NTNU
Det skapende universitet

Stående-bølge-problemer i opptaksstudioer kan minskes vha GPU-akselerert simuleringsprogram

Teori, og grep i en implementasjon

Ola Brunborg Vikholt

Master i elektronikk
Oppgaven levert: Februar 2012
Hovedveileder: Ulf R Kristiansen, IET

Norges teknisk-naturvitenskapelige universitet

NTNU – Trondheim
Norwegian University of
Science and Technology

NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

INSTITUTT FOR ELEKTRONIKK OG TELEKOMMUNIKASJON
FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

Stående-bølge-problemer i opptaksstudioer kan minskes vha GPU-akselerert simuleringsprogram

Teori, og grep i en implementasjon

Ola Brunborg Vikholt

MASTEROPPGAVE I AKUSTIKK, 30 ECTS-POENG
STUDIEPROGRAM FOR ELEKTRONIKK

13. FEBRUAR, 2012
TRONDHEIM

VEILEDER, NTNU: ULF KRISTIANSEN
1. VEILEDER, MULTICONSULT: CLAS OLA HØSØIEN
2. VEILEDER, MULTICONSULT: JENS HOLGER RINDEL

Sammendrag

I små rom, kan fenomenet resonans være årsak til ugunstige akustiske forhold ved at visse frekvenser blir overdrevent kraftige mens andre toner blir knapt hørbare. Spesielt gjelder dette rom tiltenkt akustisk bruk, som opptaksstudioer og kontrollrom. Problemet kan unngås ved akustisk behandling i ettertid og/eller arkitektonisk planlegging, og begge deler kan dra nytte av datasimulering. Tilgjengelige simuleringsverktøy er konsentrert på mellom- og høyfrekvensområdet, og dekker ikke lavfrekvente bølgers diffraktive oppførsel. Akselerasjon av parallelle beregninger på GPU tillater derimot hurtig og presis simulering, med metodene FDTD og FD**F**D. En programvare beskrives og delvis utvikles i C#. Den drar nytte av GPU-en gjennom Cudafy via CUDA. Den vil forventes å kunne assistere ved plassering av høyttaler, lyttepunkt og bassabsorbenter, såvel som i geometrisk utforming av et rom.

I denne oppgaven betraktes resonansproblemet først fra et erfaringsperspektiv, liknende situasjoner og mangelen på reelle løsninger identifiseres, og dette danner bakgrunnen for arbeidet. Videre behandles fenomenet resonans teoretisk, i enkle termer og med flere eksempler. Dette etterfølges av en matematisk-teoretisk gjennomgang av simuleringsmetodene FDTD og FD**F**D. Til slutt beskrives implementasjonsdetaljer og utarbeidede løsninger, samt de uløste utfordringene – især FD**F**D-implementasjon – som gjenstår før programmet kan bli en realitet.

Problembeskrivelse

Oppgaveformulering

Oppgaveformulering:

> Lavfrekvent lyd i små rom
>
> Utvikle et analyse-/simuleringsverktøy som kan brukes i ikke-rektangulære rom, for eksempel lydstudioer og kontrollrom, i frekvensområdet 20 - 200 Hz. Tilnærming kan for eksempel være en 2.5D-modell, dvs. en kombinasjon av analytisk løsning av bølgeligningen i z-retning (gulv og tak er parallelle) og numerisk løsning i xy-planet (FEM-verktøy, alle vegger vinkelrett på gulv/tak).
>
> De viktigste beregningsresultatene vil være:
>
> - Rommets egenfrekvenser under en gitt grense, for eksempel under 200 Hz
> - Visualisering av de modeformer som tilsvarer egenfrekvensene
> - Beregnet overføringsfunksjon mellom en eller flere kildepunkter og et mottakerpunkt.
>
> Modellen bør kunne utvides med andre egenskaper for begrensningsflatene enn fullstendig refleksjon, for eksempel ved å angi flatenes impedans (frekvensavhengig).

Oppgaven ble gitt av akustikk-avdelingen ved konsulentselskapet Multiconsult. Oppgaven ble tatt ut mandag 5. september 2011. Opprinnelig leveringsfrist var 30. januar 2012. Utsatt leveringsfrist var 13. februar 2012.

En tolkning av oppgaveformuleringen ovenfor og og en konkretisering av målene i den, utgjør denne oppgaven erklærte mål. Dette kan leses i seksjon 1.4 på side 8.

Forord

Denne rapporten sammenfatter resultatet av, og beskriver arbeidet med en 30-poengs masteroppgave som avslutter et sivilingeniørstudium i elektronikk med spesialisering i signalbehandling og akustikk, ved Norges Teknisk-Naturvitenskapelige Universitet (NTNU).

Oppgaven ble foreslått av akustikkavdelingen ved Multiconsult, og ble formidlet av professor i akustikk ved NTNU Peter Svensson. Størstedelen av arbeidet ble utført ved Multiconsults akustikkavdeling i deres kontorer på Skøyen i Oslo.

Takk

Takk til min veileder Ulf Kristiansen ved NTNU for uvurderlig hjelp med forståelsen av de ulike numeriske simuleringsalgoritmer.

En spesiell takk til min faglige veileder ved Multiconsult, Jens-Holger Rindel, for hjelp med å forstå problemet i oppgaven og hva som var ønsket av målprodukt, for mange fruktbare diskusjoner om blant annet akustisk teori, for å ha satt meg i kontakt med et faglig nettverk, og for overlevering av erfaring som ellers bare kan opparbeides over lang tid. En stor takk til min kollega og medveileder Clas Ola Høsøien ved Multiconsult, som har vært både en faglig og sosial støttespiller.

Takk til illustratøren og 3D-programmereren Arjan Westerdiep som uten videre tok av sin tid for å tilpasse sin programvare til denne oppgavens formål. Takk til medstudent Tom Alexander Solgård for ulike tips underveis om masteroppgave, Latex og ulike simuleringsprogramvare.

Jeg vil også gjerne rette en stor takk til min far Gunnar Brunborg for nyttige innspill angående strukturering av både tekst og tid, og til min mor Merete Vikholt for å ha vært svært hjelpsom og forståelsesfull i spesielt travle tider, og til dem begge for å ha lagt til rette for et miljø og en livvsstil, både før min fødsel og etter, som har vært konteksten og rammeverket dette arbeidet ble gjort i, og uten hvilket denne masteroppgaven aldri hadde blitt til.

Lørdag 7. januar, 2012
Ola Brunborg Vikholt

Innhold

Forkortelser og begreper

Modellering innebærer å bygge en modell av virkeligheten (av et roms vegger etc.) som er tilstrekkelig nøyaktig, slik at den kan brukes til å *simulere* hva som ville skjedd i et slikt rom.

Fysisk/matematisk modellering er forskjellig fra modellering, og betyr forenkling av virkeligheten (lydbølger) ned til et nivå som tillater matematisk analyse.

Simulering er *etterlikning* på en datamaskin av den prosess eller prosedyre som ville utspilt seg i et konkret tilfelle i virkeligheten (for eksempel forplantning av lyd i et rom). Simulering krever at man har en adekvat modell av, i dette tilfellet, både rommet og av lydbølgers oppførsel.

Numerisk simulering Brukes oftest der hvor simuleringen benytter flyttall med endelig presisjon, altså ikke-eksakte tall. En *numerisk* simulering vil alltid produsere en *tilnærming*.

Bass vil i denne teksten bli brukt som et synonym på lave frekvenser ved vanlige forhold, typisk 0 – 200 Hz.

Kasse Et rettvinklet parallellepiped. Også kalt kuboide (eng. cuboid), eller skoeske.

Flate Ordene begrensningflate, vegg og flate brukes om hverandre som synonymer i denne teksten. Ordet grensebetingelse derimot, reserveres utelukkende for omtale av randbetingelser i differensiallikninger.

Kontrollrom er et rom tilstøtende et studio eller opptaksrom hvorfra opptaket kontrolleres og mikses. Et rom som brukes til miksing av musikk bør det være så akustisk nøytralt som mulig.

Modeform Den tidsuavhengige funksjonen for utsvingsmønsteret i en stående bølge i et rom.

Toolchain Fra engelsk for "verktøykjede". En samling ulike programvareverktøy som anvendes *i sekvens* for å løse et bestemt problem.

FDTD Finite-Difference Time-Domain

FDFD Finite-Difference Frequency-Domain

FEM Finite Element Method

BEM Boundary Element Method

CUDA NVIDIAs parallelle beregningsarkitektur, -platform og programmeringsmodell

GPGPU General-Purpose GPU

GPU Graphical Processing Unit

Kapittel 1

Introduksjon

Denne oppgaven undersøker behovet for en ny programvare for simulering av lav-frekvent lyd i små rom, og presenterer arbeidet som har blitt gjort på vei mot en slik programvare.[1] Programvaren vil bli kalt "Saka" gjennom rapporten.

Dette kapittelet begynner med å forklare motivasjonen for oppgaven gjennom praktiske betraktninger på utfordringene man møter ved design av opptaksstudioer. Deretter setter kapittelet oppgaven i en faglig kontekst ved å undersøke hvordan problemet har vært forsøkt løst i praksis tidligere, og ved å oppsummere relatert forskning.

Foruten disse praktiske betraktningene i innledningen vil oppgaven hovedsaklig fokusere på å forstå problemet teoretisk og løse problemet analytisk. Dette i kontrast til en musikkvitenskapelig tilnærming.

1.1 Bakgrunn

1.1.1 Et illustrerende eksempel

Motivasjonen for denne oppgaven forklares aller best med et eksempel.

[1] Akustiske utfordringer ved høyere frekvenser vil ikke bli omtalt i denne oppgaven, selv om de også er mange.

Kollega Svein Are Brekke ved Multiconsult prosjekterte et lydstudio på 20 kvadratmeter og tilstøtende kontrollrom på 7 kvadratmeter i Bransehuset på Ulven i Oslo.[2]

For å gjøre konstruksjone ganske lydtette så de ikke vil forstyrre hverandre, eller forstyrre eller forstyrres av omgivelsene, ble to godt isolerte rom-i-rom tegnet, med flytende gulv[3] på skinner og gummiknotter[4] og med mineralullsisolasjon. Geometrien ble gjort irregulær for å motvirke bl.a. flutterekko og stående bølger. Dette er vanlige grep ved studiodesign [1, kap. 5].

Etterklangstiden ble målt mot byggeslutt, og absorbenter og diffusorer ble så prosjektert for innspillingsrommet for å møte kravene til etterklangstid, og for kontrollrommet for å skape en nøytral lyttesituasjon. Etter installasjon av ulike absorbenter og diffusorer samt plassering av instrumenter og utstyr, hadde begge rommene god akustikk som tilfredsstilte krav, og prosjektet var dermed vellykket.

Imidlertid var det ett unntak. Det var et *resonansproblem* ved ca 100 Hz i kontrollrommet (se figur 1.1). Dette ble lagt merke til (hørt) i løpet av etterklangstidsmålingene, og problemet ble undersøkt videre ved å spille av rene sinustoner fra de installerte høyttalerene og lytte. (Installasjon av en type lite plasskrevende membranabsorbenter ble foreslått som løsning, men dette er ikke poenget.)

Problemet kunne vært oppdaget under prosjekteringsfasen, dersom lavfrekvente fenomener i rommene hadde blitt simulert. Imidlertid er ikke profesjonell programvare for akustisk simulering (Odeon o.l.) i stand til dette, da de begrenser seg til å betrakte lyd som stråler.[5] Dette er ikke en dekkende modell for lav-frekvent lyd i små rom; altså for hvordan lyd oppfører seg når lydbølgene er lange sett i forhold til rommets dimensjoner. (Vi kommer tilbake til dette under 2.1.)

Derfor er det behov for et program, som kan simulere lavfrekvente fenomer i typiske studiorom, slik at man allerede i prosjekteringsfasen kan identifisere slike resonansproblemer som det beskrevet ovenfor.

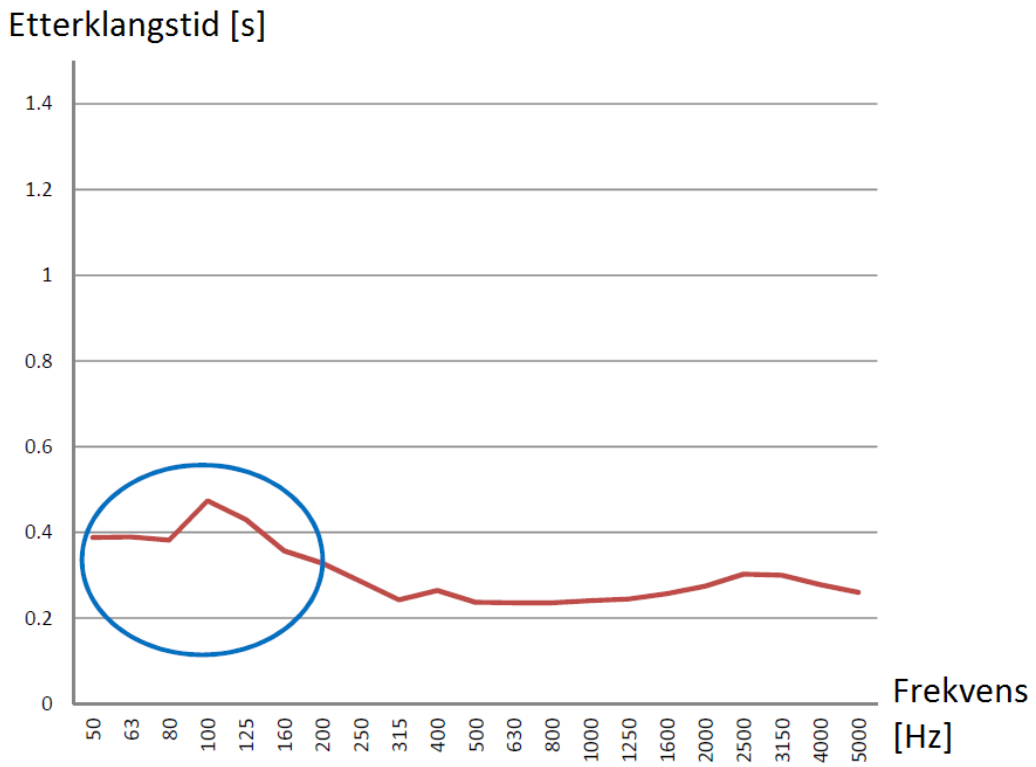
Et slikt program vil også kunne brukes til å simulere *eksisterende* rom, for å se hvilken effekt installasjon av ulike absorbenter kan ha på et roms akustikk, samt

[2] Se vedlegg A.5.

[3] Flytende gulv: se [1, avsnitt 3.2.2]

[4] Gummiknotter: se [1, figur 3.9]

[5] Dette utsagnet er en forenkling; se 1.1.6 for en mer presis gjennomgang.



Figur 1.1: Etterklangstid (0-1.4s) som funksjon av frekvens (50-5000Hz) i det omtalte kontrollrommet. Overflødig etterklang ved 100Hz er markert med blå sirkel.

hvilken effekt reposisjonering av høyttalere kan ha på overføringsfunksjonen til lyttepunktet, og dermed på det opplevde lydbildet.

1.1.2 I mer vid forstand: Dagens situasjon

Eksempelet beskrevet i forrige avsnitt, tilhører ikke de sjeldne tilfeller, men er derimot et utbredt problem. Et stort antall studioer har resonansproblemer.[6]

Mange studioer finnes hos studioutleiefirmaer, og ved skoler og kulturhus. Ofte blir det ikke investert i akustisk utbedring av disse studioene, og de blir i stedet stående med sine problemer. Multiconsult oppgir at de ikke får mange oppdrag av denne typen.

Også mange rom i private hjem blir nå gjort om til hjemme-studioer og hjemme-

[6] Kilde: muntlige kilder blant akustikere og musikkprodusenter.

kontrollrom, som et resultat av synkende kostnader på profesjonelt lydutstyr. Gunstig akustikk for avspillingsbruk og studiobruk er to svært forskjellige ting, så en alminnelig stue som begynner å brukes som studio eller kontrollrom vil, uten kostbar akustisk behandling, kunne låte dårlig og vil kunne føre til produksjon av svært ubalanserte mikser.

Både konsulenter og lekfolk vil kunne dra nytte av et dataprogram som kan identifisere slike problemer som de nevnt ovenfor, og hjelpe til på vei til en løsning av dem.

1.1.3 Moder og resonanser – hvorfor et problem?

Hva skjer når man mikser musikk i et rom med distinkte lavfrekvente resonanser?

Å mikse musikk innebærer ulik volumjustering av de ulike instrumentene og lydkildene som skal med i miksen. I tillegg benyttes tonekontroll (bass, diskant, frekvensfiltre) for å justere frekvensinnholdet i miksen for å få resultatet til å låte best mulig.

I et rom med distinkte lavfrekvente resonanser, vil man for en generell plassering av høyttalere og sitteplass/lyttepunkt, få en overføringsfunksjon som er svært ujevn i de lavere frekvensene. Dermed vil miksen som den oppfattes i lyttepunktet, være sterkt farget av rommets akustikk, og den som mikser vil, uten å legge merke til det, lage en miks som veier opp for denne fargingen. Når man så spiller av miksen i et annet rom, som ikke har den bestemte fargingen som studioet hadde, vil miksen ofte låte aldeles gal. Dette problemet er godt kjent for folk som har erfaring med å mikse musikk.

1.1.4 Andre rom og avlukker

Hva slags akustikk som er ønskelig beror på hva et rom skal brukes til, men lavfrekvent resonans vil nesten alltid være et onde som man ønsker å unngå [1, seksjon 4.1]. Slike resonanser er et problem ikke bare i studioer men også i endel andre typer rom:

Kino-saler har behov for presis lydgjengivelse for alle i salen, og siden bass ofte er en viktig del av lydbildet på kino, blir rom-moder relevant. Riktignok forsvinner utfordringene nedover mot ikke-hørbare frekvenser når kinosalen vokser, men det

er ikke *irrelevant* før vibrasjonene er så lavfrekvente at de ikke engang kan kjennes på kroppen, altså godt under 20 Hz.

Lastebil- og personbilkupeer er relativt små avlukker, og vil således bli plaget av modekollaps (sammenfall av moder) ved relativt høyere tonehøyder. For lastebiler kan motordur resonnere med moder i kuperommet og slik skape problemer. For personbiler er problemet mindre, men fortsatt aktuelt for bilstereo-entusiaster.

Større eller mindre industrihaller er ofte kasseformet og inneholder tunge maskiner som avhengig av plassering kan komme til å eksitere resonansmoder i hallen. Dette vil ikke bli hørt som lyd grunnet den lave frekvensen, men kan oppfattes som forstyrrende rumling/vibrasjoner, og kan være svært ubehagelig, eller skape svimmelhet og kvalme.

Vi ser altså at rom av ulike typer og størrelser er utsatt for de samme akustiske fenomenene, men i ulik grad og med ulikt resultat for hvordan vi oppfatter fenomenet. Mange av betraktningene og resultatene i denne oppgaven, gjelder dermed også i disse andre tilfellene.

Dataprogrammet som beskrives vil etter små eller ingen modifikasjoner kunne anvendes i disse andre tilfellene. Likevel vil denne oppgaven begrense seg til å utelukkende betrakte tilfellet med lydstudioer og tilhørende kontrollrom, og vi vil dermed få anledning til å arbeide mot en spesialtilpasset løsning. Denne løsningen kan danne grunnlag for utviding til eksplisitt støtte for andre typer rom i fremtiden.

Det er to grunner til at moder utgjør et spesielt viktig problem nettopp for studioer. Studioers små størrelser; og deres kraftige interne refleksjon. Dette vil bli forklart nærmere i Teori-kapittelet.

1.1.5 Hvorfor simulere?

Hvorfor ønsker vi å simulere og hva kan vi finne ut med simuleringer?

Rom-moder i kasseformede rom med perfekt reflekterende vegger er enkelt å uttrykke analytisk, som vi skal se i seksjon 2.1.2. Dette kan hjelpe oss til å forstå hva som gir opphav til resonanser i generelle rom. Men for å finne ut hvordan modeformene blir og ved hvilke frekvenser det oppstår resonans i rom som har komplisert geometri og virkelige materialer i veggflatene, trenger vi å løse versjoner av bølgelikningene numerisk, gjennom simulering.

1.1.6 Noen eksisterende simuleringsprogrammer

Odeon er et godt og mye brukt profesjonelt akustikk-simuleringsprogram. Odeon benytter i hovedsak versjoner av strålegangsmetoden og speilkildemetoden for å simulere diffuse lydfelts oppførsel, hvor lydbølger betraktes som stråler. Stråler beveger seg i rette linjer, men det er ingen dekkende modell for lavfrekvent lyd, i følge Kuttruff og Vorländer [2]. Lavfrekvent lyds bevegelse minner mer om skvulp i et badekar.

Selv med dagens teknologi for å simulere diffraksjon i strålegangsmetoden [3], kan ikke lavfrekvente lydbølgers fortplantning modelleres nøyaktig. (Med unntak av spredning fra rigide, konvekse objekter, hvor metoden gir korrekt svar ned til 0 Hz. [4]) Odeon er følgelig ikke anvendbart i vårt tilfelle. Det samme gjelder CATT-Acoustic [5], av samme grunn.

Finite-element-metoden (FEM) er en metode å formulere differensiallikninger på som tillater numerisk tilnærmet løsning gjennom approksimasjon. FEM er anvendelig på fysiske problemer i fluid-dynamikk, elektromagnetisme, statikk o.a. såvel som på akustikk, og er svært mye brukt. (Vi vil komme tilbake til metoden i 2.5.) ANSYS og COMSOL er to velutviklede programmer for generelle og allsidige FEM-simuleringer, og disse er ganske visst i stand til å simulere lav-frekvent lyd i små rom. Det er imidlertid to ulemper ved disse og liknende programmer: De er *for* kostbare[7]; og de er *for* generelle.[8]

Det man i stedet ønsker seg er et enkelt og spesialtilpasset program hvor det som i vårt tilfelle er vanlige operasjoner, er lette å utføre.

1.2 Liknende initiativer

Lachmanns LFtool[5] og Pafecs RoomCalc[6] er to programvarer for modeanalyse o.l. som er utviklet med samme motivasjon som den bak dette prosjektet.

I RoomCalc er det mulig å spesifisere en geometri i planet. Programmet bruker FEM-metoden og finner modene i rommet under en gitt grense, som så visualiseres. Videre kan én absorpsjonsfaktor felles for alle flatene spesifiseres, en

[7] COMSOL for akustikk kostet i 2009 ca. 240 000 kroner for en flytende lisens (se COMSOLs prisliste).

[8] i følge akustikere ved Multiconsult. COMSOLs akustikk-pakke har dog ikke blitt utførlig testet i dette prosjektet.

takhøyde spesifiseres, og kilde og mikrofon plasseres. Programmet vil da benytte 2.5D-metoden og presentere overføringsfunksjonen mellom punktene. (Se figurer 2.10 og 2.12 under avsnitt 2.1.4.)

LFtool er både mer og mindre avansert: Programmet simulerer også xy-snittflaten ved hjelp av FEM, gjennom et MATLAB-tillegg for differensialanalyse. Denne løsningen kombineres så med den analytiske løsningen av Helmholtz-likningen i z-retning. Lachmann motiveres av to andre programmer for modeanalyse, Cara og RPG, siden disse er begrenset til rektangulære rom. I LFtool kan man som i RoomCalc kun spesifisere en snittflategeometri, og det antas at gulv og himling er parallelle og at alle veggene står vinkelrett på gulvet. Inventar og møbler antas helt fraværende.

Både LFtool og RoomCalc er begrenset til rom uten endring i z-retning fordi de bruker 2.5D-metoden, LFtool støtter ikke absorpsjon i det hele tatt (harde vegger) og RoomCalc støtter bare reelle/absorberende flater. Alt i alt er det flere grunner til å arbeide mot noe mer avansert enn disse programmene.

Men, programmene er svært nyttige, og vi skal benytte oss av beregningsresultater fra dem senere i rapporten, for å illustrere og forstå nøkkelfenomener. Vi skal også hente ideer fra LFtool om utforming av brukergrensesnitt (4.2.1).

1.3 3D i stedet for 2.5D

Oppgaveformuleringen fra Multiconsult foreslår at det utvikles en *forenklet 2.5D-løsning*. 2.5D betyr at rommet "deles opp" i en todimensjonal flate som simuleres, og en z-akse som analyseres, hvorpå disse to løsningene kombineres. Dette for å spare regnekraft, etter inspirasjon fra forskningsprosjektet LFtool [5] (nevnt ovenfor). Dette minsker programmets tidsbruk: Et rom løses i løpet av noen få sekunder, i følge forfatteren M. Lachmann.

Datamaskiners regnekraft vokser fortsatt eksponensielt, og høyparallelliserende grafiske prosesseringsenheter (GPUer) som originalt ble utviklet for å effektivt kunne prosessere grafikk for dataspill, brukes nå også til generell dataprosessering og tunge databeregninger. Dette blir utnyttet i stor grad til å øke hastigheten på ulike akustiske simuleringer [7, 8, 9, 10, 11]. Vi skal vise at flytting av beregningene til GPUen øker hastigheten så mye at en *full 3D-simulering* kan gjøres innenfor tillatelig kort beregningstid, og utvikle en slik metode i stedet for 2.5D. Motivasjonene for en full 3D-simulering er forøvrig:

- En full 3D-simulering er konseptuelt enklere å analysere og utvikle.
- Muligheter for visualisering av vilkårlige snittflater er flere, og volumvisualisering er mulig.
- Vi kan simulere vilkårlige geometrier (i stedet for å være begrenset til å simulere slike hvor rette vegger strekker seg fra gulv til tak). Dette er viktig fordi godt prosjekterte studioer ofte har skråstilte vegger og/eller skrå tak. Dessuten vil det bli mulig å simulere rom med møbler og instrumenter installert.
- Forutsatt at beregningene går svært raskt, vil det bli mulig å se *kontinuerlig oppdaterte* beregningsresultater av de endringene man gjør i geometri o.a. "On-line" eksperimentering med slik *responsiv* programvare vil tillate en bruker å utvikle en god *intuisjon* for de simulerte fenomener.

1.4 Mål for oppgaven

Målet for denne oppgaven er en beskrivelse av en programvare for dataassistert basshøyttalerplassering og bassabsorbentinstallasjon i opptaksstudioer og kontrollrom, med de elementer dette måtte bygge på. I denne oppgaven har dette blitt tolket til å medføre de følgende konkrete delmålene for oppgaven:

- Gjøre en praktisk undersøkelse av motivasjonen bak oppgaveformuleringen.
- Forstå og beskrive nødvendig teori; derunder beskrive og vurdere numeriske modeller mhp. teoretisk fundament, beregningskompleksitet, beregningsnøyaktighet, konfigurasjonsfleksibilitet og implementasjon.
- Beskrive og, i den grad tiden tillater det, utvikle en prototype eller tidlig versjon av et program, med alt det omfatter av såvel beregningsalgoritme som grafisk utforming. Algoritmer for tunge beregninger skal implementeres i GPU-kode. Se forøvrig 4.2.1 for utarbeidede kravspesifikasjon og programdetaljer.
- Verifisere utviklet verktøy ved å sammenlikne resultat av måling av et eksisterende rom med simulering av en modell av det samme rommet.

Alle målene nevnt ovenfor er steg på vei til en *praktisk* og *helhetlig* løsning av problemet. De skal alle forsøkes løst i like stor grad, uten spesiell vekt på noen av punktene.

1.5 Rapportens struktur

Rapporten er organisert som følger. Kapittel 2 Teori inneholder en enkel introduksjon av modeteori, en teoretisk analyse av problemet for hånden; det utleder simuleringsmetodene FDTD og FD**F**D og sammenlikner dem, diskuterer romgeometri og implementasjon av impedans i begrensningsflater, og det diskuterer implementasjon av algoritmer for GPU. Kapittel 3 Metoder presenterer programvareverktøy, utviklingsverktøy, programvarebiblioteker og måleutstyr som ble brukt, og diskuterer bakgrunnen for valg av simuleringsmodell og programmeringsspråk og -metode. Kapittel 4 Resultater presenterer måleresultater, beregningsresultater fra ulike programvare, prosjektets utviklede programvare, og brukergrensesnitt-designet som foreslås på bakgrunn av brukerscenarioer. Kapittel 5 Diskusjon diskuterer resultatene fra kapittel 4, vurderer arbeidsmetodene brukt i prosjektet og konkluderer til slutt masterprosjektet og denne rapporten.

Kapittel 2

Teori

Dette kapittelet gjennomgår den teoretiske utledningen og matematiske analysen som er nødvendig for å forstå resten av rapporten. Det antas at leseren er kjent med grunnpillarene i akustikk og kalkulus, og at han har en interesse for bygningsakustikk samt for programvareutvikling.

Kapittelet går enkelte steder i noe mer detalj enn hva som kan virke nødvendig for en profesjonell akustiker. En slik leser anmodes derfor om å hoppe over de avsnitt som behandler kjent stoff.

For en alternativ praktisk betraktning av lav-frekvente utfordringer ved studiodesign, se [1, kap. 4]. Dette er en svært god ressurs for å få en intuitiv forståelse for feltet.

2.1 Lav-frekvent lyd. En introduksjon med praktiske betraktninger

I denne seksjonen vil vi undersøke hvordan lav-frekvent lyd oppfører seg i små rom. Men la oss aller først definere hva som menes med uttrykket *lav-frekvent*, som så langt har blitt brukt uten definisjon.

Lav-frekvent lyd er et relativt begrep og har forskjellige betydninger avhengig av kontekst. I sammenheng med miksing av musikk for eksempel, betyr lav-frekvent lyd det vi oppfatter som dyp lyd og bass, altså lyd fra omlag 300 Hz ned til 30 Hz eller helt ned til 20 Hz.

