

Utvikling av simuleringsverktøy for lavfrekvent lyd i små rom

Truls Vidarssønn Klami

Master i elektronikk

Innlevert: juni 2013

Hovedveileder: Ulf R Kristiansen, IET

Medveileder: Jens Holger Rindel, Multiconsult
Clas Ola Høsøien, Multiconsult

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Oppgavetekst

Utvikle et analyse-/simuleringsverktøy for lydfelt med lav modetetthet som også kan brukes i ikke-rektangulære rom, noe som ofte er tilfelle i lydstudioer og kontrollrom i frekvensområdet 20 – 200 Hz. Tilnærming kan for eksempel være en 2.5 D 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 mot-takepunkt.

Modellen bør ta høyde for andre egenskaper for begrensingsflatene enn fullstendig refleksjon, for eksempel ved å angi flater med frekvensavhengig impedans.

Forord

Denne rapporten er resultatet av en masteroppgave i akustikk, skrevet ved institutt for elektronikk og telekommunikasjon i samarbeid med Multiconsult AS. Masteroppgaven avslutter sivilingeniørstudiet i elektronikk ved NTNU.

Jeg vil takke min veileder Jens Holger Rindel fra Multiconsult både for faglig hjelp og for nyttige råd om fremgangsmåte gjennom arbeidet med oppgaven. Sist men ikke minst vil jeg takke min faglærer, professor Ulf Kristiansen ved NTNU, for faglig hjelp og for givende diskusjoner rundt oppgaven.

Truls Vidarssønn Klami

Trondheim, 17. juni 2013

Sammendrag

Målet med denne oppgaven har vært å lage et effektivt simuleringsverktøy for lavfrekvent lyd i små rom. Verktøyet løser bølge-ligningen numerisk ved endelige elementers metode (FEM), og har som mål å kunne beregne lavfrekvent lyd nøyaktig. Dagens mest utbredte simuleringsmetode for romakustikk, raytracing, gir store feil ved lave frekvenser, og oppgavens verktøy utvikles for å komplimentere raytracing-metoden.

Simuleringsverktøyet som er utviklet er begrenset til kun å gjelde for rom formet som rettvinklede prizmer, dette for å kunne løse bølge-ligningen separat i horisontalplan og z-retning i rommet. I praksis byr dette skjeldent på problemer, da de aller fleste rom uansett har slik form.

Akustiske egenskaper for rommets flater implementeres i verktøyet gjennom en forenklet impedansmodell, der hver flate beskrives ved en masse-fjær-demping-analogi. Denne modellen beskriver faktiske lydabsorbenter godt ved lave frekvenser. Resultatene fra simuleringen visualiseres i MATLAB, der både transferfunksjon mellom en kilde og en mottaker og modeformer ved gitte frekvenser plottes.

Verifisering og evaluering av verktøyet er gjort ved sammenligning med det eksisterende FEM-simuleringsprogrammet COMSOL Multiphysics, som er antatt å gi så korrekte resultater som et FEM-verktøy kan gjøre. I de tilfeller der analytisk løsning av bølge-ligningen finnes er verktøyet også vurdert opp mot disse.

Sammenligninger med konvensjonelle FEM-simuleringer gjort direkte i et tredimensjonalt rom har vist en tidsbesparelse på 80 -90 % ved bruk av verktøyet utviklet i denne oppgaven. Det har dessuten ytterligere potensiale, ettersom verktøyet bare utnytter én prosessorkjerne.

Med helt harde vegger gir simuleringsverktøyet tilnærmet perfekt transferfunksjon og modeplot. Med endelige veggimpedanser fås derimot et avvik sammenlignet med konvensjonell FEM-simulering, som øker når impedansene går mot 0. Det er konkludert med at verktøyet gir pålitelige resultater for rom der alle veggimpedanser har imaginærdel $|\text{Im}(z)| > 10000 \text{ kg}/(\text{m}^2\text{s})$, men langt lavere impedansverdier kan anvendes på en liten andel av overflatearealet i rommet. I et rom med ellers harde vegger er resultatet funnet pålitelig med $\text{Im}(z) = 1000 \text{ kg}/(\text{m}^2\text{s})$ på én vegg.

Lydabsorpsjon undervurderes av verktøyet, og i dempede rom fremstilles dermed resonansene som skarpere enn ved konvensjonell tredimensjonal FEM-simulering.

Verktøyet finner dessuten egenfrekvenser uavhengig av dempingen, slik at resonansene også fremstilles noe høyere enn i COMSOL.

Tidsbesparelsen i forhold til konvensjonell tredimensjonal FEM-simulering gjør at simuleringsverktøyet kan være aktuelt å bruke, og i moderat dempede rom bør heller ikke de nedre grensene for impedanser utgjøre noe problem. Verktøyet har stort forbedringspotensiale med tanke på blant annet kjøretid og brukergrensesnitt. Allerede i sin nåværende form anses det likevel som fullt anvendelig for romakustiske simuleringer.

Abstract

The goal of this thesis has been to develop an effective tool for acoustic simulation of low frequencies in small rooms. The tool solves the wave equation numerically by the finite element method (FEM) in order to predict low frequency sound accurately. The most common method for room acoustic simulation today, the raytracing method, gives large errors at low frequencies, and the method of this thesis is intended to compliment raytracing.

The tool developed is limited to rooms in the shape of right prisms. This enables solving the wave equation separately in the horizontal plane and the vertical direction, reducing computation time. In most applications this should not be a major limitation, as rooms typically are shaped like this.

Acoustical properties of the room surfaces are implemented through a simplified impedance model, in which each surface is described by a mass-spring-damping analogy. This model works well for physical sound absorbers at low frequencies, as modelled in this thesis. Simulation results are displayed in MATLAB by plotting both the transfer function between a source and a receiver and the mode shapes at certain frequencies.

Testing of the tool is done by comparisons with simulations in the existing FEM tool COMSOL Multiphysics, which is assumed to provide results that are as good as they get within the limitations of FEM. The tool is also compared with the analytical solution to the wave equation, in the rare cases where the solution is known.

Compared with conventional, three-dimensional simulations in COMSOL, the tool developed reduces the simulation time by 80 - 90 %. The tool has potential of further reductions of simulation time, as it only utilizes one CPU core.

Simulation of a room with perfectly hard walls resulted in nearly perfect transfer functions and mode plots. With finite wall impedances, on the other hand, the results diverged from the ones from COMSOL. The tool developed provides trustworthy results for rooms where all surfaces have an impedance with imaginary part $|\text{Im}(z)| > 10000 \text{ kg}/(\text{m}^2\text{s})$. One can, however, use significantly lower impedances on small areas in the room. In a room with otherwise hard walls, results were found trustworthy with the imaginary part of the impedance $\text{Im}(z) = 1000 \text{ kg}/(\text{m}^2\text{s})$ on one wall.

The tool underestimates the influence of sound absorption, causing the transfer function peaks in damped rooms to be sharper than in the equivalent COMSOL

simulations. Furthermore, the tool ignores the sound absorption when calculating eigenfrequencies. The resonances are therefore found at slightly higher frequencies than in COMSOL.

The reduced time consumption compared with conventional FEM simulations is an attractive property of the simulation tool, and in moderately damped rooms the impedance limitations should not be any problem. Although there is great room for improvement for the tool, both in terms of simulation time and of user interface, it is in its current shape considered applicable for simulation of room acoustics.

Ordliste

1D, 2D, 3D - endimensjonal, todimensjonal, tredimensjonal

COMSOL - FEM-basert simuleringsverktøy

Eigenfrekvens - Frekvensene som gir stående bølger i et rom

Eigenmode - Stående bølge-mønster, altså lydtrykk som funksjon av posisjon, ved de enkelte egenfrekvensene

FEM - Finite element method, på norsk endelige elementers metode, en numerisk metode for å løse differensialligninger

FreeFem++ - Et gratis FEM-basert verktøy for numerisk løsning av differensialligninger

FreeFem++-cs - Skall til FreeFem++ som gir en viss grad av grafisk brukergrensesnitt

Hard vegg - Perfekt reflekterende flate, altså med uendelig stor impedans

Impedans - Refererer til spesifikk akustisk impedans, $z = \frac{p}{u}$, der p og u er henholdsvis lydtrykk og partikkelfart.

Mesh - Inndeling av rommet som simuleres med FEM, i denne oppgaven triangler i 2D og tetraeder i 3D

Mode - Lydtrykk som funksjon av posisjon ved en gitt frekvens

Resonans - Samme betydning som egenfrekvens

Rettvinklet prisme - Geometrisk figur som ligner en rett sylinder, men som har polygonformet grunnflate og tverrsnitt

Innhold

1	Innledning	1
2	Teori	3
2.1	Bølgeligningen og Helmholtz ligning	3
2.2	Egenmoder, egenfrekvenser og stående bølger i rektangulære rom med harde vegger	3
2.3	Endelige elementers metode	5
2.4	Svak formulering	8
2.5	Grensebetingelser	8
2.6	Kombinasjon av løsninger i 2D og 1D	10
2.7	Veggimpedanser	12
3	Simuleringsverktøy og skript	14
3.1	Simuleringsverktøyet FreeFem++	14
3.2	Om den fysiske modellen	14
3.3	Om beregningsmodellene	15
3.4	Generelt om skriptene	16
3.5	2D-simulering med kilde	17
3.5.1	Parametre, variabler og konstanter	18
3.5.2	Mesh-generering	18
3.5.3	For-løkke med løsning av bølgeligningen	18
3.5.4	Behandling og eksportering av simuleringsresultater	19
3.6	Modebasert 3D-simulering	19
3.6.1	Parametre, variabler og konstanter	19
3.6.2	Mesh-generering	19
3.6.3	For-løkke med løsning av bølgeligningen	20
3.6.4	Behandling av simuleringsresultater	20
3.6.5	Eksportering av simuleringsresultater	21
3.7	COMSOL	21
3.8	Visualisering av simuleringsresultater	22
4	Simuleringer	23
4.1	CPU-test	23
4.2	Rektangulært rom med konstante veggimpedanser	23
4.2.1	Harde vegger, gulv og tak	24
4.2.2	Store, imaginære impedanser	25
4.2.3	Mindre, imaginære impedanser	26
4.2.4	Store, komplekse impedanser	26
4.2.5	Mindre, komplekse impedanser	26

4.2.6	Én vegg med liten, kompleks impedans	26
4.3	Annen romgeometri	27
4.4	Impedanser gitt ved parameterne R , m og d	27
4.5	Sammenligning mellom kilde- og 3D-simulering i FreeFem++	28
5	Resultater	30
5.1	CPU-test	30
5.2	Rektangulært rom med konstante impedanser	30
5.2.1	Harde vegger, gulv og tak	30
5.2.2	Store, imaginære impedanser	32
5.2.3	Mindre, imaginære impedanser	34
5.2.4	Store, komplekse impedanser	35
5.2.5	Mindre, komplekse impedanser	36
5.2.6	Én vegg med liten, kompleks impedans	37
5.3	Annen romgeometri	38
5.4	Impedanser gitt ved parameterne R , m og d	39
5.5	Sammenligning mellom kilde- og 3D-simulering i FreeFem++	40
6	Diskusjon	42
6.1	Sammenligning mellom 3D-modell i FreeFem++ og COMSOL	42
6.2	Vurdering av kilde- og 3D-modellen i FreeFem++	43
6.3	Brukergrensesnitt	44
6.4	Visualisering	44
7	Konklusjon	46
8	Videre arbeid	48
	Referanser	49
A	HMS-vurdering	50
B	Brukermanual	52
B.1	Bruk av eigen-original.edp	52
B.1.1	Bestemmelse av parametre	52
B.1.2	Kjøring av skript	53
B.2	Bruk av eigen-ui.edp	53
B.3	Endring av antall hjørner	54
B.4	Visualisering	56
B.5	Annet	56
C	MATLAB-kode	57

C.1	Visualisering av resultater	57
C.2	Kode for beregning av egenfrekvenser i rektangulært rom med harde vegger	59
C.3	cpu-test	60
D	FreeFem++-kode	61
D.1	Kildemodell	61
D.2	3D-modell	66
D.3	3D-modell med brukerinput gjennom dialogbokser	77
E	COMSOL-simulering	88

1 Innledning

Å kunne forutsi hvordan et rom lyder før det bygges er meget ønskelig for en romakustiker, og det er essensielt når et rom skal prosjekteres.

Før numeriske beregninger på datamaskin var mulig ble romakustiske parametre som etterklangstid beregnet ved svært enkle modeller, slik som Sabines formel for etterklangstid [2, s. 336]. Denne beskriver etterklangstiden som funksjon av romvolum V og total lydabsorpsjon A i rommet, og er formulert som $T = \frac{0.161V}{A}$. Uttrykket sier ingenting om hverken absorberenes posisjon eller rommets form.

I dag brukes vanligvis datamaskiner for å finne rommets akustiske egenskaper gjennom numeriske beregninger. Teknikken som hovedsakelig brukes kalles raytracing, og er anvendt i romakustikksimuleringsverktøy som Odeon og CATT-acoustic. Raytracing brukt i romakustikk baserer seg på betraktning av lydølgene som svært mange diskrete lydstråler der banen til hver stråle beregnes separat, og forutsetter at lydølgene kun beveger seg rett frem. I åpent rom er dette riktig, men når lydølgene treffer enten hjørner eller objekter som er små i forhold til bølgelengden bøyer bølgene seg rundt disse. Ved høye frekvenser og tilhørende korte bølgelengder skjer dette kun i liten grad, mens feilen ved raytracing blir vesentlig større i bassen [4, s. 57].

Fra analyse av blant annet rektangulære rom i Kinsler et al. [2, s.246] vet vi at lydtrykket ved lave frekvenser varierer mye mellom ulike posisjoner i et rom. Dette skyldes stående bølger, et fenomen som beskrives i kapittel 2.2. Kartlegging av hvordan lydtrykket fordeles i et rom er derfor ønskelig, både for å unngå utforming av rom som gir ekstreme lydtrykksforskjeller og for enkelt å finne for eksempel fordelaktige høyttalerposisjoner i rommet. Med fordelaktige posisjoner menes posisjoner der rommets forskjellige stående bølger har forholdsvis like utsving.

For å eksakt beregne lydtrykket i et rom er en nødt til å løse bølgeligningen. Dette er mulig analytisk kun for noen få romformer [4, s. v], men løsningen kan tilnærmes ved hjelp av numerikk. En numerisk metode som kan gjøre dette er endelige elementers metode, kalt FEM for finite element method. Med FEM kan bølgeligningen løses numerisk for vilkårlige romformer, og veggens akustiske egenskaper kan inkluderes. Metoden har dessuten fordelen at behøvd regnekraft øker med frekvensen, og den virker derfor å komplimentere raytracing-metoden godt.

