis ser «rein» raud, grøn og blå ut? ([255 0 0] etc.)
Korleis ser det ut når de tar med «full pott» av to grunnfargar, men ingenting av den tredje? ([255 255 0] etc.)
Kva kode kan de bruke for å få lyseblått? Enn rosa?

Når vi forstår RGB-kodar og korleis vi kan «herje» litt med matriser, har vi også fått på plass litt av grunnlaget for å manipulere bilde. Vel dykk ei rekke, vassrett eller loddrett, og plott raud, grøn og blå-verdiane i bildet langs denne rada:

```
S=1000;
figure(2)
plot(V(:,S,1),'r')
hold on
plot(V(:,S,2),'g')
plot(V(:,S,3),'b')
hold off
```

Her har vi plotta RGB-koden langs søyle nr. S i bildet, som her er tilordna variabelen V.

Når de går langs denne rada/søyla i bildet og samanliknar med bildet, kjenner de igjen korleis dei ulike fargane varierar i styrke? Prøv gjerne med eit par andre rekker/søyler også.

På på nettsida, under "bildefiler", finn de bildet *red-eye.jpg*. Blant skripta finn de også skriptet FilterRaudtAuga.m. Dette skriptet fjernar, om enn på ein lite sofistikert måte, det raude i pupillane. Køyr skriptet, les kjeldekoden og forsøk å forstå korleis vi fjernar det raude.

Etterpå: Finn dykk eit eller anna portrett kor de ønsker å endre auge-fargen. Ta gjerne det nemnde skriptet til hjelp.

Bildet Julestjerne.jpg viser ei julestjerne med nesten heilt svart bakgrunn. Prøv å gjere denne *heilt* svart. Prøv også lage ein heilt annan bakgrunnsfarge.

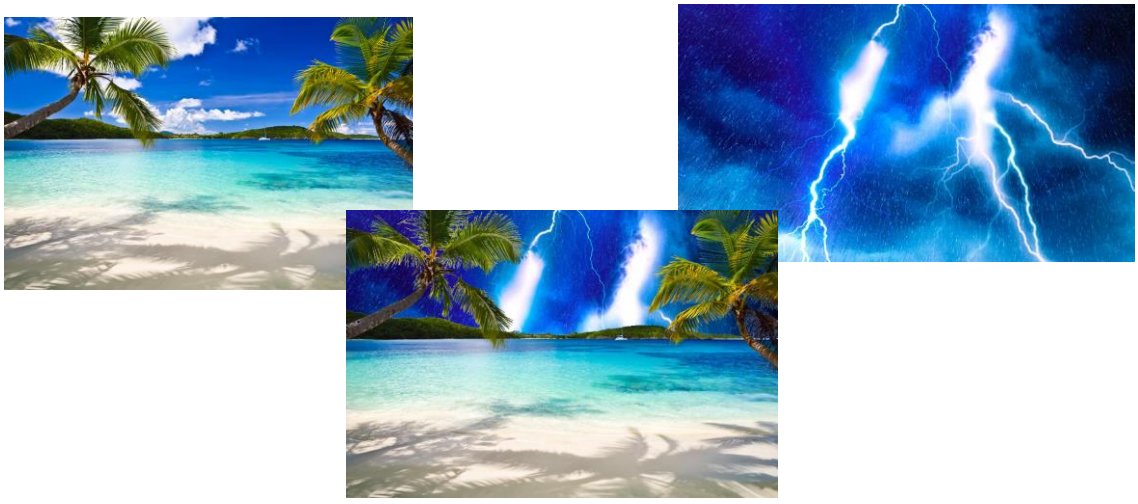
Innan marknadsføring er sjølvsagt manipulering av bilde ikkje heilt uvanleg. Her er nokre «kreative» eksempel frå eigedoms-bransjen:
https://www.vg.no/nyheter/innenriks/bolig/i-bergen-maa-man-bruke-paraply-selv-om-megler-legger-paa-blaa-himmel/a/10114063/

https://www.bt.no/bolig/Megler-far-kritikk-for-falske-flammer-104b.html

https://www.vg.no/nyheter/innenriks/boligmarkedet/fjernet-fjell-fra-boligannonse/a/23297872/

No skal de prøve sjølv!

Til hjelp skal de få mitt skarve forsøk på å gjere noko liknande – om enn med «motsett forteikn»:



Skriptet som gjer dette, heiter ManipulereBilde.m. De finn dette, saman med dei aktuelle bilda, på nettsida.

Forsøk å gjere noko liknande sjølve – altså å sette saman deler frå (minst) to ulike bilde til eitt – på ein meir eller mindre «kreativ» måte. Det nye bildet kan lagrast ved å bruke MATLAB-funksjonen `imwrite`; sjekk dokumentasjonen for å finne ut korleis de brukar den.