På den annen side, i denne teksten vil *lav-frekvent lyd* referere til et frekvensområde som vil være forskjellige avhengig av størrelsen til avlukket vi diskuterer i hvert enkelt tilfelle. Se figur 2.3. Siden et mindre rom vil "oppfatte" en bølgelengde som relativt lenger enn et stort rom vil, må bølgelengdene være kortere i dette mindre rommet for å oppføre seg slik de gjør i det store rommet. Eller som uttrykt i Bygningsakustikk av Vigran, s. 98: "I måleteknisk sammenheng, det være seg ved bestemmelse av lydisolasjon, av lydeffekt, av absorpsjon m.m. er [egenfrekvensenes Hz-verdi] i seg selv ikke så viktige. Den relative tettheten, hvor mange egenfrekvenser det er innenfor en gitt båndbredde, er imidlertid av avgjørende betydning for målenøyaktigheten." For et lite rom refererer derfor *lave frekvenser* til høyere frekvenser enn i tilfelle med et større rom. For et veldig stort rom vil nesten alle bølgelengder "se" korte ut, og her refererer begrepet lav-frekvent lyd til kun de aller laveste frekvensene.

Vi ønsker at begrepet lav-frekvent lyd skal omfatte de frekvensene hvor det er lav modetetthet i et rom, altså få distinkte modefrekvenser per oktav. Det er denne lave modetettheten som representerer det problemet som undersøkes i denne oppgaven. Ved høyere frekvenser ligger modene tett og utgjør ikke noe problem på den samme måten. La oss undersøke den siste påstanden nærmere i det følgende.

2.1.1 Impulsresponsens to deler

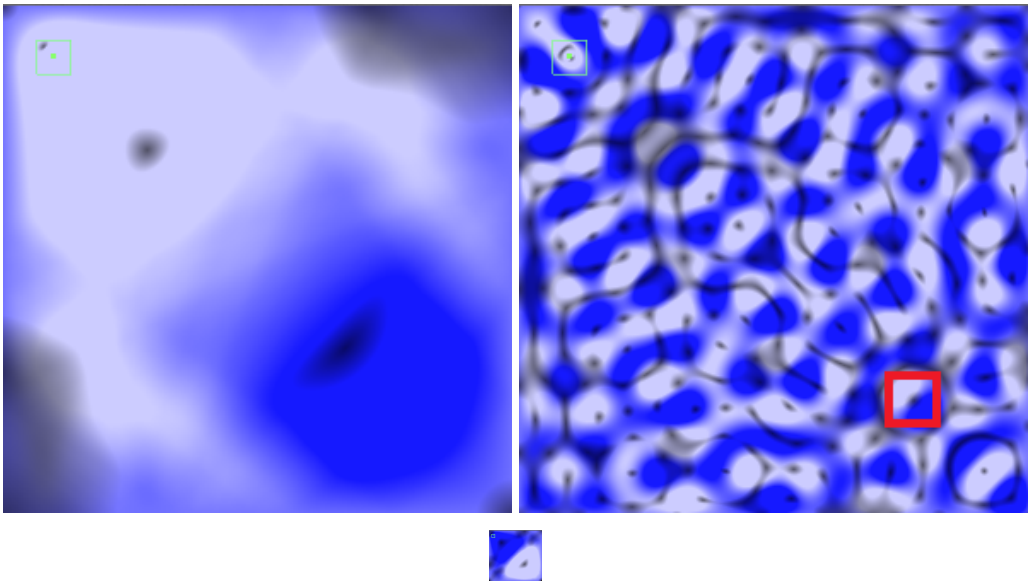
La oss se på en typisk impulsrespons[1] mellom to vilkårlige punkter i et praktisk rom. Fra det ene punktet sendes ut et (tilnærmet) flatspektret signal, og signalet mottatt i det andre punktet registreres og lagres.

Frekvensinnholdet i det mottatte signalet betegner overføringsfunksjonen mellom kilde og mottaker. Som eksempel er det i figur 2.2 (frekvens-magnitude-plot[2]) vist en måling fra en eksisterende stue. Overføringen er symmetrisk, slik at den vil være den samme om man bytter plass på høyttaleren og mikrofonen. Overføringsfunksjonen tegnes på en logaritmisk frekvens-akse hvor det er lik avstand mellom hver oktav/frekvensdobling, fordi det likner på hvordan vi oppfatter tonehøyde.

Vi ser at overføringsfunksjonen består av et område med lav modetetthet (få, spredte topper), og et område med høy modetetthet (tett samlede topper). Dette tilsvarer lavfrekvent- og høyfrekvent-delene av overføringsfunksjonen, og de

[1] Impulsrespons brukes her synonymt med både transferfunksjon og overføringsfunksjonen.

[2] Absoluttverdien eller magnituden til overføringsfunksjonen, som kan tegnes på en frekvens/styrke-graf, viser hvor mye av hver frekvens som når fram til mottakeren.

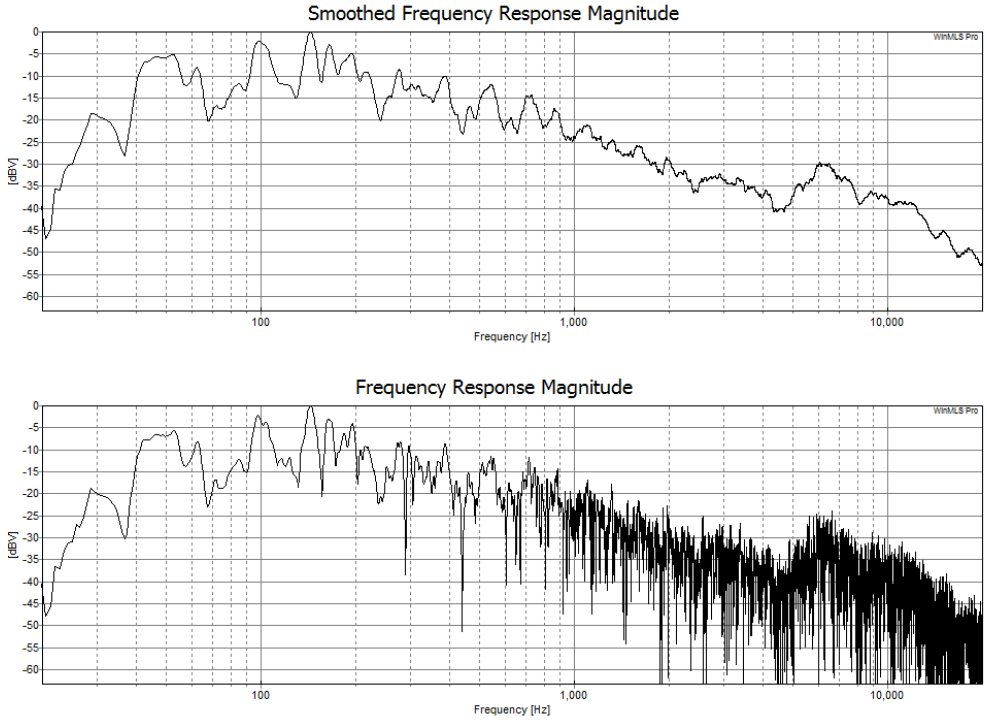


Figur 2.1: Øverst til venstre: Et $3x3m$ rom med en 110 Hz sinuskilde.
Øverst til høyre: Et $3x3m$ rom med en 1.1 kHz sinuskilde.
Nederst: Et $30cmx30cm$ rom, altså $\frac{1}{10}$**-dels sidelengder**, med en 1.1 kHz sinuskilde.
Legg merke til at 1 kHz-bølgens oppførsel i det *lille* rommet tilsvarer 110 Hz-bølgens oppførsel i det store rommet. Det lille rommet er eksitert med det samme signalet som i rommet til *høyre* – det kunne saktens se ut som om det var et utsnitt av det høyre rommet (rødt). Bølgenes oppførsel i det lille og det venstre rommet likner fordi, så mye mindre som det lille rommet er enn det venstre, så mye kortere er også bølgelengden til signalet som fyller det. *Illustrasjonene er laget med programmet "qtsaka", fra forfatterens masterprosjektoppgave [12].*

skilles av Schroeder-frekvensen (se seksjon 2.1.2) som er ca. 140 Hz for dette rommet. Toppene – kun synlige i det uprosesserte plottet (a) – vil i realiteten oppfattes som i det jevnede plottet (b).[3]

Disse få, spredte toppene betyr at vi har mye sterkere overføring ved akkurat toppens frekvens, enn i de omkringliggende frekvenser. Det opprinnelige signalet blir utsatt for sterk farging i lavfrekvens-spekteret når det registreres i lyttepunktet, ved enkelte frekvenser såpass mye at rommet "gjaller". Dette problemet kalles *bass booming* eller *booming effect* eller *boomyness* av studio-brukere, og er noe

[3]Disse toppene er egentlig spredt omlag like jevnt utover overalt, i lineær frekvens, mens per oktav – altså når vi betrakter frekvensaksen logaritmisk – ligger toppene fler og tettere sammen i høye frekvenser.



Figur 2.2: Over (a): Målt overføringsfunksjon mellom to punkter i et studio med kun porøse absorbenter (for høye frekvenser) installert. Under (b): Samme overføringsfunksjon, uten jevning (smoothing) av måledata. Impulsresponsen kan deles i to deler: en sterkt varierende del i de lavere frekvenser, som har sin årsak i moder, og til sammenlikning jevnt avtagende del i de høyere frekvenser. (Målt med WinMLS, eksitasjon var et 10 s sinussveip.)

Figur 2.3: Rom for målingene i figur 2.2.

man ønsker å unngå [2].

En slik overføringsfunksjonen vil være forskjellig i ett og samme rom, avhengig av hvilke to punkter man velger. Derimot er rommets *egenfrekvenser* noe som beskriver selve rommet i seg selv: problemene i overføringsfunksjon kan spores tilbake til problemer i rommets egenfrekvenser. Rommet egenfrekvenser er derfor det neste vi skal se på.

[13]

2.1.2 Et roms egenfrekvenser

Et kasseformet rom med harde, 100% reflekterende vegger støtter et bestemt sett av stående bølger (under en gitt frekvens). Frekvensene til disse bølgene er gitt av

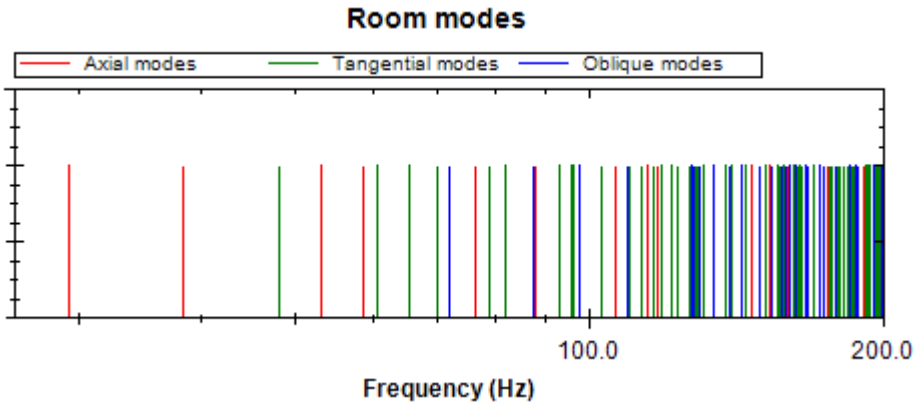
$$f_{x_n,y_n,z_n} = \frac{c}{2}\sqrt{\left(\frac{n_x}{L_x}\right)^2 + \left(\frac{n_y}{L_y}\right)^2 + \left(\frac{n_z}{L_z}\right)^2}$$

, hvor f_{n_x,n_y,n_z} er settet av modefrekvenser gitt av c som er er lydens hastighet ($340^{m}/_{s}$) og L_i som er rommets sidelengder, og hvor hver frekvens er identifisert av en *mode-tuppel* (n_x,n_y,n_z) hvor $n_i \in \mathbb{R}$ ($i \in \{x,y,z\}$).

Utledningen av uttrykket kan leses i introduksjonsbøker for akustikk, for eksempel [14] eller [15]. (For de som kjenner analysen av hvilke moder/svingeformer som støttes av en streng, er analysen av et hardt kasseformet rom simpelthen en utvidelse av streng-betraktningen til tre dimensjoner, og med en litt annen grensebetingelse.)

Her skal vi bare se på et par eksempler for å bli minnet på hvordan modeformene ser ut og på hvordan modefrekvensene sammenhenger med rommets dimensjoner.

Figur 2.4 viser resonansfrekvensene opp til 200 Hz i et kasseformet rom i typisk studio-størrelse med idealiserte harde vegger.[4] De innbyrdes forholdene mellom rommets sidelengder er relativt gode med tanke på å unngå sjenerende moder. Likevel er det få modefrekvenser under 100 Hz: 19 i antall. Under 100 Hz er det klart at moderesonans må undersøkes og eventuelt begrenses. Mellom 100 og 200 Hz er det uklart om vi vil snakke om moder eller diffuse lydfelt.

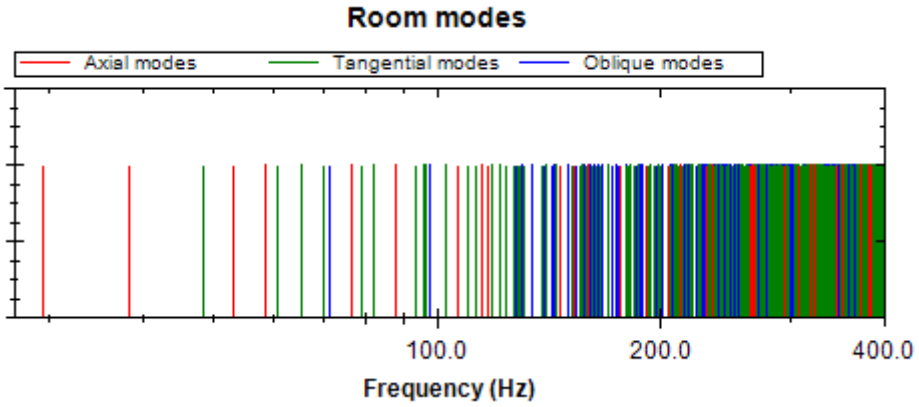


Figur 2.4: Modeplot for et hardt kasseformet rom med dimensjoner $4.44 x 5.8 x 3.2 m$. Rommet har 19 moder under 100 Hz og 108 moder under 200 Hz. Frekvensområde: 30 - 200 Hz.

Figur 2.5 viser modene i det samme rommet for opp til litt over 400 Hz. Over 200 Hz er modene så mange og ligger så tett innenfor oktaven, at her er det helt klart at vi ikke lenger kan snakke om et modeområde. Det er ca 600 moder totalt, inkludert sammenfallende, mellom 200 Hz og 400 Hz i dette rommet.

Når et roms sidelengder er heltallsmultipler av hverandre får vi *sammenfall av moder*. Dette kan sees i figur 2.6, som viser modefrekvensene for et *kubisk* og to nesten-kubiske rom, tegnet oppå hverandre, og med det tilhørende rommets dimensjoner notert ved siden av. Figuren viser at dersom en eller to av kubens

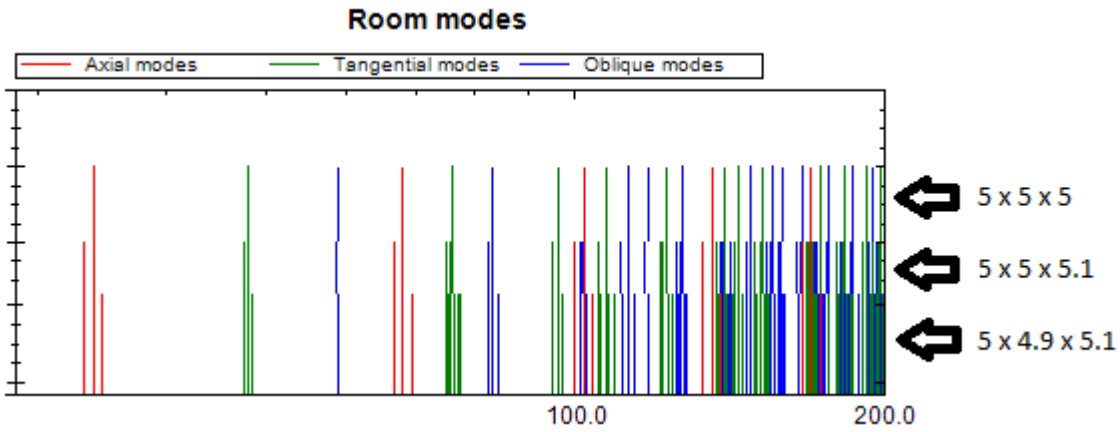
[4] Dimensjonene til rommet er hentet fra studioet beskrevet under introduksjonen.



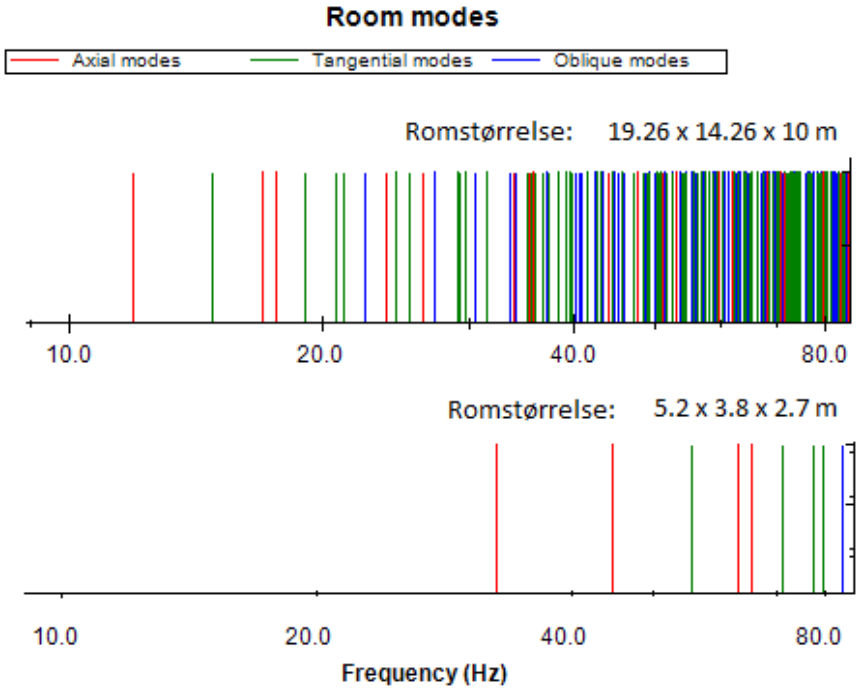
Figur 2.5: Modeplot for et hardt kasseformet rom med dimensjoner $4.44x5.8x3.2m$. Plottet viser at dette rommets lydfelt over 200 Hz bør betraktes som diffust. Frekvensområde 30 - 400 Hz.

dimensjoner endres litt, slik at det ikke lenger er helt kubisk, *separeres* modene som før overlappet hverandre i frekvens, og en jevnere graf kommer til syne. Den perfekte kuben har **like mange** moder som en uperfekt kube, men de overlapper altså hverandre – de har samme frekvens – og ser dermed ut til å være færre.

Sammenfall av moder er noe vi ønsker å unngå fordi det gir opphav til kraftig resonans eller "gjenklang" i et rom rundt de aktuelle frekvenser. Tilsvarende gjelder det at en jevnere mode-graf gir flere ulike resonansfrekvenser og med mindre kraftig resonans ved hver enkelt frekvens – noe som er å foretrekke i akustisk sammenheng.



Figur 2.6: Modeplot for et hardt kasseformet rom med dimensjoner 5 x 5 x 5 m. Med endringer i modeplottet tegnet inn for endringer i romstørrelsen.



Figur 2.7: Modetettheten er høyere i et stort rom (øverst) enn i et lite rom (nederst). Det vil si, modemønsteret er det samme i de to rommene, men er forsøvet mot lavere frekvenser for det store rommet. (Her er det store rommet omlag fire ganger større enn det lille i x-, y- og z-retning.) *Gjengitt med tillatelse fra Martin Lachmann.*

Det er ikke umiddelbart enkelt å fastslå hva som skiller mode-delen og diffus-delen av et roms frekvensrespons. Det er et overgangsbånd heller enn en diskret frekvens, men for å likevel sette en grensefrekvens for overgangen, utarbeidet Manfred Schroeder (1962) det teoretiske uttrykket $F_s = 2000 \cdot \sqrt{\frac{T_{60}}{V}}$ for overgangsfrekvensen F_s – gitt av T_{60} som er etterklangstiden i rommet, og rommets volum V. [16]

Schroeder-frekvensen søker egentlig bare å fastslå over hvilken frekvens lydfeltet i et rom kan betraktes som diffust (HF-regionen) og dermed beskrives statistisk, og sier ikke noe om mengden modeoverlapp i LF-regionen. Men la oss anta at potensielt problematisk modeoverlapp begrenser seg til LF-regionen, altså under Schroeder-frekvensen. Dette vil være tilstrekkelig og i tillegg noe på den sikre siden, i og med at modetettheten antakeligvis er høy nok også et stykke *under* Schroeder-frekvensen, for de fleste praktiske rom.

Slike mode-/resonansfrekvenser som presentert ovenfor finnes i harde rom av

enhver geometri, men modene blir straks vanskelige å regne ut når rommet blir litt mer komplisert. Da kan vi bruke numeriske approksimasjonsmetoder som FDTD eller FEM. Disse metodene vil bli presentert senere i dette kapittelet. Analysen ovenfor gjelder ikke dette dersom veggene i et rom absorberer lyd i større eller mindre grad, men modenes plassering i en hard-vegget idealisering av virkelig rom gir en god indikasjon på ved hvilke frekvenser det virkelige rommet kan by på akustiske utfordringer.

2.1.3 Illustrasjon av moder

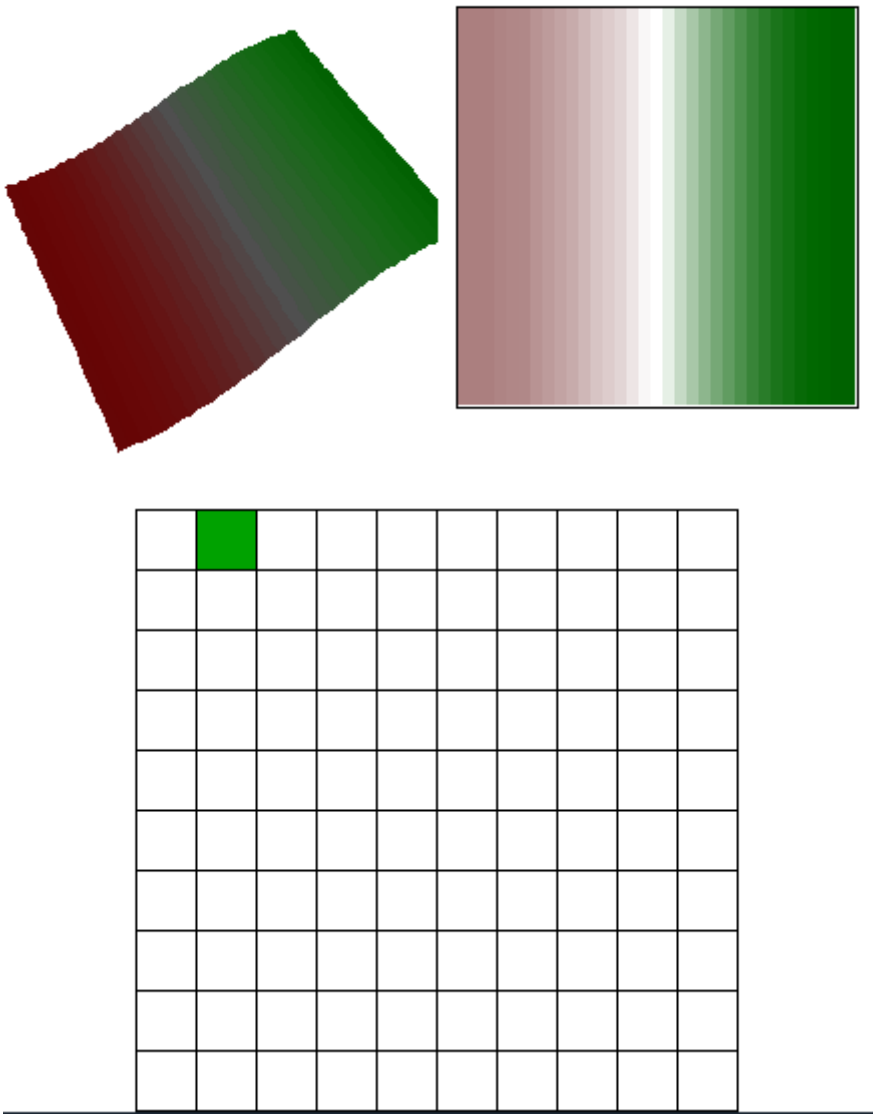
Vi kan nærme oss en intuitiv forståelse av hvordan moder oppfører seg i et rom ved å betrakte modeformer på en todimensjonal membran. Membranens grensebetingelse settes (kunstig) til fri/åpen, og dermed kan membranens svingeutslag betraktes som utslag i lydtrykket i et rom. I de to figurene 2.8 og 2.9 ser vi en visualisering av eksitasjon av to ulike moder i et kvadratisk rom. Til å produsere følgende visualisering har vi brukt java-applet-programmetmembrane skrevet av Paul Falstad.

2.1.4 Impedansens innvirkning på overføringsfunksjoner

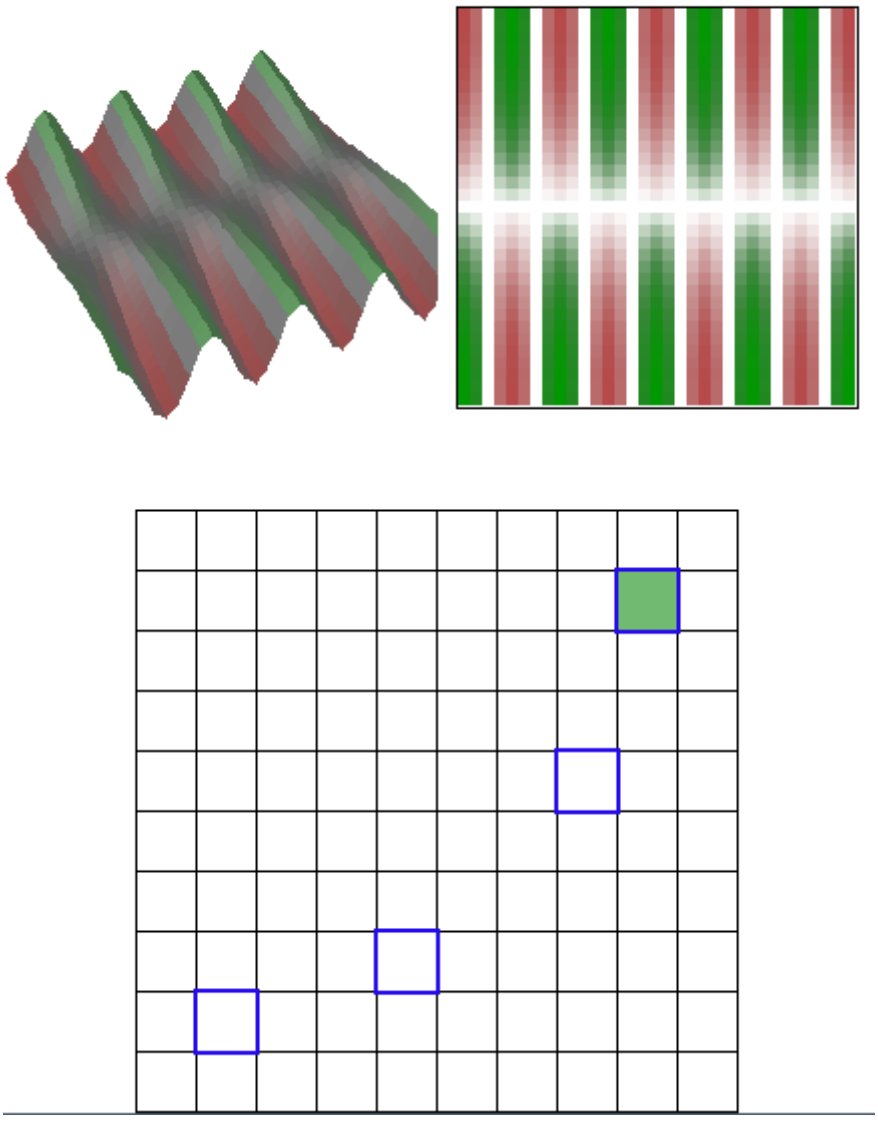
I denne underseksjonen skal vi undersøke sammenhengen mellom moder og impulsresponser, altså hvordan moder i et rom påvirker overføringsfunksjoner mellom par av punkter i rommet.

Overføringsfunksjoner i lave frekvenser avhenger av begrensningsflatenes impedans, både dersom den er reell, imaginær, eller kompleks. [14] For å forstå at dette er tilfellet er det tilstrekkelig å betrakte noen eksempler. *PAFEC RoomCalc* heter et program som beregner rom-moder i 2-dimensjonale rom samt overføringsfunksjoner mellom vilkårlige punkter. I dette programmet kan man spesifisere en enkelt reell (i motsetning til kompleks) absorpsjonsverdig som vil gjelde for alle veggene i rommet. Programmet tillater dog ikke spesifisering av ulik absorpsjon for forskjellige vegger. Det er ikke klart hvordan denne absorpsjonsverdien blir brukt, men det framgår klart at impulsresponsen mellom to punkter – selv om den fortsatt er påvirket av de tilstedeværende modene – blir svært mye jevnere når absorpsjonen øker litt.