Selv ved lave frekvenser er metoden likevel ansett som beregningsmessig kostbar. Derfor er det i denne oppgaven valgt å gjøre separate FEM-analyser i horisontalplanet og i z-retning for deretter å kombinere disse løsningene til å gi lydtrykket

i hele rommet. Dette er antatt å være beregningsmessig vesentlig lettere enn å gjøre FEM-analysen direkte på et tredimensjonalt rom.

For selve FEM-analysen i en, to eller tre dimensjoner (1D, 2D eller 3D) finnes det i dag fungerende programvare, slik som for eksempel COMSOL Multiphysics. Disse er dog gjerne svært kostbare, blant annet fordi de har svært mye funksjonalitet også for andre fagområder. COMSOL har dessuten ikke funksjonalitet for å kombinere 1D- og 2D-løsning slik det ønskes i denne oppgaven. Derimot finnes det enklere, gratis programvare som i langt større grad kan tilpasses av brukeren, og dermed anvendes både for løsning av bølgeligningen og for kombinasjon av løsningene.

I denne oppgaven skal et slikt gratisprogram, kalt FreeFem++, anvendes for å lage et helhetlig simuleringsverktøy som løser bølgeligningen i horisontalplan og z-retning for deretter å kombinere disse løsningene til en fullstendig løsning for rommet. For at en slik kombinasjon av løsninger skal være korrekt må rommets egenskaper i horisontalplan og z-retning være uavhengige av hverandre, og verktøyet lages derfor kun for rom med parallelle og flate gulv og tak, samt vertikale vegger. De aller fleste rom har uansett slik form, så denne begrensningen bør ikke være noe stort problem. Verktøyet skal brukes ved lave frekvenser, hvilket i denne oppgaven er definert som opp til 200 Hz, for å komplimentere eksisterende raytracing-modeller.

Ved å velge kilde- og mottakerposisjon skal en med det utviklede simuleringsverktøyet kunne beregne transferfunksjonen mellom disse. I tillegg skal verktøyet gi ut modeplot, samt en liste over rommets egenfrekvenser.

Hovedvekten i oppgaven er lagt på utvikling og validering av verktøyet. Valideringen skal i all hovedsak gjøres ved sammenligninger med simuleringer i COMSOL, som uten videre er antatt å gi så pålitelige resultater som et FEM-verktøy kan gjøre. Det legges lite vekt på visualisering av resultatene, men som eksempel på hvordan resultatene kan vises gjøres det en kombinert plotting av transferfunksjon og moder i MATLAB.

2 Teori

2.1 Bølgeligningen og Helmholtz ligning

Bølgeligningen er ligningen som beskriver lydtrykket som funksjon av tid og rom. Denne ligningen, utledet blant annet i Kinsler et al. [2, Kap. 5, s.113-119], er i 2D, i kartesiske koordinater, gitt som

$$\nabla^2 p(x,y,t) - \frac{1}{c^2} \frac{\partial^2 p(x,y,t)}{\partial t^2} = 0 \quad (1)$$

der $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, c er lydfarten og $p(x,y,t)$ er lydtrykket, altså avviket fra stasjonærtrykket.

I rom uten tap av lydenergi har vi fra Kinsler et al. [2, s. 247] at lydtrykkets rom- og tidskomponent er uavhengige av hverandre. I den videre utledningen antas et slikt rom, og lydtrykket kan skrives som $p(x,y,t) = p(x,y) \cdot T(t)$, der $T(t)$ er en vilkårlig funksjon. Ved en bestemt frekvens f vil tidsleddet i trykket svinge som en sinus, og med kompleks notasjon får vi $T(t) = e^{j\omega t}$, der $\omega = 2\pi f$. Lydtrykket blir da $p(x,y,z) = p(x,y) \cdot e^{j\omega t}$. Ved å dobbelderivere dette uttrykket for p med hensyn på t kan vi deretter sette det inn i bølgeligningen, ligning 1, og få

$$\nabla^2 p(x,y) \cdot e^{j\omega t} + \frac{\omega^2}{c^2} p(x,y) \cdot e^{j\omega t} = 0$$

Tidsleddet kan så divideres bort, og vi får Helmholtz ligning:

$$\nabla^2 p(x,y) + k^2 p(x,y) = 0 \quad (2)$$

der $k = \frac{\omega}{c}$ er bølgetallet. Denne ligningen gir lydtrykkets utsving som funksjon av posisjon i et tapsfritt rom.

2.2 Egenmoder, egenfrekvenser og stående bølger i rektangulære rom med harde vegger

I Kinsler et al. [2, s. 247] vises det at ligning 2, i et rektangulært rom med harde vegger, kun har en løsning for enkelte, diskrete verdier av bølgetallet k . Dette er for øvrig tilfellet for alle rom [5, s. 68], men videre i delkapittelet tas det for enkelhets skyld utgangspunkt i et rektangulært rom. Hvert av bølgetallene

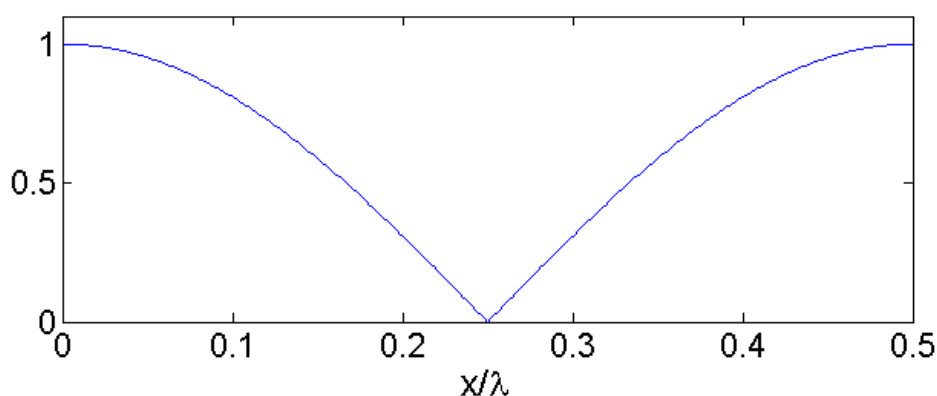
representerer en av rommets egenmoder og gjennom relasjonen $k = \frac{\omega}{c} = \frac{2\pi f}{c}$ en av egenfrekvensene. En beskrivelse av egenmoder og -frekvenser er gitt videre i dette delkapittelet.

Fra Kinsler et al. [2, s. 247] har vi følgende uttrykk for egenfrekvensene i et rektangulært rom med harde vegger:

$$f_{lmn} = \frac{c}{2} \sqrt{\left(\frac{l}{L_x}\right)^2 + \left(\frac{m}{L_y}\right)^2 + \left(\frac{n}{L_z}\right)^2} \quad (3)$$

der L_x , L_y og L_z er rommets dimensjoner, og l , m og n er modenummer i henholdsvis x-, y- og z-retning. I rektangulære rom angir modenumrene det antallet halvbølger stående bølge-mønsteret får i de respektive retningene ved egenfrekvens f_{lmn} . Et eksempel som forklarer dette følger:

Egenfrekvenser kan forstås som de frekvenser der lydbølgene, etter å ha blitt reflektert frem og tilbake i rommet, er i fase med seg selv. Dette skjer for eksempel ved frekvensen der rommet i x-retning er en halv bølgelengde langt. Inntil de to veggene i x-retning fås da maksimalt lydtrykksamplitude, ettersom en lydbølge som går ut fra den ene veggen vil treffe den andre veggen i motfase, og den første i fase igjen. Midt mellom veggene fås derimot et nullpunkt, ettersom en lydbølge som går ut fra midtpunktet vil ha beveget seg en halv bølgelengde, altså være 180° ut av fase, når den kommer tilbake til midtpunktet etter en veggrefleksjon. Fra Kinsler et al. [2, s. 247] har vi at lydtrykksamplituden langs rommet i x-retning ved denne frekvensen går som $|p| \propto \left| \cos\left(\frac{\pi x}{L_x}\right) \right|$. Denne trykkfordelingen er vist i figur 2.1.



Figur 2.1: Relativ lydtrykksamplitude i x-retning

Lydtrykkfordelingen vist i figur 2.1 gjelder for alle egenfrekvensene f_{1mn} , etter som den kun viser fordelingen i x-retning. Ved egenfrekvensen f_{100} får vi så 0 halvbølger i y- og z-retning i stående bølge-mønsteret, slik at lydtrykket er konstant i disse retningene. Denne trykkfordelingen i rommet kalles egenmoden til egenfrekvens f_{100} , og betegnes i denne oppgaven som ψ_{100} .

Videre vil begrepet resonans ofte bli brukt om egenfrekvensene, dette har i denne oppgaven samme betydning.

2.3 Endelige elementers metode

Endelige elementers metode, forkortet FEM for finite element method, er en numerisk metode for å løse differensialligninger. Dette er metoden som benyttes i denne oppgaven. Generelt er metoden forholdsvis komplisert, og det gjøres her kun en begrenset gjennomgang av metoden for bruk i 2D, basert på den skrevet i Kristiansen & Viggen [4, Kap. 3]. I grove trekk kan metoden beskrives som følger:

Området som skal studeres deles opp i elementer med et mesh, altså en ruteinndeling, der formen og størrelsen på rutene kan variere. Området kan for eksempel være gulvplanet i et rom. Dersom variabelen, her lydtrykket, er kjent i nodene, altså hjørnene i rutene, kan verdier inne i hver rute tilnærmes ved interpolasjon mellom nodeverdiene til den gjeldende ruten. I 2D-tilfellet er tilnærmingen til p , kalt \tilde{p} , gitt som

$$\begin{aligned}\tilde{p}(x,y) &= N_1(x,y)p_1 + N_2(x,y)p_2 + \dots + N_M(x,y)p_M \\ &= \vec{N}(x,y) \cdot \vec{p}\end{aligned}\tag{4}$$

der M er antall noder i elementet som inneholder punktet (x,y) , og N_i og p_i er henholdsvis interpolasjonsfunksjonen, kalt basisfunksjonen, og trykket til node nummer i . \vec{N} og \vec{p} er vektorer som inneholder basisfunksjonene N_i og nodetrykkene p_i for alle nodene til et element.

Basisfunksjonene i \vec{N} defineres slik at de er $\neq 0$ kun i elementene som berører basisfunksjonens node, og ligning 4 er dermed like gyldig dersom M er antall noder i meshet. Videre omdefineres derfor \vec{N} og \vec{p} til å være vektorer som inneholder basisfunksjoner og nodetrykk i alle meshets noder, og M angir videre antall noder i hele meshet.

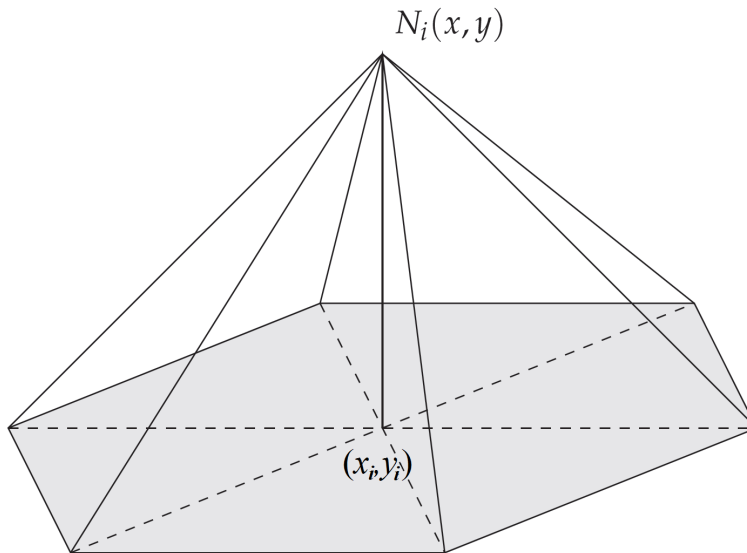
Ytterligere krav til basisfunksjonene i \vec{N} , som kreves for at interpolasjonen skal bli korrekt i nodepunktene, er som følger:

$$N_i(x_j, y_j) = \begin{cases} 1, & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases}$$

$$\sum_{i=1}^M N_i(x, y) = 1$$

for $i, j \in 1, M$ i området som studeres, og der (x_j, y_j) er koordinatene til node j .

Basisfunksjonene \vec{N} kan formuleres på ulike måter, og kan for eksempel defineres som lineært avtagende fra sin egen node til nabolodene. I 2D vil basisfunksjonen N_i da se ut som i figur 2.2.



Figur 2.2: Eksempel på basisfunksjon til node i , figur er hentet fra [4, s. 36], men noe modifisert

Ved å sette tilnærmingen til p gitt i ligning 4 inn i ligning 2 får vi følgende approksimasjon til lydtrykket i et 2-dimensjonalt område:

$$\nabla^2 \tilde{p}(x, y) + k^2 \tilde{p}(x, y) = \nabla^2 (\vec{N}(x, y) \cdot \vec{p}) + k^2 \vec{N}(x, y) \cdot \vec{p} = R(x, y) \approx 0 \quad (5)$$

der $R(x, y)$ er feilen som skyldes tilnærmelsen \tilde{p} for p . Det videre målet er da å gjøre feilledet $|R(x, y)|$ så lite som mulig for alle aktuelle x og y , altså over hele

området. Dette kan oppnås ved å integrere ligning 5 over hele området og sette dette integralet lik 0, men for å minimere feilen også lokalt i de enkelte elementene brukes vanligvis den såkalte Galerkin-metoden [4, s. 40]. Dette innebærer at ligning 5 multipliseres med basisfunksjonene \vec{N} før integrasjonen gjøres. Vi får da

$$\int_{\Omega} \vec{N} \left(\nabla^2(\vec{N} \cdot \vec{p}) + k^2 \vec{N} \cdot \vec{p} \right) d\Omega = 0 \quad (6)$$

som altså er ligningssettet

$$\begin{bmatrix} \int_{\Omega} N_1 \left(\nabla^2(\vec{N} \cdot \vec{p}) + k^2 \vec{N} \cdot \vec{p} \right) d\Omega = 0 \\ \int_{\Omega} N_2 \left(\nabla^2(\vec{N} \cdot \vec{p}) + k^2 \vec{N} \cdot \vec{p} \right) d\Omega = 0 \\ \vdots \\ \int_{\Omega} N_M \left(\nabla^2(\vec{N} \cdot \vec{p}) + k^2 \vec{N} \cdot \vec{p} \right) d\Omega = 0 \end{bmatrix}$$

der Ω er området som studeres. Merk at M er antall noder, slik at vi har like mange ligninger som det er ukjente p_i . Av notasjonshensyn er (x,y) -avhengigheten til \vec{N} gitt implisitt.