Oppgåve 6: Temperaturen i eitt rom

Vi skal tilbake til temperaturar og omnar – men no skal vi berre sjå på eitt rom med éin omn. Sånn sett er denne oppgåva enklare enn oppgåve 4. Men i oppgåve 4 såg vi berre på situasjonar der systemet var i jamvekt – der det gjekk like mykje varme inn i rommet som ut av det. Slik er det ikkje alltid.

Vi skal her sjå på ein vanleg modell kor ein tenkjer seg at farten som temperaturen endrar seg med, er proporsjonal med temperaturforskjellen ute. Om vi i tillegg har ein omn i rommet, kan denne kompensere for noko av varmetapet, eller den kan overkompensere slik at temperaturen går opp. Omnen er regulerbar med ein maksimal effekt på 600 Watt.

Modellen kan settast opp slik

$T'(t) = -k_1(T-T_{ute}) + k_2 P(t)$.

Her er $T'(t)$ (den deriverte) farten som temperaturen endrar seg med – målt i Celsiusgrader per time. Denne kan vere negativ eller positiv – avhengig av om temperaturen avtar eller aukar. For konstantane k_1 og k_2 set vi verdiane 0,3 og 0,008.

Jobben dykkar er å regulere effekten på omnen, $P(t)$, slik at vi får passe temperatur i rommet.

I funksjonfila OmnFunk.m har vi laga eit forslag til korleis omnen kan regulerast. Denne styringsmekanismen er slik at effekten blir sett til maksimal når temperaturen blir mindre enn 18 grader og omnen blir skrudd av igjen når temperaturen når 20 grader.

Med kjend tidsavhengig effekt, $P(t)$, og utetemperatur, $T_{ute}(t)$, og kjende konstantar k_1 og k_2, kan vi no bestemme kva temperaturutvikling vi vil få rommet – etter denne modellen. Dette er gjort i skriptet TemperaturUtvikling.m. Denne brukar funksjonsfilene SolveTempEq.m og TempFart.m, kor verdiane av k_1 og k_2 er lagde inn. Utetemperaturen som funksjon av tid, $Tute(t)$, er den same som i oppgåve 4 – altså TempFunk.m. Alle desse filene finn de på nettsida.

Skriptet TemperaturUtvikling.m plottar resultatet – altså temperaturutviklinga i rommet. Vi startar ved midnatt, t=0, det første døgnet og held på til klokka 12 døgnet etterpå, altså t=36. I tillegg plottar det ute-temperaturen og effekten omnen leverer som funksjonar av tid. SolveTempEq.m løyser sjølve likninga over, som er ei *differensiallikning*, og TempFart.m er ei funksjonsfil som gir høgresida i uttrykket over.

De kan godt ta ein kikk i desse filene. Men de kan også late dei vere «svarte boksar» (enn så lenge, i alle fall). Det er *omnen* vi skal regulere – altså: vi skal leike litt med OmnFunk.m. Og så skal vi justere litt på tala i skriptet TemperaturUtvikling.m.

Køyr dette skriptet og sjekk at plotta de får ut gir meining. Ser de korleis styringsmekanismen/ termostaten for omnen er implementert? Merk at det kan ta eit par sekund å køyre skriptet.

Starttemperaturen ved midnatt det første døgnet er sett til 19 grader. Dette er gjort i skriptet TemperaturUtvikling.m. Dersom vi koplar ut omnen, altså set $P=0$, kor lang tid går det før det er like kaldt inne som ute?

Dersom vi set starttemperaturen til å vere lik ute-temperaturen, korleis vil temperaturen i rommet følgje utetemperaturen? Vil dei vere like heile tida?

Tenk deg no at rommet ikkje skal brukast før klokka 12:00 neste dag – altså ved t=36. Det er viktig at temperaturen er 18 grader då. Om de ventar i det lengste med å skru på omnen, når må du seinast skru han på? Vi minner om at maksimal effekt er 600 W.

Kor mykje energi har de brukt i løpet av desse 36 timane? Tips: Forbruket er *integralet* av $P(t)$. Ein tilnærma verdi for dette kan reknast ut slik: trapz(TidVektor,EffektVektor). (Då får du svaret i watt-timar, Wh. Kanskje vil du heller ha det i *kilo*-watt-timar, kWh?)

Om vi går tilbake til den opphavlege styringsmekanismen, kor mykje energi har de då brukt i løpet av desse 36 timane?

Ut frå vår modell og vår omn, kva er den største temperaturforskjellen vi klarar å «halde i sjakk»?

Gå inn i OmnFunk.m og implementér andre måtar å regulere omnen på. Klarar de å få ned straumforbruket utan at det blir urimeleg kaldt i rommet? Hugs at omnen leverer *opptil* 600 W; den kan godt levere mindre også, og den treng ikkje nødvendigvis levere de same heile tida når den er på.