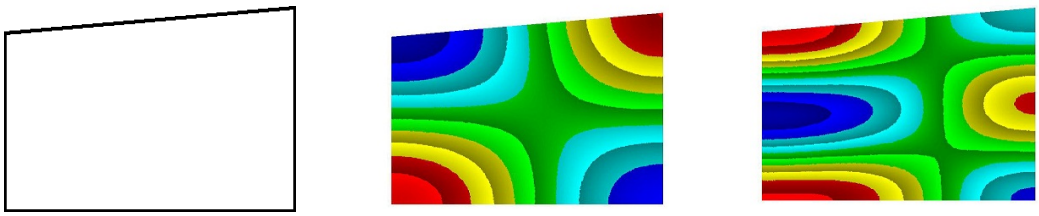
For å kunne forutsi og behandle lav-frekvente lydfelt i små rom trenger vi derfor å kjenne til eller finne impedansene til alle rommets flater.



Figur 2.8: *Mode 1,0 i et todimensjonalt kvadratisk ideelt rom. Grønt felt i koordinat 1,0 representerer at mode 1,0 er eksitert. Gjengitt med tillattelse fra Paul Falstad.*

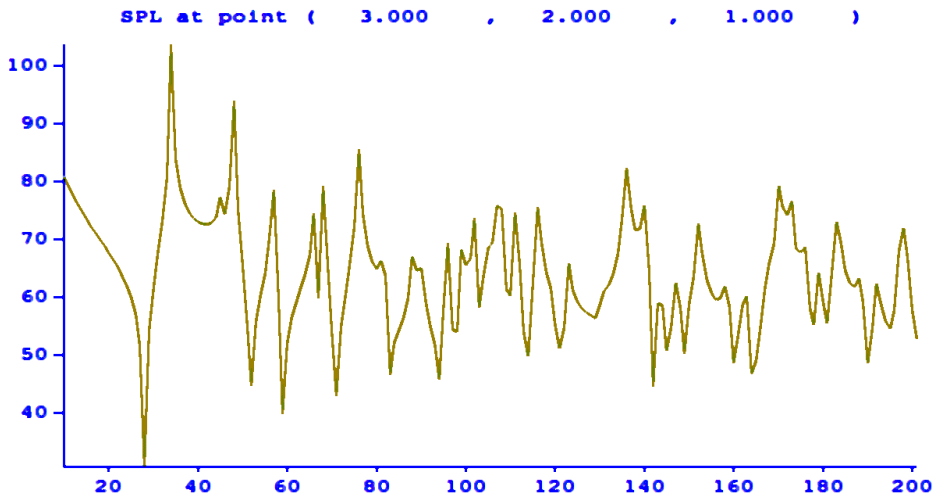


Figur 2.9: *Mode 8,1 i et todimensjonalt kvadratisk ideelt rom. Gule felter markerer moder med like frekvenser (sammenfallende moder) - for mode 8,1 er disse 7,4, 4,7 og 1,8. Gjengitt med tillattelse fra Paul Falstad.*

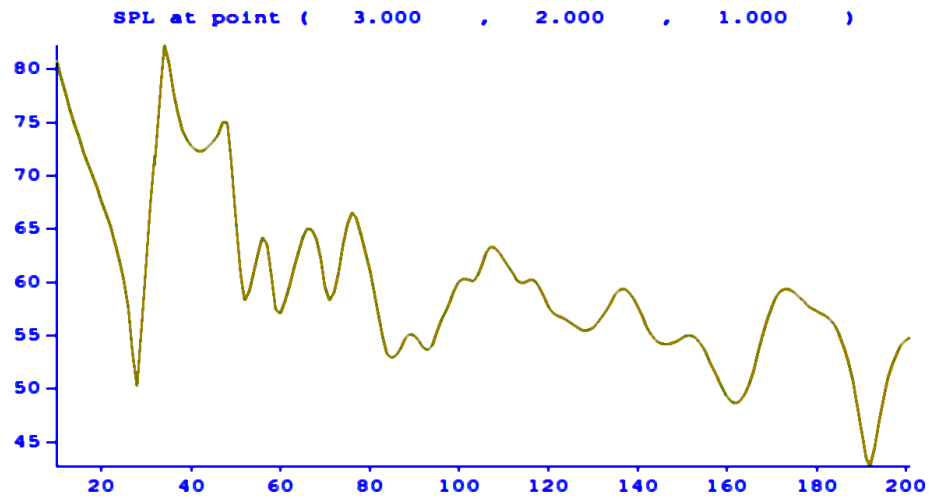


(a) Rommets form. (b) 57 Hz (c) 98 Hz

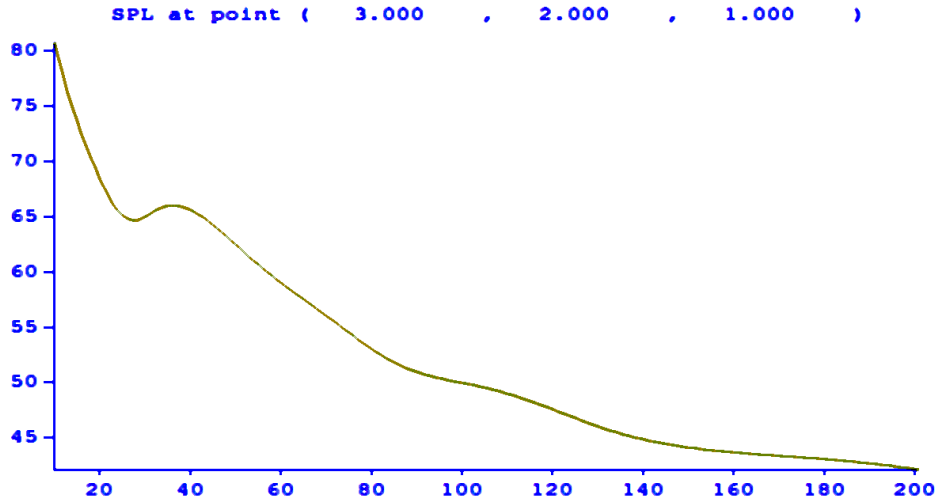
Figur 2.10: Viser rommets form (a) samt to av rommets mange moder (b; c). Dimensjoner: x-sidelengde 5 m; y-sidelengder 3.5 og 4 m



Figur 2.11: Overføringsfunksjonen mellom punktene (1,1,1) og (3,2,1) i rommet i figur 2.10, med $demping = 0.3$.



Figur 2.12: Overføringsfunksjonen mellom punktene (1,1,1) og (3,2,1) i rommet i figur 2.10, med $demping = 3$.



Figur 2.13: Overføringsfunksjonen mellom punktene (1,1,1) og (3,2,1) i rommet i figur 2.10, med $demping = 30$.

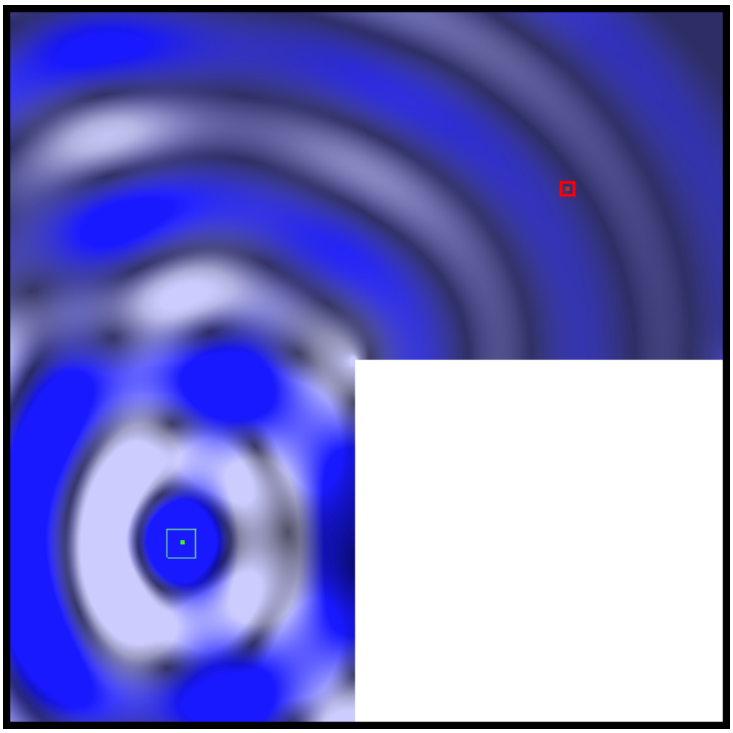
Vanligvis kreves en komplisert målemetode for å anslå et materiales impedans, for eksempel to-mikrofon-rørmetoden. Imidlertid er en rekke vanlige bygningsmaterialer og akustiske materialer blitt kartlagt, men da bare med deres *statistiske absorpsjonskoeffisient*, α_{stat}[5]. Dette enhetsløse forholdstallet måles for hvert hele frekvensbånd mellom 63 Hz og 80 kHz og noteres sammen med en utvetydig beskrivelse av materialet. Dette gjøres i hovedsak av produsenter av ulike akustiske materialer samt av forskninglaboratorier og bygningsakustikere for alminnelige byggematerialer.

2.2 Begrensninger ved geometriske metoder

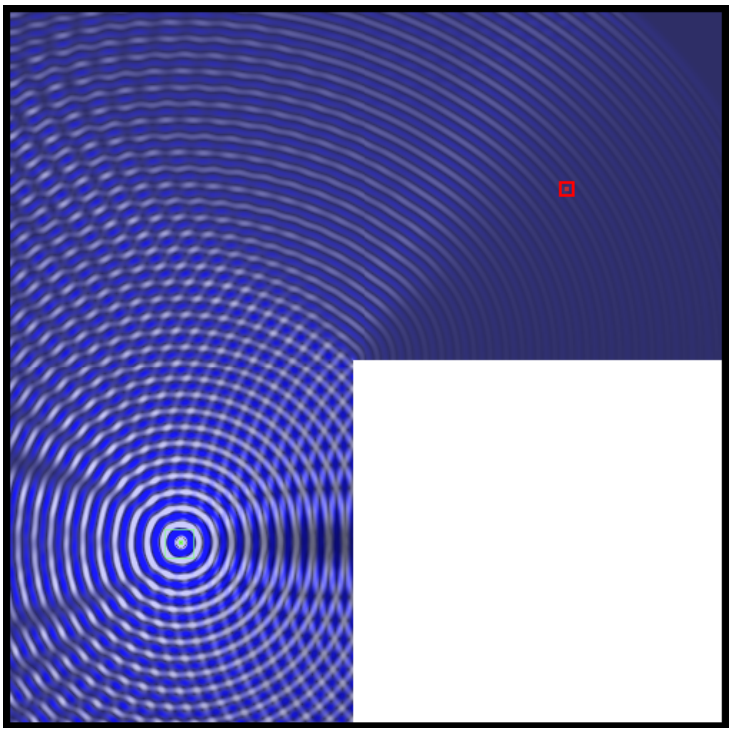
Som nevnt under introduksjonen dekkes ikke lav-frekvent lyd tilstrekkelig godt av programvare som antar en strålegangsforenkling av lydbølgene. Hvorfor dette er slik kan enkelt forstås gjennom å betrakte følgende eksempel: En 300Hz sinusbølge og en 3kHz sinusbølge sendes ut fra en rundtstrålende kilde som vist henholdsvis i figur 2.14 og 2.15, i fritt rom og med en 2.5 x 2.5 m rigid vegg som vist i hvitt. Vi sammenlikner utbredelsen av de to bølgene, spesielt i hvordan de påvirkes av den rigide veggen (i hvitt). Vi at den lavfrekvente bølgen påvirkes i mindre grad av veggen og bøyes rundt den, mens den høyfrekvente bølgen oppfører seg mer som en stråle, slik at det som ligger bak veggen sett fra kilden, kommer i skyggen av den lydmessig. Diffraksjonseffekten for den lavfrekvente bølgen er enda tydeligere for lavere frekvenser enn 300Hz. Mottakerpunktet markert i rødt i figurene, mottar et 11dB svakere lydtrykk fra den høyfrekvente kilden enn fra den lavfrekvente kilden.

Dersom den lavfrekvente lydens utbredelse modelleres med stråler, er det klart at vi vil få store feil. Vi presenterer derfor i det følgende tre numeriske modelleringsmetoder som kan produsere presise resultater for lavfrekvent lyd.

[5] α_{stat}, eller bare α, er et mål på hvor mye energi som gjennomsnittlig blir absorbert per areal av et flatt stykke materiale, ved tilfeldig lydinnfall. I følge standarden måles tallet for et $10-12\,m^2$ stort stykke material.



Figur 2.14: Sterk diffraksjon av 300Hz (rundtstrålende) sinusbølge rundt kant (i hvitt) i ellers fritt felt markert med svart kant.



Figur 2.15: Svak diffraksjon (stråleoppførsel) av 3kHz sinusbølge rundt kant (i hvitt) i ellers fritt feltmarkert med svart kant.

2.3 Simuleringsmodeller for lav-frekvent lyd

Det finnes tre typer[6] akustiske beregningsmetoder som gir presise resultater for lav-frekvent lyd: FDTD, FEM, og FD**F**D. De er alle numeriske beregningsmetoder for å finne tilnærmede løsninger på differensiallikninger. Anvendt på akustikk, produserer de numeriske approksimasjoner av derivater av bølgelikningen, gitt et konkret tilfelle av et rom hvor flatene har endelige, komplekse impedanser. FDTD løser bølgelikningen i tid, FD**F**D løser den i frekvens, og FEM løser den gjennom variasjonsanalyse. FD**F**D kan betraktes som en undertype av FEM. En utmerket utledning av bakgrunnen for numeriske simuleringsmetoder for akustikk finnes i [18, s. 29-30]. Utledningen er konsis, men svært presis og lett forståelig.

Anvendelser: FDTD har blitt akselerert med GPU og anvendt på akustikk utallige ganger før. Etter forfatterens kjennskap har FD**F**D blitt anvendt lite på akustikk, men i noe grad på elektromagnetisme [19, 20].

Metodene vil i de følgende seksjonene gjennomgås hver for seg.

2.4 FDTD

I denne seksjonen vil vi forklare Finite-Difference Time-Domain (FDTD)-metoden ved å utlede algoritmens "oppdateringslikninger", med utgangspunkt i bølgelikningen. Deretter vil vi vurdere fordeler og ulemper ved metoden, dens parallelliserbarhet, dens vanlige bruksformål og plass i litteraturen, og dens nytte i denne oppgaven basert på hva slags beregningsresultater den produserer.

2.4.1 Utledning av FDTD

Dette er bølgelikningen slik den ofte brukes i akustikk:

$$c^2\nabla^2 p(\bar{r},t) - \ddot{p}(\bar{r},t) = 0 \tag{2.1}$$

En forklaring av uttrykket følger:

[6] En fjerde metode, Boundary Element Method (BEM) vil ikke bli betraktet i denne oppgaven. Interesserte refereres til for eksempel [17, 13.3.1].

$p(\bar{r},t)$ er lydfeltet i rommet som betraktes (definert for $\bar{r} \in D$ i et lukket luftrom D, for tiden $t \in R_{\geq 0}$). p er lydfeltets relative trykk i punktet $\bar{r}$ i rommet[7,8]; $\bar{r}$ kan dekomponeres i kartesiske komponenter, $\bar{r} = \langle r_x, r_y, r_z \rangle$. For eksempel, hvis $p(\langle 1,2,\frac{1}{2} \rangle) = 20\text{mPa}$ betyr dette at det relative trykket i punktet lengde = 1m, bredde = 2m og høyde = $\frac{1}{2}$m er 20 mPa ved tidspunktet t. (20 mPa tilsvarer $60\,dB_{ref20\mu Pa}$[9].)

∇^2 er den romlige Laplace-operatoren.[10] Dens fysiske tolkning i akustikk er relatert til endring av ekspansjon/kontraksjon til luft-gassen i det aktuelle punktet. Siden vi uttrykker p i kartesiske koordinater kan vi simpelthen erstatte $\nabla^2 p(\bar{r},t)$ med summen av de dobbelderiverte av p i x-, y- og z-retning:

$$\frac{\delta^2 p(\bar{r},t)}{\delta x^2} + \frac{\delta^2 p(\bar{r},t)}{\delta y^2} + \frac{\delta^2 p(\bar{r},t)}{\delta z^2} \tag{2.2}$$

$\ddot{p}$ er en forkortelse[11] for $\frac{\delta^2 p(\bar{r},t)}{\delta t^2}$, som er den dobbelt tidsderiverte av p.

c er lydhastigheten. Vi har antatt at mediet er ikke-dispersivt, dvs at bølgeforplantningshastigheten $c(p)$, som i mer generelle tilfeller er en funksjon av p, er konstant og uavhengig av bølgens frekvens, altså simpelthen c.

Det bølgelikningen i (2.1) sier, er altså at for alle de infinitisemale små gassvolumene δV som D består av, er δVs akselerasjon i en retning lik δVs ekspansjon i den retningen ganget med en konstant c^2, hvor symbolet c er valgt fordi denne konstanten i likningen, bestemmer forplantningshastigheten, som er ca. $340^{m}/_{s}$ i romtemperert luft. "Bølgelikningen beskriver et lydtrykksfelt persis dersom $|\text{p}(\bar{r},\text{t})| \ll \rho_0 c^2$, hvor ρ_0 er tettheten til mediet ved likevektstrykk." – **(author?)** [18].

The wave equation accurately describes the pressure in the sound eld provided jp(r; t)j 0c2, where 0 is the density of the propagation medium at equilibrium.

I FDTD approksimerer man *differensial*likningen (2.1) med en likning mellom endelige *differanser*, som relaterer trykket i små, men endelig store volumelement og deres forandring etter endelig store tidssteg, altså ΔV i stedet for δV, og Δt i stedet for δt. Disse erstatningene krever følgende analyse: Differensialet $\frac{\delta f(a)}{\delta x}$ (for

[7] Se 1 for en forklaring på relativt ift. absolutt trykk.

[8] Med p skal vi alltid mene $p(\bar{r},t)$, og med $p(\bar{r})$ (ett argument) skal vi alltid mene $p(\bar{r},t)$.

[9] Se 2 og 3 for en forklaring på denne notasjonen.

[10] alternative notasjoner for $\nabla^2 f$ (for en generell funksjon f) er Δf og $\nabla \cdot \nabla f$ og $\text{div}(\text{grad}\, f)$.

[11] brukt spesielt mye av kyberteknikere

en kontinuerlig differensierbar funksjon f av x, i punktet $x = a$) er definert som (grenseuttrykket) $\lim_{h\to 0} \frac{f(a+h)-f(a)}{h}$. Dette kan approksimere med $\frac{f(a+\Delta h)-f(a)}{\Delta h}$, hvor Δh er et endelig stort tall. Man kan med Taylor-ekspansjon vise at feilen i approksimasjonen blir $O(\Delta h)$[12], og at denne kan minskes ved å minske Δh, eller ved å ta med flere punkter på f med i approksimasjonen.

Bølgelikningen blir da (etter innsetting av (2.2) for ∇^2, samt to ganger applikasjon av approksimasjonsmetoden ovenfor)

$$c^2\left(\frac{p_{\bar{x}} + p_{-\bar{x}} - 2\cdot p}{\Delta x^2} + \frac{p_{\bar{y}} + p_{-\bar{y}} - 2\cdot p}{\Delta y^2} + \frac{p_{\bar{z}} + p_{-\bar{z}} - 2\cdot p}{\Delta z^2}\right) - \frac{p(\bar{r}, t+\Delta t) + p(\bar{r}, t-\Delta t) - 2\cdot p}{\Delta t^2} = 0$$

hvor p_b er innført som forkortelse for $p(\bar{r}+b,t)$ for et symbol b. Videre er $\bar{x}$ etc. innført som vektorer med lengde Δx slik at $p_{\bar{x}}$= p($\langle r_x + \Delta x, r_y, r_z \rangle$) etc. Dersom uttrykket løses for $p(\bar{r}, t+\Delta t)$ har man en likning, en *oppdateringslikning* for *det nye trykket* p i *punktet* $\bar{r}$ på *tidspunkt* $t+\Delta t$ basert på tidligere trykk (ved tid t og $t-\Delta t$) i samme punkt og trykket i de umiddelbart omkringliggende punkter. I programkode kan det se slik ut (vi lar $\Delta x = \Delta y = \Delta z$ og kaller dem `dx`):

```
p[x,y,z] = c*c*dt*dt/dx/dx*(
                p[x+1,y,z]+p[x-1,y,z]+
                p[x,y+1,z]+p[x,y-1,z]+
                p[x,y,z+1]+p[x,y,z-1]
                -6*p[x,y,z])
        +2*p[x,y,z]-last p[x,y,z]
```

For ytterligere detaljer om representasjon av lydfelt og kilder, og algoritmeimplementasjonsdetaljer henvises leseren til forfatterens prosjektoppgave [12].

FDTD-implementasjonen i denne oppgaven benytter ikke oppdateringslikningen ovenfor, men i stedet en litt annen variant, hvor et partikkelhastighetfelt også lagres, og hvor partikkelhastighetfelt og trykkfelt oppdateres annen hver gang, basert på hverandre.

[12] O: *asymptotisk notasjon*.

2.4.2 Fordeler og ulemper ved FDTD

La oss kort gå igjennom de viktigste fordelene og ulempene ved metoden, både når det gjelder enkelhet i implementasjon, beregningshastighet og -letthet, type resultater den produserer og nyttigheten av disse i dette prosjektet.

FDTD-metoden er en enkel metode. Det vil si, utledningen ovenfor er forholdsvis enkel, og implementasjon av metoden er rett-fram. FDTD er i sin natur veldig parallelliserbar, derfor er det naturlig å ønske å implementere den for GPU. Men, tross tilsynelatende enkelhet er implementasjon av algoritmen *på GPU* likevel ikke rett fram, men i stedet meget utfordrende.

Diskretisering av luftrommet som skal simuleres er enkelt for FDTD-metoden; det samme gjelder diskretisering av rigide begrensningsflater. På den annen side er representasjon av *begrensningsflater med frekvensavhengig impedans* ikke uten videre enkelt å implementere, da FDTD-metoden opererer i tidsdomenet. Da må celler med "minne" benyttes – celler som *aktivt modellerer* impedansen, *i tid*.

Vanlig for metoden er en uniform diskretisering av rommet. Dermed er metoden ikke passende i tilfeller hvor man er interessert i ulik presisjon for utvalgte deler av lydfeltet.

FDTD-metoden er mye brukt, også på akustikk, men spesielt på problemer i elektromagnetisme for å simulere bølgeledere, antenner og liknende, og er dermed beskrevet godt i litteraturen. Men, litteraturen er altså dominert av applikasjon på problemer i *elektromagnetisme*, slik at mange resultater og metoder må tolkes og "oversettes" til et akustikk-perspektiv før de kan anvendes.

FDTD produserer en tidsutvikling av et lydfelt, og er således nyttig for direkte visualisering av bølgeforplantning over tid. Dette vil dog ikke direkte benyttes i denne oppgaven.

Lydfeltet i ethvert punkt i det diskretiserte luftrommet kan enkelt leses av ved hvert tids-steg, slik at det er trivielt å "plassere mikrofoner" og slik ta opp (mottatt) signal i et punkt eller et vilkårlig antall punkter i det simulerte rommet, for direkte analyse eller lagring og off-line analyse. I denne oppgaven vil nettopp denne metoden kunne brukes for å kalkulere impulsresponsen for et antall mottakerpunkter, for eksempel for en samling punkter innenfor et aktuelt lytteområde, som i sittehøyde i et studio.

FDTD kan ikke uten videre finne generelle akustiske egenskaper ved et rom, som hvilke resonansmoder det støtter. Til dette trenger vi FEM/FD**F**D. Dette vil bli diskutert nærmere i neste seksjon.

FDTD-metoden er til forskjell fra geometriske beregningsmetoder spesielt nøyaktig ved lave frekvenser, og siden den modellerer lydbølger likt slik de er i virkeligheten, uten betydelig forenkling, kan den nøyaktig modellere diffraksjon og egenskaper ved lydbølger som ikke dekkes av en stråletilnærming av lydbølgene.

FDTD-metodens krav til minne og beregningskravt avhenger av ønsket beregningspresisjon, øvre beregningsfrekvens og størrelsen på rommet som skal simuleres. Dessverre vokser problemstørrelsen (både minne og beregningsmengde) kubisk med øvre frekvens.

I FDTD kan høyttaler- og mikrofondirektivitet implementeres, og metoden lager bedre auraliseringer, i følge [18] [21]: "The FDTD method produces impulse responses that are better suited for auralization than FEM and BEM." Dette får ikke anvendelse i denne oppgaven fordi direktivitet er av liten betydning for lave frekvenser [22], men poenget kan være av betydning for en eventuell forlengelse av dette prosjektet i fremtiden.

Representasjon av akseptable kilder i FDTD-metoden er uten videre enkelt, men disse såkalte “harde” kildene vil ha en overvekt av lave frekvenser, men som kan veies opp for i etterbehandling. En ny metode for bedre kildeimplementasjon i FDTD-metoden ble nylig publisert: [23].

2.5 FEM

For å forstå Finite Element Method (FEM)-metoden trengs det at man er kjent med funksjonell analyse, eller variasjonsanalyse. Dette ble ansett som utenfor rekkevidde i denne oppgaven, og i stedet skal vi ta for oss FD**F**D-metoden, og utlede og benytte denne. Videre er det vanskelig å finne litteratur som beskriver implementasjon av FEM-algoritmen, i det hele tatt med spesielt for akustikk, og en implementasjon måtte derfor ha blitt bygget opp fra grunnen gjennom analyse. I denne metoden deles simuleringsrommet opp i polygoner som Delaunay-trianguleres, som igjen er en komplisert prosess. Før vi går videre til FDTD-metoden, la oss liste opp de viktigste punktene ved FEM-metoden:

2.5.1 Fordeler og ulemper ved FEM

Fordeler:

- Bruker Delaunay-triangulering:
  - Ingen vokseliseringskvantiseringsfeil
  - Mulighet for økt gridoppløsning i interessante områder
- Finner resonansmoder

Ulemper:

- Sparse Matrix må bygges og løses
- Polygon må Delaunay-trianguleres
- Vanskelig å forstå

Andre bruksområder:

- Elektromagnetisme

2.6 FDFD

Finite-Difference *Frequency*-Domain (FD**F**D)-metoden kan betraktes som en undertype av FEM-metoden (om hyperbolske interpolasjonsfunksjoner benyttes), men kan betydelig enklere utledes og forstås uavhengig av FEM. FD**F**D-metoden kan benytte samme, enkle rom-diskretisering som FDTD-metoden, altså vokselisering i stedet for polygontriangulering (se (2.7.2)). Disse to faktorene gjør metoden attraktiv for dette prosjektet.

Metoden går ut på å løse bølgelikningen for det aktuelle domenet, én harmonisk frekvens av gangen. Vi kan finne denne frekvensdomeneversjonen av bølgelikningen, eller Helmholtz-likningen som den kalles, på to måter: Enten ved å Fourier-transformere bølgelikningen [18], eller ved å bruke *separasjon av variable* (se sep). Vi skal i det følgende utlede Helmholtz-likningen fra bølgelikningen ved å bruke separasjon av variable, før vi forklarer hvordan likningen kan diskretiseres for å komme fram til likningssettet som utgjør kjernen i FD**F**D-metoden.

På samme måte som i utledningen av FDTD-metoden har vi bølgelikningen:

$$c^2\nabla^2 p(\bar{r},t) - \ddot{p}(\bar{r},t) = 0 \tag{2.3}$$

Vi antar at p kan separeres, i et produkt av en posisjons-avhengig del $A(\bar{r})$ og en tids-avhengig del $T(t)$, slik:

$$p = A(\bar{r})T(t).$$

Applikasjon av Laplace-operatoren vil da gi:

$$\nabla^2 p(\bar{r},t) \longrightarrow \nabla^2 A(\bar{r})T(t) \longrightarrow T(t)\cdot\nabla^2 A(\bar{r}).$$

siden $T(t)$ er uavhengig av $\bar{r}$ og dermed ikke påvirkes av Laplace-operatoren. Det samme gjelder tilsvarende omvent for $A(\bar{r})$ slik at:

$$\ddot{p}(\bar{r},t) \longrightarrow \frac{\partial^2}{\partial t^2}[A(\bar{r})T(t)] \longrightarrow A(\bar{r})\ddot{T}(t).$$

Dermed blir bølgelikningen

$$c^2 T(t)\nabla^2 A(\bar{r}) - A(\bar{r})\ddot{T}(t) = 0$$

som kan omskrives til

$$\frac{\nabla^2 A(\bar{r})}{A(\bar{r})} - \frac{\ddot{T}(t)}{c^2 T(t)} = 0.$$

Da de to leddene er avhengig av to ulike variable men likevel alltid skal være like, forstår vi at de begge må være *konstante*, og vi får følgende likninger:

$$\frac{\ddot{T}(t)}{c^2 T(t)} = -k^2$$

og

$$\frac{\nabla^2 A(\bar{r})}{A(\bar{r})} = -k^2,$$

hvor $-k^2$ er valgt som konstant ($k \in C$), uten at vi har mistet noe generalitet. Et hvilket som helst symbol kan velges, men dersom $-k^2$ velges vil som vi skal se simpelthen k dukke opp senere i våre kalkulasjoner.

Vi skriver om likningen for A for å komme fram til Helmholtz' likning, som er:

$$\nabla^2 A(\bar{r}) + k^2 A(\bar{r}) = 0.$$

(Dette er forøvrig et egenverdiproblem hvor $A(\bar{r})$ som løser likningen er egenfunksjoner, og hvor k^2 da er egenfunksjonens tilhørende egenverdi.)