Ettersom elementene i \vec{p} ikke har noen (x,y) -avhengighet er det ønskelig å få satt \vec{p} utenfor integralet i ligning 6. Vi tar for oss den første ligningen i ligningssettet og observerer at den kan skrives som

$$\begin{aligned} & \int_{\Omega} N_1 \nabla^2 \sum_{i=1}^M N_i p_i d\Omega + \int_{\Omega} k^2 N_1 \sum_{i=1}^M N_i p_i d\Omega \\ &= \sum_{i=1}^M \int_{\Omega} N_1 \nabla^2 (N_i p_i) d\Omega + \sum_{i=1}^M \int_{\Omega} N_1 k^2 N_i p_i d\Omega \\ &= \sum_{i=1}^M p_i \int_{\Omega} \left(N_1 \nabla^2 N_i + N_1 k^2 N_i \right) d\Omega = 0 \end{aligned}$$

Tilsvarende kan gjøres for resten av ligningssystemet, slik at vi får M ligninger på formen

$$\sum_{i=1}^M p_i \int_{\Omega} \left(N_j \nabla^2 N_i + N_j k^2 N_i \right) d\Omega = 0, \quad j \in 1, M \quad (7)$$

2.4 Svak formulering

Dersom basisfunksjonene er lineære ser vi i ligning 7 at ∇^2 -leddet blir 0. For å unngå dette tapet av informasjon ønskes det å omformulere denne ligningen til svak form [4, s. 41]. Svak form innebærer at ∇^2 -leddet erstattes med kun ∇ -ledd.

Fra Green's formel [3, s. 466] har vi at

$$\int_{\Omega} N_j \nabla^2 N_i d\Omega + \int_{\Omega} \nabla N_j \cdot \nabla N_i d\Omega = \oint_{\Gamma} N_j \nabla N_i \cdot \vec{n} d\Gamma$$

der Γ er randen på området Ω og \vec{n} er enhetsnormalvektoren ut fra Γ . $\int_{\Omega} N_j \nabla^2 N_i d\Omega$ -leddet i ligning 7 kan dermed skrives som

$$\int_{\Omega} N_j \nabla^2 N_i d\Omega = \oint_{\Gamma} N_j \nabla N_i \cdot \vec{n} d\Gamma - \int_{\Omega} \nabla N_j \cdot \nabla N_i d\Omega$$

og ligning 7 kan skrives som

$$\sum_{i=1}^M p_i \left(\oint_{\Gamma} N_j \nabla N_i \cdot \vec{n} d\Gamma - \int_{\Omega} \nabla N_j \cdot \nabla N_i d\Omega + \int_{\Omega} N_j k^2 N_i d\Omega \right) = 0$$

Vi får med det ligningssettet

$$\left[\begin{array}{l} \sum_{i=1}^M p_i \left(\oint_{\Gamma} N_1 \nabla N_i \cdot \vec{n} d\Gamma - \int_{\Omega} \nabla N_1 \cdot \nabla N_i d\Omega + \int_{\Omega} N_1 k^2 N_i d\Omega \right) = 0 \\ \sum_{i=1}^M p_i \left(\oint_{\Gamma} N_2 \nabla N_i \cdot \vec{n} d\Gamma - \int_{\Omega} \nabla N_2 \cdot \nabla N_i d\Omega + \int_{\Omega} N_2 k^2 N_i d\Omega \right) = 0 \\ \vdots \\ \sum_{i=1}^M p_i \left(\oint_{\Gamma} N_M \nabla N_i \cdot \vec{n} d\Gamma - \int_{\Omega} \nabla N_M \cdot \nabla N_i d\Omega + \int_{\Omega} N_M k^2 N_i d\Omega \right) = 0 \end{array} \right] \quad (8)$$

som er settet av ligning 7 på svak form.

2.5 Grensebetingelser

For å beskrive veggens innvirkning på lydfeltet i rommet benyttes impedanseverdiene til veggene. Merk at vegg i denne betydningen også kan være et gulv

eller tak. Veggenes spesifikke akustiske impedans z er fra Kinsler et al. [2, s. 286] gitt ved

$$z = \frac{p}{u}$$

der p og u er lydtrykk og partikkelfart ved veggoverflaten. Dette tar da form som grensebetingelser, og vil bli implementert gjennom integralene over randen Γ i ligning 8.

Fra Kinsler et al. [2, s. 119] har vi Eulers ligning

$$\rho_0 \frac{\partial \vec{u}}{\partial t} = -\nabla p \quad (9)$$

der ρ_0 er massetettheten til det akustiske mediet, her luft. Antakelsen om harmonisk tidsvariasjon for p i kapittel 2.1 gjøres også for u , slik at $\frac{\partial \vec{u}}{\partial t} = j\omega \vec{u}$. Ligning 9 tar da formen

$$j\omega \rho_0 \vec{u} = -\nabla p \quad (10)$$

slik at

$$\vec{u} = -\frac{\nabla p}{j\omega \rho_0} \quad (11)$$

I denne oppgaven gjøres antakelsen om at alle overflater er såkalt lokalt reaktive, hvilket vil si at normalkomponenten av hastigheten, u_n , i ethvert punkt langs veggene kun er avhengig av trykket ved det samme punktet [5, s. 44]. Man får da fra Kuttruff [5, s. 44] at impedansen er uavhengig av lydets innfallsvinkel.

For å implementere impedansene i kurveintegralet i ligning 8 er det ønskelig å få med et prikkprodukt med enhetsnormalvektoren til kurven det integreres over. Dette oppnår vi ved å bruke uttrykket for veggimpedanser gitt for normalt innfallende lydbølger. Vi får da impedansen gitt som $z = z_n = \frac{p}{u_n}$, der n angir normalkomponent. u_n finnes ved å skalarmultiplisere med enhetsnormalvektoren \vec{n} i ligning 11, og vi får

$$u_n = \vec{u} \cdot \vec{n} = -\frac{\nabla p \cdot \vec{n}}{j\omega \rho_0} \implies z = \frac{p}{u_n} = -\frac{j\omega \rho_0 p}{\nabla p \cdot \vec{n}} \quad (12)$$

Merk at antakelsen om lokalt reaktive flater kan være langt fra korrekt [6, s. 110]. Antakelsen gjøres likevel, for enkel implementasjon av impedanser i modellen.

Ved å innføre tilnærmingen til p fra ligning 4 i ligning 12 får vi

$$\begin{aligned}
 z &= -\frac{j\omega\rho_0(\vec{N} \cdot \vec{p})}{\nabla(\vec{N} \cdot \vec{p}) \cdot \vec{n}} = -j\omega\rho_0 \frac{\sum_{i=1}^M N_i p_i}{\nabla(\sum_{i=1}^M N_i p_i) \cdot \vec{n}} \\
 \implies \sum_{i=1}^M (p_i \nabla N_i \cdot \vec{n}) &= -\frac{(\sum_{i=1}^M N_i p_i) j\omega\rho_0}{z}
 \end{aligned} \tag{13}$$

og innsetting av ligning 13 i ligning 8 gir det endelige ligningssettet

$$\left[\begin{array}{l}
 \sum_{i=1}^M p_i \left(\oint_{\Gamma} j\omega\rho_0 \frac{N_1 N_i}{z} d\Gamma + \int_{\Omega} \nabla N_1 \cdot \nabla N_i d\Omega - \int_{\Omega} N_1 k^2 N_i d\Omega \right) = 0 \\
 \sum_{i=1}^M p_i \left(\oint_{\Gamma} j\omega\rho_0 \frac{N_2 N_i}{z} d\Gamma + \int_{\Omega} \nabla N_2 \cdot \nabla N_i d\Omega - \int_{\Omega} N_2 k^2 N_i d\Omega \right) = 0 \\
 \vdots \\
 \sum_{i=1}^M p_i \left(\oint_{\Gamma} j\omega\rho_0 \frac{N_M N_i}{z} d\Gamma + \int_{\Omega} \nabla N_M \cdot \nabla N_i d\Omega - \int_{\Omega} N_M k^2 N_i d\Omega \right) = 0
 \end{array} \right] \tag{14}$$

som kan løses direkte. Dette er altså ligningssettet som skal løses med FEM. Merk at forskjellige impedanser på forskjellige veggseksjoner kan implementeres ved å la z være stykkevis konstant langs randen og dele opp integralene over Γ tilsvarende.

2.6 Kombinasjon av løsninger i 2D og 1D

I rom der gulv og tak er parallelle, alle vegger står normalt på gulv- og takplanet og alle flater er harde kan modeformene i gulvplanet og i z -retning kombineres til rommoder uten å innføre noen feil [2, s. 249]. Vi får da lydtrykksfordelingen i rommet gitt som

$$p(x,y,z) = XY(x,y) \cdot Z(z) \tag{15}$$

der $XY(x,y)$ og $Z(z)$ angir løsning i henholdsvis xy -planet og i z -retning. I denne oppgaven benyttes dette uttrykket også ved rom med ikke-harde flater, altså flater med endelig impedans. Feilen dette innfører vil bli undersøkt.

Fra Kuttruff [5, s. 83] har vi følgende uttrykk for lydtrykket som funksjon av frekvens og mottakerposisjon i et rom gitt av rommets egenfrekvenser, dempingskonstanter og egenmoder:

$$p(x,y,z,\omega) = \sum_{mn} \frac{A_{mn}(x,y,z,\omega)}{\omega^2 - \omega_{mn}^2 - 2j\delta_{mn}\omega_{mn}} \quad (16)$$

ω_{mn} er vinkelhastigheten ved egenfrekvens nummer mn og $\delta_{mn} = \frac{cA}{8V}$ er dempingskonstant gitt av rommets volum V og absorpsjonsareal A ved samme egenfrekvens [5, s. 131]. Absorpsjonsarealet A forklares i kapittel 2.7. Merk for øvrig at A_{mn} og A er to helt forskjellige variabler. A_{mn} er en funksjon av egenmode mn , ψ_{mn} , samt et ledd som beskriver kildens frekvensavhengighet. Uttrykket bygger på en antakelse om at dempingskonstantene er små, altså at impedansene er store.

A_{mn} har form i rommet som

$$A_{mn}(x,y,z,\omega) \propto \psi_{mn}(x,y,z) \cdot \psi_{mn}(x_0,y_0,z_0)$$

der (x_0,y_0,z_0) er posisjonen til en akustisk punktkilde og (x,y,z) er posisjonen lydtrykket beregnes i. Med separeringen i ligning 15 får vi så at $\psi_{mn}(x,y,z) = \psi_{h,m}(x,y) \cdot \psi_{z,n}(z)$, der $\psi_{h,m}$ og $\psi_{z,n}$ er egenmode nummer m og n i henholdsvis horisontalplanet og z -retning.

Fra Pierce [6, s. 287] har vi også at egenmodene må normaliseres slik at

$$\int_{romvolum} \psi_{mn}^2 dV = \int_{gulvareal} \psi_{h,m}^2 dS \cdot \int_z \psi_{z,n}^2 dz = 1$$

Relativ lydtrykkfordeling i rommet kan altså beregnes hvis rommets egenfrekvenser og -moder, samt dempingskonstantene ved disse, er kjent.

Rommets egenverdier $k_{mn} = \frac{\omega_{mn}}{c}$ kan beregnes ut fra egenverdiene i horisontalplanet og i z -retning, og er gitt som

$$\begin{aligned} k_{mn} &= \sqrt{k_{h,m}^2 + k_{z,n}^2} \\ \implies f_{mn} &= \sqrt{f_{h,m}^2 + f_{z,n}^2} \end{aligned} \quad (17)$$

der $k_{h,m}$ og $k_{z,n}$ er egenverdi m og n i henholdsvis horisontalplan og vertikalretning, og $f_{h,m}$ og $f_{z,n}$ er de tilsvarende egenfrekvensene [2, s. 249]. $k_{h,m}$ og $k_{z,n}$, samt modeformene ved disse, kan finnes ved FEM-simulering.

2.7 Veggimpedanser

Begrepet veggimpedans betegner i denne oppgaven det samme som flateimpedans, altså impedans på enten vegg, gulv eller tak. For beskrivelse av vegger, gulv og tak benyttes metoden beskrevet i [7]. I denne artikkelen brukes det en enkel masse-fjær-demping-modell for veggimpedans. Impedansen til en vegg blir da

$$z(\omega) = R + j \left(\omega m - \frac{\rho c^2}{\omega d} \right) \quad (18)$$

der R , m og d er veggens resistans-, masse- og luftsjikt-komponent, og der luftsjikt-komponenten representerer fjær-leddet. R , m og d har henholdsvis enhetene $[\text{kg}/(\text{m}^2\text{s})]$, $[\text{kg}/\text{m}^2]$ og $[\text{m}]$. Det vises i artikkelen at en slik modell samsvarer godt med faktisk absorpsjonskoeffisient og faseskift til veggen for frekvenser opp til rundt 500 Hz, slik at modellen er god i frekvensområdet som er aktuelt for denne oppgaven. For en grundigere innføring i modellen vises det til nevnte artikkel.

For beregning av absorpsjonsarealet A til hver overflate i rommet har vi fra Kinsler et al. [2, s. 337] at $A = \sum_i \alpha_i \cdot S_i$, der α_i og S_i er absorpsjonskoeffisient og areal til overflate i .

Fra Kinsler et al. [2, s. 339] har vi at absorpsjonskoeffisienten er gitt ved

$$\alpha = 1 - R_{\Pi i}$$

der $R_{\Pi i}$ er effektrefleksjonskoeffisienten til flate i , som sier hvor stor del av innfallende lydeffekt som reflekteres. Fra Kinsler et al. [2, s. 150 og 161] har vi videre at

$$R_{\Pi i} = \left| \frac{z_i - r_1}{z_i + r_1} \right|^2$$

der r_1 er impedansen i luft og z_i er impedansen til flate i , slik at

$$A = \sum_i \alpha_i \cdot S_i = \sum_i \left(1 - \left| \frac{z_i - r_1}{z_i + r_1} \right|^2 \right) \cdot S_i \quad (19)$$

3 Simuleringsverktøy og skript

3.1 Simuleringsverktøyet FreeFem++

FreeFem++ er, som navnet spiller på, et fritt tilgjengelig verktøy for å løse differensialligninger ved hjelp av FEM. Det kan betraktes som et programmeringsspråk med innebygde funksjoner for blant annet løsning av differensialligninger og for plotting. FreeFem++ er forholdsvis likt C++, da det er skrevet i og bygger på dette språket.

Beskrivelse av problemet som skal løses gjøres i et skript, der blant annet form på området som skal simuleres, mesh-størrelse og selve differensialligningen på svak form, beskrevet i kapittel 2.4, skrives inn. I formuleringen av differensialligningen inngår også grensebetingelsene gitt som impedanser på vegger, gulv og tak, som beskrevet i kapittel 2.5.

For valget av simuleringsverktøy ble FEM-programmene listet opp i [8] kort evaluert. FreeFem++ ble valgt grunnet blant annet at det ikke var knyttet til noe annet, bestemt fagfelt, på grunn av dets slektskap med C++ og fordi det kjørte på Windows. Det ble senere gjort grundigere undersøkelser av programmet, og det ble konkludert med at det hadde de nødvendige funksjonene.