Dersom vi skriver ut Laplace-operatoren på samme måte som i utledningen av FDTD, og så diskretiserer differensialuttrykkene til differanseapproksimasjoner, får vi i eksempelpunktet $0,0,0$ en likning som relaterer punktet til nabopunktene og k, slik:

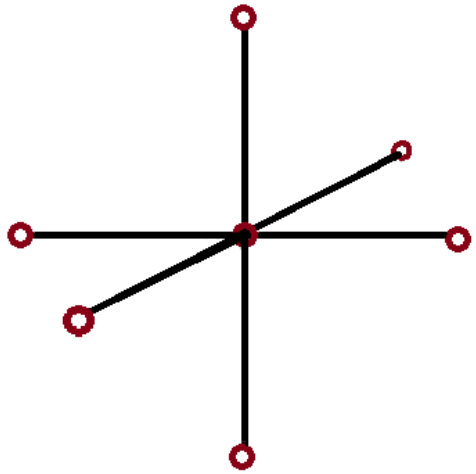
$$\frac{A_{1,0,0} + A_{-1,0,0} - 2 \cdot A_{0,0,0} + A_{0,1,0} + A_{0,-1,0} - 2 \cdot A_{0,0,0} + A_{0,0,1} + A_{0,0,-1} - 2 \cdot A_{0,0,0}}{\Delta x^2} + k^2 A_{0,0,0} = 0,$$

eventuelt slik:

$$A_{1,0,0} + A_{-1,0,0} - 2 \cdot A_{0,0,0} + A_{0,1,0} + A_{0,-1,0} - 2 \cdot A_{0,0,0} + A_{0,0,1} + A_{0,0,-1} - 2 \cdot A_{0,0,0} + \Delta x^2 k^2 A_{0,0,0} = 0.$$

Her er punktet $0,0,0$ tatt som eksempel ($A_{0,0,0}$ er en forkortelse for $A(\langle 0,0,0 \rangle)$). Likningen må settes opp og løses for hver eneste punkt x,y,z i luftrommet. I tillegg, dersom grenseflatene har endelige impedanser får vi en rekke ekstra, spesielle likninger for de punktene som ligger nær grenseflatene. I denne utledningen har vi benyttet den enkleste vanlige diskrete approksimasjonen av Laplace-operatoren i tre dimensjoner. Nabopunktene vi tar med i beregningen av trykket i et punkt i neste iterasjon, er illustrert i figur 2.16. Mange andre og mer kompliserte såkalte *stensiler* med tilhørende interpolasjonslikninger er mulige – se for eksempel [24] for detaljert gjennomgang og analyse av feilrate ved ulike interpolasjonsmetoder.

Dette likningssettet kan med fordel skrives opp i matriseform $B \times \underline{\mathrm{A}} + k^2 \underline{\mathrm{A}} = 0$ hvor B er koeffisientmatrisen (som for likningen ovenfor inneholder (1, 1, 1, 1, 1, 1 og 6 samt en rekke 0-er) og A er en vektor som enumererer alle punktene i luftrommet. Matrisens størrelse blir $(x \cdot y \cdot z)^2$, dvs at for en rom med sidelengder 5 meter og 0.2 m store celler, får vi en matrise med 244 millioner elementer. Hver



Figur 2.16: "Stensil" som illustrerer punktene som relateres i 7-punktsversjoner av både FDTD og FD**F**D.

likning relaterer bare et fåtall punkter (7; se figur (2.16)), så matrisen blir glissen (eng: *sparse*), altså det blir bare et fåtall oppføringer i matrisen som ikke er 0.

k-verdiene som løser likningssettet svarer til modefrekvensene som rommet støtter.

Løsningens $\underline{A}$ gir modeformen, altså fordelingen av trykkutslag/defleksjon i rommet.

2.7 Romgeometri og veggmaterialer. Import og behandling

Når vi gjennom simulering skal regne ut et bestemt roms ulike akustiske egenskaper, må først rommets konfigurasjon noteres i datamaskinen. Med rommets konfigurasjon menes rommets geometri, altså alle veggenes plassering og størrelse; og hvilke materialer de ulike veggflatene består av.

2.7.1 Geometri

Geometrien kan tegnes direkte inn i det aktuelle simuleringsprogrammet, eller det kan importeres fra fil.

Direkte inntegning kan være en fordel i anvendelser hvor hyppig endring av geometrien er ønskelig. I tilfellet med denne oppgaven derimot, vil situasjon ofte være den, at brukeren (dvs. akustikk-konsulenten) allerede har en 3D-modell av rommet som skal simuleres, eller at en slik modell kan hentes fra en arkitekt. Dette gjelder både enten rommet eksisterer eller det er på planleggingsstadiet. (Man kan naturligvis få undersøkt akustikken for rom som ennå ikke er bygget!)

I de tilfellene hvor det ikke finnes noen modell av rommet, må det legges til rette for at man rimelig enkelt kan lage en. Da det allerede finnes flere programvare tilgjengelig som er spesialisert for dette formålet, og siden notasjon av tredimensjonale geometrier er komplisert å få til på en datamaskin, tenker man seg denne oppgaven løst ved at brukeren tegner modellen inn i en dertil egnet eksisterende programvare. Deretter må modellen kunnes hentes ut av tegneprogrammet og introduseres i simuleringsprogrammet.

Se avsnitt 4.3.1 for en mulig – og praktisk implementert – løsning på utfordringen beskrevet i dette avsnittet.

2.7.2 Vokselisering

Modellen må deretter tilpasses for inkludering i den aktuelle simuleringsmodellen. Modellen foreligger på dette stadiet på polygon-form[13], med en materialtype tilordnet hvert polygon. En representasjon må produseres som passer med hvordan simuleringsrommet er diskretisert.

Ved FDTD-metoden deles rommet opp i et kartesisk gitter, og hvert diskretiserte romelement blir da en kube som til sammen fyller rommet som en kubisk honningkube. Den samme diskretiseringen brukes også gjerne for FD**F**D-metoden.

Vokselisering[14] er navnet på en prosess som produserer en ekvivalent representasjon som kan brukes i denne typen gitter: Alle kubene som skjæres av et polygon, tilordnes materialet eller veggtypen til polygonet den skjæres av, mens de resterende kubene ikke får tilordnet noen veggtype, og dermed betraktes som luft eller åpent rom.

Denne prosessen er ikke eksakt men en tilnærming. Det vil i denne oppgaven ikke bli undersøkt kvantitativt hvor stor beregningsfeil denne unøyaktigheten

[13] Polygon-form vil si at modellen er beskrevet som en samling flater, hvor hver flate er definert som og spenner ut mellom en samling punkter.

[14] Voksel, eller voxel, betyr volumelement.

medfører, men en kort kvalitativ betraktning følger: Kvantiseringen av veggenes koordinater vil medføre forskyvning av rette vegger samt ujevthet i skrå vegger, i vokselrepresentasjonen i forhold til den svært presise polygon-modellen. Her vil lik forskyvning av parallelle vegger ikke medføre noe problem, mens ulik forskyvning vil medføre forstørring eller forminsking av rommet, noe som vil utgjøre en viss forskjell i de beregnede rommoder og impulsresponser. Feilen i veggposisjon vil avhenge av den enkelte algoritme, men vil være i størrelsesorden kubens halve størrelse.

Se avsnitt 4.3.2 for en mulig – og implementert – løsning på vokseliseringsoppgaven beskrevet i dette avsnittet.

Kapittel 3

Materialer og metoder

3.1 Måleutstyr

Under måling av Fagerengs rom (se 4.4) ble følgende utstyr benyttet:

- To stykk Genelec 8040A høyttalere (se figur 3.1) i sin opprinnelige konfigurasjon i studioet. Innbyrdes avstand var 945 mm mellom akustiske sentra. Lavere 3dB cut-off-frekvens for høyttaleren er 45 Hz.
- Kondensatormikrofon, mikrofonforsterker og lydkort fra Multiconsults akustikkavdelings måleutstyrssett.

Programvare

- Under manuell lyttesesjon ble SigGen (for Mac) av Joe Cavers brukt til å sveipe gjennom sinustoner.
- WinMLS 2004 lisensiert av Multiconsult ble brukt til å eksitere og koordinere impulsresponsmålinger, og til å signalprosessere målinger for å produsere og plotte resultater.

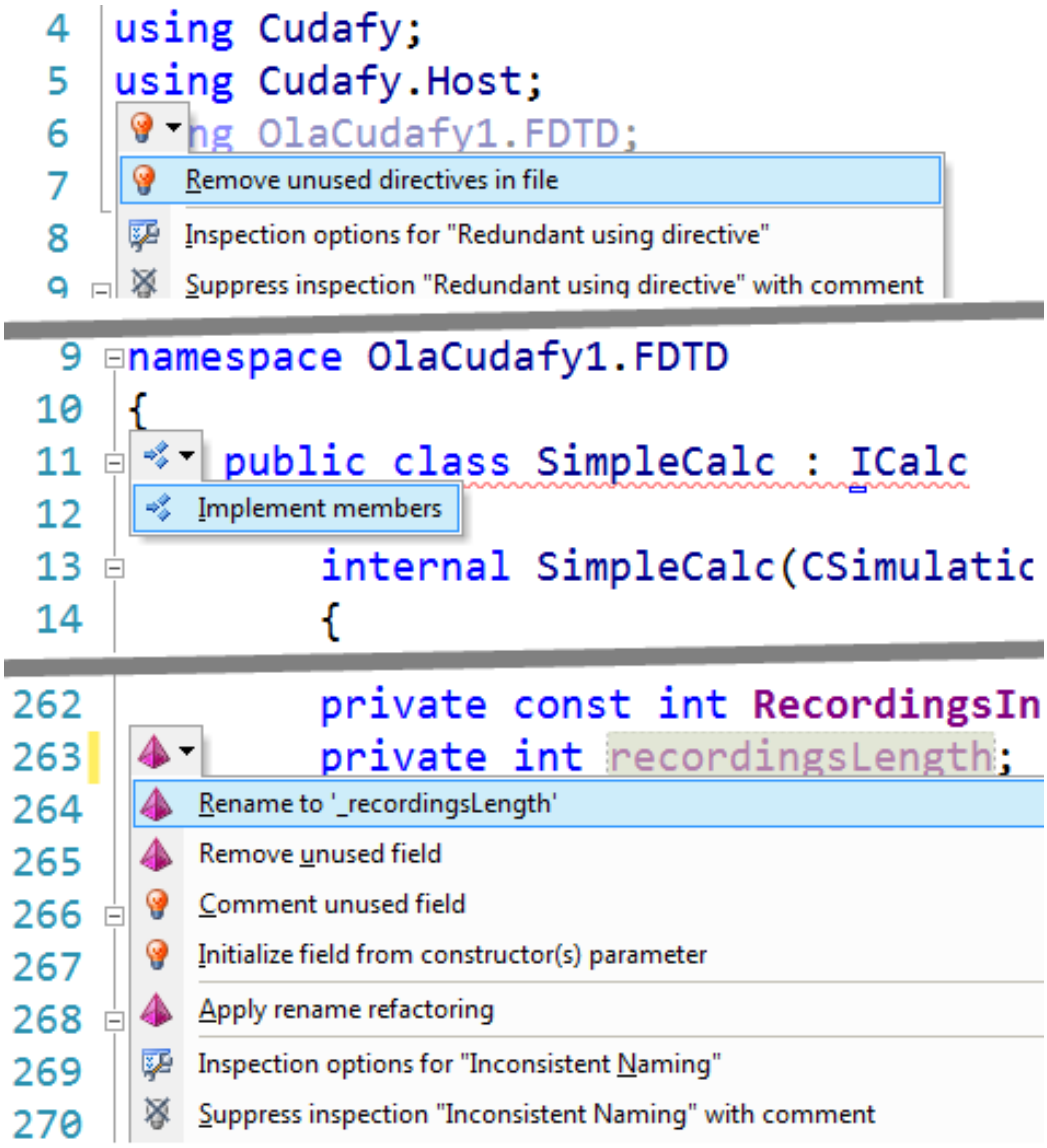
3.2 Programvare

Programmering De følgende verktøy og programvarer ble brukt til programmering:

Figur 3.1: Høyttaler brukt under måling av Fagerengs rom: Genelec 8040A

- Utviklingsmiljøet Microsoft Visual Studio (VS) 2010 på studentlisens gjennom NTNU ble benyttet til all utvikling. Programmeringsspråket C# var det mest naturlige valget fordi dette er godt integrert i VS, svært populært, moderne og under stadig utvikling og utbedring. Forfatteren har mye erfaring med C# og Visual Studio. VS ble valgt fordi det er en av de beste og mest profesjonelle utviklingmiljøene som finnes. Delfunksjonaliteten IntelliSense, som automatisk predikerer og foreslår utskrivinger av variabler, funksjonsnavn og liknende, har en stor del av fortjenesten for dette. VS er vanligvis svært kostbart, men da studenter ved NTNU får gratis lisenstilgang, var dette ikke en hindring. Den andre store ulempen ved VS og C# er at både utviklingen og kjøring av programmet i realiteten må skje på Windows. (Selv om en portabel, alternative implementasjon av C# finnes (Mono), er denne ikke en fullgod erstatning, spesielt fordi den ikke støtter Windows Forms.) Siden Multiconsult som den første store brukeren av programmet likevel kun bruker Windows, ble OS-portabilitet ansett som et mindre problem som det ikke kunne tas hensyn til denne gang.
- Egenbetalt studentversjon av VS-pluginen ReSharper fra JetBrains, som utvider VSs standardfunksjoner. Denne programvare ble brukt til å assistere utvikling, gjennom intelligent fargekoding, auto-completion, refactoring, enkel kodenavigering, linting og til å forsikre overhold av såsom gode kodevaner, navneregler og hvitespace-formattering.
- En prøveversjon av JetBrains dotTrace ble brukt til å *profilere*[1] C#-programmet.
- Biblioteket Cudafy ble brukt for å programmere GPU-en direkte via C#. Cudafy genererer CUDA-kode automatisk. Se underseksjon 4.2.4.

[1]Profilere: identifisere et programs ulike delfunksjoners tidsbruk for å kunne identifisere og nøytralisere flaskehalser.



Figur 3.2: Noen av funksjonene til ReSharper.

Figur 3.3: Programmer og verktøy benyttet. Fra venstre: Visual Studio, TortoiseHG, Dropbox, NVIDIA & Nsight, SketchUp, Zotero, LyX, Paint.Net & MS Paint, og WinMLS.

Andre programvareverktøy

- Google SketchUp 8 ble brukt til å tegne inn modell av Fagerens rom og L-formet rom. En prøveversjon av Pro-versjonen av samme programvare var nødvendig for eksportering av modeller til .obj-format.
- Paint.NET og Microsoft Paint ble mye brukt til å redigere figurene for inkludering i rapporten.
- TortoiseHG ble brukt som grafisk frontend for Mercurial, et versjonskontrollsystem som ble brukt på både programmet og rapportens kildekode.
- I tillegg til versjonering med TorgoiseHG ble alle filer relatert til masterprosjektet (inkludert for eksempel SketchUp-modeller, WinMLS-målinger, bilder av rom, målenotater, og relevante artikler), synkronisert mellom ulike datamaskiner og operativsystemer, og sikkerhetskopiert, vha. "cloud storage"-tjenesten Dropbox.
- NVIDIA Parallel Nsight 2.0 ble installert i den hensikt å debugge og profile kode i CUDA-kjerner via VS. Grunnet blokkerende problemer og manglende progresjon med Cudafy (se underseksjon 4.2.4), samt ingen tidligere erfaring med Nsight, ble ikke Nsight brukt.
- Nødvendige utviklings- og kjøretidsbiblioteker for CUDA ble hentet fra NVIDIAs nettsider og installert.

Rapport Rapporten er er et middel for formidling av masteroppgavens resultater heller enn et forskningsmål i seg selv. Men den representerer en ikke ubetydelig teknisk utfordring som må løses på lik linje med andre utfordringer i løpet av en sivilingeniørs masteroppgave. Derfor et par setninger om rapportskriving:

Konvensjonen blant sivilingeniørstudenter ved NTNU er å skrive i typesettings- og dokumentproduksjonssystemet, Latex, som minner mer om et programme-

ringsspråk. Dette "kompileres" til en PDF, og i prosessen inkluderes og plasseres figurer og litteraturhenvisninger, seksjoner nummereres, og innholdsfortegnelse genereres – alt automatisk. Mange ulike skrivemiljøer for Latex ble forsøkt, men samtlige hadde alvorlige feil og mangler som gjorde dem vanskelige å bruke eller forstyrrende under skriveprosessen, deriblant Texmaker, Bakoma og LEd. Til slutt ble LyX oppdaget, som er en WYSIWYM[2]-editor som produserer Latex-kode *for deg*, før denne blir kompilert til PDF. LyX tillot dramatisk økning av produktiviteten, men også dette skriveprogrammet har utbedringspotensiale og et godt stykke igjen før det kan anbefales for profesjonelt bruk.

Automatisk referansebehandling Referansebehandlingsprogrammet Zotero ble brukt både "stand-alone" og som plugin til Google Chrome og Firefox, til å hente referanser fra Google Scholar-søkeresultater og liknende litteratursøk. Zotero lagrer ønsket referanse i en database, sammen med metadata, artikkel-PDF og link til opprinnelsessted, og synkroniserer automatisk databasen blant dine datamaskiner. Zotero kan eksportere databasen sin til .bib-format for enkel og direkte inkludering i Latex.

3.3 Programmeringsmetoder

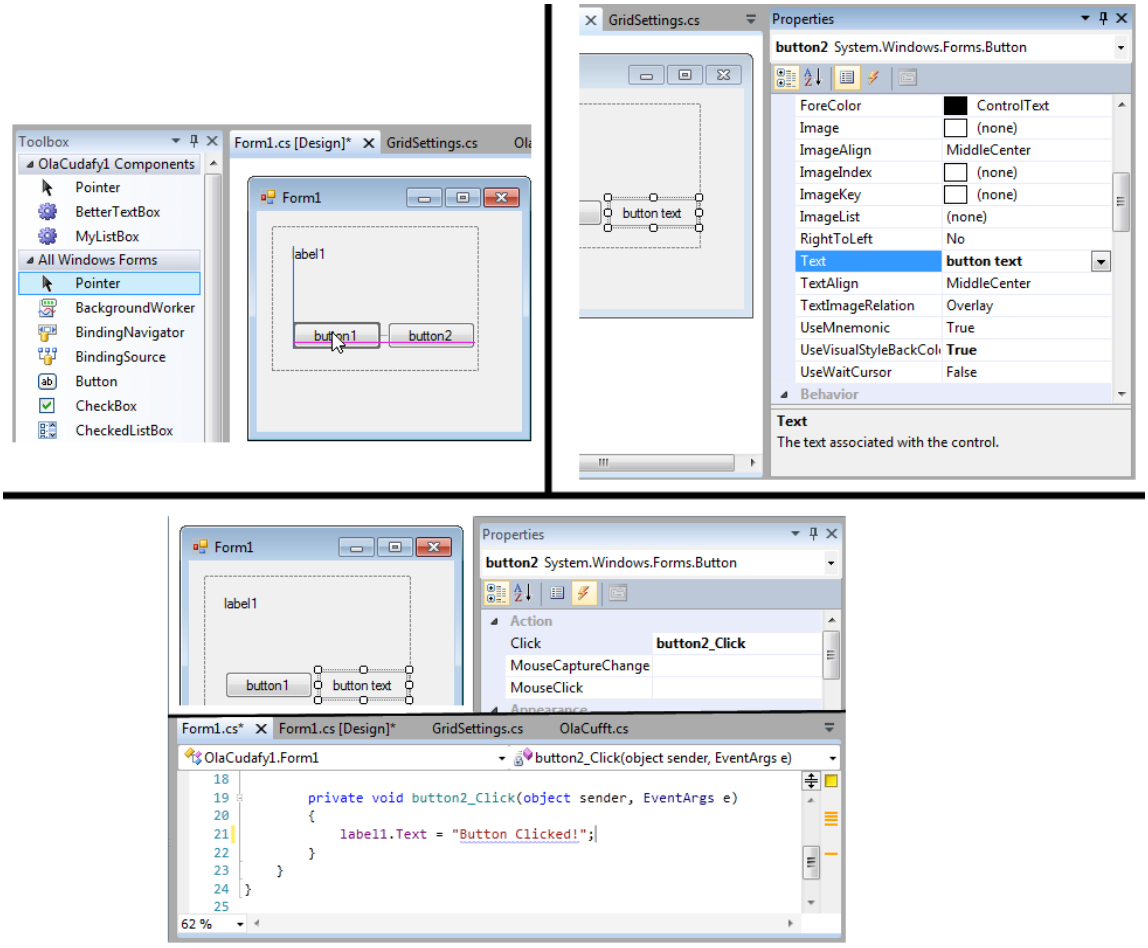
Brukergrensesnittet for programmet ble laget med Windows Forms. Grafisk layout-editor, innstillinger for komponenters egenskaper, og sammenkobling av "events" og funksjoner, er alt bygget inn i Visual Studio – se figur 3.4.

3.3.1 Cudafy

CUDA er på nåværende tidspunkt den mest populære måten å programmere GPUer på, dessuten har jeg erfaring med CUDA. Dermed ble CUDA foretrukket framfor alternativer som OpenCL og Microsofts DirectCompute. I utgangspunktet må CUDA programmeres fra C++, noe som gjør det veldig seint å utvikle og vanskelig å vedlikeholde.

Cudafy, eller mer presist CUDAfy.NET (se figur 3.5), er et bibliotek som lar deg bruke CUDA fra C#. Biblioteket Cudafy er ungt, men lovende og under stadig utvikling. På nåværende tidspunkt støttes tilsynelatende de viktigste delene av

[2] WhatYouSeeIsWhatYouMean



Figur 3.4: Skjermbilder fra Windows Forms-funksjonalitet i Visual Studio: GUI-editor (venstre), instillinger for egenskaper (høyre) og event-sammenkobling (under).

Figur 3.5: CUDAfy.NET-logo og CUDA-logo

CUDA, men Cudafy har fortsatt stort behov for utvikling av dokumentasjon og eliminasjon av bugs.

Kapittel 4

Resultater

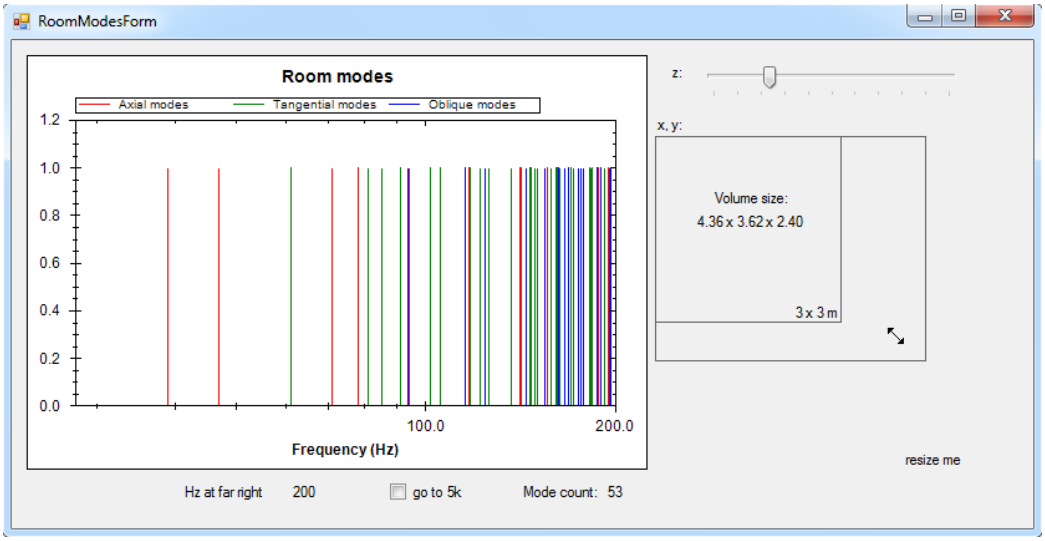
4.1 Utforsking av rom-moder

For å kunne få en forståelse for hvordan rom-moder avhenger av et roms dimensjoner, ble programmet RoomModes laget (se figur 4.1). Grafen i programmet er tegnet med ZedGraph, som er et gratis bibliotek. Modene er regnet ut med de velkjente modelikningene for kasseformede rom, og vist i ulike farger avhengig av type (se figuren). Rommets dimensjoner kan enkelt justeres ved å dra i boksen synlig til høyre, og modene vil "flytte seg" - altså oppdateres - umiddelbart slik at man kan få en følelse av sammenhengen. Grafen kan zoomes og pannes som man ønsker vha. mushjulet, slik at et hvilket som helst frekvensområde av interesse kan vises i grafen (opp til en grense på 5 kHz). Antall (totale) moder telles og kan leses av under "Mode count:" nederst i vinduet.

RoomModes har blitt brukt til å produsere flere av figurene i denne rapporten.

4.2 Programvare

I henhold til oppgavens mål om å produsere en programvare, vil denne seksjonen først oppsummere ønsket funksjonalitet i et idealisert program, og dernest i lys av dette, presentere det programmet som det har lyktes å implementere, gitt tiden som var til rådighet.



Figur 4.1: Programmet RoomModes viser aksielle moder i rødt, tangentielle i grønt og diagonale moder i blått. Modene oppdateres straks rommets dimensjoner endres, ved å dra på firkanten og justere z-slideren. Grafen kan zoomes og pannes for å vise ønsket frekvensområde i det synlige vinduet.

4.2.1 Designforslag

Kravspesifikasjon

Det følgende er en tentativ oversikt over funksjonalitet som bør være tilgjengelig i programmet, og brukergrensesnittet disse skal tilbys gjennom. Den søkte funksjonaliteten har blitt kartlagt gjennom samtaler med akustikk-konsulenter ved Multiconsult om forestilte og antatte bruksscenarioer, hvor liknende programvare også har vært en inpirasjon.

- Modell av rom kan lastes inn fra vanlig 3D-format. Maks-frekvens bestemmes automatisk (Schroeder-frekvens) og rommet vokseliseres deretter i en passende oppløsning.
- Modeformer/-mønstre ($A(\bar{r})$) i snittflate i rommet, for valgt frekvens, tegnet inn i snitt-plan av rommet. Frekvens velges ved å klikke (evt dra) i overføringsfunksjonsgraf (se A.5 for illustrasjonsvideo). Spesielt viktig er visualisering av trykkfordeling i xy-snittflate i ørehøyde for sittende person: 1.2 m. Justerbar høyde og eventuelt snitt-retning.
- Programmet "konverterer"[1] automatisk fra absorpsjonsfaktor til impedans,

[1] Seksjon 4.3.7

og abstraherer dette vekk for brukeren, som kun spesifiserer *absorpsjon for rommets flater* uten å trenge å tenke på impedans. Dette fordi absorpsjonsfaktor α er mer intuitivt forståelig enn impedans, spesielt for akustikk-konsulenter som er erfarne med slike verdier. Absorpsjonsfaktorer kan hentes fra søkbar absorbent- og material-database. Skal kunne endre på materialtilordninger og observere forandringer i akustikken.

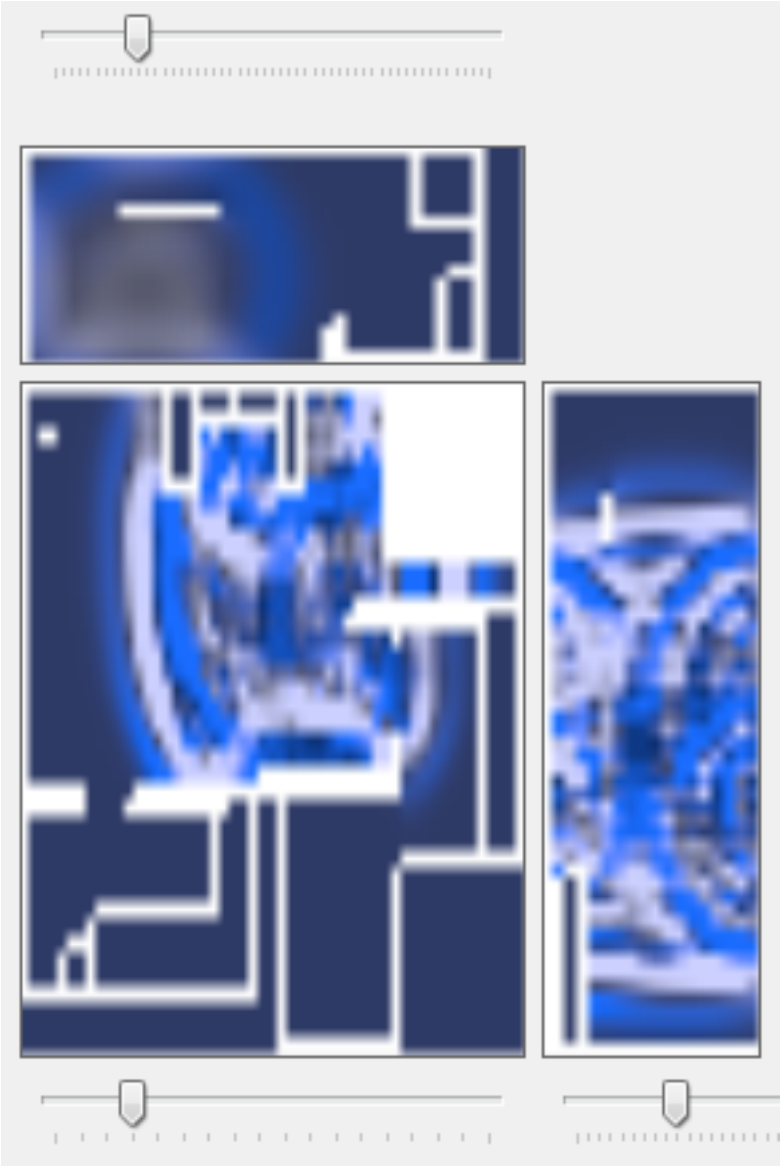
- Et antall kilder kan plasseres forskjellige steder rundt omkring i rommet. Videre; en eller flere lyttepunkter kan spesifiseres. (Kun *rundtstrålende*, da dette antas å gi tilstrekkelig nøyaktig modellering av modeller av høyttalere og lyttende personer ved de aktuelle frekvenser.) Kilder og lyttepunkter kan flyttes på med musen og endring i impulsrespons reflekteres umiddelbart.
- Graf som viser overføringsfunksjon fra høyttalere til lyttepunkt, og hvor lydstyrke i dB kan leses av.
- Frysing av aktuelle plot for sammenlikning av ulike konfigurasjoner.
- Regne ut globale aktuelle tall, for eksempel G (styrke) eller kvalitativt mål på studio-kvalitet.

Kravene er basert på antakelser og vil nødvendigvis endre seg. Nye brukerscenarioer vil alltid bli oppdaget under bruk av et program. God programvare utvikles trinnvis, og helst i tett samarbeid med brukeren.

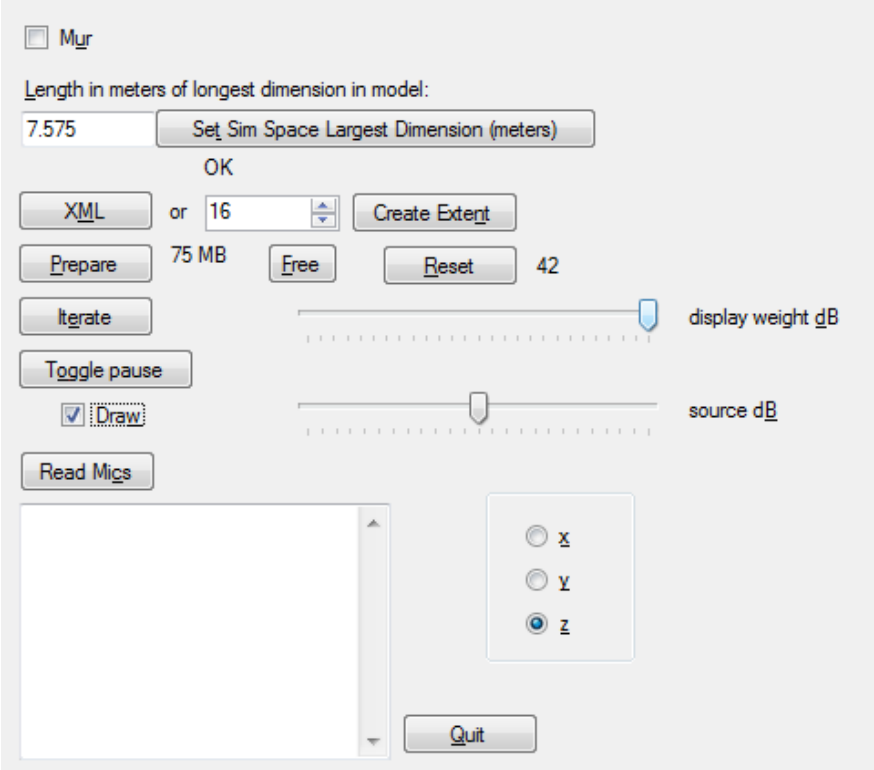
4.2.2 Funksjonalitet i programmet Saka

Det utviklede programmet "Saka" inneholder følgende funksjonalitet:

- Lasting av rom-modell fra XML-fil – krever på forhånd applisering av vokseliserings-verktøykjeden beskrevet under 4.3.1. Rom-dimensjoner må også spesifiseres manuelt.
- Lasting av absorbent-database – beskrevet i 4.3.6.
- Tilordning av akustiske egenskaper fra absorbenter til materialtyper i rom-modellen – (4.3.9).
- Snittflater av rommet i x-, y- og z-retning vises samtidig i tre vinduer (se figur 4.2). Snitthøyden kan justeres fritt. Tidsutviklingen av FDTD-simuleringen vises



Figur 4.2: Saka: Snittflater av lastet rom, med vegger i hvitt, og propagerende lydfelt i blått. Lastet rom er Fagerengs rom: vi kjenner igjen rommet fra snittene.



Figur 4.3: Saka: Knapper, kun for det formål å aksessere så langt implementert funksjonalitet.

- Rammeverket støtter utplassering av et vilkårlig antall kilder og mikrofoner. (*GUI* for dette eksisterer ennå ikke.) Alle mikrofonsignaler registeres og lagres.
- Materialimpedans støttes i FDTD i den grad at vegger kan ha frekvensuavhengig, reell absorpsjon – se 4.2.3.
- Ovenfor nevnte funksjonalitet er tilgjengelig gjennom knapper og felter vist i figur 4.3.

Mye av denne funksjonaliteten overlapper med "qtsaka", programmet som ble utviklet under prosjektoppgaven min. Likevel, Saka er en fullstendig omskriving av "qtsaka", og er bedre designet og planlagt, og legger slik bedre til rette for utvidelse av GUI, og så som FDFD-simulering. "qtsaka" ble skrevet i C++ og er derfor svært krevende å utvikle og vedlikeholde.

4.2.3 Implementasjonsdetaljer (`C#`-implementasjon av simuleringsprogram)

Ønske i punkt 3 i tilbakemeldingen fra engasjerende akustiker (se A.7) lyder "[Det er] viktig å kunne endre på overflatenes impedans; først og fremt på tap, dvs. realdelen. Dette har stor betydning for hvor glatt eller skarp overføringsfunksjonen blir. Se eksemplene til slutt i vedlagte PowerPoint." For å møte dette ønsket ble absorpsjon i FDTD ble implementert i henhold til metoden beskrevet i Hills AES-artikkel[25].

Metoder for å implementere *fullstendig frekvensavhengig impedans* ble undersøkt og forsøkt utarbeidet, med inspirasjon fra masse-fjær-demper-metoden beskrevet i [26]. Det ble dessverre ikke anledning til å fullføre spesifikasjon og implementasjon av en slik modell i GPU-kode.

4.2.4 Cudafy

Cudafy (3.3.1) ble utforsket gjennom eksempler som følger med biblioteket. Eksemplene var lette å forstå, få til å virke samt modifisere, og Cudafys programmeringmodell var også rimelig lett å forstå. Det ble besluttet å gå videre med Cudafy. Bruken ville bestå i å først forsøke å "porte" FDTD-algoritmen fra forfatterens prosjektoppgave i programmet "qtsaka", fra CUDA til Cudafy.

Indekserings-bug Etter nøye porting av algoritmene for FDTD-iterasjonskalkulasjon og lydfelt-visualisering, viste visualiseringene at noe åpenbart måtte være feil på et eller annet punkt. Unit-tester ble utarbeidet for å kunne identifisere feilen. Kernelene ble kjørt *som C#-kode* slik at de kunne stegvis debugget, og resultatet ble sammenliknet med manuelle kalkulasjoner. Dette viste ingen feil. Dermed måtte uoverensstemmelsen ligge mellom resultat fra emulert kode og resultat fra CUDA-oversettelse på GPU. Etter mye leting, ble det oppdaget at CUDA-oversettelsen (CUDAFYSOURCETEMP.cu) som Cudafy produserer, inneholdt en svært alvorlig bug: I CUDA er man nødt til å aksessere et array i minnet via lineære indekser, og med et forskyvningstillegg ("offset" eller "stride") i tilfelle flerdimensjonale arrayer. For eksempel må element [x, y] ([2, 5]) i et 2-dimensjonalt array av dimensjoner [xSize, ySize] ([10, 20]), indekseres med følgende lineære indeks: x+y*xSize (2+5*10). Cudafy abstraherer vekk dette slik at man fra C# kan behandle arrayene som flerdimensjonale og aksessere dem på vanlig måte: array[x, y] vil da automatisk regnes om til x+y*xSize. Men, i Cudafys oversettelse til CUDA-kode var det ikke tatt høyde for at indeksene kunne inneholde sammensatte uttrykk (som må evalueres først). Dermed ble uttrykket array[x, y+1] som brukes i FDTD, uheldigvis oversatt til x+y+1*xSize. Denne feilen er såpass åpenbar, at det ikke skulle tenkes å gå an å lage den. Av den grunn ble andre potensielle feil undersøkt først, og det tok dermed lang tid før denne feilen ble funnet. Rundt årsskiftet ble denne buggen fikset i neste versjon av Cudafy og utvikling kunne dermed fortsette.

FFT går veldig raskt på GPU, gjennom delfunksjonaliteten CUFFT i CUDA. Denne funksjonaliteten skal i følge API-dokumentasjon også være reflektert i Cudafy, men da hverken eksempler eller dokumentasjon for bruk av Cudafy sin FFT var å finne, lot det seg ikke gjøre å ta frekvenstransformasjonen av det registrerte mikrofonsignalet i Saka.

4.2.5 Profiling

For å øke hastigheten på FDTD-simuleringen, ble JetBrains dotTrace (3.2) brukt til å profilere Saka. Dette ledet til identifikasjon av en forsinkende faktor ved visning av DataGridView: Dersom både rader og kolonner settes til å auto-skaleres, tar utfylling av tabellen uforholdsmessig lang tid. Det samme gjelder om autoskalerings-feltet settes per kolonne i stedet for for hele tabellen som helhet. Disse problemene ble nøytralisert.

4.3 Geometri og grensebetingelser

FDTD og FD**F**D er to ulike metoder med ganske ulikt resultat, men de kan bruke nøyaktig samme type diskretisering av simuleringsrommet i kuber/voksler/celler. Derfor vil de begge dra nytte av de verktøy som beskrives i det følgende.

4.3.1 Verktøy for import av geometrimodell

SketchUp[2] ble valgt som tegneprogram for modellene som skal simuleres; dette av tre grunner:

- Mange modeller av eksisterende såvel som planlagte rom finnes i SketchUp-format allerede. Alternativt er SketchUp ofte i stand til å importere eller konvertere modellen fra dens nåværende format.
- SketchUp er enkelt å bruke: det er enkelt å tegne inn kompliserte geometrier, i 2D og ikke minst i 3D.
- SketchUp tillater eksportering av modellen til mange ulike formater[3].

Med andre ord oppfyller SketchUp kravene som ble beskrevet tidligere, under 2.7.1.

4.3.2 Westerdieps *voxelizer*

Det ble vurdert å implementere en relativt enkel vokseliseringsalgoritme. En slik implementasjon ville i så fall ha blitt gjort så enkel som mulig, ved for eksempel å ha minimal støtte for spesialtilfeller som løse vegger[4]. Riktignok kunne selv en enkel implementasjon bli tidkrevende, derfor ble det besluttet å først undersøke eksisterende løsninger.

[2]Sketchup, nytt anno 1999, ble overtatt av Google i 2006. SketchUp er et tegne- og skisseprogram for 2D og 3D-skisser som har vokst sterkt i både evne og popularitet de senere år. SketchUp brukes nå av profesjonelle arkitekter såvel som amatører til å skissere alt fra enkle geometrier til hele bygninger, og også modeller for bruk i animert film.

[3]Dette gjelder riktignok bare Pro-versjonen. Men, siden Pro-versjonen kan lastes ned og brukes gratis for en viss periode, samt fordi prisen på fullversjonen er svært akseptabel, sees det ikke på som noen begrensning å anta at fullversjonen av SketchUp foreligger for brukeren av simuleringsprogrammet.

[4]Det vil si polygoner som, ikke betegner en begrensningsflate i ytterkant av simuleringsrommet, men som står "midt i løse lufta" inne i simuleringsrommet.

Det ser merkelig nok ikke ut til å være (etter forfatterens beste kjennskap) noen utbredt bruk av vokseliseringsprogramvare i FDTD-litteraturen. Det var derfor ikke mulig å finne noen ferdig løsning på denne utfordringen. I stedet ble et utvalg vokseliseringsprogramvare som var tilgjengelig på internett for andre formål, undersøkt som kandidater.

Kravet til et tilfredsstillende program var

- Intuitivt i bruk, eller ekvivalent: tilstrekkelig brukerdokumentasjon
- God voksel-gjengivelse av polygonmodellen uten synlige feil
- Stand-alone, altså ikke plugin til 3D-programmer, Matlab e.l.
- I stand til å registrere material-informasjon i vokslene

Av de undersøkte programmer (se neste avsnitt), ble Arjan Westerdieps **voxelizer**[5] funnet å være enklest i bruk og samtidig produsere den riktigste gjengivelsen av test-modellen.

Programmet er et javascript- og flash-program som kjører gjennom nettleseren, og krever ingen ekstra programvare. Det importerer .dxf og .obj-filer, og tillater justering av oppløsningen, det vil si størrelsen på vokslene. Som tidligere nevnt støtter pro-versjonen av SketchUp eksport til flere formater, deriblant .obj[6]. Modellen ble derfor overført fra SketchUp til **voxelizer** i dette formatet.

Imidlertid eksporterte originalversjon av **voxelizer** en *polygonrepresentasjon* av de produserte vokslene – ikke simpelthen koordinatene til vokslene, som er det vi ønsker. Programmet lagret heller ikke materialinformasjon i vokslene.

Programmets opphavsmann ble derfor kontaktet, og fortalt om denne masteroppgavens intensjoner. Westerdiep produserte så en ny versjon, som nå er tilgjengelig på www.drububu.com/ola. Denne versjonen eksporterer vokselenes *posisjonener* samt *materialdata* til filer av typen XML[7].

Lisensen for bruken av Westerdieps program er ikke blitt diskutert, og må naturligvis avklares før programmet kan brukes i produksjon.

[5] www.drububu.com/voxelizer

[6] .obj er et enkelt men tilstrekkelig format som er viden kjent og brukt, og støttet av mange programmer

[7] XML er et godt standardisert og svært utbredt format.

4.3.3 Undersøkte programvarepakker for vokselisering

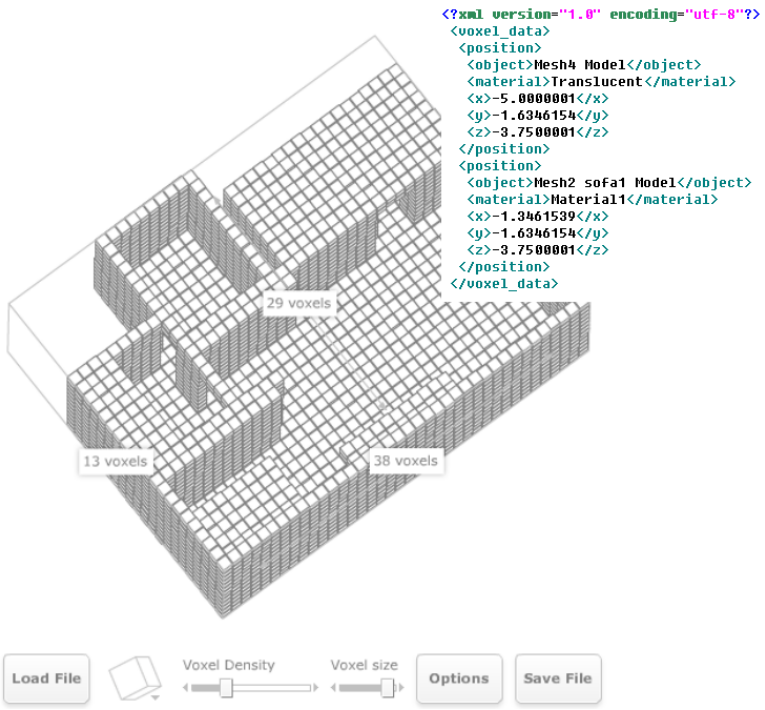
Som beskrevet i avsnitt 4.3.2 ble også andre programmer fra internett undersøkt som kandidater for vokseliseringsprogramvare. Her er disse listet, sammen med en begrunnelse for hvorfor de ikke ble valgt.

- 3D voxelizer[8], av Carlos Martinez-Ortiz. Dette programmet avhenger av Matlab. Dette er uheldig fordi Matlab er både kostbart og tidkrevende og vanskelig å installere, noe som vil medføre en hindring om 3D voxeliser var en nødvendig del av vår verktøykjede. Videre ser 3D voxeliser ikke ut til å støtte ulike materialer i begrensningsflatene.
- Voxelizer, av Dan Morris ved Stanford Haptics Lab[9]. Dette programmet virker bare for "vanntette" overflater (surfaces) i rommet, og "maler" dem med vokseler, i stedet for å fylle dem som er det vi ønsker å gjøre.
- binvox[10]. Dette er et kommandolinjeprogram med flere opsjoner. Under test produserte programmet upresise gjengivelser av test-modellen, for alle kombinasjoner av opsjonsvalg utprøvd.
- Luis Manuel Morillo (Luima) luima.com/voxpro.htm. Dette programmet er et skript for 3D studio max 2008. Her gjelder det samme argumentet som for 3D voxeliser men i enda større grad: vi bør ikke gjøre oss avhengig av store, eksterne programmer.
- CVMLCPP: Voxelizer, fra tech.unige.ch/cvmlcpp/source/doc/Voxelizer.html. Dette er et programvarebibliotek (library) for C++, og må følgelig inkluderes i en implementasjon før det kan brukes. Biblioteket ble av den grunn ikke vurdert denne gang, men kan være en kandidat for inkludering i en fremtidig versjon.
- I tillegg ble enkelte andre programmer også såvidt vurdert, men disse ble fort forkastet da de enten ikke ville kompilere eller manglet tilstrekkelig brukerdokumentasjon.

4.3.4 Tolkning av XML-filen

Filene som produseres av Westerdieps program inneholder ingen direkte informasjon om vokseloppløsningen, altså antallet kuber, i de tre vektorretningene,

[8] www.mathworks.com/matlabcentral/fileexchange/21044-3d-voxel
[9] techhouse.org/dmorris/projects/voxelizer
[10] www.cs.princeton.edu/min/binvox



Figur 4.4: Vokselisering av Fagerengs rom i Westerdieps *voxelizer*-program. Taket er klippet vekk for gjennomsynlighet.

og spesifiserer heller ikke hva slags lengdeenhet koordinatene er notert i. Derfor var det behov for programlogikk som kunne utlede dette. Det ble implementert logikk som løser denne oppgaven gjennom en kombinasjon av telling og gjetting – implementasjonen er å finne i den vedlagte programkoden; se avsnitt **??**. Denne metoden antas å virke for de fleste praktiske tilfeller, men en framtidig, fullgod løsning krever ikke mer enn at vokseliseringsprogrammet modifiseres til å i stedet notere vokselers koordinater som x-, y- og z-indekser.

Videre produserer den implementerte logikken en representasjon av material/voksel-dataene, som består i en liste over vegg-celler, hvor hver vegg-celle har en tredimensjonal koordinat og et material.

4.3.5 Vokselisering av Fagerengs rom

4.3.6 Absorbent-database

Multiconsult vedlikeholder en absorpsjonsdatabase med ca. 13000 oppføringer, som inneholder statistisk absorpsjonsfaktor i de åtte frekvensbåndene 63, 125, 250, 500, 1k, 2k, 4k og 8k Hz, for en rekke ulike materialer, sammen med en kort, presis beskrivelse av hvert material.[11] Absorpsjonstall for akustiske materialer fra ulike produsenter, samt tall for vanlige byggematerialer, er med i databasen. Absorpsjontall for 63Hz-båndet er unøyaktige eller fraværende for de fleste materialene[12] – dette fordi klangromsmetoden som vanligvis benyttes til å finne absorpsjonstall, ikke er presis for frekvenser under noen hundre Hz. [15, seksjon 5.1 avsnitt 5.]

Parsing av databasen Databasen er en tekstfil med filendelse .LI8. Hver oppføring går over to linjer, og er adskilt av en tredje, tom linje. En oppføring følger alltid følgende format: Linje 1: ID (heltall) – mellomrom – beskrivelse (tekst) – linjeskift. Linje 2: mellomrom – 8 absorpsjonskoeffisienter som IEEE-flyttalls-streng adkilt av mellomrom – linjeskift.

En programsnutt ble implementert for å parse (syntaksanalyse) denne databasen og lagre den i en veldefinert datastruktur. Kildekoden er gjengitt i vedlegg A.2. Implementasjonen benytter såkalte "states", og er robust mot feil i databasen ved at den vil hoppe over eventuelle deler som ikke følger syntaksen.

4.3.7 Fra statistisk absorpsjonskoeffisient α til materialimpedans Z

J. H. Rindel beskriver i [27] en prosedyre som tillater anslag av et materials impedans gjennom å oversette fra materialets absorpsjonskoeffisient α_{stat}. [13] Noe

[11] Et eksempel på en oppføring er
ID: 2343; Absorpsjonstall: 0.16 0.16 0.15 0.07 0.08 0.05 0.06 0.06; Beskrivelse: Walls, 13 mm plaster on 25 mm studs (nomineral wool) Ref. Dalenbäck, Datensatz der CATT-Software, November 2000. Data-Category in CATT: PLASTER3, C: Walls, D: 13

[12] i følge J. H. Rindel

[13] J. H. Rindel er administrerende direktør i Odeon A/S som produserer Odeon, et profesjonelt akustikk-simuleringsprogram som benytter en hybridmodell hovedsaklig basert på strålemetoden. Materialers impedans, og da spesielt dens imaginære del, er av betydning i strålemetoden da en

liknende ser ikke ut til å være gjort tidligere i litteraturen, tross behovet for kunnskap om impedans både i geometriske og numeriske simuleringsmodeller. Prosedyren for slik oversetting oppsummeres i det følgende.

NB: For at det skal gi mening å anvende denne metoden i programmet, må materialene i rommet som skal simuleres først være utvetydig spesifisert; men la oss inntil videre anta at dette problemet er løst: Hver type flate i modellen er tilordnet det materialet i databasen[14], som har α_{stat}-verdier så nært som mulig det de er i virkeligheten.

Prosedyren beskrevet i Rindels artikkel, som er begrenset til å gjelde ved lave frekvenser, går i korte trekk ut på følgende.

4.3.8 Materialklassifikasjon på bakgrunn av α_{stat}

For et gitt materiale og dets α_{stat}-verdier, gjetter man hvilken av de tre absorbentkategoriene materialet tilhører. Kategoriene er de vanlige absorbentkategoriene *porøs*, *membran* og *resonator* [15, avs. 5.2]. Gjetningen baserer seg på en erfaringsmessig kartlagt sammenheng mellom absorbentkategori og α_{stat}s størrelse og variasjon over de lavere frekvensbåndene.

Deretter beregnes impedansen utfra en modell for den aktuelle absorbenttype. (Se [15, avs. 5.4] som beskriver modeller for sammenhengen mellom Z og α_{stat} i de ulike absorbenttypene.)

Siden impedansen er frekvensavhengig og følgelig må beskrives som en funksjon av ω, antar Rindel en enkel førstegrads svingemodell felles for de tre materialtypene, og beskriver impedansen gjennom de tre frihetsgradene i denne modellen: masse, demping og fjæring. Fra artikkelen:

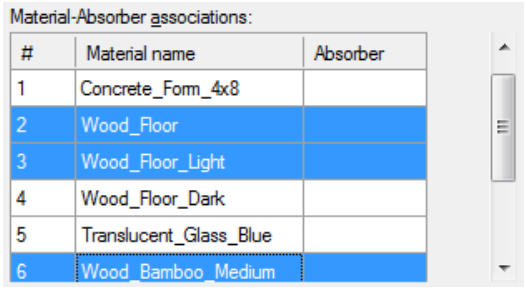
$$Z_a(\omega) = R + jX(\omega),$$

hvor $Z_a(\omega) \in \mathbb{C}$ er impedansen bestående av $R \in \mathbb{R}$ – den akustiske resistansen, og $jX(\omega) \in \mathbb{C}$:

$$X(\omega) = \omega m - \frac{\rho c^2}{\omega d},$$

"lydstråle" vil bli utsatt for en fase- eller tidsforskyvning etter refleksjon fra et materiale med kompleks impedans.

[14] absorpsjonsdatabasen, beskrevet under (4.3.7).



Figur 4.5: Materialnavn fra importert modell i Saka, merket for tilordning av absorbenter.

hvor m er flatemassetettheten (kg/m^2) og d er dybden til luftrommet bak massen. Konstanten ρ er lufttettheten. (Se [27] for ytterligere detaljer.)

Artikkelen tar videre for seg et eksempel fra hver av absorbenttypene, beregner konstantene i svingemodellen i henhold til formlene, og estimerer så på nytt α_{stat}-verdier for den enkle modellen av materialet. Rindel viser at alle de tilbakeberegnede verdiene stemmer godt, samt argumenterer for at absorpsjonsoppførselen til modellen kan ekstrapoleres nedover i frekvens for å få et anslag for det tidligere ukjente 63Hz-båndets α_{stat}.

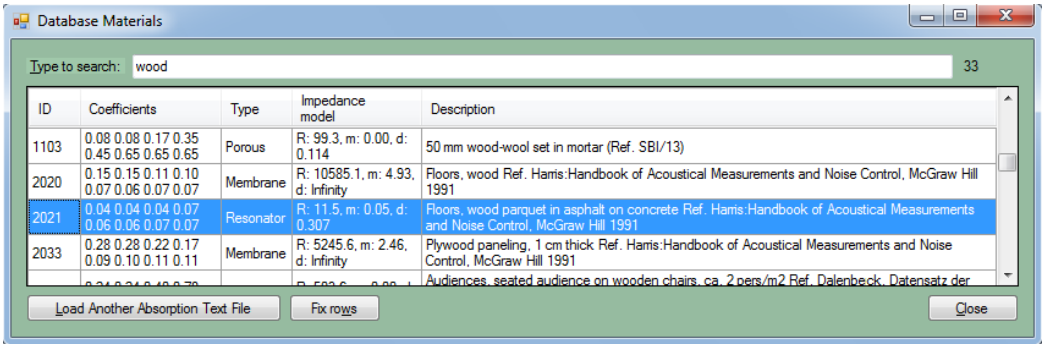
Rindels metode har blitt implementert, og implementasjonen har blitt "unit"-testet med eksempeltilfellene fra artikkelen. Kildekode finnes i vedlegg A.3. Rindels metode blir benyttet på alle oppføringene i absorbentdatabasen, slik at absorbenttype og koeffisienter fra impedansmodellen kan listes sammen med absorpsjonskoeffisientene i Saka – se seksjon 4.3.9.

I løpet av prosessen ble forøvrig en feil oppdaget i et av eksempeltallene i artikkelen, og feilen ble rettet i samarbeid med forfatteren.

4.3.9 Brukergrensesnitt for materialassosiasjoner / Beskrivelse av delfunksjonalitet

Det følgende forklarer prosedyren for materialtilordning i Saka, det vil si hvordan materialene i den lastede rom-modellen gis de ulike akustiske egenskapene de skal modelleres med under simuleringen.

Når en modell av et rom er lastet i Saka (fra .xml), vil materialene i modellen vises i en liste (se figur 4.5) i hovedvinduet. Disse representerer kun navn opprinnelig tilordnet de ulike polygonene i SketchUp-modellen, og er ennå ikke tilknyttet noen informasjon om akustisk oppførsel. Disse kan så tilordnes ulike

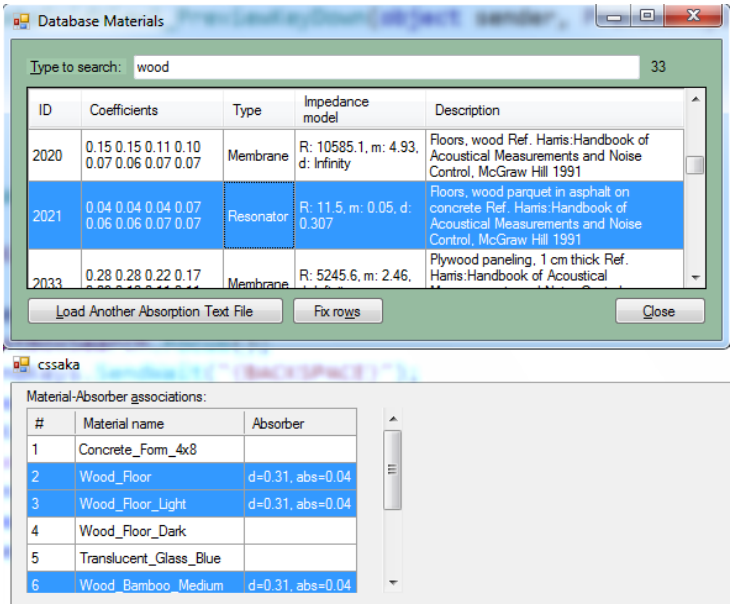


Figur 4.6: Absorbenter fra database vises søkbar tabell i eget vindu i Saka

absorbenter og materialer fra en absorbentdatabasen beskrevet i seksjon 4.3.6. Et antall materialer kan tilordnes samtidig, på følgende måte: De materialene som skal tilordnes velges med Ctrl- eller Shift-klikk, og så trykker man Enter. Da vil et nytt vindu åpnes (figur 4.6), som inneholder et søkefelt, samt alle oppføringene i databasen listet i et DataGridView – en generell og svært konfigurerbar Forms-komponent ("control") for visning av tabelldata. Vinduet vil automatisk laste Multiconsults absorbent-database, men andre tilsvarende .li8-databaser kan eventuelt lastes manuelt. Deretter vil koeffisientene i Rindels impedansemodell (se seksjon 4.3.7) regnes ut for alle oppføringene i databasen. Søkefeltet tillater en å søke i beskrivelsene og ID-nummerene, og søkeresultatet etter hvert tastetrykk vises umiddelbart i tabellen. Denne "search while typing"-funksjonaliteten er implementert som et DataView-filter. (Søk vil oppleves som responsivt nok for databaser opp til en viss størrelse; for større databaser må en ny implementasjon benytte indeksering.) Relevant kildekode er gjengitt i A.1.