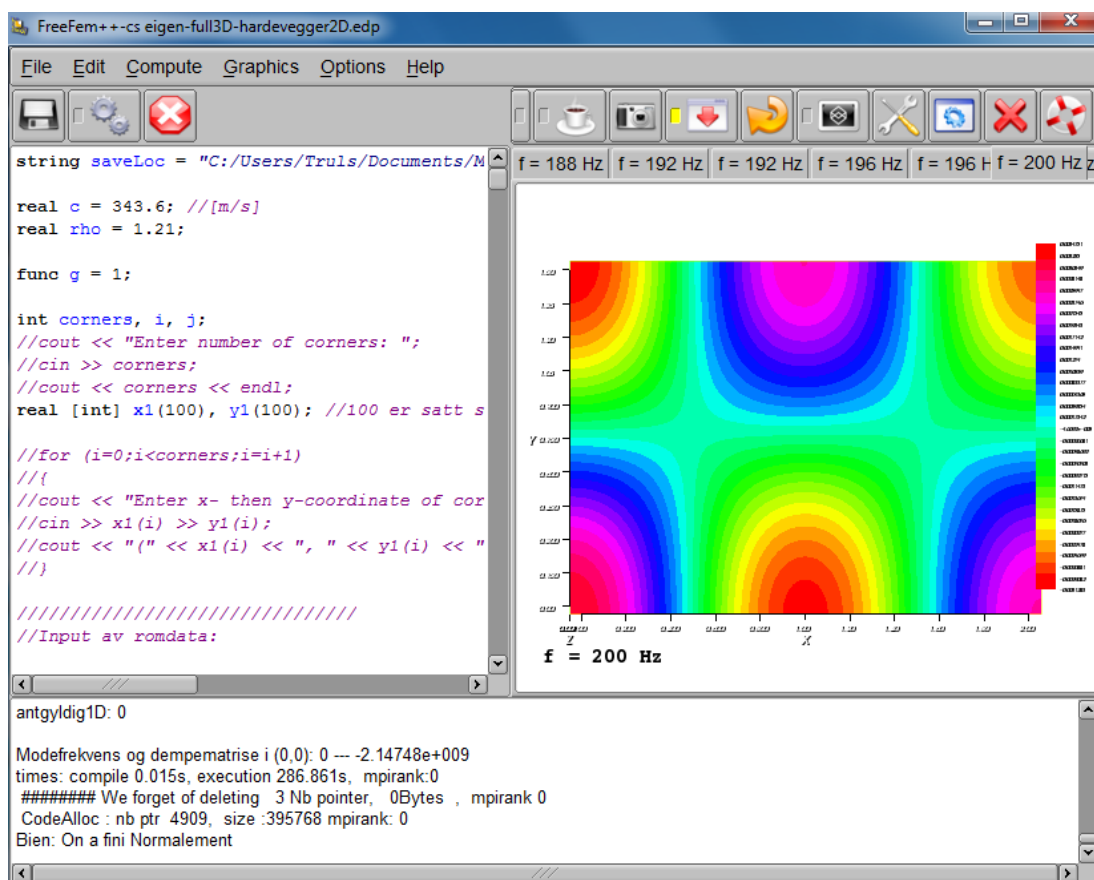
I denne oppgaven brukes FreeFem++-cs, som fungerer som et skall på FreeFem++ og gir en viss grad av grafisk brukergrensesnitt. Brukergrensesnittet i FreeFem++-cs er som vist i figur 3.1, og er tredelt med skript- og plot-vindu øverst, og informasjon om kjøring av skriptet i vinduet nederst.

For detaljer om virkemåte og anvendelse av FreeFem++ og FreeFem++-cs vises det til programmets brukermanual [1].

Skriptene som er skrevet for kjøring i FreeFem++-cs vil videre i denne oppgaven bli referert til som FreeFem++-skript, ikke som FreeFem++-cs-skript. Dette både fordi selve språket skriptene er skrevet i er FreeFem++ og for å få marginalt enklere notasjon.

3.2 Om den fysiske modellen

Den fysiske modellen som simuleres er begrenset til å kunne brukes kun på rom formet som rettvinklede prizmer, altså rom der alle vegger er vertikale og gulv og tak er flate og står normalt på veggene. Videre kreves det at veggene ikke endrer egenskaper i z-retning. Disse begrensningene settes for at kombinasjonen



Figur 3.1: Brukergrensesnitt i FreeFem++-cs, plottet til høyre er lydtrykksfordelingen i horisontalplanet i et rom ved 200 Hz

av løsning i horisontalplan og z -retning skal være mulig, en grundigere forklaring av dette er å finne i kapittel 2.6. Dette bør ikke legge store begrensninger på anvendelse av simuleringverktøyet, da de fleste rom følger disse kriteriene.

Impedanser på vegger, gulv og tak er implementert etter modellen beskrevet i kapittel 2.7, altså gitt ved parameterne resistans, masse per areal og tykkelse på et eventuelt luftsjikt i absorbenten.

3.3 Om beregningsmodellene

Det er laget to forskjellige modeller i FreeFem++ for beregning av lydtrykk, skrevet i hvert sitt skript. Begge modellene tar utgangspunkt i separate simuleringer i xy -planet og z -retning, videre i oppgaven også referert til som henholdsvis 2D-

og 1D-simulering. I 2D-simuleringen settes rommets veggimpedanser som grensebetingelser, i 1D-simuleringen settes rommets gulv- og takimpedans som de to grensebetingelsene.

Den første modellen er relativt enkel. Det simuleres med en omnidireksjonell punktkilde i henholdsvis xy -planet og z -retning, og denne kilden sender lyd ved en og en frekvens. Effekten kilden stråler ut er ikke kjent. Lydtrykket i 2D og 1D finnes ut fra dette som funksjon av frekvens. Den store svakheten med denne modellen er at løsningene i 2D og 1D ikke kan kombineres riktig, da de kombinerte egenfrekvensene beskrevet i ligning 17, kapittel 2.6, med både m og $n \neq 0$, ikke inkluderes. Denne modellen er likevel laget, for å fungere som backup dersom den adskillig mer kompliserte andre modellen skulle vise seg ikke å fungere tilfredsstillende.

Det er verdt å merke seg at en full 3D-simulering på form som beskrevet i forrige avsnitt, altså med kilde simulert i 3D-rom, ville gitt ønsket resultat, men 3D-simulering er antatt å være beregningsmessig for tungt. Det er for så vidt heller ikke trivielt å implementere 3D-simulering i FreeFem++, da dette programmet ikke kan generere 3D-mesh.

Den andre modellen er basert på beregning av egenfrekvenser og -moder i horisontalplanet og i z -retning. Egenfrekvensene kombineres som beskrevet i kapittel 2.6 og gir dermed lydtrykket i rommet som funksjon av frekvens. Svakheten med denne modellen er at dempingen i veggene ikke kan implementeres ved beregning av egenfrekvensene. Dette skyldes at egenfrekvensløseren i FreeFem++ ikke godtar tap av energi. Dermed inngår dempingen kun gjennom totaldempingsleddet i ligning 16. Feilen dette gir vil bli studert ved sammenligning med COMSOL, som ikke har nevnte begrensning i egenfrekvensløseren. I så måte kan den første modellen virke komplimenterende, da denne støtter forskjellig demping på veggene og kan si noe om hvordan lydtrykket blir i nærheten av ulikt dempede vegger.

3.4 Generelt om skriptene

I de følgende delkapitler forklares de viktigste sidene ved skriptene. Skriptene i sin helhet er lagt i vedlegg D. For en gjennomgang av anvendelse av simuleringsverktøyene vises det til brukermanualen i vedlegg B.

Med tanke på praktisk anvendelse av simuleringsverktøyet er skriptet som er basert på beregning av egenfrekvenser og -moder laget i to utgaver. I den ene hardkodes alle variabler inn i selve skriptet, mens variablene i den andre bestemmes gjennom dialogbokser under kjøring av programmet. Utgaven med dialogbokser er naturligvis enklere å bruke, da en ikke behøver å endre på selve skriptet.

Dessverre mister brukeren da muligheten til å bestemme antall hjørner og vegger i rommet, ettersom dette er en parameter som må hardkodes i skriptet. Dette skyldes meshgenereringen og ligningsformuleringen, som dessverre må kodes for et bestemt antall vegger. I tillegg må en ved denne løsningen skrive inn alle variablene hver gang skriptet kjøres.

Begge utgaver av dette skriptet er vedlagt, men i testingen som gjøres i denne oppgaven brukes utgaven der bruker-input hardkodes.

Overordnet struktur er lik i alle skriptene, og er som følger:

1. Brukerbestemte parametre som veggegenskaper og hjørneposisjoner, enten hardkodet i skriptet eller innhentet gjennom dialogbokser
2. Simuleringsparametre som meshstørrelse og frekvensoppløsning
3. Initialisering av øvrige variabler for simuleringen
4. For-løkke der impedanser beregnes og ligningene løses
5. Eventuell behandling av resultater, samt skriving av data til tekstfiler

Merk at alle tall i skriptene er i SI-enheter.

Skriptene skal generere en transferfunksjon mellom kilde og mottaker og plot av lydtrykkfordelingen, videre kalt modeplot, ved forskjellige frekvenser.

I det følgende gjennomgås de forskjellige delene av skriptene. For fullstendig oversikt vises det til selve skriptene i vedlegg D, leseren bør i så fall ha noe kjennskap til C++.

Til slutt kan det nevnes at selv om FreeFem++ har noe støtte for parallellberegninger, altså bruk av flere prosessorkjerner samtidig, er dette ikke forsøkt implementert. Skriptene er basert på en for-løkke der de numeriske beregningene gjøres, og det burde være mulig å gjøre parallellberegninger her ved å kjøre for eksempel 2 eller 4 iterasjoner parallelt. Det kan dermed antas at en vesentlig tidsbesparelse i forhold til hva som oppnås i denne oppgaven er realistisk.

3.5 2D-simulering med kilde

Videre i oppgaven vil denne modellen også refereres til som kildemodellen i FreeFem++.

3.5.1 Parametre, variabler og konstanter

I begynnelsen av skriptet defineres koordinatene til kilden, mottakeren og rommets hjørner, og takhøyden bestemmes. Deretter bestemmes resistans-, masse- og sjikttykkelseverdiene for gulv, tak og alle vegger. Så settes parameterne for selve simuleringen, altså frekvensoppløsningen for transferfunksjonen, frekvensoppløsning for modeplot, frekvensområde som skal studeres og plassering for datafilene som skrives ut av programmet. En boolsk variabel, kalt `printResults`, som angir om simuleringsresultatene skal eksporteres til eksterne filer ligger også her.

Variabler og konstanter som vanligvis ikke behøver å bestemmes av bruker, altså lydfart, massetetthet i luft og grovhet i mesh, defineres til slutt. Grovhet i mesh er i skriptet gitt som en funksjon av høyeste frekvens det simuleres for, slik at en bruker ikke behøver å endre på denne selv om høyeste simuleringsfrekvens økes.

3.5.2 Mesh-generering

Mesh-generering i FreeFem++ gjøres ved at meshets rand defineres som en eller flere kurver. I skriptet lages derfor linjer mellom hjørnene i rommet, og det lages mesh fra dette både for 2D- og 1D-løsning. 1D-løsningen beregnes i FreeFem++ ved 2D-simulering, med to harde, parallelle vegger som står 10 cm unna hverandre, og gulv og tak med tilhørende impedanser. Så lenge avstanden mellom veggene er mye mindre enn $\lambda/2$, der λ er bølgelengden, vil en få stående bølger kun i z-retning, og simuleringen i 2D gir kun 1D-løsningen.

Finheten i begge mesh bestemmes av antall noder på hver linje i meshets ytterkant, og er her definert med et visst antall noder pr bølgelengde ved høyeste simulerte frekvens. Parameteren `meshfact` i skriptet angir dette. Verdien på `meshfact` er bestemt gjennom simuleringene beskrevet i kapittel 4.2.1.

I tillegg inkluderes punktkilden i meshet, som en liten sirkel med sentrum i gitt kildeposisjon. Radius på kilden er satt til 5 mm. Etersom kilden dermed får en liten utstrekning må den plasseres mer enn 5 mm ut fra alle rommets vegger. Merk at mottakeren ikke finnes fysisk i modellen, og dermed heller ikke har noen utstrekning eller plasseringsbegrensninger.

3.5.3 For-løkke med løsning av bølgeligningen

I denne løkken gjøres løsningen av problemet for en frekvens per iterasjon. Problemet er her bølgeligningen, som løses med hensyn på trykket p . Impedansene

for vegger, gulv og tak beregnes for hver gjennomkjøring, da disse er frekvensavhengige.

Selve bølgeligningen skrives inn på svak form, omtrent som formulert i ligning 14, bortsett fra at summingen er gjort implisitt og at p_i og N_i er kombinert til variabelen pN i skriptet. FreeFem++ løser så ligningen ved FEM slik det grovt beskrives i kapittel 2.3. Eksakt hvordan dette gjøres i FreeFem++ er usikkert, da brukermanualen [1] ikke er funnet å beskrive dette i detalj.

3.5.4 Behandling og eksportering av simuleringsresultater

I enkelte av iterasjonene av for-løkken, gitt av frekvensoppløsningen på modeplottene, plottes lydtrykket i horisontalplanet og i z-retning. Avhengig av verdien på `printResults`-variabelen skrives disse plottene ut som `.jpg`-filer.

Transferfunksjon genereres ved at lydtrykket i mottakerposisjonen lagres for hver iterasjon av for-løkken. Avhengig av verdien på variabelen `printResults` skrives så en liste med trykk og tilhørende frekvens ut til en `.txt`-fil.

3.6 Modebasert 3D-simulering

Videre i oppgaven vil denne modellen også refereres til som 3D-modellen i FreeFem++. Det minnes om at simuleringen også her gjøres i horisontalplan og z-retning, men ettersom den gir et 3D-resultat omtales den likevel som en 3D-modell eller -simulering.

3.6.1 Parametre, variabler og konstanter

Denne seksjonen av skriptet er i hovedsak lik den i kapittel 3.5, og beskrivelsen gjentas derfor ikke.

3.6.2 Mesh-generering

Også denne seksjonen er i hovedsak lik, bortsett fra at kilden, ettersom den ikke benyttes i selve simuleringen, heller ikke inkluderes i meshet.

3.6.3 For-løkke med løsning av bølgeligningen

Selve løsningen av bølgeligningen er her forskjellig fra løsningen i 3.5, ettersom den ukjente i dette tilfellet ikke er lydtrykket, men egenverdiene $k = \frac{\omega}{c}$ og de tilhørende egenmodene. For å finne disse brukes FreeFem++ sin egen EigenValue-funksjon.