Akustikeren får så utfordringen med å finne oppføringer i absorbentdatabasen som samsvarer godt med veggene i det planlagte eller eksisterende rommet. Når absorbenten som ønskes tilordnet er funnet, velger man denne ved å dobbeltklikke eller trykke enter, og den tilordnes dermed til de valgte materialene. Tilordningen blir synlig i hovedvinduet (figur 4.7). Deretter kan prosedyren gjentas for å tilordne resterende materialer.

Hvordan absorbent-dataene brukes (eg. absorpsjonskoeffisient eller koeffisient fra Rindels impedansmodell) er opp til simuleringsmetoden å velge, og blir ikke bestemt her. Verdiene under *Absorber*-kolonnen i figuren er hentet ut kun for illustrasjonsformål, og materialnavnet er i stedet faktisk tilordnet selve databaseoppføringen.



Figur 4.7: Absorbent er blitt assosiert med materialnavn.

4.4 Måling av rom

For å ha et realistisk mål å teste beregningene mot, ble det på et tidlig stadium besluttet å gjøre målinger i et eksisterende studio. Tidligere kollega ved Statsbygg Morten Halmrast introduserte student Rune Fagereng, som på samme tidspunkt også skrev en masteroppgave om lavfrekvent lyd i studioer, men fra et musikkteknisk perspektiv. Fagereng hadde et studio (i Hønefoss) som han ønsket å gjøre akustiske målinger av før og etter installasjon av absorbenter. Målet med installasjon av absorbenter ville være, 1) å vise at problematiske moder kan forminskes ved riktig plassering av passende bassabsorbenter, gjennom å sammenlikne frekvensresponser fra før og etter installasjon, og 2), å benytte Saka til å assistere plassering av absorbenter. BBC radios membranabsorbenter beskrevet i [28] er små, enkle i konstruksjon, og effektive for svært lave frekvenser (godt under 100 Hz). Et samarbeid ble satt i gang, som gikk ut på at jeg skulle skaffe til veie måleutstyr og utføre målingene, mens Fagereng skulle tilby rommet og bygge bassabsorbenter.

Fagerens rom, som var planlagt brukt som hjemme-studio, viste seg å avvike sterkt fra et profesjonelt studio på mange måter: Geometrien er komplisert. Et kjøkken er koblet til stuen og det finnes ingen dør som kan lukkes. Det samme gjelder entreen. (Badet har riktignok en dør, selv om dette ikke går fram av SketchUp-figuren.) Stuen er kasseformet og har uheldige dimensjoneringsforhold

mtp. modesammenfall. Stuen er omsluttet av betong i gulv, én langvegg og én kortvegg, av en tilsynelatende lett eller hul vegg mot kjøkkenet, og har en skillevegg stikkende langt inn i rommet (se figur). Det var en seng og en sofa installert i stuen som ytterligere ville komplisere påfølgende modellering av rommet. Tross rommets kompleksitet ble det besluttet å fortsette samarbeidet og begynne målingene som planlagt, fordi tilfellet representerer en realistisk situasjon som likevel ønskes løst.

Rommet ble målt opp med målebånd og en modell av rommet ble tegnet i SketchUp (se figur 4.9 og 4.10). I begynnelsen ble lengder målt nøye (ned til halve cm), men etterhvert ble avvik på 1-3 cm ignorert, når det ble innsett at kravet til målenøyaktighet i lengder er lavt for de bølgelengdene vi skal betrakte. Teksturene synlig i modellen var på dette punktet kun ment å likne flatenes bekledning i utseende. Grunnet manglende erfaring i å fastslå materialers typer eller egenskaper, ble en filmsnutt tatt opp hvor alle veggtyper ble banket på etter tur, for siden å konsultere erfarne personer om veggenes beskaffenhet.

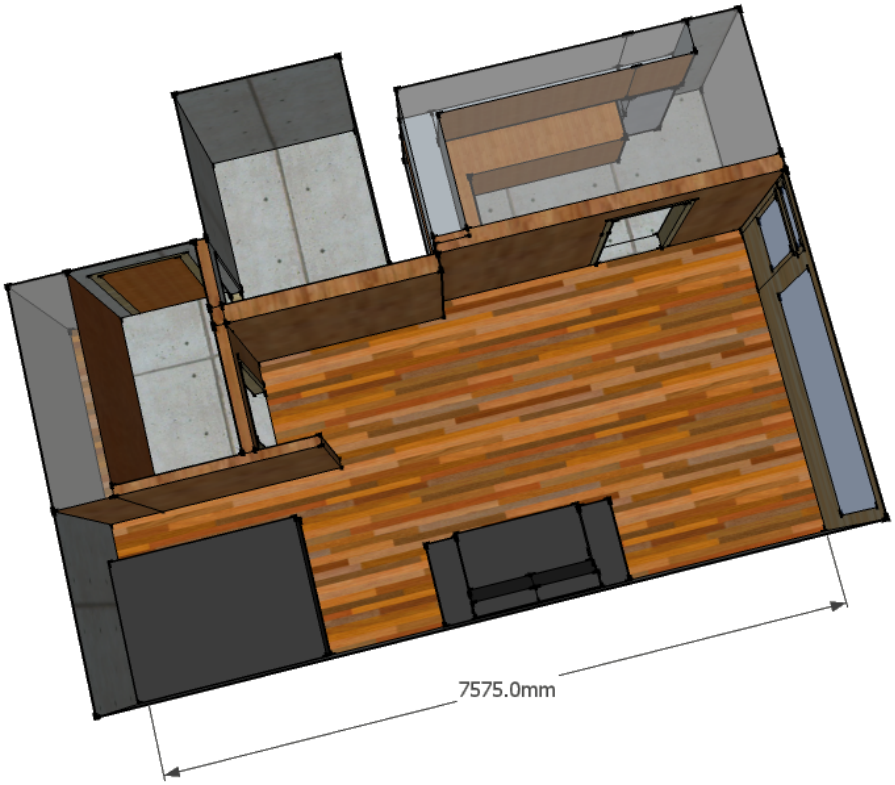
For å etterlikne en realistisk situasjon, ble de opprinnelig installerte høyttalerne brukt som lydkilder (se 3.1). Mikrofonen ble plassert på ulike steder og høyder rundt omkring i rommet, samt på en strategisk samling punkter (foroverlent; høyre øre o.l.) rundt lytteområdet foran miksepulten. Sinussveip ble eksitert fra én høyttaler av gangen (L+R) og målingene registrert og lagret, alt vha. WinMLS. Mikrofonposisjonen ble målt med lasermåler og notert med millimeterpresisjon for hver måling, slik at målingene kunne gjentas med likt målingsoppsett etter absorbenter var installert. (Høy presisjon ble ivaretatt fordi Fagereng potensielt ønsket å benytte denne informasjonen.) Til sammen ble omlag 63 individualle målinger utført. Notatene fra målingene (samt høyttalerposisjonene) er tilgjengelig i vedlegg A.6.

Fotografier ble tatt av rommet fra alle retninger, av hvert måleoppsett samt av enhver egenskap ved rommet og interiør som kunne tenkes å være av interesse for fremtidig modellering og tolkning av resultater.[15]

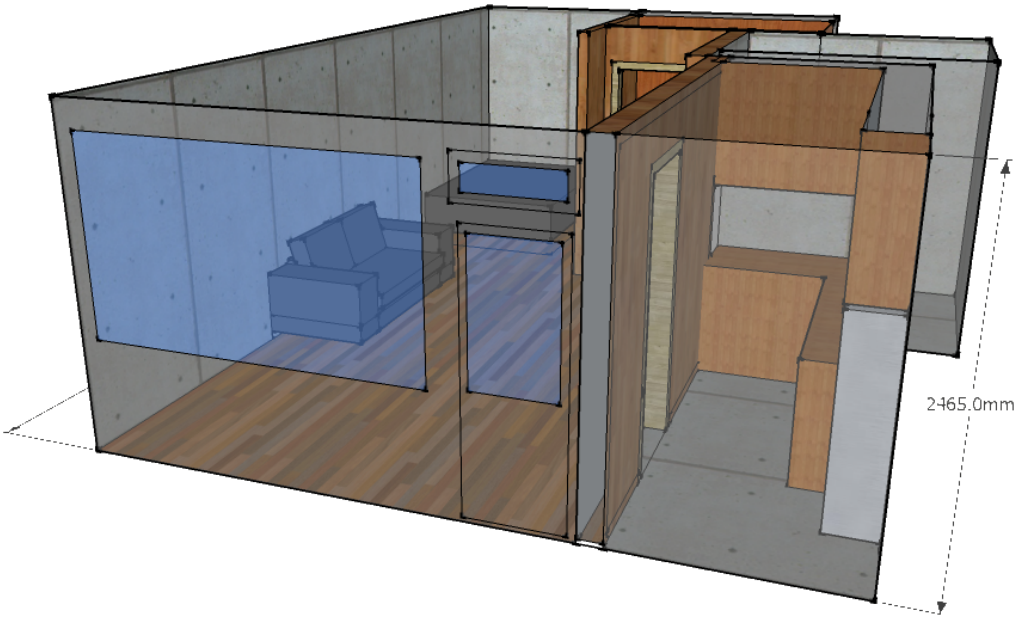
4.5 Utprøving av Odeon for lave frekvenser

For å vurdere Odeons evne til å produsere presise simuleringsresultater for lave frekvenser, ble følgende eksperiment utført: Impulsresponsen mellom to punkter i et L-formet rom med 10% absorberende vegger beregnes, først i Odeon og så

[15] Alle fotografiene er vedlagt i egen fil ved innlevering til sensur.



Figur 4.8: SketchUp-modell av Fagerengs rom (perspektiv 1)



Figur 4.9: SketchUp-modell av Fagerengs rom (perspektiv 2)

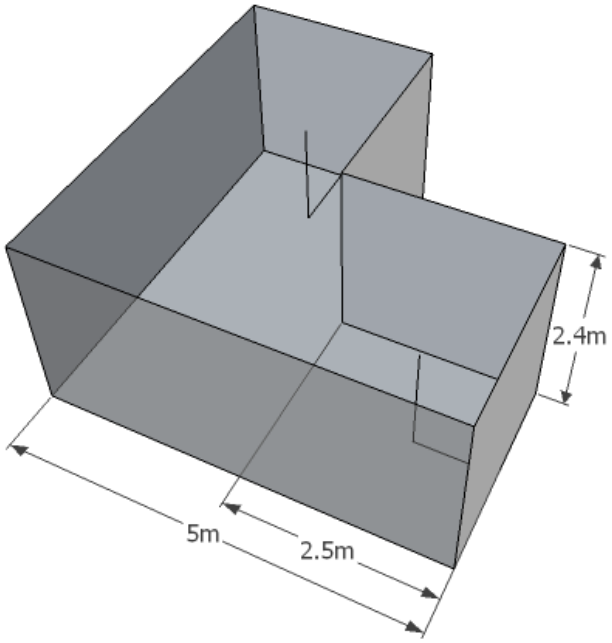
Figur 4.10: SketchUp-modell av Fagerengs rom (perspektiv 3)

Figur 4.11: Fagerengs rom: Måling ved miksepult

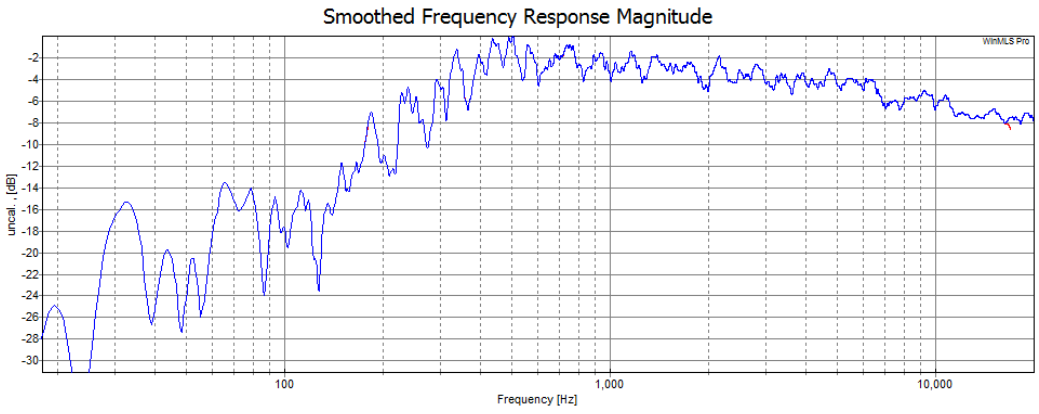
med oppgavens FDTD-implementasjon. De lav-frekvente delene (under 200Hz) av impulsresponsene sammenliknes i de følgende figurer.

I figur (4.12) ser vi SketchUp-modellen av det L-formede rommet, som var utgangspunkt for eksport til både Odeon gjennom Odeons SketchUp-plugin, og til FDTD: SketchUp-modellen ble konvertert for bruk i FDTD via obj-xml-verktøykjeden beskrevet under (4.3.2). Vokseliseringen av rommet for dette formålet er vist i figur 4.14.

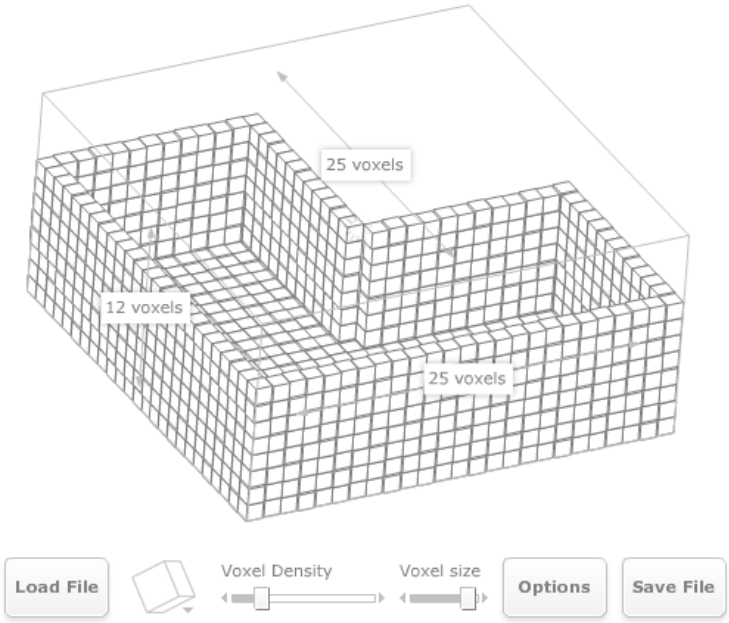
I ettertid ble det innsett at, selv om FDTD-beregningen må tenkes å produsere et resultat for lave frekvenser som er betydelig overlegent Odeons, er det uten et referansepunkt – som eksempel en måling, en svært nøyaktig beregning eller en analytisk løsning – vanskelig å si hvilken av de to beregningsmetodene som gir et bedre resultat. Dette problemet kunne ha vært unngått dersom et kasseformet rom hadde blitt valgt i stedet, siden det som vi har sett er enkelt å finne modene for et slikt rom analytisk. Da kunne man ha sett resultatene fra Odeon og FDTD i lys av de analytisk beregnede modene.



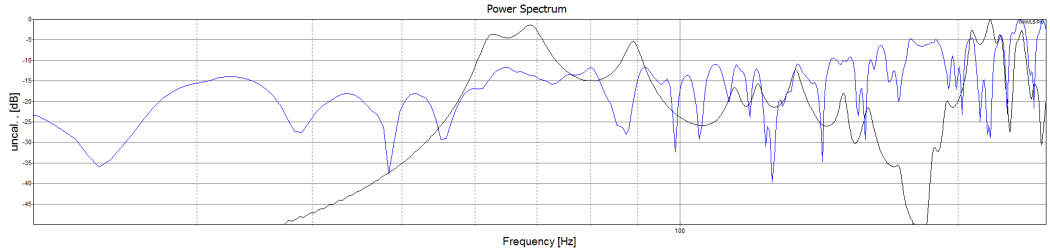
Figur 4.12: Sketchup-modell av L-formet rom. Mål: 5 x 5 m, høyde: 2.4 m



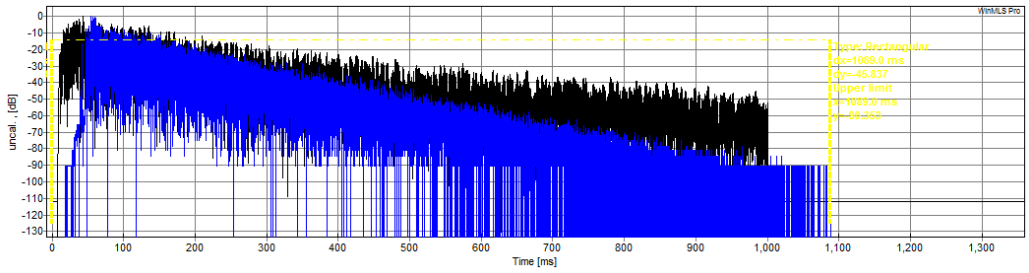
Figur 4.13: Overføringsfunksjon for rommet i figur 4.12.



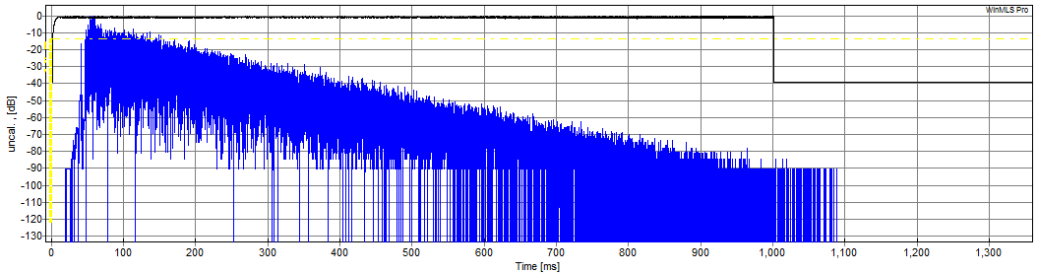
Figur 4.14: Vokselisering av L-formet rom



Figur 4.15: Impulsrespons mellom punktene (4.00, 1.25, 1.50) og (1.25, 4.00, 1.50) i L-formet rom. Resultat fra Odeon i blått, og resultat fra FDTD i svart. Frekvensområde: 20 Hz til 249 Hz (rommets omtrentlige Schroeder-frekvens); logaritmisk. Lydnivåområde: 0dB til -50dB, med gitterlinjer hver 5. dB.



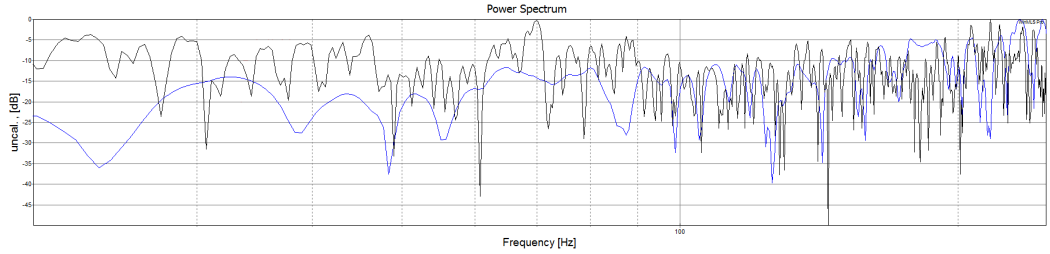
Figur 4.16: WinMLS' tolkning (absoluttverdi) av impulsrespons-tidsdata fra Odeon-simulering (i blått) og FDTD-simulering (i svart). Begge er avtagende med tid som man forventer da veggene er absorberende.



Figur 4.17: WinMLS' trolig feilaktige tolkning av tidsdata for Odeon- og FDTD-målinger (i hhv. blått og svart). Dette skjer kun dersom Odeon-resultatet lastes først.

Bug i WinMLS Resultatet i figur (4.15) er trolig basert på en riktig tolkning av tidsdataene fra Odeon og FDTD, fordi WinMLS' tolkning av de to impulsresponsene likner hverandre, i vinduet som viser tidsdataene (se figur (4.16)): begge er avtagende.

Derimot, dersom man laster resultatet fra Odeon-simuleringen (fra .wav-fil) *først*, og deretter laster resultatet fra FDTD-simuleringen (fra .txt-fil), blir WinMLS' tolkning av tidsdataene som man kan se i figur (4.17), og impulsresponsene som WinMLS kalkulerer er da som vist i figur (4.18). I tillegg blir her frekvensresponsplottet hetende Power Spectrum i stedet for Frequency Response Magnitude som i figur (2.2), selv om det tilsynelatende er samme beregningsmetode som ligger bak, i og med at frekvensresponsen for Odeon-beregningen jo ser nøyaktig lik ut som i figur (2.2).



Figur 4.18: (Feilaktig for FDTD) impulsrespons mellom punktene (4.00, 1.25, 1.50) og (1.25, 4.00, 1.50). Resultat fra Odeon i blått, og resultat fra FDTD i svart. Dette er resultatet produsert av WinMLS dersom Odeon-resultatet lastes først. Sammenlikn med figur 4.15.

4.6 Tidsbruk

La oss se på noen eksempler på anslått tidsbruk T_{sim} ved beregning av impulsrespons med FDTD-metoden. I det følgende vil beregningsmodellen først forklares (med tall for et eksempeltilfelle i parentes), før tall for et utvalg representative tilfeller presenteres i en tabell. Gitt et kasseformet rom med gitte sidelengder L, M og N (for eksempel 5 m, 5 m og 3 m), bestående av indre flater S_i (sum: S (110 m^3)) med gitte absorpsjonskoeffisienter α_i (for eksempel 0.3; alle like). Vi antar at vi får en tilstrekkelig god approksimasjon av impulsresponsen ved å la simuleringen vare så lenge som rommets etterklangstid, T_{60} (0.37 s). Vi beregner denne med Sabines formel:

$$T_{60} = \frac{4 \cdot ln(10^6)}{c} \frac{V}{A} \approx 0.161 \frac{V}{A},$$

hvor ln er den naturlige logaritmen, c er lydhastigheten, V er rommets volum og A er rommets ekvivalente absorpsjonsareal, $A = \sum S_i \cdot \alpha_i$ (33 m^3). Egentlig gjelder T_{60} kun for frekvenser hvor lydfeltet er diffust, så denne beregningstiden kan tenkes å måtte utvides i en mer presis beregningsmodell.

For å finne ut hvor lang tid denne beregningen vil ta, er vi nødt til å finne ut hvor raskt rommet kan simuleres. Dette vil avhenge av problemstørrelsen (antall celler), og maskinens celleberegningshastighet.

Vi antar (fra seksjon 2.1.1) at problematiske moder begrenser seg til under rommets Schroeder-frekvens F_s (140 Hz), og beregner denne med Schroeders formel

L	M	N	α	S	A	F_s	T_{60}	T_{sim}
5	5	3	0.2	110	22	171	0.55	0.49
5	5	3	0.3	110	33	140	0.37	0.15
5	5	3	0.4	110	44	121	0.27	0.06
6	7	2.8	0.2	157	31	143	0.6	0.42
6	7	2.8	0.3	157	47	117	0.4	0.12
6	7	2.8	0.4	157	63	101	0.3	0.05

Tabell 4.1: Simuleringstider for rom av ulike kombinasjoner av dimensjoner og ekvivalente absorpsjonsareal.

$$F_s = 2000 \cdot \sqrt{\frac{T_{60}}{V}}.$$

(Her er T_{60} etterklangstiden som vi allerede har beregnet, mens V (75 m^3) er rommets volum $V = L \cdot M \cdot N$.) Den korteste bølgelengden av interesse er dermed $\frac{c}{F_s}$. Anta at 6 celler er tilstrekkelig for å approksimere bølgen nøyaktig nok. Da må cellenes sidelengde være $\Delta x = \frac{c}{6F_s}$ (0.41 m). Antall celler C i rommet blir dermed $C = \frac{V}{\Delta x^3}$ (1081). CourantFriedrichsLewy setter krav til akseptable verdier for simuleringens tidssteg, som tilfredsstilles av $\Delta t = \frac{\Delta x}{2c}$ [12]. Med andre ord kreves $\frac{1}{\Delta t} = \frac{\Delta x}{2c}$ (1675) iterasjoner per simulerte sekund. På MacBook Pro brukes algoritmen ca. 220 ns per celle, og altså $T_{\text{rom}} = 220\,\text{ns} \cdot C$ (238 s) per iterasjon av rommet. Tid som kreves for å simulere ett sekund blir simtime ratio = $F_s * T_{\text{rom}}(0.40)$, og tid som kreves for å simulere i T_{60} blir $T_{sim} = \text{simtime ratio} \cdot T_{60}$ (0.15 s).

Dette vil oppleves som nesten instantant av en bruker. Tabell 4.1 viser simuleringstider for noen andre kombinasjoner av romdimensjoner og gjennomsnittlig absorpsjonskoeffisient.

Et regneark ble produsert som utfører utregningene ovenfor (se A.5).

Disse resultatene beviser påstanden om hurtighet presentert innledningsvis i rapporten, i seksjon 1.3.

Tidsbruk i FDFD-metoden Det har ikke vært mulig å anslå tidsbruk for FD**F**D-metoden i og med at ingen algoritme foreligger.

Kapittel 5

Diskusjon

Dette kapittelet tar et overblikk over masterprosjektet, og setter arbeidet gjort i denne oppgaven i et større perspektiv.

Rapportens introduksjonen belyser aktuelle utfordringer innen akustikk og spesielt studiodesign. Med betraktningene i Kapittel 2 Teori, har vi kartlagt finite-difference-metodenes anvendelighet på utfordringene innen småroms- og lavfrekvent-akustikk. Som omtalt i 2.4.2 omhandler en overvekt av den eksisterende vitenskapelige litteraturen om simuleringsmetoder, applikasjoner i elektromagnetisme. Gjennomgangen i denne oppgaven kan derfor være nyttig for andre akustikk-studenter i fremtiden. Det kunne også være en idé å lage et instruksjonsskriv som presist beskriver hvordan de ulike elektromagnetiske feltene og likningene skal tolkes ved oversetting til akustikk. Fagmiljøet og spesielt studenter vil trolig kunne dra nytte av en slik ressurs.

Utviklingen av programmeringsverktøy spesielt for GPGPU har sammen med GPU-ers beregningskapasitet og evne vokst fenomenalt den senere tid, og anvendelsene likeså[1]. Om et drøye års tid vil vi antakeligvis få oppleve at GPGPU er *standard* og *forventet* måte å gjøre en stor klasse av beregninger på. Utviklingen på GPGPU-området skjer svært raskt, også for akustikk. Som nevnt under innledningen skjer det mye i akademia i dette skjæringspunktet. Overraskende og kanskje uheldigvis: den raske utviklingen skaper en fare for at arbeidet i denne masteroppgaven og liknende arbeid utført av kun én person, nesten vil være utdatert i løpet av den tiden det tar å utvikle. Dersom Saka skal

[1] Cuda Showcase lister hele 1300 imponerende GPGPU-prosjekter av varierende profesjonalitet (sist lastet 12. februar 2012).

utvikles videre og publiseres eller selges, er det altså nødt til å skjer innen rimelig tid.

Framtiden innen tungregning er GPGPU. Men framtiden i datamaskiner er også utvilsomt Web – at programmer lastet som nettsider i nettlesere, og ikke Windows Forms og liknende systemspesifikke vindussystemer. God framsynthet i prosjekter av denne typen bør derfor satse på å aksessere GPU-datakraft via web-applikasjoner. Nettlesere er ikke klare for dette ennå, men vil bli det i nær framtid, med adopsjon av HTML5-standarden.

Som nevnt i 4.4 ble det planlagt å bygge og installere bassabsorbenter mellom første og andre runde av målinger i Fagerengs rom. Dessverre ble absorbentene aldri bygget, og andre runde av målinger i Fagerengs rom er derfor bare endret i mellomtonefrekvenser, siden enkelte skumgummiabsorbenter ble kjøpt og installert.

I oppgaveformuleringen er det spesifisert at man ønsker å se modeformene i et rom, og det er for å fylle dette målet at FDFD-metoden trengs. Men, det kan tenkes at det ikke er nødvendig å se modeformene, når man har tilgang til en raskt oppdatert overføringsfunksjon. En modeform identifiserer problemer ved romlig fordeling av utsving, men gjelder likevel bare den enkelte frekvens man betrakter i ethvert tilfelle, mens en overføringsfunksjon synliggjør hele frekvensområdet av interesse, med begrensningen i stedet knyttet til høyttaler- og lytte-posisjon. I denne oppgaven ble akustikeres ønsker om å se modeformene antatt og lagt til grunn (selv om problemet ikke ble fullstendig løst), men en alternativ tilnærming ville vært å stille spørsmål ved dette behovet.

En viktig lærdom fra dette prosjektet er at ved valg av programvare, -bibliotek o.l., bør dens modenhet og brukerbase (antall brukere) tas meget alvorlig. Unge, umodne programmer, nisjeprogrammer og liknende bør i de fleste tilfeller unngås, om et prosjekt kan tolerere kun en begrenset mengde uforutsette programvareproblemer.

Kilderepresentasjon i FDTD-metoden har ikke blitt diskutert tidligere i oppgaven men blitt tatt for gitt. Sannheten er at en “hard kilde” i ett punkt, som her implementert, ikke overraskende skaper imperfeksjon, nærmere bestemt lav-frekvens-artefakter. I en fersk artikkel [23] presenterer Jeong en ny kildetype for akustisk FDTD som er påstått å omgå tidligere kildemodellers problemer.

Programmering og programvareutvikling har vært en stor del av denne oppgaven, men GPU-programmering var opprinnelig ikke tenkt som et av hovedpunktene. Programmer av denne typen (høyparallelliserbare) kan gå svært mye raskere

dersom beregningsintensive deler blir implementert i GPU-kode, men på tross av dette vil det ikke nødvendigvis alltid være en fordel å velge GPGPU. Fagfeltet er nemlig fortsatt relativt nytt, slik at verktøyene og programmeringsspråkene som må brukes, ofte har såkalte "barnesykdommer" og "bugs", siden de er unge og under stadig utvikling. (Dette gjelder i høy grad både CUDA og Cudafy.) Om man likevel velger GPGPU, må man høyde for at utvikling av denne programvaren er ekstra kompetanse- og tidkrevende, og kan by på overraskende og uforutsette utfordringer.

5.1 Konklusjon

Denne oppgaven har tatt for seg problemet med lav-frekvente resonanser i små rom, spesifikt studioer og mikserom. (Sammenlikn med målene i 1.4.)

- Motivasjonen bak oppgaveformuleringen har blitt undersøkt, og oppgaven har blitt funnet å representere en reell utfordring. Problemet er tidsaktuelt og interessant, fordi kraftige verktøy i fysisk-matematisk analyse, og i datamaskiners evne, tillater tilnærmingsmetoder for løsning av problemet som tidligere i historien har vært utenkelige, umulige eller upraktiske.
- Teorien bak problemet har blitt gjennomgått: Fra en hverdagslig og intuitiv forståelse av mode-/resonansproblemer har vi beveget oss mot en teoretisk forståelse av hva resonans er, hvorfor og når det oppstår, og hva vi kan gjøre med det. Videre har programmeringsrelaterte emner (numeriske modeller; beregningskompleksitet; bruk av biblioteket Cudafy m.m.) blitt forklart teoretisk eller gjennom intuitive og erfaringsmessige betraktninger. GPU-programmeringsdetaljer og implementasjonsdetaljer for FDTD-metoden har blitt utelatt, da disse gjennomgås i forfatterens prosjektoppgave [12, 12]. Behoved for og teorien bak FDFD-metoden har blitt gjennomgått, men en implementasjon av metoden mangler.
- En tidlig versjon av simuleringsprogrammet som skisseres, har blitt implementert (“Saka”). Implementasjonen gjør bruk av GPGPU-akselerasjon. Programmet har løst flere viktige oppgaver (import og tilordning av absorpsjonsdata, import av modell av rom, visualisering av rom, plassering av kilder og mikrofoner, og beregning av impulsresponser), mens en av hovedmålene med programmet i følge oppgaveformuleringen – beregning og visualisering av modeformer – ikke har blitt løst.
- Selv om et eksisterende rom har blitt grundig modellert og grundig akustisk målt, har det ikke lyktes å sammenlikne målingene med simuleringer av

rommet. Dette skyldes 1) manglende implementasjon av *impedans*-støtte i FDTD-metoden, 2) manglende kunnskap om veggene i rommet, og 3) manglende mulighet for utregning av frekvensrespons (FFT) grunnet feil og mangler ved valgte beregningsbibliotek (Cudafy). I stedet ble simulering i Saka av et L-formet rom med 10% absorberende vegger sammenliknet med en Odeon-simulering av det samme rommet. Få konklusjoner kan trekkes av denne sammenlikningen blant annet fordi den ikke har noen fasit / referanse.

- Det utviklede programmet, designspesifikasjonene, målingene og modellene danner et godt grunnlag for videreutvikling av dette prosjektet og programmet Saka. Akustikkavdelingen ved Multiconsult, som var engasjerende instans i denne masteroppgaven, har uttrykt at de er interessert i å anskaffe programmet når det blir klart til bruk.

5.2 Framtidig arbeid

Hill et. al. presenterer i [29] en metode kalt "Chameleon Subwoofer Array", hvor 2-4 subwoofere kombineres i et array for å få en ønsket direktivitet, og hvor et flertall slike arrayer plasseres rundt i et rom. Videre beskrives hvordan forskjellig equalizering for hver høyttaler ("Multiple point equalization") kan benyttes for å i kombinasjon oppnå en overlegent flat og god frekvensrespons i lav-frekvensområdet. Programmet Saka kunne benytte metoden beskrevet i artikkelen, som gjør stor bruk av FDTD, og slik ta adaptiv equalizering med i betraktning når programmet foreslår plasseringer for basshøyttalerene. Om nevnte høyttaler-*arrayer* også skal støttes, må kilde*direktivitet* inkluderes i simuleringsmodellen.

Vokselisering: I 2.7.2 ble kvantiseringsfeil ved plassering av voksler i heltallskoordinater antatt å være ubetydelig for simuleringsresultat, og ble sett bort i fra. En strengere behandling av dette punktet krever en undersøkelse av hvor stor feil som introduseres som et resultat av avrundingene. En analytisk betraktning bør kunne fastslå en øvre grense for denne feilens utslag i beregningsresultater. Om feilen viser seg å være av vesentlig betydning, bør vokseliseringsprogrammet produsere et mål på total avrundingsfeil begått samt forsøke å minimere denne.

Impedans ikke dekkende: I midten av Fagerengs rom er det en hul skillevegg i løse lufta (se figur 4.9). Veggen modelleres muligens ikke tilstrekkelig godt av "impedanser på hver side", siden lavfrekvente lyder kan tenkes å gå igjennom veggen til en viss grad. Dersom akustikken på begge sider av slike vegger er interessant i et stort nok antall reelle tilfeller som skal simuleres med Saka, og

dersom modellen "impedanser på hver side" introduserer uakseptabelt store avvik, må metoder for å ta i betraktning forplantningen av lyd inni veggen undersøkes og eventuelt implementeres.

Cudafy har siden versjon 1.6 støtte for "Bi-Conjugate Gradient"-metoden (BiCG) for løsing av glisne matriser (sparse matrices). En FD**F**D-implementasjon består av å løse glisne matriser og kan således ta i bruk denne BiCG-funksjonaliteten.

Impedans kontra absorpsjon: Hvor mye har presis modellering av frekvensavhengig impedans å si for lavfrekvent-delen av overføringsfunksjoner fra simuleringer, sammenliknet med kun reellverdi-absorpsjon? Hill påstår i [25, 25] at impedans er av liten betydning og kan ignoreres, og det samme gjør LFTool og RoomCalc, mens J. H. Rindel påstår det motsatte. Denne rapporten undersøker ikke hvordan kompleks impedans påvirker overføringsfunksjonen, men dette bør kartlegges. Det vil være viktig å få klarhet på dette punktet.

5.3 Etterord

Denne seksjonen lister noen ting som skulle vært gjort annerledes i arbeidet med prosjektet, og som kan læres av ved å gjøres bedre en annen gang:

Jeg skulle satt meg mer inn i teori om bølgeanalyse og diverse simuleringsmetoder, på et tidlig stadium. Wikipedia og andre websider ble mye brukt som ressurs i begynnelsen, men etter hvert ble det oppdaget at bøker dedikert til behandling av akustikk er både mer presise, mer pedagogiske og en utmerket samling av nødvendig informasjon. Et lite utvalg av relevante bøker er langt overlegent selv gode søk på internett. En annen gang vil jeg forsøke å "se meg mer rundt" og planlegge detaljerte analytiske oppgaver i større grad på forhånd, før jeg fordyper meg i disse analytiske problemene.

Jeg skulle ha vurdert nøye hva å implementere før programmeringsarbeid ble satt i gang. Selv om det er sant at programmering handler mye om å prøve seg fram for å finne ut om ting lar seg gjøre enkelt og greit eller ikke, ville dette prosjektet ha gagnet av mer planlegging som oppsats til programmering. Muligens ville endel andre valg ha blitt tatt på tidligere stadium. Dette illustreres av implementasjonen av metoden i Rindels artikkel, som gikk veldig fort og greit, siden alt allerede var nøye beskrevet i artikkelen. Når jeg programmerer i fremtiden vil jeg forsøke så langt det er mulig å planlegge og få overblikk, og spesielt stille meg selv spørsmålet om hvilket mål programmeringen fyller, og hvilken grad av viktighet dette målet har.

Jeg skulle ha gått i gang med testing umiddelbart. Testing er blitt svært populært i programmering i senere år, og ikke uten grunn. Ved nøye uttenkte tester kan korrekt funksjon av et programs byggestener testes programmatisk, helt automatisk. Man vil få beskjed fra en test om en funksjon ikke lenger har den samme funksjonalitet, og test av funksjoner med eksempeltilfeller er en svært effektiv måte å finne feil på, sammenliknet med stegvis debugging.

Utfordringen med å utvikle brukbar simuleringsprogramvare ble til en viss grad undervurdert. En annen gang bør det undersøkes om man heller bør ta utgangspunkt i eksisterende simuleringsprogramvare, og eventuelt lisensiere deler av denne.

Jeg skulle ha gjort en risikoanalyse, det vil si en vurdering av sannsynligheten for og alvorligheten ved at enkelte ting ikke gikk som planlagt, som bygging av bassabsorbenter.

Fremfor alt, flere møter skulle blitt arrangert. Jeg skulle tatt enda mer kontakt med veiledere og kolleger, titt og ofte. Det var i samtale med andre at de største problemene ble oppdaget og at de største innsiktene ble oppnådd.

Referanselitteratur

[1] P. Newell, *Recording studio design*, 1st ed. Focal Press, 2003. 1.1.1, 3, 4, 1.1.4, 2

[2] A. Weisser and J. Rindel, "Evaluation of sound quality, boominess, and boxiness in small rooms," *JOURNAL-AUDIO ENGINEERING SOCIETY*, vol. 54, no. 6, p. 495, 2006. 1.1.6, 2.1.1

[3] U. Svensson, R. Fred, and J. Vanderkooy, "An analytic secondary source model of edge diffraction impulse responses," *The Journal of the Acoustical Society of America*, vol. 106, p. 2331, 1999. 1.1.6

[4] U. Svensson and A. Asheim, "Edge-diffraction based integral equation for the scattering from convex rigid polyhedra," Aalborg, Denmark, Jun. 2011. 1.1.6

[5] M. Lachmann and R. Pieren, "Simulationswerkzeug fr tieffrequente schallfelder in rumen – entwicklung, test und anwendung," Aug. 2011. [Online]. Available: (takontakt) 1.1.6, 1.2, 1.3

[6] PAFEC, "PAFEC RoomCalc," Strelley Hall, Nottingham NG8 6PE, United Kingdom, Apr. 2011, roomcalc@pafec.info. [Online]. Available: www.pafecfe.com 1.2

[7] N. Tsingos, "Using programmable graphics hardware for acoustics and audio rendering," in *127th Audio Engineering Society Convention*, 2009. 1.3

[8] N. Raghuvanshi, B. Lloyd, N. Govindaraju, and M. Lin, "Efficient numerical acoustic simulation on graphics processors using adaptive rectangular decomposition," in *Proc. EAA Symp. on Auralization*, 2009, p. 1517. 1.3

[9] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *13th Int. Conf on Digital Audio Effects*, 2010. 1.3

[10] L. Savioja, D. Manocha, and M. Lin, "Use of GPUs in room acoustic modeling and auralization," in *Proc. Int. Symp. on Room Acoustics*, 2010. 1.3

[11] L. Savioja, "Audio signal processing using graphics processing units," *Audio Engineering Society*, vol. 59, no. 1/2, pp. 3–19, 2011. 1.3

[12] O. B. Vikholt, "GPU-accelerated FDTD modelling and visualization of horn loudspeaker acoustics," NTNU, Trondheim, Norway, Master project, Dec. 2010. 2.1, 2.4.1, 4.6, 5.1

[13] M. Skålevik, "Small room AcousticsThe hard case," 2011. 2.1.1

[14] H. Kuttruff, *Room acoustics*, 4th ed. London: Spon Press, 2000. 2.1.2, 2.1.4

[15] T. Vigran, *Bygningsakustikk: et grunnlag*. Tapir Akademisk Forlag, 2002. 2.1.2, 4.3.6, 4.3.8

[16] M. Skålevik, "Schroeder frequency revisited," in *Forum Acusticum*, 2011. 2.1.2

[17] M. Crocker, "Handbook of acoustics," *A Wiley-Interscience Publication*, 1998. 6

[18] E. Habets, "Room impulse response generator," *Technische Universiteit Eindhoven, Tech. Rep*, 2006. 2.3, 2.4.1, 2.4.2, 2.6

[19] S. Zainud-Deen, E. Hassan, M. Ibrahim, K. Awadalla, and A. Botros, "Electromagnetic scattering using gpu-based finite difference frequency domain method," *Progress In Electromagnetics Research*, vol. 16, p. 351369, 2009. 2.3

[20] S. H. Zainud-Deen, "Graphical processing units (GPU) acceleration of finite-difference frequency-domain (FDFD) technique," *2009 National Radio Science Conference*, p. 1, Mar. 2009. [Online]. Available: http://resolver.scholarsportal.info/resolve/11106980/v2009inone/1_gpuaofft.xml 2.3

[21] P. Naylor, *Speech dereverberation*. Springer Verlag, 2010. 2.4.2

[22] L. Beranek, L. Beranek, and L. Beranek, *Acoustics*. McGraw-Hill New York, 1954. 2.4.2

[23] H. Jeong and Y. W. Lam, "Source implementation to eliminate low-frequency artifacts in finite difference time domain room acoustic simulation," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, p. 258268, 2012. [Online]. Available: http://asadl.org/jasa/resource/1/jasman/v131/i1/p258_s1 2.4.2, 5

[24] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, Jan. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5440917 2.6

[25] A. Hill and M. Hawksford, "Visualization and analysis tools for low frequency propagation in a generalized 3D acoustic space," in *Audio Engineering Society Convention*, vol. 127, 2009. 4.2.3, 5.2

[26] S. Sakamoto and H. Nagatomo, "Calculation of impulse responses and acoustic parameters in a hall by the nite-difference time-domain method," *Acoustical Science and Technology*, vol. 29, no. 4, pp. 256–265, 2008. 4.2.3

[27] "Rindel 2011 - an impedance model for estimating the complex pressure reflection factor.pdf," 2011. 4.3.7, 4.3.8

[28] J. A. Fletcher, "The design of a modular sound absorber for very low frequencies," *NASA STI/Recon Technical Report N*, vol. 93, p. 25170, 1992. [Online]. Available: http://adsabs.harvard.edu/abs/1992STIN...9325170F 4.4

[29] A. Hill and M. Hawksford, "Chameleon subwoofer arrays - generalized theory of vectored sources in a closed acoustic space," in *Journal of the Audio Engineering Society*, vol. 128th Convention, London, May 2010, p. 23. 5.2

Tillegg A

Vedlegg

A.1 Kildekode for "search while typing"-funksjonalitet

(Seksjon 4.3.9 omtaler følgende kildekode.)

```
106         /// <summary>
107         /// Refilters the contents of the data grid view
108         /// to reflect the search keywords
109         /// </summary>
110         private void TextBoxSearchKeyUp(object sender, KeyEventArgs e)
111         {
112             if (_myView == null) return;
113 
114             string outputInfo = string.Empty;
115             string[] keyWords = textBoxSearch.Text.Split(' ');
116 
117             foreach (string word in keyWords)
118             {
119                 if (word.IndexOfAny(new[] {'\'', '"'}) > -1)
120                     continue;
121                 if (outputInfo.Length != 0)
122                     outputInfo += " AND ";
123                 outputInfo += "(";
124 
125                 outputInfo += string.Format("Description LIKE '%{0}%'", word);
126 
127                 int wordAsInt;
128                 if (int.TryParse(word, out wordAsInt))
129                     outputInfo += string.Format(" OR ID = '{0}'", wordAsInt);
130 
131                 outputInfo += ")";
132             }
133 
134             //Applies the filter to the DataView
135             _myView.RowFilter = outputInfo;
136 
137             labelResultCount.Text = _myView.Count.ToString();
138         }
```

A.2 Kildekode for parsing av absorbent-database i .LI8-fil

(Underseksjon 4.3.6 omtaler følgende kildekode.)

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.RegularExpressions;

namespace OlaCudafy1
{
    internal class AbsorberDatabase
    {
        private const int NumberOfCoefficients = 8;

        public AbsorberDatabase(string path)
        {
            Database = new List<AbsorberDatabaseEntry>();

            // Read in every line in the file.
            using (var reader = new StreamReader(path))
            {
                string line;
                var readState = new ReadState();
                const string headerPattern = @"^\s*([0-9]+)\s(.*)\s*$";
                const string separatorPattern = @"^\s*$";

                int materialNumber = 0;
                string materialDescription = null;

                while ((line = reader.ReadLine()) != null)
                {
                    Match match;
                    switch (readState.WhichLine)
                    {
                        case EWhichLine.Default:
                            match = Regex.Match(line, headerPattern, RegexOptions.IgnoreCase);

                            if (match.Success)
                            {
                                if (!int.TryParse(match.Groups[1].Value, out materialNumber))
                                    readState.EntryFailed_NeedsReset = true;
                                materialDescription = match.Groups[2].Value;

                                readState.Increment();
                            }
                            else
                                readState.Jammed();
                            break;

                        case EWhichLine.DescriptionLineRead: // read data:
                            var coeffs = new float[NumberOfCoefficients];
                            bool allParseOK = true;
                            int i = 0;
                            foreach (string s in line.Split(new[] {'\t', ' '},
                                StringSplitOptions.RemoveEmptyEntries))
                            {
                                float result;
                                if (!float.TryParse(s.Trim(), out result))
                                {
                                    allParseOK = false;
                                    break;
                                }
                                coeffs[i] = result;
                                ++i;
                                if (i >= coeffs.Length)
                                    break;
                            }
```

```
64                              }
65                              if (allParseOK && !readState.EntryFailed_NeedsReset)
66                                  Database.Add(new AbsorberDatabaseEntry(
67                                                  materialNumber, materialDescription, coeffs));
68                              else
69                  // could not correctly interpret this material specification entry found in file
70                                  readState.Jammed();
71                              readState.Increment();
72                              break;
73                          default: //case EWhichLine.DataLineRead:
74                              match = Regex.Match(line, separatorPattern);
75                              if (match.Success)
76                                  readState.Reset();
77                              break;
78                      }
79                  }
80              }
81          }
82
83          internal List<AbsorberDatabaseEntry> Database { get; set; }
84
85          #region Nested type: EWhichLine
86
87          private enum EWhichLine
88          {
89              Default,
90              DescriptionLineRead,
91              WaitingForBlankLine
92          }
93
94          #endregion
95
96          #region Nested type: ReadState
97
98          private class ReadState
99          {
100             public ReadState()
101             {
102                 Reset();
103             }
104
105             internal bool EntryFailed_NeedsReset { get; set; }
106             internal EWhichLine WhichLine { get; private set; }
107
108             public void Jammed()
109             {
110                 WhichLine = EWhichLine.WaitingForBlankLine;
111             }
112
113             internal void Reset()
114             {
115                 EntryFailed_NeedsReset = false;
116                 WhichLine = EWhichLine.Default;
117             }
118
119             internal void Increment()
120             {
121                 if (WhichLine == EWhichLine.Default)
122                     WhichLine = EWhichLine.DescriptionLineRead;
123                 else if (WhichLine == EWhichLine.DescriptionLineRead)
124                     WhichLine = EWhichLine.WaitingForBlankLine;
125             }
126         }
127
128         #endregion
129     }
130 }
```

A.3 Kildekode for Rindels metode

(Seksjon 4.3.7 omtaler følgende kildekode.)

```
using System;
using System.ComponentModel;
using Cudafy;

namespace OlaCudafy1
{
    /// <summary>
    /// The RindelImpedanceEstimation class
    /// provides logic for estimating the absorber type and
    /// the three parameters of a simple material impedance model
    /// of the absorber, given the absorption coefficients of
    /// the material at 125, 250, 500 and 1000 Hz.
    ///
    /// This can be used in for example the FDTD simulation method
    /// to model the impedance of a material for which only
    /// the absorption coefficients are known. The absorption
    /// coefficients for many materials are easily available
    /// through large databases.
    ///
    /// This implementation is based on the method described
    /// by J. H. Rindel in 'An impedance model for estimating
    ///  the complex pressure reflection factor', presented at
    /// Forum Acusticum 2011 in Aalborg, Denmark.
    /// </summary>
    internal class RindelImpedanceEstimation
    {
        #region AbsorberType enum

        public enum AbsorberType
        {
            Porous, // section 3.4 in the article
            Membrane, // section 3.5 in the article
            Resonator, // section 3.6 in the article
        }

        #endregion

        private readonly AbsorberType _absorberType;
        private readonly float[] _absorptionCoefficients;
        private readonly float _acousticImpedance;
        private readonly BandCoefficients _bandCoefficients;
        private readonly FrequencyBand _whatBandToUse;
        private BandCoefficients _imaginaryTerms;

        /// <summary>
        /// </summary>
        /// <param name="absorptionCoefficients">
        /// The statistic absorption coefficients alpha_stat of the material.</param>
        /// <param name="acousticImpedance">
        /// The acoustic impedance to use in the calculations.</param>
        internal RindelImpedanceEstimation(float[] absorptionCoefficients, float acousticImpedance)
        {
            _absorptionCoefficients = absorptionCoefficients;
            _absorberType = GuessAbsorberType(_absorptionCoefficients);
            _bandCoefficients = new BandCoefficients(_absorptionCoefficients);
```

```
56              _acousticImpedance = acousticImpedance;
57              if (_absorberType == AbsorberType.Resonator)
58                  _whatBandToUse = _bandCoefficients.BandOfMaximum;
59              else
60                  _whatBandToUse = WhatBandToUse(_absorberType);
61              EstimatedResistanceTerm = EstimateResistanceTerm();
62              EstimateImaginaryTerms();
63              EstimatedImaginaryTermsArray = _imaginaryTerms.ToArray();
64          }
65
66          internal float EstimatedResistanceTerm { get; private set; }
67          internal float[] EstimatedImaginaryTermsArray { get; private set; }
68
69          private float FrequencyBandFrequencyValue(FrequencyBand band)
70          {
71              switch (band)
72              {
73                  case FrequencyBand.Band125Hz:
74                      return 125;
75                  case FrequencyBand.Band250Hz:
76                      return 250;
77                  case FrequencyBand.Band500Hz:
78                      return 500;
79                  case FrequencyBand.Band1000Hz:
80                      return 1000;
81                  default:
82                      throw new InvalidEnumArgumentException();
83              }
84          }
85
86          /// <summary>
87          /// Guesses the absorber type according to the method described by
88          /// J. H. Rindel, on the basis of the supplied absorption coefficients.
89          /// The supplied float values should be the absorption coefficient
90          /// (alpha_stat) values for the four frequency bands
91          /// 125, 250, 500 and 1000 Hz, upon which the absorber type will be
92          /// guessed.
93          /// </summary>
94          /// <returns></returns>
95          private static AbsorberType GuessAbsorberType(
96              BandCoefficients bandCoefficients)
97          {
98              float alpha0125HzBand = bandCoefficients.Alpha125HzBand;
99              float alpha0250HzBand = bandCoefficients.Alpha250HzBand;
100             float alpha0500HzBand = bandCoefficients.Alpha500HzBand;
101             float alpha1000HzBand = bandCoefficients.Alpha1000HzBand;
102             if (alpha0125HzBand <= alpha0250HzBand &&
103                 alpha0250HzBand <= alpha0500HzBand &&
104                 alpha0500HzBand <= alpha1000HzBand) // monotonically increasing absorption
105                 return AbsorberType.Porous;
106             if (alpha0125HzBand >= alpha0250HzBand &&
107                 alpha0250HzBand >= alpha0500HzBand) // monotonically decreasing
108                 return AbsorberType.Membrane;
109             return AbsorberType.Resonator; // otherwise
110         }
```

```
111
112         /// <summary>
113         /// Guesses the absorber type according to the method described by
114         /// J. H. Rindel, on the basis of the supplied absorption coefficients.
115         /// The supplied float array should be of length 5 but may be longer,
116         /// and will be interpreted as follows:
117         ///   1st element: 63 Hz: ignored;
118         ///   2nd-5th element: 125, 250, 500 and 1000 Hz: used for estimation;
119         ///   any susequent elements if present: ignored;
120         /// </summary>
121         /// <param name="absorptionCoefficients"></param>
122         /// <returns></returns>
123         public static AbsorberType GuessAbsorberType(float[] absorptionCoefficients)
124         {
125             var bandCoefficients = new BandCoefficients(absorptionCoefficients);
126             return GuessAbsorberType(bandCoefficients);
127         }
128
129         /// <summary>
130         /// Estimates the imaginary term X(f) for each of the bands (f=)
131         /// 125, 250, 500 and 1000 Hz and places them in a float array looking like this:
132         /// [0, (125Hz-term), (250Hz-term), (500Hz-term), (1000Hz-term)].
133         /// Estimation is done for an absorber having the supplied coefficients.
134         /// Estimation is done according Rindel's method, section 3.2, equation (9).
135         ///
136         /// X(f) can be positive or negative, but only the positive values are returned.
137         /// The other values can be found by taking -X(f)
138         /// </summary>
139         /// <returns></returns>
140         private void EstimateImaginaryTerms()
141         {
142             float R = EstimatedResistanceTerm;
143             _imaginaryTerms = new BandCoefficients();
144             foreach (FrequencyBand band in Enum.GetValues(typeof (FrequencyBand)))
145             {
146                 float zfStar = ZfStar.Get(band, _acousticImpedance);
147                 float nominator = 4f*R*zfStar;
148                 float denom = _bandCoefficients[band];
149                 float term = (R + zfStar);
150                 var x = (float) Math.Sqrt(nominator/denom - term*term);
151                 _imaginaryTerms[band] = x;
152             }
153             // TODO
154             // In addition, imaginaryTerms[0] should contain data for the 63Hz band,
155             // (and possibly imaginaryTerms[5] for the 2000Hz band)
156             // estimated by using Rindel's equation (3) in section 2.1, using:
157             // omega=2*PI*63Hz, and with 'd' and 'm' as first estimated according to the
158             // scattered instructions in the article which differ depending on absorber type.
159         }
160
161         /// <summary>
162         /// Estimates 'd' according to equation (13)
163         /// </summary>
164         /// <returns></returns>
165         internal float EstimateD(float speedOfSound)
```

```
166         {
167             switch (_absorberType)
168             {
169                 case AbsorberType.Porous:
170                     float nom = _acousticImpedance*speedOfSound;
171                     float freq = FrequencyBandFrequencyValue(FrequencyBand.Band125Hz);
172                     float den = freq*2f*
173                                 (float) Math.PI*Math.Abs(_imaginaryTerms[FrequencyBand.Band125Hz]);
174 
175                     float nom250 = _acousticImpedance*speedOfSound;
176                     float freq250 = FrequencyBandFrequencyValue(FrequencyBand.Band250Hz);
177                     float den250 = freq250*2f*
178                                    (float) Math.PI*Math.Abs(_imaginaryTerms[FrequencyBand.Band250Hz]);
179 
180                     return (nom/den + nom250/den250)*.5f; // average
181                 case AbsorberType.Membrane:
182                     return float.PositiveInfinity;
183                 case AbsorberType.Resonator: // equation (16)
184                     //// TODO consolidate with EstimateM
185                     float nomres = _acousticImpedance* speedOfSound;
186                      // dupe from estimate m -> res
187                     float omega1 = GMath.PI*2f*FrequencyBandFrequencyValue(FrequencyBand.Band125Hz);
188                      // dupe from estimate m -> res
189                     float nom1 = Math.Abs(_imaginaryTerms[FrequencyBand.Band125Hz])*omega1;
190                     float denres = omega1*omega1*EstimateM() + nom1;
191                     return nomres/denres;
192                 default:
193                     throw new NotImplementedException();
194             }
195         }
196 
197         internal float EstimateM()
198         {
199             switch (_absorberType)
200             {
201                 case AbsorberType.Porous:
202                     return 0f;
203 
204                 case AbsorberType.Membrane:
205                     float omega500 = 2f*GMath.PI*FrequencyBandFrequencyValue(FrequencyBand.Band500Hz);
206                     float nom500 = Math.Abs(_imaginaryTerms[FrequencyBand.Band500Hz]);
207                     float den500 = omega500;
208 
209                     float omega250 = 2f*GMath.PI*FrequencyBandFrequencyValue(FrequencyBand.Band250Hz);
210                     float nom250 = Math.Abs(_imaginaryTerms[FrequencyBand.Band250Hz]);
211                     float den250 = omega250;
212 
213                     return (nom500/den500 + nom250/den250)*.5f; // average
214 
215                 case AbsorberType.Resonator: // refer to equation (15)
216                     float omega1 = GMath.PI*2f*FrequencyBandFrequencyValue(FrequencyBand.Band125Hz);
217                     float omega2 = GMath.PI*2f*FrequencyBandFrequencyValue(FrequencyBand.Band1000Hz);
218                     float nom1 = Math.Abs(_imaginaryTerms[FrequencyBand.Band125Hz])*omega1;
219                     float nom2 = Math.Abs(_imaginaryTerms[FrequencyBand.Band1000Hz])*omega2;
220                     float nom = nom1 + nom2;
```