Et problem med denne funksjonen er at den ikke tar høyde for at enkelte komponenter i ligningen, her impedansen z , er avhengig av frekvensen og dermed egenverdien k .

Som en løsning på dette brukes det også i dette skriptet en for-løkke som itererer gjennom frekvensene, og impedansverdier beregnes i hver iterasjon med den gjeldende frekvens. Egenverdier beregnes, og i utgangspunktet regnes egenfrekvensene som har mindre avvik fra senterfrekvensen enn halve frekvenssteget som gyldige. Med senterfrekvens menes frekvensen impedansene i den gitte iterasjonen er beregnet for, og med frekvenssteg menes økningen i senterfrekvens fra en iterasjon til den neste.

Ettersom lydtrykket finnes som en veid sum av alle rommets egenfrekvenser beregnes egenfrekvensene opp til en frekvens 100 Hz over ønsket høyeste frekvens i transferfunksjonen. Det antas her at egenfrekvensene som er mer enn 100 Hz høyere har tilstrekkelig liten innvirkning på lydtrykket gitt i ligning 16 til at de kan sees bort fra. Korrektheten i dette vil bli undersøkt ved sammenligning med simuleringsresultater fra COMSOL.

3.6.4 Behandling av simuleringsresultater

Som en konsekvens av frekvensavhengige impedanser vil de beregnede egenverdiene flytte seg noe i frekvens for hver iterasjon. Dermed vil man med prosedyren beskrevet i forrige avsnitt risikere at enkelte egenverdier ikke registreres som gyldige, eller blir registrert flere ganger. For eksempel kan en med 5 Hz frekvenssteg finne en egenfrekvens på 73 Hz når impedansene er beregnet ved 70 Hz, mens den har flyttet seg til 72 Hz når impedansene er beregnet ved 75 Hz. Ingen av disse funnede egenfrekvensene vil da betegnes som gyldige.

Dette problemet er noe uelegant løst ved å godta egenfrekvenser med litt større avvik fra senterfrekvensen, for deretter å la brukeren studere plot av modene for å vurdere hvilke moder som er registrert flere ganger. Brukeren får med dette også mulighet til også å forkaste moder som ikke nødvendigvis er registrert flere ganger, men som kan være ugyldige.

I simuleringene gjort i denne oppgaven er det antatt at egenfrekvenser > 0 Hz der lydtrykket har samme fortegn overalt er ugyldige. Hvorvidt denne antakelsen er korrekt vites dessverre ikke, men det virker ulogisk at en egenmode med konstant fortegn skal være ved høyere frekvens enn 0 Hz. Derimot regnes 0-moden, altså moden ved 0 Hz som har konstant lydtrykk i hele rommet, automatisk med blant de godkjente modene. Årsaken til dette forklares i kapittel 5.2.1.

1D-situasjonen gir naturligvis samme problem, men ettersom man her kun har moder i en dimensjon fås ikke sammenfallende moder. Dermed kan det sjekkes automatisk om de funnede egenfrekvensene ligger tilstrekkelig langt unna hverandre. I skriptet er differansekravet satt til $\Delta f \geq \frac{c}{4h}$, der h er takhøyden. Kravet om avstand mellom egenfrekvensene settes også i forhold til 0-moden, slik at løsninger i nærheten av 0 Hz, der lydtrykket altså har konstant fortegn, ikke godkjennes.

I henhold til ligning 17 lages det så en matrise med de forskjellige egenfrekvensene til rommet og en tilsvarende matrise med tilhørende dempingskonstanter.

Til slutt beregnes lydtrykket i mottakerposisjon etter ligning 16. For å få trykket som funksjon av frekvens gjøres dette i en for-løkke der frekvensen det løses for øker lineært. De tilhørende modeformene beregnes ved en tilsvarende veid summering av alle godkjente egenmoder.

3.6.5 Eksportering av simuleringsresultater

Transferfunksjon mellom kilde og mottaker og plot av modeformer eksporteres på samme måte som i kildemodellen, altså ved skriving til henholdsvis .txt- og .jpg-filer. I tillegg eksporteres noe metadata, slik som frekvens- og modeplot-oppløsning, til en annen .txt-fil. Disse metadataene benyttes ved visualiseringen av dataene i MATLAB. Matrisen med de godkjente egenfrekvensene skrives til slutt til en egen .txt-fil.

3.7 COMSOL

Simuleringene i COMSOL gjøres hovedsakelig med en 3D-modell der avstanden mellom nodene i meshet er i størrelsesorden lik den som brukes i FreeFem++.

For beregning av transferfunksjon brukes en metode tilsvarende kildemetoden i FreeFem++, altså med en punktkilde som stråler ved en og en frekvens, men det gjøres her i 3D. Kilden sender 1W lydeffekt. I modellen settes også et punkt i mottakerposisjon. Dette eksisterer ikke fysisk i modellen, men brukes for å lese

ut lydtrykket i riktig posisjon. Impedansverdier settes individuelt for hver flate. I enkelte tilfeller er det ønskelig å finne egenfrekvenser også i COMSOL. COMSOL har en egen metode, kalt Eigenfrequency Study, som benyttes for dette både i 2D og 3D.

Videre i oppgaven vil 3D-modellen i COMSOL referere til modellen med punktkilde. Der det er snakk om modellen basert på EigenFrequency Study vil dette bli spesifisert.

Simuleringsparameterne i disse modellene, slik som lyd hastighet og lufttetthet, beholdes som default-verdier. Unntaket er mesh-størrelsen, som settes til å være så lik som mulig den brukt i FreeFem++.

Fullstendige data for en av COMSOL-simuleringene er eksportert som MATLAB-kode, og er vedlagt i vedlegg E. De eneste parameterne som endres fra dette er de som beskriver romgeometri og veggimpedanser, som alle vil bli nevnt.

3.8 Visualisering av simuleringsresultater

Som et eksempel på mulig visualisering av simuleringsresultatene, både transferfunksjon og modeplot, er det laget et MATLAB-skript for dette. Skriptet er vedlagt i vedlegg C.1, og gjennomgås kun kortfattet her.

Skriptet leser inn trykk-, frekvens- og eventuell metadata fra .txt-filene som genereres ved kjøring av FreeFem++-skriptene. I tillegg henter det inn bildene av modeplottene. Skriptet plotter så transferfunksjonen samt stående-bølge-mønster i både 2D og 1D ved en og en frekvens. For hvert bilde med mode i 2D og 1D markeres det i transferfunksjonen hvilken frekvens mønsteret tilhører. Et eksempel på dette er vist i figur 5.12.

4 Simuleringer

I dette kapitlet presenteres simuleringene som er gjort for testing av modellene. Hovedsaklig testes 3D-modellen i FreeFem++ opp mot 3D-modellen i COMSOL, med forskjellige impedansverdier og romgeometrier. I tillegg evalueres meshstørrelse og korrektheten til modeplot kort ved sammenligninger mellom 3D- og kildemodellen i FreeFem++.

For enkelhets skyld er det kun simulert rom med firkantet gulv og tak. Veggene er ikke delt opp ytterligere, slik at hver vegg kun har ett materiale og en impedans.

Simuleringene som beskrives i dette kapitlet er gjort med skriptene i vedlegg D og E, og kun romgeometri, kilde- og mottakerposisjon og impedanser varierer i forhold til disse. Unntaket er simuleringen av rektangulært rom med harde vegger, der meshstørrelsen varieres.

Der annet ikke spesifiseres er modellene kun simulert i 3D, både i FreeFem++ og i COMSOL, for å få ut transferfunksjon mellom kilde og mottaker. I simuleringene i 3D i FreeFem++ brukes konsekvent en frekvensoppløsning på 0,1 Hz ved beregning av egenfrekvensene.

Ved eventuelle 2D-simuleringer ses naturligvis z -verdiene bort fra. I FreeFem++-modellene settes takhøyden da til 10 cm for å unngå stående bølger i z -retning.

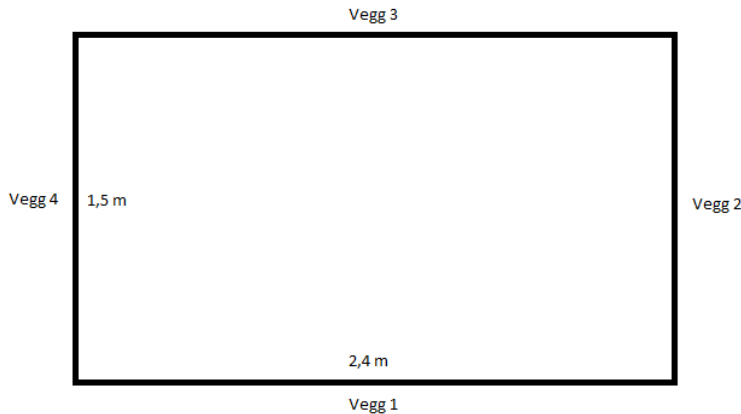
Gulvplanet er i alle simuleringer satt ved $z = 0$.

4.1 CPU-test

Simuleringene i FreeFem++ og COMSOL er kjørt på forskjellige pc-er, og tidsbruken ved de forskjellige simuleringene er dermed ikke direkte sammenlignbar. For likevel å kunne sammenligne tidsbruken ble det gjort en svært enkel cpu-test, der et simpelt men prosessorkrevende matlab-skript ble kjørt på begge pc-er. Merk at dette MATLAB-skriptet bruker flerkjerneteknikk, slik at det tester prosessorene, ikke deres enklelte kjerner. Skriptet er vedlagt i vedlegg C.3.

4.2 Rektangulært rom med konstante veggimpedanser

Det første rommet som simuleres er rektangulært. Det er plassert utelukkende i 1. oktant, altså med $x, y, z \geq 0$, og med ett hjørne i $(0,0,0)$. Rommet sett ovenfra, med veggnummerering, er vist i figur 4.1.



Figur 4.1: Romgeometri og veggnummerering, rektangulært rom

Kilde- og mottakerposisjon, samt romdimensjoner, er i tillegg gitt i tabell 4.1. I alle simuleringer av rektangulært rom der annet ikke er spesifisert brukes disse kilde- og mottakerposisjonene.

Tabell 4.1: Geometri, rektangulært rom

Geometri	x [m]	y [m]	z [m]
Romdimensjoner	2,4	1,5	2
Kildeposisjon	0,01	0,01	0,01
Mottakerposisjon	2,39	1,49	1,99

De følgende simuleringer gjøres med konstante veggimpedanser av hensyn til sammenligning med COMSOL-simuleringer, der simulering med frekvensavhengige impedanser er svært tungvint. FreeFem++-skriptene er derfor kodet om for å håndtere konstante veggimpedanser. Dette er gjort ved at alle linjer der impedansene beregnes på nytt er kommentert ut, og impedansverdiene er bestemt direkte.

4.2.1 Harde vegger, gulv og tak

Den første modellen som er simulert har harde vegger, gulv og tak. Denne modellen vil sammenlignes både med COMSOL og med eksakt løsning, da den analytiske løsningen for egenfrekvensene i denne situasjonen er kjent [2, s. 247]. Beregning av de korrekte egenfrekvensene ut fra ligning 3 er gjort i MATLAB, koden for dette er å finne i vedlegg C.2.

Impedansparameterne er gitt i tabell 4.2, og er satt store nok til å gi en i praksis uendelig impedans. Modellen som er omkodet til å benytte konstante veggimpedanser er ikke brukt her, da de frekvensavhengige impedansene uansett blir tilnærmet uendelig store i dette tilfellet. Merk at også luftsjiktet d er satt til $1e10$ m, for at det aktuelle leddet skal falle bort i beregning av impedansene gitt av ligning 18.

Tabell 4.2: Simuleringsparametre, rom med harde vegger

Impedanser	R [kg/(m ² s)]	m [kg/m ²]	d [m]
Alle flater	$1e10$	$1e10$	$1e10$

Det er først simulert med kildemodellen i FreeFem++, med varierende oppløsning i meshet for å vurdere nødvendig finhet. Oppløsningen er satt til henholdsvis 2, 5, 10 og 25 noder pr bølgelengde ved høyeste simulerte frekvens. I alle videre simuleringer er meshstørrelsen satt i henhold til konklusjonen av disse simuleringene.

Deretter er det gjort 3D-simulering i FreeFem++ og COMSOL for å sammenligne transferfunksjonene disse simuleringene gir.

4.2.2 Store, imaginære impedanser

Det simulerte rommet har her impedanser som gitt i tabell 4.3. Her, og i alle videre simuleringer med konstante impedanser, brukes FreeFem++-skriptet for konstante impedansverdier.

Tabell 4.3: Impedansverdier, rom med store, imaginære impedanser

Impedanser	z [kg/(m ² s)]
Gulv	$7000j$
Tak	$10000j$
Vegg 1	$10000j$
Vegg 2	$8000j$
Vegg 3	$5000j$
Vegg 4	$7000j$

En tilsvarende simulering ble gjort med tilsvarende, negative impedanser på alle flater, altså impedansverdier lik de komplekskonjugerte av verdiene tabell 4.3.

4.2.3 Mindre, imaginære impedanser

Som nevnt i kapittel 2.6 er metoden som brukes for å kombinere 2D- og 1D-løsningene i FreeFem++-modellen utledet for harde vegger. For å vurdere hvor godt denne modellen fungerer med lavere impedanser gjøres en simulering lik den i delkapittel 4.2.2, men der impedansverdiene settes lik 1/10 av verdiene i tabell 4.3.

4.2.4 Store, komplekse impedanser

Det er her lagt en realdel til impedansene i tabell 4.3. Det minnes om at egenverdiløseren i FreeFem++ ikke støtter absorpsjon i veggene, slik at realdelen av impedansen kun inngår ved dempeleddet δ_{mn} i ligning 16. Impedansverdiene for denne simuleringen er gitt i tabell 4.4.

Tabell 4.4: Impedansverdier, rom med store, komplekse impedanser

Impedanser	z [kg/(m ² s)]
Gulv	10000+7000 <i>j</i>
Tak	1000+10000 <i>j</i>
Vegg 1	1000+10000 <i>j</i>
Vegg 2	30000+8000 <i>j</i>
Vegg 3	2000+5000 <i>j</i>
Vegg 4	10000+7000 <i>j</i>

Det gjøres også en simulering av samme rektangulære rom, men der kilde- og mottakerposisjon er henholdsvis (1, 1, 1) og (1, 0,7, 1,5).

4.2.5 Mindre, komplekse impedanser

Tilsvarende som i kapittel 4.2.3 brukes her impedansverdier lik 1/10 av verdiene i tabell 4.4. Simuleringen gjøres likt som i delkapittel 4.2.4.

4.2.6 Én vegg med liten, kompleks impedans

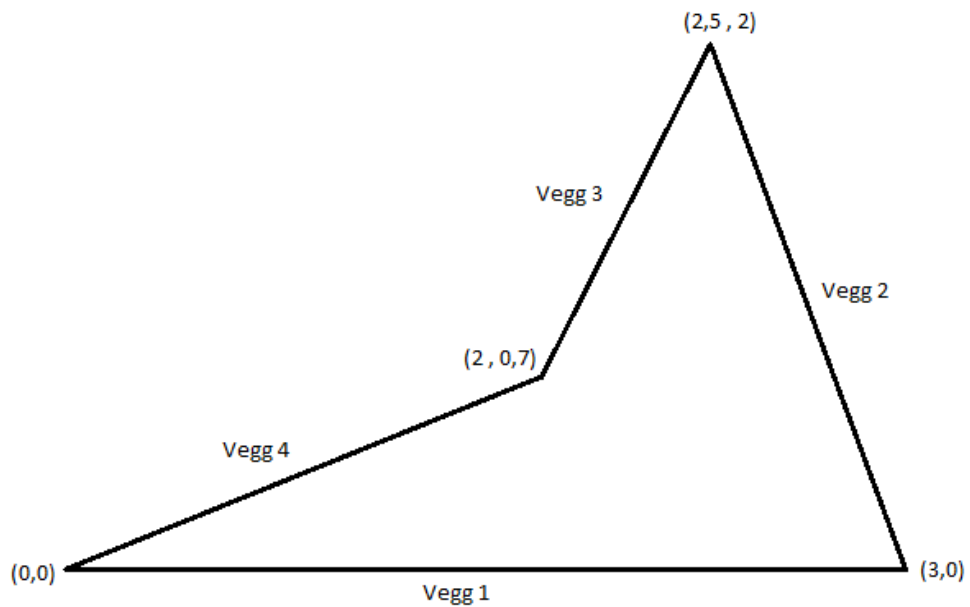
For å undersøke effekten av bare én flate med lav impedans simuleres det rektangulære rommet med impedansverdiene gitt i tabell 4.5.

Tabell 4.5: Impedansverdier, én vegg med liten, kompleks impedans

Impedanser	z [kg/(m ² s)]
Gulv	$1e10j$
Tak	$1e10j$
Vegg 1	$500+1000j$
Vegg 2	$1e10j$
Vegg 3	$1e10j$
Vegg 4	$1e10j$

4.3 Annen romgeometri

Det gjøres også en simulering med annen romform, for å undersøke hvorvidt 3D-modellen i FreeFem++ fungerer like godt på andre geometrier. Impedansverdiene er satt som i tabell 4.4, og form i horisontalplanet er som vist i figur 4.2. Takhøyden er 2 meter, kilde- og mottakerposisjon er gitt i tabell 4.6.



Figur 4.2: Romgeometri og veggnummerering, rom med annen form

4.4 Impedanser gitt ved parameterne R , m og d

Beskrivelse av impedansene ved parameterne R , m og d som beskrevet i kapittel 2.7 er en viktig egenskap ved simuleringsverktøyet. Dette gir frekvensavhengige

Tabell 4.6: Kilde- og mottakerposisjon, rom med annen form

Posisjon	x [m]	y [m]	z [m]
Kilde	0,01	0,01	0,01
Mottaker	2,5	1,99	1,99

impedanser som ifølge Rindel [7] sammenfaller svært godt med impedanseegenskapene til fysiske absorberter.

Grunnen til at dette likevel vektlegges lite i testingen av skriptet er rett og slett at COMSOL ikke er funnet å støtte frekvensavhengige impedanser, hvilket vanskeliggjør sammenligninger. Sammenligning med måledata på eksisterende rom, der impedansparameterne R , m og d for flatene i rommet er kjent, ville selvfølgelig vært mulig, men dette er av tidsmessige årsaker dessverre ikke blitt gjort.

Det gjøres derfor kun én simulering for å undersøke om modellen ser ut til å oppføre seg riktig. Simuleringsresultatet vil vises ved et enkeltbilde fra visualiseringen beskrevet i kapittel 3.8.

Form og kilde- og mottakerposisjon i rommet er gitt i tabell 4.1. Rommet har harde gulv og vegger, og membranabsorbent i taket. Parameterne R , m og d er da gitt som i tabell 4.7, der verdiene for takabsorbenten er hentet fra eksempelet på en membranabsorbent beskrevet i Rindel [7]. For at fjærleddet i ligning 18 skal falle bort er d satt til $1e10$ for membranabsorbenten.

Tabell 4.7: Impedansparametre, rektangulært rom med frekvensavhengige impedanser

Impedanser	R [kg/(m ² s)]	m [kg/m ²]	d [m]
Gulv - hardt	$1e10$	$1e10$	$1e10$
Tak - membranabsorbent	2513	2,139	$1e10$
Vegg 1 - hard	$1e10$	$1e10$	$1e10$
Vegg 2 - hard	$1e10$	$1e10$	$1e10$
Vegg 3 - hard	$1e10$	$1e10$	$1e10$
Vegg 4 - hard	$1e10$	$1e10$	$1e10$

4.5 Sammenligning mellom kilde- og 3D-simulering i FreeFem++

En simulering er blitt gjort både i kilde- og 3D-modellen i FreeFem++ for å vurdere hvordan transferfunksjon og deformasjon påvirkes av ulike demping på veggene.

Det simulerte rommet ligner det i kapittel 4.4, men den absorberende flaten er i stedet satt på en vegg. Geometri inklusive kilde- og mottakerposisjon er satt som i kapittel 4.4 med unntak av at takhøyden, som for å tilnærme 2D er satt til 10 cm. Mottakerens z-koordinat er tilsvarende satt til $z = 0.09$ m.

Rommets impedansparametre er som vist i tabell 4.8.

Tabell 4.8: Impedansparametre, sammenligning mellom kilde- og 3D-simulering i FreeFem++ i rektangulært rom

Impedanser	R [kg/(m²s)]	m [kg/m²]	d [m]
Gulv - hardt	1e10	1e10	1e10
Tak - hardt	1e10	1e10	1e10
Vegg 1 - membranabsorbent	2513	2,139	1e10
Vegg 2 - hard	1e10	1e10	1e10
Vegg 3 - hard	1e10	1e10	1e10
Vegg 4 - hard	1e10	1e10	1e10

5 Resultater

COMSOL- og FreeFem++-simuleringene er ikke kalibrert i forhold til hverandre, og gir i utgangspunktet ut svært forskjellige lydtrykksamplituder. I alle tilfeller der transferfunksjon fra FreeFem++ og COMSOL plottes sammen er derfor nivået på transferfunksjonen fra COMSOL justert til omtrent samme nivå som FreeFem++-transferfunksjonen, for enklere å kunne sammenligne resultatene. I plottene av transferfunksjonene er det kun kurveformene som er interessante, da lydnivået ikke svarer til noen bestemt lydeffekt fra kilden.

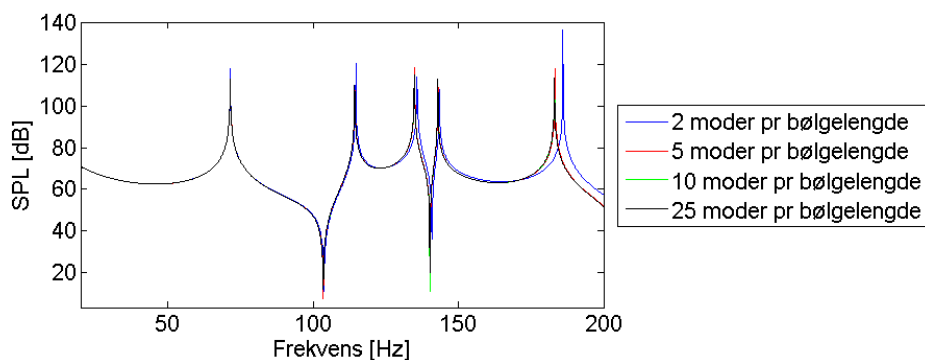
5.1 CPU-test

Kjøringen av testskriptet i vedlegg C.3 i MATLAB tok ca. 8,5 sekunder på FreeFem++-pc-en, mens den tok ca. 25,6 sekunder på COMSOL-pc-en. Ut fra dette antas det derfor at prosessoren i FreeFem++-pc-en er omlag 3 ganger raskere enn den i COMSOL-pc-en. Det minnes dog om at dette er en svært simpel test, og resultatet gir kun en indikasjon på de to prosessorenes relative ytelse.

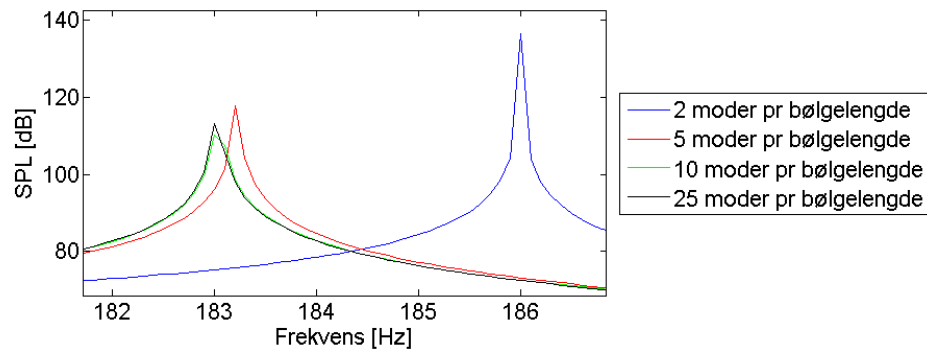
5.2 Rektangulært rom med konstante impedanser

5.2.1 Harde vegger, gulv og tak

Bruk av forskjellige meshstørrelser i kildemodellen i FreeFem++ ga transferfunksjonene i figur 5.1. Et utsnitt av området med de største forskjellene er gitt i figur 5.2. Det simulerte rommet er beskrevet i kapittel 4.2.



Figur 5.1: Transferfunksjoner, FreeFem++ kildemodell med harde vegger, varierende finhet i mesh



Figur 5.2: Utsnitt av transferfunksjonene i figur 5.1, området med størst avvik

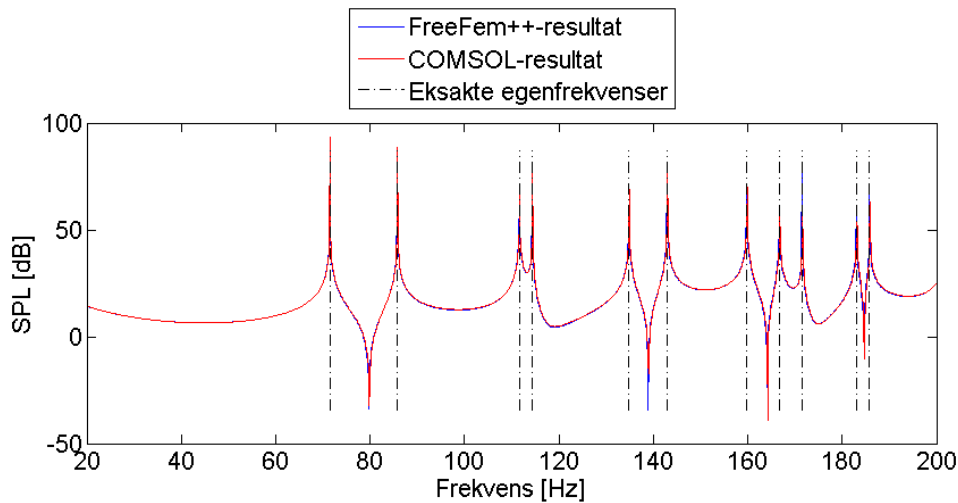
Vi ser at kurvene er tilnærmet like for 10 og 25 noder pr bølgelengde, mens avvik kan sees både ved 5 og spesielt 2 noder pr bølgelengde. I de videre simuleringer vil derfor 10 noder pr bølgelengde benyttes. Dette viser naturligvis ikke at 10 noder pr bølgelengde i alle tilfeller er tilstrekkelig, men med tanke på det relativt lille avviket ved 5 noder per bølgelengde antas det å være godt nok.

Simulering av rommet beskrevet i kapittel 4.2.1 både med modebasert 3D i FreeFem++ og punktkilde i 3D i COMSOL ga transferfunksjonene gitt i figur 5.3. Eksakte egenfrekvenser, funnet analytisk, er også markert i plottet. 0-moden, der lydtrykket er konstant i rommet, er inkludert i FreeFem++-modellen både her og i alle andre simuleringer, da det i dette tilfellet resulterte i en transferfunksjon tilmerket lik den fra COMSOL.

Ettersom alle vegger i rommet er perfekt reflekterende fås naturligvis enorme lydnivåer ved rommets egenfrekvenser, og eksakte nivåer på disse er verken interessante eller sammenlignbare mellom plottene. Vi ser dog at de i frekvens sammenfaller nesten perfekt med de analytisk beregnede egenfrekvensene, og at FreeFem++ og COMSOL gir de samme. I tillegg er lydtrykket utenom egenfrekvensene tilnærmet helt likt mellom simuleringene.

Lydtrykkene beregnet i FreeFem++ og COMSOL er tilnærmet like helt opp til 200 Hz, hvilket viser at det her er tilstrekkelig med egenfrekvenser opp til 300 Hz for beregning etter ligning 16 opp til 200 Hz. Dette overrasker for så vidt ikke, da dempingsleddene i ligning 16 er 0 og burde gjøre at kun de nærmeste egenfrekvenser bidrar til lydtrykket i vesentlig grad.

Kjøretid for FreeFem++-skriptet var ca 220 sekunder med oppløsning i transferfunksjon på 0,1 Hz, og med plot av modeformer for hver tiende iterasjon, altså med en oppløsning på 1 Hz. For COMSOL-simuleringen, med samme oppløsning på transferfunksjon som i FreeFem++, var kjøretiden ca 3500 sekunder. Under



Figur 5.3: Transferfunksjon pluss analytisk løsning, rektangulært rom med harde vegger

antakelsen om at FreeFem++-pcen er omlag 3 ganger raskere enn COMSOL-pcen, gir dette altså en tidsbesparelse på omtrentlig $1 - \frac{3 \cdot 220s}{3500s} \approx 81\%$.