```
221                 float den = omega2*omega2 - omega1*omega1;
222                 return nom/den;
223
224             default:
225                 throw new NotImplementedException();
226         }
227     }
228
229     /// <summary>
230     /// Taken from definitions at Rindel's article section 3.4 and 3.5.
231     /// </summary>
232     /// <param name="absorberType"></param>
233     /// <returns></returns>
234     private static FrequencyBand WhatBandToUse(AbsorberType absorberType)
235     {
236         switch (absorberType)
237         {
238             case AbsorberType.Porous:
239                 return FrequencyBand.Band1000Hz;
240             case AbsorberType.Membrane:
241                 return FrequencyBand.Band125Hz;
242             default:
243                 throw new InvalidEnumArgumentException("Illegal AbsorberType enum type passed");
244         }
245     }
246
247     /// <summary>
248     /// Estimates the resistance term R of the simple impedance model for the
249     /// absorber with supplied coefficients, according to Rindel's method,
250     /// in Rindel's article section 3.3 equation (12).
251     ///
252     /// The unit of the returned R is kg per square meter per second.
253     /// </summary>
254     /// <returns></returns>
255     private float EstimateResistanceTerm()
256     {
257         if (_absorberType == AbsorberType.Porous || _absorberType == AbsorberType.Membrane
258             || _absorberType == AbsorberType.Resonator)
259         {
260             float g = G(_whatBandToUse, _bandCoefficients[_whatBandToUse], _acousticImpedance);
261             var expression = (float) Math.Sqrt((
262                                 g*g - ZfStar.GetSquared(_whatBandToUse, _acousticImpedance)));
263             ResistanceTermSolutionType resistanceTermSolutionType =
264                 ResistanceTermSolutionType.Hard;
265             if (_absorberType == AbsorberType.Porous)
266                 resistanceTermSolutionType = _bandCoefficients.Maximum > 0.20
267                                              ? ResistanceTermSolutionType.Soft
268                                              : ResistanceTermSolutionType.Hard;
269             else if (_absorberType == AbsorberType.Membrane)
270                 // always hard in case of membrane
271                 resistanceTermSolutionType = ResistanceTermSolutionType.Hard;
272             else if (_absorberType == AbsorberType.Resonator)
273                 resistanceTermSolutionType = ResistanceTermSolutionType.Soft;
274             float R = g +
275                       (resistanceTermSolutionType == ResistanceTermSolutionType.Hard
```

```
276                             ? expression
277                             : -expression);
278                 return R;
279             }
280             throw new InvalidEnumArgumentException(
281                 string.Format("absorberType can only be one of {0}, {1} or {2}, " + "but was {3}.",
282                               AbsorberType.Porous.ToString(),
283                               AbsorberType.Membrane.ToString(),
284                               AbsorberType.Resonator.ToString(),
285                               _absorberType.ToString()));
286         }
287 
288         /// <summary>
289         /// Implements the G abbreviate function from section 3.2.
290         /// </summary>
291         /// <returns></returns>
292         private static float G(FrequencyBand omega, float alphaS, float acousticImpedance)
293         {
294             return ZfStar.Get(omega, acousticImpedance)*(2f/alphaS - 1);
295         }
296 
297         #region Nested type: ResistanceTermSolutionType
298 
299         private enum ResistanceTermSolutionType
300         {
301             Hard,
302             Soft
303         }
304 
305         #endregion
306 
307         #region Nested type: ZfStar
308 
309         /// <summary>
310         /// Picks out values from the table in section 3.1 in Rindel's article.
311         /// </summary>
312         private class ZfStar
313         {
314             /// <summary>
315             /// Get the value of Z_f_star, i.e. the equivalent field impedance
316             /// relative to the impedance of air, for the given frequency band
317             /// </summary>
318             /// <returns></returns>
319             public static float Get(FrequencyBand band, float acousticImpedance)
320             {
321                 // CAUTION! assumes a value for rho*c
322                 //const float rhoC = 1.2041f * 343.26f;
323                 //const float rhoC = 1.2250f * 340.31f;
324                 //return GetOverRhoC(band)*rhoC;
325                 return GetOverRhoC(band)*acousticImpedance;
326             }
327 
328             private static float GetOverRhoC(FrequencyBand band)
329             {
330                 switch (band)
```

```
331                 {
332                     case FrequencyBand.Band125Hz:
333                         return 1.04f; // see the article
334                     case FrequencyBand.Band250Hz:
335                         return 1.35f;
336                     case FrequencyBand.Band500Hz:
337                         return 1.53f;
338                     case FrequencyBand.Band1000Hz:
339                         return 1.62f;
340                     default:
341                         throw new InvalidEnumArgumentException(
342                             "Illegal FrequencyBand enum type passed");
343                 }
344             }
345 
346             public static float GetSquared(FrequencyBand band, float acousticImpedance)
347             {
348                 return Get(band, acousticImpedance)*Get(band, acousticImpedance);
349             }
350         }
351 
352         #endregion
353     }
354 }
```

A.4 Ytterligere kildekode

Utover kildekoden i de foregående seksjoner, er kildekoden til programmet Saka er ikke vedlagt i sin helhet, av tre grunner:

- De viktigste delene av programmet er gjennomgått i tilstrekkelig detalj i denne rapporten.
- Deler av programvaren kan være beskyttet av kopirettigheter.
- Uferdig programvare som ikke er utførlig testet skaper ofte mer frustrasjon enn nytte.

En fullstendig kopi av programmets kildekode og nødvendige biblioteker og filer er vedlagt oppgaven ved innlevering til sensur, i mappen “OlaCudafy1-Saka-”. (Se forøvrig A.5.) Andre interesserte bes skrive til ovikholt@gmail.com.

A.5 Ekstramateriale

Ekstramateriale består av:

- Skisse, notater og evalueringspresentasjon fra Svein Are Brekke sitt studioprosjekt beskrevet i innledningen: “Svein-Are_Mål studio.pdf” og “Svein-Are_Studio_-_Teori_og_praksis.pdf”.
- Regneark for tidsbruk i FDTD: “ola’s FDTD Scheme Spreadsheet (under development) tags- cuda fdtd qtsaka nvidia calculation simulation memory processing requirements volume size.xlsx”.
- Filmsnutt som illustrerer “dragging” over frekvensaksen i brukergrensesnittet i LFTool: “LFTool_FreqSweep_Zuend.mpg”.
- Absorpsjonsdatabasen til Multiconsult: “absorpsjonsdatabasen til multiconsult MATERIAL-MC-jan2008.LI8”.
- Bilder, WinMLS-målinger, SketchUp-modeller og vokseliseringer av Fagerengs rom, i mappene “fagerengs-rom—bilder”, “fagerengs-rom—winmls-målinger” og “fagerengs-rom—modeller-og-voxelisering”.
- Kildedata fra Odeon- og FDTD-simulering av det L-formede rommet, i mappen “odeon-rom-test”.

Ekstramaterialet er kun vedlagt oppgaven ved innlevering til sensur, som filen “vikholt-masteroppgave-tillegg.zip”. Andre interesserte bes skrive til ovikholt@gmail.com.

A.6 Notater fra målinger i Fagerengs rom

```
Manuelt sveip og lytting:
frekvens symptom
----------------
140 resonans
159 node
191 node
117 node
100 resonans
60 minus
53 romling
42 resonans



Mlinger
```

```
---------

hyttalerplasseringer
venstre hyttaler:
1.28 m til venstre vegg
1.11 m til vindu
0.88+0.24 ned (0.24 er fra 240 mm for akustisk sentrum, se datablad)
hyre hyttaler:
1.26 til hyre vegg
1.12 til vindu
0.86+0.24 ned

mling 1
22:14-22:40
se bilder
se fagereng/mling1.wmb
micplassering
1.67 til venstre vegg (alts for venstre re)
2.09 til vindu
1.13 ned til gulv
med vindhette

2 personer i rommet

mling 2 (verifisering av mling 1)
samme som mling 1

mling 3
micplassering
(bare endring registreres -- de andre avstandene for mikrofonen er de samme
  som ovenfor)
1.81 til venstre vegg (alts for hyre re)

mling 4 (verifisering av mling 3)
samme oppsett som for mling 3


mling 5
23:05
mic
0.67 til venstre
```

```
2.48 til vindu
(samme som tidligere) ned

mling 6
23:10
mic
0.65 til venstre
0.67 til vindu
0.85 ned

mling 7
mic nederst i venstre hjrne ved vinduet, inni vindhetta (se bilde)

mling 8
foran vindu
23:20
1.75 til venstre
0.63 til vindu
1.42 ned

mling 9
ved lampe thf vindu
13:23
0.47 til hyre
6.76 til bakre vegg
1.69 ned til gulv

(ved alle mlinger: 1 eller 2 personer i drpning til kjkken)

mling 10
13:29
ved skillevegg
1.11 til venstre
2.68 til skap (vindu->skap 7.00)
0.78 til tak (takhyde 2.462)
ola satt i stol

mling 10-2
ola sto i drpningen som vanlig

mling 11, i seng
```

```
13:41
1.12 til venstre
1.00 til bak
1.13 til tak
ola i stol


mling 1 til 11 skjedde med begge hyttalere spillende samme sweep-signal
  (volumkontroll satt til mono)
heretter mler vi med hyttalerne uavhengig av hverandre, alts hver for seg:

(ved alle flgende mlinger satt ola i mixestolen)

mling 12
14:28
i seng (samme som mling 11)
mono, venstre kanal

mling 13
mono, hyre kanal

mling 14
14:41
ved skillevegg (samme som mling 10)
venstre kanal

mling 15
hyre kanal

mling 16 og 17
(samme som mling 9)
14:45

(fagereng flytta seg til sofaen)

mling 18 og 19
(samme som mling 8)
14:52

mling 20 og 21
(samme som mling 7, med mic helt i hjrnet)
```

15:02

mling 22 og 23
(samme som mling 6, alts bak hyttalerne og til venstre)
15:15

mling 24 og 25
(samme som mling 5, alts
0.67 til venstre
2.48 til vindu
1.13 ned til gulv)
15:26

heretter ble mlingenes tid endret fra 20s til 10s og samtidig ble volumet
  skrudd opp ca 3 dB
(mling25_dobbel-fart er mlt p denne mten)

mling 26 og 27
(samme som mling 3)
15:33

mling 28 og 29
(samme som mling 1)
15:39

mling 30 og 31
i SWEETSPOT
868 millimeter til hver hyttaler
1.12 meter over bakken
1.75 til venstre vegg
1.95 til vindu
1.02 mellom hyttalerne

nye mlepunkter rundt sweetspot:

mling 32 og 33
tilbakelent tilstand

```
1.34 opp til tak (samme som sweetspot)
2.33 til vindu
ellers som sweetspot

mling 34 og 35
lent til venstre
1.53 fra venstre vegg
ellers som sweetspot

mling 36 og 37
1.92 til venstre
ellers som sweetspot


heretter er mic-plasseringene mindre nyaktig oppmlt

mlinger med rundtstrlende hyttaler
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

hyttaler plasser nederst i hyre hjrne ved vinduet, bak hyttalerne
p en bok slik at den kommer like langt fra vegger og gulv
se bilde

mling 38
i sweetspot

mling 39
tilbakelent

mling 40
hyrelent

mling 41
venstrelent

mling 42
posisjon samme som for mling 1

mling 43
posisjon samme som for mling 3
10++ dB lavere output (ett hakk p forsterkeren)
```

```
s installerte vi absorbentene
------------------------------

svarte absorbenter
dimensjoner
1.20 hye
0.60 brede
0.05-0.06 tykke

opphenging
0.78 over gulvet
6 stk, 3 p hver side av rommet (ikke symmetrisk, se bilder)

venstre side:
hengt opp fra vinduet med avstander (kanten til absorbentene som er nrmes
  vinduet)
0.22
1.95
2.60

hyre side:
fra vindu:
1. absorbent: 0.42
2. absorbent: 1.34
3. absorbent: 2.26

"bass"-absorbenter plassert i de to hjrnene p hver side bak hyttalerne:
hyde 1 stk: 0.60
kvadratiske sider:  0.30
rd: hyde: 0.30

diffusorer i bakkant, over senga (se bilde). ml:
0.60 kvadratiske
0.22 dype (ved hyeste topp)

mlinger
^^^^^^^^

(punkt # refererer til plasseringen under de frste mlingene 1-11 over)
```

```
mling med absorbenter, mling 1 (se abs1.wmb)
i pos. 1 (se mling 1 verst)
venstre hyttaler
abs 2: hyre hyttaler

abs 2-2
samme som abs 2 men ca +3 dB p ekstern volumkontroll


abs 3
sweetspot L

abs 4
sweetspot R

abs 5 og 6
punkt 3 L og R

abs 7 og 8
punkt tilbakelent L og R

abs 9 og 10
venstrelent L og R

abs 11 og 12
hyrelent L og R

abs 13 og 14
punkt 5 L og R

abs 15 og 16
punkt 6 L og R

abs 17 og 18
punkt 7 (helt i hjrnet der)

abs 19 og 20
punkt 8 L og R

abs 21 og 22
```

```
punkt 9 L og R

abs 23 og 24
18:41
punkt 10 L og R

(hyttalerne sender ut chirps og klikk fra tid til annen, under mling.
ser dog merkelig nok ikke ut til  pvirke mlingene nevneverdig.)

abs 25 og 26
18:49
punkt 11 (i senga) L og R
ca. samme som punkt 11 (feil under mling av punkt 11? kan ikke huske at
  den var S langt fra venstre vegg...)
```

A.7 Midtveis statusrapport og tilbakemelding

Følgende statusrapport ble sendt til mine 3 veiledere i slutten av November. Tilbakemelding fra J. H. Rindel er flettet inn i teksten.

Ola Vikholt <ovikholt@gmail.com>

Statusrapport

jens.holger.rindel@multiconsult.no <jens.holger.rindel@multiconsult.no> **00:41 24. november 2011**
Til: ovikholt@gmail.com
Kopi: ulf.kristiansen@iet.ntnu.no, clas.ola.hosoien@multiconsult.no

Hei Ola,

Takk for statusrapport. Her kommer mine kommentarer (innflettet i teksten).

Jeg har vedheftet en lille PowerPoint, hvor jeg har eksperimentert med en enkel 2.5 D FEM modell.

Mvh
Jens Holger

From: Ola Vikholt [mailto:ovikholt@gmail.com]
Sent: 22. november 2011 11:43
To: Ulf Kristiansen; Rindel, Jens Holger; Høsøien, Clas Ola
Subject: Statusrapport

Statusrapport, 22. nov 2011

Hei Ulf Kristiansen, Clas Høsøien og Jens Holger Rindel

(Ulf, jeg skulle ha sendt dette for lenge siden, men det har vært mye å gjøre i forberedelsene til flytting til Japan.)

Ved nåværende tidspunkt er jeg /midt i/ arbeidet med min masteroppgave. Jeg har samlet masse informasjon, lest artikler og kommunisert med ulike personer og fagmiljøer. Nå har jeg mange baller i luften samtidig (eller parallelt løpende tråder, som det heter i programmering).

I det følgende ønsker jeg å dokumentere min progresjon og nåværende situasjon. Dette vil klargjøre for dere og for meg selv hvor langt arbeidet er kommet. Deretter kan jeg, men litt assistanse fra dere, finne ut hvilke delprosjekter det vil være fornuftig å fokusere på videre, og hvilke delprosjekter som av relevans- og tidshensyn bør tilsidesettes.

Jeg ønsker videre å utarbeide en foreløpig tidsplan for de resterende 38 arbeidsdagene (+ evt. helgarbeid) som gjenstår før den 18. januar, som jeg tenker som tentativ leveringsfrist. (Den formelle

helgarbeid) som gjenstår før den 18. januar, som jeg tenker som tentativ leveringsfrist. (Den formelle fristen er 30. januar, men da er jeg i Japan.)

Kom gjerne med hva det måtte være av synspunkter, kommentarer og kritikk. Trykk svar, og skriv kommentarer inn rett etter avsnittet det gjelder. Jeg setter pris på såvel uferdige tanker og spørsmål som advarsler og kritikk.

Mål

Opprinnelig beskrivelsestekst utarbeidet av Multiconsult:

```
Lavfrekvent lyd i små rom

Utvikle et analyse-/simuleringsverktøy for lydfelt med lav modetetthet
som også kan brukes i ikke-rektangulære rom, for eksempel lydstudier og
kontrollrom, i frekvensområdet 20 - 200 Hz. Tilnærming kan for eksempel
være en 2.5D-modell, dvs. en kombinasjon av analytisk løsning av
bølgeligningen i z-retning (gulv og tak er parallelle) og numerisk
løsning i xy-planet (FEM-verktøy, alle vegger vinkelrett på gulv/tak).

De viktigste beregningsresultatene vil være:
* Rommets egenfrekvenser under en gitt grense, for eksempel under 200 Hz
* Visualisering av de modeformer der tilsvarer egenfrekvensene
* Beregnet overføringsfunksjon mellom en eller flere kildepunkter og et
  mottakerpunkt.

Modellen bør kunne utvides med andre egenskaper for begrensningsflatene
enn fullstendig refleksjon, for eksempel ved å angi flatenes
impedans (frekvensavhengig).
```

Jeg planlegger å fravike fra forslaget i oppgavebeskrivelsen, som følger: En 2.5D-modell er riktignok svært interessant og har utvilsomt anvendelser i andre problemer, men er i vårt tilfellet faktisk unødvendig, når man sammenlikner ulempene ved denne modellen med fordelene ved en ren 3D-modell:

- En 2.5D-modell krever at det aktuelle rommet er tilnærmet konstant i z-retning.
- En 2.5D-modell vil kreve spesiell teoretisk forståelse
- En 3D-modell kan simulere lydfeltet i en vilkårlig geometri presist
- En 3D-modell kan ha vilkårlig fordeling av impedanser rundt om i rommet
- En 3D-modellering ved aktuelle frekvenser og romstørrelser, går svært raskt på dagens (parallelliserende) maskinvare.
- En 3D-modellering er beregningsmessig mer intensiv, men er teoretisk enklere, og dermed enklere å forstå, implementere og vedlikeholde.

Hva slags 3D-modellering som skal brukes vil bli diskutert et stykke nedenfor.

Konkret produkt

Såvidt jeg forstår, ønsker vi å lage et /dataprogram/ som gjør det lett for konsulentene ved Multiconsult å finne gode plasseringer for høyttalere og lyttepunkter i små rom som studioer og mindre kinosaler. Med gode plasseringer menes at den lavfrekvente delen av overføringsfunksjon mellom høyttalernes påtrykk og mottatt lyd i lytteområdet, er så /jevn/ som mulig og med så mange resonansfrekvenser som mulig. I tillegg vil vi kunne evaluere lydkvaliteten (konkret: lavfrekvensresponsen) ved ulike lytteposisjoner. Programmet vil også kunne brukes i en arkitektfase, til å forstå hvilke lavfrekvente fenomen som kan oppstå i et rom som ennå ikke er bygget.

- Sammenheng mellom romform, modeformer, og egenfrekvensenes fordeling. Spesielt viktig å unngå sammenfall av egenfrekvenser / store intervaller mellom egenfrekvenser.

Brukerscenarioer

(Forklaring av "brukerscenario": Brukerscenario vil si en oppgave eller en rekke oppgaver som en bruker ønsker utført av et program: veien fra problemstilling -- hvor brukeren har et ønske eller et problem -- til løsning -- hvor brukeren er fornøyd med svaret produsert av programmet.)

Hyppige brukerscenarioer bør forsøkes forestilt og samles i en liste. Programmet bør utformes slik at de hyppigste eller mest sannsynlige brukerscenarioene, er de som det er enklest få programmet til å utføre. Dette gjenstår å gjøre.

Funksjonalitet

Som konkrete oppgaver programmet skal kunne utføre, tenkes følgende:

- Gitt en lytteposisjon og flere høyttalerposisjoner, få en umiddelbar forståelse for overføringsfunksjonen.
- Kunne flytte på lyttepunkt-/høyttalerplassering og observere endring i overføringsfunksjon.
- Bruke konseptet lytteområde i stedet for lyttepunkt, da lytteområde er et mer realistisk konsept.

- Lytteområde isf lyttepunkt er OK, men lavere prioritet

- Viktig å kunne endre på overflatenes impedans; først og fremt på tap, dvs. realdelen. Dette har stor betydning for hvor glatt eller skarp overføringsfunksjonen blir. Se eksemplene til slutt i vedlagte PowerPoint.

Og eventuelt:

- Kunne spesifisere lovlige områder for høyttalere og lytteområde, hvorpå programmet vil finne den kombinasjonen som gir best resultat.
- Få beregnet individuell equalization for hver av høyttalerne, som vil gjøre mottatt respons så flat

som mulig, i henhold til den imponerende "Chameleon Subwoofer Arrays"-metoden som er beskrevet i en artikkel av A.J. Hill, 2010.

Brukergrensesnitt

Ønsker å gjøre beregningsresultater umiddelbart enkelt å tolke, gjennom visualisering av 3D lyd-volum, snittflater, tidsutvikling av lydfelt, og gjennom tegning av frekvensresponsgrafer.

Det bør være mulig å lytte til frekvensresponsen. Dette er relativt lett og raskt å få til programmatisk.

- Dette kan komme senere. Ikke prioritert hvis beregningene er begrenset til under 200 Hz.

Ellers planlegger jeg å hente inspirasjon fra Hill og Hawksfords "Visualization and Analysis Tools for Low-Frequency Propagation in a Generalized 3D Acoustic Space", 2009, hvor de presenterer et

liknende program det vi ønsker å lage, som er skrevet i MatLab, men som ikke ser ut til å bruke GPU eller å tillate importering av modeller. Programmet kan gi ideer om andre nyttige beregninger, funksjonaliteter, visualiseringsmetoder og modelleringsteknikker.

- NB: Referansen er ikke fra 2009, men 2011 i JAES, vol. 59.

Sammenlikne med målinger

Som test av programvaren, samt for verifisering av beregningsresultatene, ønsker vi å modellere ett eller to eksisterende rom, og gjøre målinger i de samme rommene:

1. Rune Fagereng studerer musikkvitenskap ved Universitetet i Oslo og skriver nå masteroppgave om lavfrekvent lyd, fra et musikkperspektiv. Tor Halmrast har introdusert meg til ham. Jeg har modellert hans hybel i Hønefoss, hvor han tenker å installere diffusorer og absorbenter i løpet av neste uke, og jeg håper å kunne gjøre målinger i rommet med og uten akustiske elementer installert. Rommet er noe komplisert (hule trevegger, komplisert geometri), så et betydelig avvik må forventes, men dette vil kunne gi en indikasjon på hvor anvendelig programmet vil være i realistiske rom.

2. Ellers har det blitt forslått at jeg kan måle i et rom i kjelleren i Multiconsult, som skal være betydelig enklere (betongflater etc.).

(Lage program eller måle først? Dette bestemmer om man kan benytte programmet under målingene.)

Implementeringsdetaljer

Modell av rom

Først må en modell av rommet importeres inn i dataprogrammet. Dette gjøres enklest ved å hente en modell fra SketchUp, som er det desidert enkleste og mest tilgjengelige programmet for tegning av tredimensjonale modeller av rom. SketchUp kan også åpne mange eksisterende modeller.

Modeller kan hentes ut av SketchUp på to måter:

1. Ved å lage et skript for SketchUp (i Ruby) som benytter seg av SketchUps innebygde funksjoner for å kunne danne en vilkårlig representasjon av modellen. Jeg har lite erfaring her, men Jonathan Scheaffer og Odeons utikler Claus Lynge Christensen, som jeg er i kontakt med, har erfaring med å skrive slike plugins.

2. Ved å lagre modellen i et eller annet format, for eksempel .obj eller .dae, .3ds eller .dwg, som så kan leses og tolkes av mitt program. (Dersom modellen skal lagres i .3ds eller .dwg, trengs en Pro-lisensversjon av SketchUp.)

- Ikke bruk tid på å lage egen skript for SketchUp. Foreslår å bruke den, der er laget til Odeon (SU2Odeon). Den håndterer også de kompliserte geometriske problemer, der vil oppstå i mer vilkårlige rommodeller.
- Fra Odeon er det uproblematisk å eksportere videre i dxf format, om det ønskes.
- Fra Odeon kan rommodellen fås som en text-fil med punkt-koordinater og flatedefinisjoner.

SketchUp-modeller er polygonmodeller, som må Delaunay-trianguleres for bruk i FEM (og BEM?), mens dersom FDTD eller FDFD skal brukes må man voxelisere [*] modellen, og dette kan være litt vanskelig.

- Dette er et sentralt problem (også for BEM), som er nødvendig å løse.

[*] (Når det gjelder voxelisering av modeller har jeg kikket på et Constructive Solid Geometry (CSG) MatLab-script som Schaeffer refererte meg til. Dette skriptet har jeg foreløpig ikke fått til å virke.)

I tillegg til modell, må plassering av høyttalere og mikrofoner spesifiseres. Dette tenkes gjort direkte i programmet, under kjøring.

Beregningsmodell

Det finnes flere ulike måter å modellere lydfeltet og veggene på. Jeg vil vurdere metodene FDTD, FDFD, FEM og BEM.

Faktorer å ta hensyn til for å velge:

- Eksisterende kode og kompetanse (stor overvekt i favør av FDTD)
- Tidsbruk, dvs. beregningskompleksitet (gjenstår å finne ut av)
- Nøyaktighet? (antar foreløpig tilfredsstillende nøyaktighet for alle metoder)
- Impdedans: hvor lett det er å implementere en tilfredsstillende impedansmodell
- Direktivitet: hvor lett det er å modellere kilder med ulik direktivitetsmønster (kardioide etc.) (Må kunne modellere faktiske høyttalere -- Odeon kan visstnok hente data om høyttalere...)

- Modell: hvor lett det er å generere modell (polygon- versus voxel-modell), som diskutert over

- Foreslår en enkel impedansmodell med en realdel og to led i imaginærdelen, se slutning av vedheftede PowerPoint. Fordelen sammenlignet med ande impedansmodeller er, at de tre led kan antas tilnærmet uavhengige av frekvensen i aktuelt frekvensområde, ca. 20 – 200 Hz.

- Direktivitet for lydkilder kan vente; ikke høyt prioritert for lave frekvenser.

Impedans

Modellering av impedans vil avhenge av beregningsmodell.

Impedans-data for ulike fysiske materialer som vi ønsker å modellere, kan hentes ut fra Odeon eller en Odeons kunnskapsdatabase -- dette må undersøkes nærmere.

Aktuelle spørsmål som bør besvares om tiden tillater det:

- Hvor mye har impedans å si for resultatet, under 200 / under 100 Hz -- altså, hva er en øvre grense for feil dersom impedans ignoreres eller forenkles fullstendig?

- Impedans kan forenkles, men må ikke ignoreres, jevnfør tidligere. Spesielt realdelen er viktig.

Beregningsmetode

Jeg ønsker å gjøre selve beregningen på GPU, datamaskinens grafiske prosesseringsenhet. Dette vil få beregningene til å gå raskt. Det krever spesiell programmering, men det er ikke altfor vanskelig eller tidkrevende. Jeg ønsker ikke å fokusere på at GPU er brukt (til forskjell fra i prosjektoppgaven min, hvor det var et poeng), siden parallell-prosessering blir mer og mer vanlig. (Vil bruke API-et "CUDA". Andre API-er for parallelle beregninger er CUDA, OpenCL og Microsoft DirectCompute)

Ved å beregne på GPU, går beregningene forholdsmessig svært raskt. I stedet for å forenkle, og beregne bare 2.5D, blir man da i stand til å modellere alle geometrier. Dermed trengs ikke en avansert 2.5D-kombinasjonsmodell.

Med svært raske beregninger, vil programmet bli responsivt -- resultater vil bli beregnet tilsynelatende umiddelbart, og dette vil gjøre at programmet kan brukes på en annen måte, ved for

eksempel å gi inntrykk av at man kan /dra/ på et lyttepunkt og se overføringsfunksjonen i kontinuerlig endring.

Programmering

Tidligere har jeg programmert i C++, på min private Mac. Det grafiske brukergrensesnittet har da blitt laget vha. et tynt lavnivå-API som heter GLUT. Dette gjør at alt brukergrensesnitt må utvikles fra

laget via: et tynt lavnivå-API som heter GLUT. Dette gjør at alt brukergrensesnitt må utvikles fra bunnen av. Videre er C++ et gammelt og ineffektivt språk som gjør at utvikling går tregt. (Fordelen er at GLUT og C++ virker på Windows, og programmet mitt kan derfor kjøres på Windows uten videre.)

I stedet ønsker jeg nå å bruke C#, som er et nyere og mye mer moderne programmeringsspråk. C# har et utmerket API for grafisk brukergrensesnitt, men dette virker bare på Windows. Om man bruker et godt utviklerverktøy vil det være svært raskt å utvikle i C# - her er Microsoft Visual Studio 2010 det riktige valget.

(Skulle jeg programmert i C++ ville gjenværende tid vært knapp, men, med de riktige verktøyene kan jeg programmere svært raskt, og kan lage et program som vil være mer enn godt nok. Et program skrevet i C# vil dessuten være lettere å vedlikeholde og utvide for Multiconsult i fremtiden.)

(Datamaskinen det utvikles på må ha et høy-ytelses-grafikk-kort av en viss type (se over), for å kunne gjøre de parallelle beregningene. - Vil ikke dette gjøre at programmet bare vil kunne kjøre på denne spesielle PC-en? - Bare helt i starten. GPGPU, som det heter når man gjør ikke-grafiske beregninger på grafikk-kort, er i ferd med å bli svært utbredt, og som en følge av dette blir slike grafikk-kort raskt mer og mer vanlig i alle typer datamaskiner, også i bærbare. Min Mac har et slikt grafikk-kort, og for eksempel Lenovo ThinkPad W510 til 8700 kr har et slikt kort.)

I det siste har Matlab introdusert biblioteker som lar deg utføre parallelle beregninger. Så det kunne tenkes at man kunne utvikle det ettelyste programmet i MatLab. Men, jeg kan ikke Matlab på langt nær så godt som C#, og spesielt ikke GUI-produksjon. Dessuten er Halls noe ekvivalente program (nevnt tidligere) utviklet i Matlab -- og dette programmet er ikke spesielt behagelig å bruke.

- I Multiconsult brukes bare Windows.
- C# er nok et gott valg.

Roomcalc Ex-01.pptx
1794K