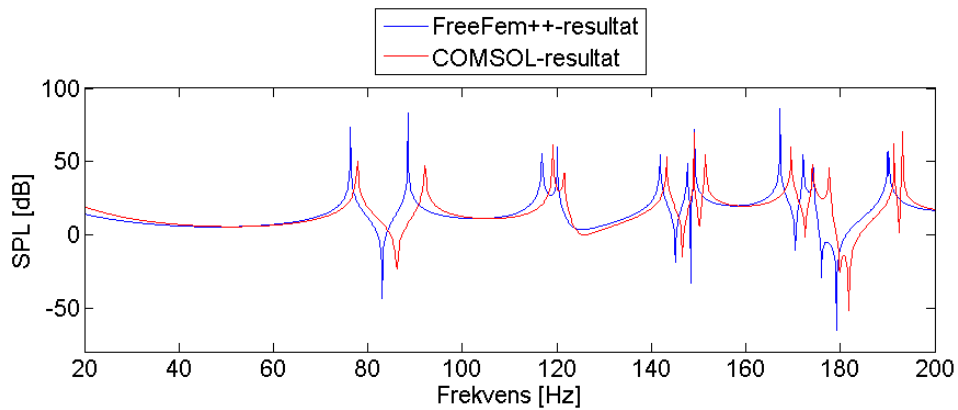
Uten plot av modeformer var kjøretiden i FreeFem++ ca. 84 sekunder. Dersom kun transferfunksjonen er av interesse kan altså en vesentlig større tidsbesparelse, da på $1 - \frac{3 \cdot 84s}{3500s} \approx 93\%$, oppnås.

Det må dog minnes om at disse tallene grunnet den simple cpu-testen er meget usikre.

5.2.2 Store, imaginære impedanser

Simulering av modellen beskrevet i kapittel 4.2.2 i både FreeFem++ og COMSOL ga transferfunksjonene vist i figur 5.4.

Vi ser her et visst avvik mellom kurvene. Avviket er i stor grad i frekvensretning, selve kurveformene er svært like. For å undersøke hvorfor egenfrekvensene er skjøvet noe ned i frekvens i FreeFem++-simuleringen gjøres egenverdisimulering i både 2D og 3D i COMSOL med samme rom- og simuleringsparametre som tidligere. Dette gir egenfrekvensene i tabell 5.1, der også egenverdiene fra FreeFem++-simuleringen er vist:



Figur 5.4: Transferfunksjon, rektangulært rom med konstante, store, imaginære impedanser

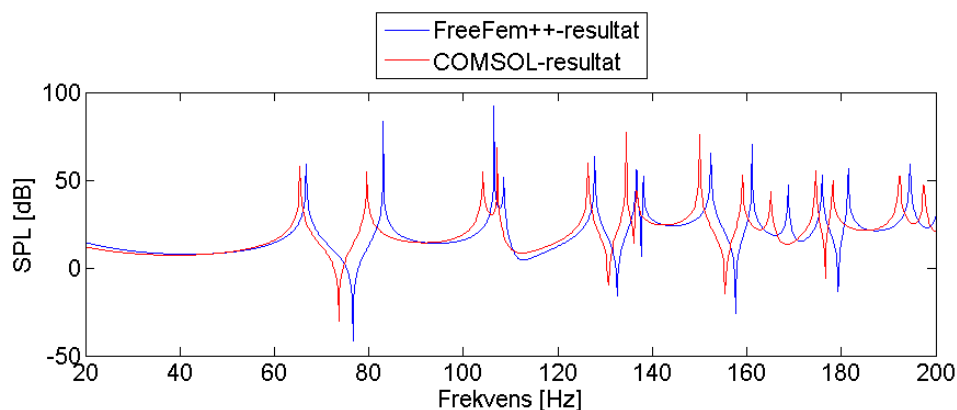
Tabell 5.1: Egenfrekvenser i Hz fra FreeFem++ og fra 2D og 3D COMSOL-simuleringer, rektangulært rom med store, imaginære impedanser

Mode (xy,z)	FreeFem++ 3D	COMSOL 2D	COMSOL 3D
(0,0)	0		9.74
(1,0)	76.3	76,4	77.8
(0,1)	88.5		92.1
(1,1)	116.8		119,2
(2,0)	120.1	120,3	121,5
(3,0)	141.9	142,1	143,3
(4,0)	147.7	147,9	149,1
(2,1)	149.2		154,5
(3,1)	167.2		169,6
(4,1)	172.1		174,2
(0,2)	174,2		177,7
(5,0)	190.0	190,4	191,5
(1,2)	190.2		193,3

Vi ser her at COMSOL finner andre egenfrekvenser ved simulering i 2D enn det finner i 3D, og at modene COMSOL finner i 2D er svært nære de FreeFem++ finner i xy-planet. Dette tyder på at den anvendte formelen for kombinasjon av egenfrekvenser og -moder til 3D-modere ikke er gyldig når veggimpedansene er endelige. Derimot ser FreeFem++-skriptet ut til å finne modene ved riktige frekvenser i xy-planet, og følgelig antakeligvis også i z-retning.

For øvrig kan det påpekes at COMSOL sin 0-mode ikke er ved 0 Hz, men litt høyere. Hva dette skyldes er usikkert, men det antas å være galt. I 3D-modellen i FreeFem++ beholdes 0-moden ved 0 Hz.

Med negative impedanser blir transferfunksjonene som vist i figur 5.5.



Figur 5.5: Transferfunksjon, rektangulært rom med konstante, store, negative, imaginære impedanser

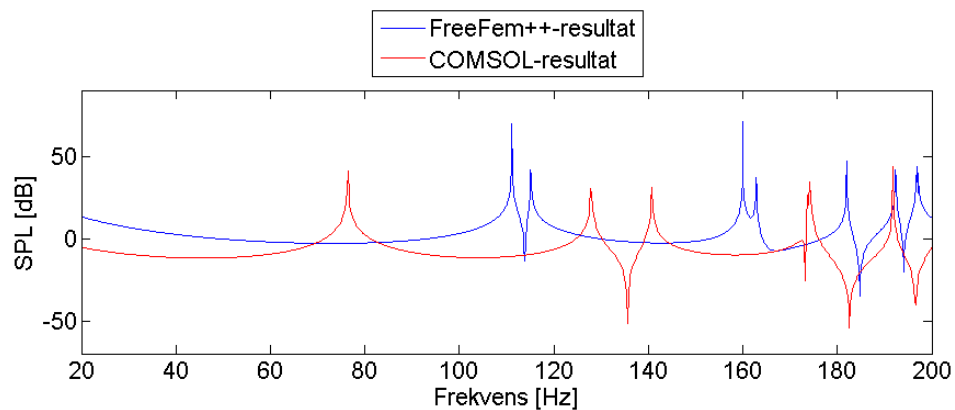
Her er avviket motsatt, og FreeFem++ gir ut en transferfunksjon som er forskjøvet noe oppover i frekvens i forhold til COMSOL.

Ved sammenligning med simuleringen med harde vegger, figur 5.3, ser vi at FreeFem++-simuleringen i begge tilfeller gir mindre frekvensforskyving enn det COMSOL-simuleringen gjør. Det antas at avviket også her skyldes kombinasjonen av 1D- og 2D-løsningene.

5.2.3 Mindre, imaginære impedanser

Med impedansene på alle flater redusert med 90 %, slik det beskrives i kapittel 4.2.3, ble transferfunksjonen som vist i figur 5.6.

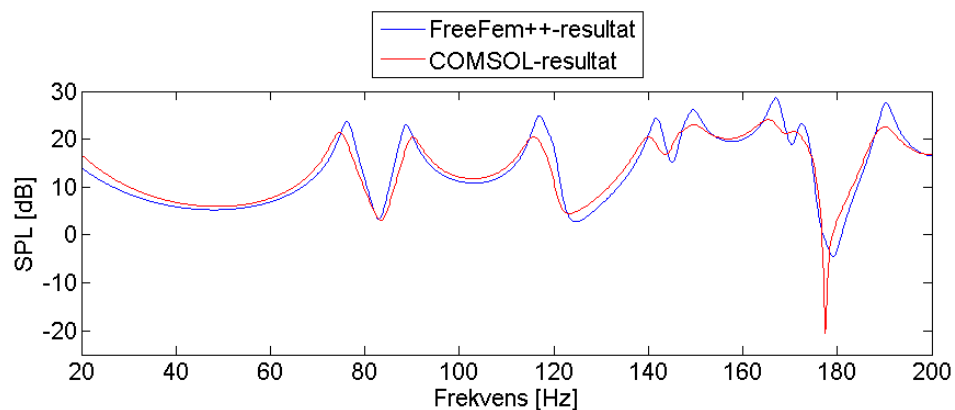
Her er det svært få likheter mellom FreeFem++- og COMSOL-simuleringen å spore. Med tanke på resultatene vist i tabell 5.1, spesielt at 2D-simuleringen i COMSOL ga egenfrekvenser svært nære dem fra FreeFem++-simuleringen, ser det ut til at selve modellen med kombinasjon av egenfrekvenser svikter fullstendig ved lave impedanser. Dersom en legger godviljen til kan det også se ut som at FreeFem++-resultatet er forskjøvet langt nedover i frekvens i forhold til COMSOL-resultatet, gitt at den dypeste toppen fra COMSOL er 0-moden forskjøvet langt oppover.



Figur 5.6: Transferfunksjon, konstante, imaginære impedanser

5.2.4 Store, komplekse impedanser

Med impedanser som beskrevet i kapittel 4.2.4 fås transferfunksjonene vist i figur 5.7.

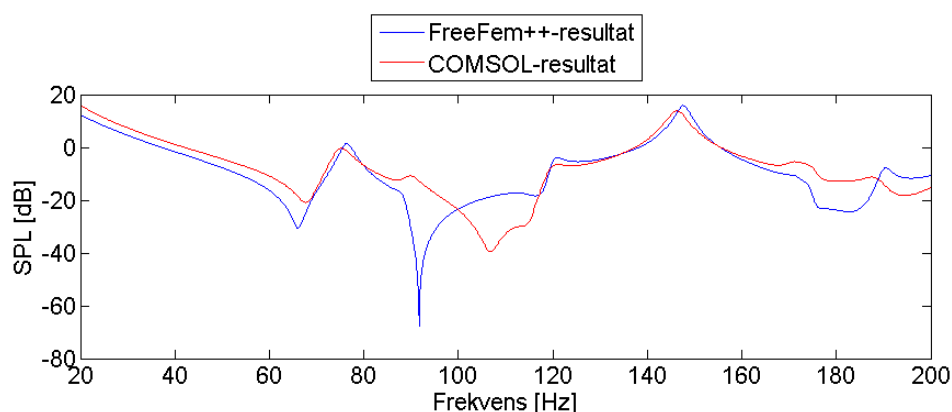


Figur 5.7: Transferfunksjon, konstante, store, komplekse impedanser

Vi ser at frekvensavviket på egenfrekvensene faktisk blir mindre her enn i den tilsvarende simuleringen uten demping i veggene, altså i figur 5.4. Hva dette skyldes vites ikke sikkert, men det antas at egenfrekvensene i rommet blir lavere grunnet dempingen. Ellers ser vi at FreeFem++ gir vesentlig større peaks enn COMSOL, spesielt ved høyere frekvenser. Det tyder altså på at dempingen ikke vektlegges nok i FreeFem++-modellen, ettersom demping vil gi bredere, mindre skarpe resonanser. Det er i COMSOL-modellen valgt ikke å implementere luftabsorpsjon,

slik at årsaken må være en annen. Det kan tenkes at bruken av impedanser for normalt lydinnfall i ligningsformuleringen i FreeFem++-modellen gir denne forskjellen fra COMSOL, men dette vites ikke.

Med endrede kilde- og mottakerposisjoner ble transferfunksjonen som vist i figur 5.8.



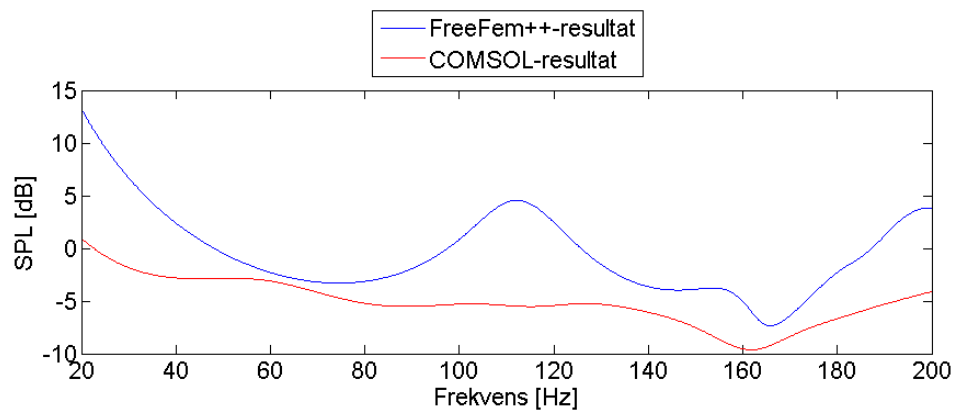
Figur 5.8: Transferfunksjon, konstante, store, komplekse impedanser - kilde og mottaker plassert langt fra hjørnene

Kilde og mottaker plassert i nye posisjoner, langt fra rommets hjørner, resulterer her i at mange av rommets egenfrekvenser eksiteres og oppfanges i vesentlig mindre grad. Som konsekvens av dette kan det også observeres at lydnivået her er langt under det i figur 5.7. Kurvene fra FreeFem++ og COMSOL-simuleringen er i store deler svært like, men har enkelte meget store avvik, for eksempel mellom 90 og 115 Hz. Ved 90 Hz gir faktisk COMSOL en peak, mens FreeFem++ gir en meget markant dip. Hva dette skyldes er dessverre uvisst.

5.2.5 Mindre, komplekse impedanser

Med impedansene på alle flater redusert med 90 % fra modellen i kapittel 4.2.4, og kilde og mottaker tilbake i hjørneposisjonene, ble transferfunksjonen som vist i figur 5.9.

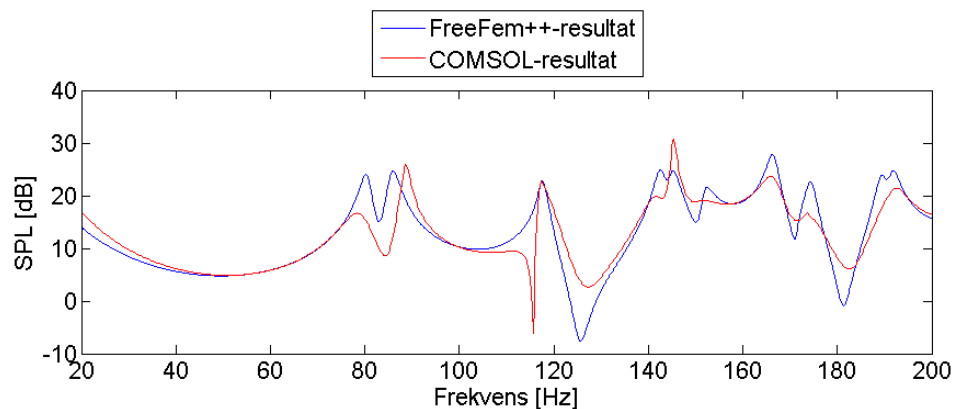
Igen ser vi at COMSOL gir vesentlig flatere transferfunksjon enn FreeFem++, altså at dempingen har mindre innflytelse i simuleringen i FreeFem++. Utover dette er det vanskelig å avgjøre om likhetene mellom kurvene, slik som rundt 160 Hz, er tilfeldige, eller om FreeFem++-modellen faktisk gir bedre resultat med demping på veggene.



Figur 5.9: Transferfunksjon, konstante, mindre, komplekse impedanser

5.2.6 Én vegg med liten, kompleks impedans

Simulering av rommet med liten impedans på én vegg, slik beskrevet i kapittel 4.2.6, ga transferfunksjonene i figur 5.10.



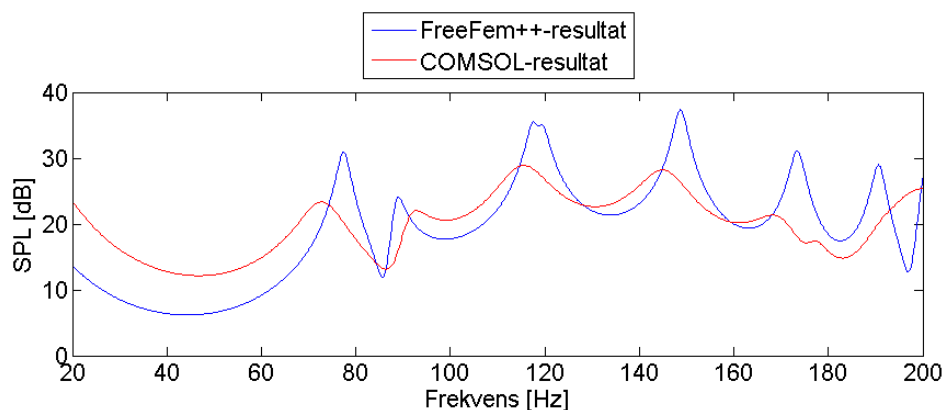
Figur 5.10: Transferfunksjon, konstant, liten, kompleks impedans på en vegg, alle andre flater harde

De to kurvene har store likheter, og topp- og bunnpunkter i de to ligger nære hverandre i frekvens. Den tidligere omtalte forskjellen i forskyvning av egenfrekvenser er altså ikke spesielt markant her. Ellers ser vi igjen at COMSOL-kurven ved høye frekvenser er adskillig mer dempet. Unntaket er ved 145 Hz, der det kan tyde på at de to distinkte egenfrekvensene funnet i FreeFem++ sammenfaller i COMSOL. Hva den skarpe dipen i COMSOL-kurven ved 116 Hz skyldes er uvisst.

Det er interessant å observere den første resonansen. Dette er antakeligvis halv-bølgeresonansen i x-retning, da rommet har størst utstrekning i denne retningen. I så fall er dette en endimensjonal mode med den dempede veggen i den ene enden. Det er naturlig at denne moden dempes mer enn moder i andre retninger i rommet, og vi ser at den dempes vesentlig mer i COMSOL enn i FreeFem++. Dette skyldes sannsynligvis at modellen i FreeFem++ ikke tar hensyn til dempingens plassering i rommet.

5.3 Annen romgeometri

Med romform, kilde- og mottakerposisjon og impedanser som beskrevet i kapittel 4.3 fås transferfunksjonene vist i figur 5.11.



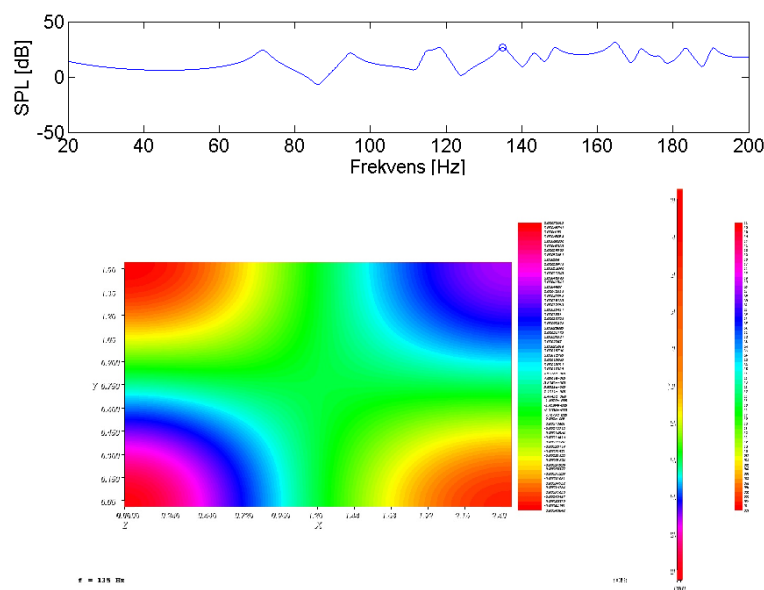
Figur 5.11: Transferfunksjon, annen romform med konstante, store, komplekse impedanser

Igjen ser vi at COMSOL gir en vesentlig gjevnere transferfunksjon, som viser større innflytelse av dempingen.

Eigenfrekvensene fra FreeFem++-simuleringen er her for det meste høyere enn de tilsvarende fra COMSOL. Dette kan skyldes at FreeFem++ ikke tar hensyn til realdelen av impedansene ved beregning av egenfrekvenser. I et smalt og avlangt rom som dette, altså med stort overflateareal per volum, vil veggabsorpsjonen ha større innvirkning. Dette kan dermed være årsaken til at egenfrekvensene er mer forskjellige her enn i figur 5.7.

5.4 Impedanser gitt ved parameterne R , m og d

Parametre som beskrevet i kapittel 4.4 i rektangulært rom ga transferfunksjon og modeplot, beregnet i 3D FreeFem++, som vist i figur 5.12.



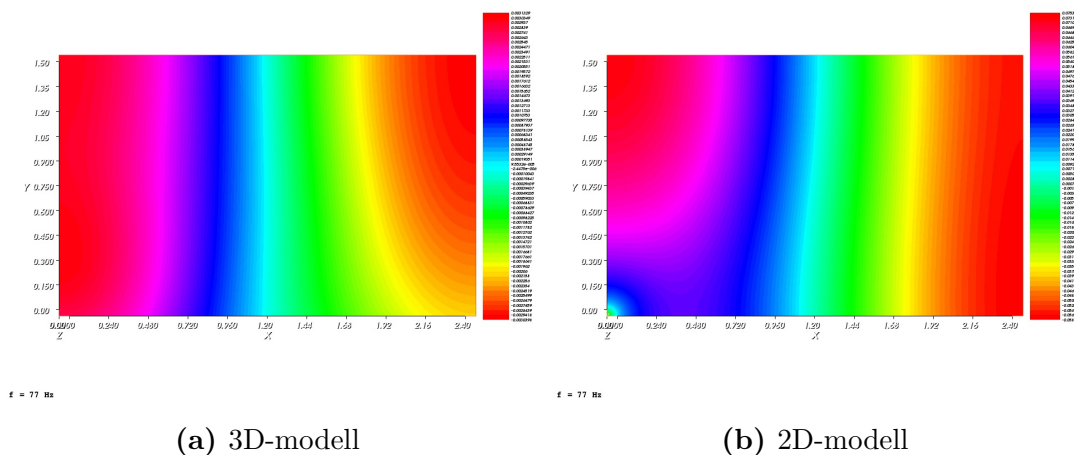
Figur 5.12: Transferfunksjon og modeplot, rektangulært rom med frekvensavhengige impedanser

Modeplottene i horisontalplan og z-retning er ved 135 Hz, hvilket kan sees av markeringen på transferfunksjonen. Tallverdiene de forskjellige fargene i modeplottene representerer er dessverre uleselige. Det er valgt å ikke fokusere på å rette opp dette da plottet uansett kun er en skisse, ment som et forslag til et fremtidig visualiseringsverktøy. Når faktisk visualisering gjøres i MATLAB vises ett og ett slikt bilde med modeplot for økende frekvens.

Transferfunksjonen er det vanskelig å si noe spesifikt om, men den ser ut til å være noe mer dempet ved lavere frekvenser, som stemmer med membranabsorbentmodellen fra Rindel [7].

5.5 Sammenligning mellom kilde- og 3D-simulering i FreeFem++

Modeformen ved kilde- og 3D-simulering i FreeFem++, i rommet beskrevet i 4.5, ble ved 77 Hz som vist i figur 5.13.



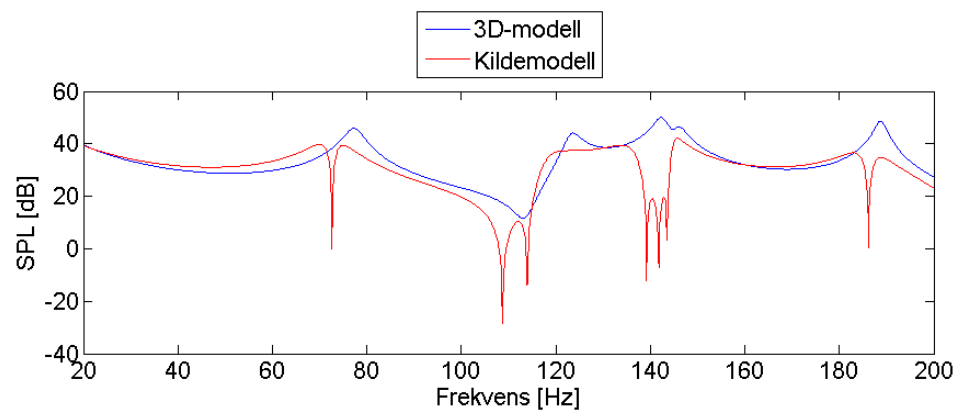
Figur 5.13: Modeform ved 77 Hz, 2D-rom, frekvensavhengig impedans

Vi ser av plottene at det hovedsakelig er samme egenmode som eksiteres i begge tilfellene, men det er likevel vesentlige forskjeller mellom plottene. Kilden, altså punktet nederst i venstre hjørne i figur 5.13b, ser her faktisk ut til å være i motfase med selve moden. Hvordan dette skjer er uvisst, men det har muligens å gjøre med at kildemodellen også har en direktelyd-komponent.

Forskjellene mellom plottene like ved den nederste veggen, altså den med absorpsjon, er forholdsvis små. Noe overraskende ser faktisk 3D-modellen ut til å gi et lavere lydtrykk ved denne veggen enn kildemodellen.

Transferfunksjonen for begge modellene ble som vist i figur 5.14.

Vi ser at transferfunksjonene fra de to modellene er relativt like, men kildemodellen har noen svært markante dips. At disse til dels er plassert der 3D-modellen gir resonanser er unektelig mystisk. Det er mulig at disse dippene skyldes feil i koden på kildemodellen, men dette vites ikke. Uansett betraktes 3D-modellen som mest pålitelig.



Figur 5.14: Transferfunksjon fra FreeFem++-modellene, rektangulært 2D-rom med frekvensavhengig impedans

6 Diskusjon

6.1 Sammenligning mellom 3D-modell i FreeFem++ og COMSOL

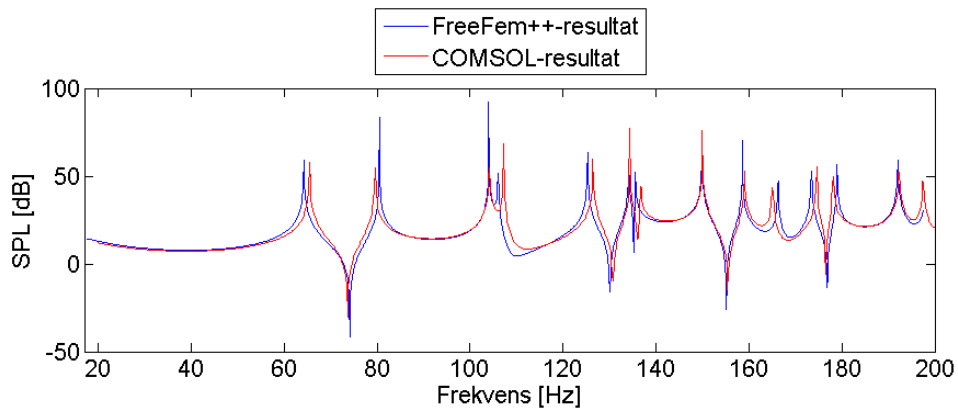
Ved simulering av rom med kun helt harde flater gir FreeFem++-modellen en transferfunksjon så godt som identisk med den COMSOL gir, med resonanser som nærmest perfekt sammenfaller med de analytisk beregnede. Når veggimpedansene derimot er endelige fås til dels store avvik mellom FreeFem++ og COMSOL.

Sammenligning mellom figur 5.3, 5.4 og 5.5 viser at modene forskyves oppover i frekvens med positive, imaginære impedanser på veggene, og nedover med negative. Dette stemmer overens med teorien i Kuttruff [5, s. 81], der akkurat dette fenomenet beskrives.

FreeFem++-simuleringen gir mindre frekvensforskyvning enn den i COMSOL gjør, både opp og ned. Fra tabell 5.1 har vi dog at egenmodene FreeFem++ finner i horisontalplanet stemmer godt overens med de COMSOL finner ved 2D-simulering. Ettersom modellen for kombinasjon av egenfrekvenser er perfekt kun ved harde vegger er det rimelig å anta at det er bruk av denne modellen som gjør at egenfrekvensene funnet i FreeFem++ ikke forskyves tilstrekkelig når impedansene er endelige.

I simuleringene der impedansene beskrives som store er forskjellen i frekvensskift mellom FreeFem++ og COMSOL relativt små, slik at resultatene fra disse FreeFem++-simuleringene kan være anvendelige. Avlest fra figur 5.4 og 5.5 kan det observeres at avviket i dette spesifikke tilfellet ligger på bare noen få Hz. Der impedansen har samme fortegn på alle rommets flater kan en dessuten ta høyde for at forskyvningen opp eller ned relativt egenfrekvensene ved harde flater er for liten, og kompensere for dette ved avlesning av resultatene. For eksempel får vi ved å forskyve transferfunksjonen fra FreeFem++ i simuleringen med negative, imaginære impedanser nedover med ytterligere 2,5 Hz et resultat svært likt det fra COMSOL. Et plot av dette er gitt i figur 6.1. Som vi ser blir feilen da svært liten.

Likevel er det åpenbart en grense for hvor lave impedanser 3D-modellen i FreeFem++ gir anvendelige resultater for. Blant annet har vi sett av transferfunksjonene i figur 5.6 og 5.9 at impedansverdier i størrelsesorden 500 - 1000 kg/m²s på alle flater gir resultater meget ulike de fra COMSOL. Derimot har vi sett av figur 5.10 at resultatet med kun en vegg med en slik liten impedans blir relativt likt COMSOL-resultatet. Dette er meget positivt, da en uansett svært sjeldent dekker alle flater med lydabsorbenter.



Figur 6.1: Transferfunksjon konstante, store negative imaginære impedanser

Det ser fra COMSOL-resultatene ut til at damping, altså realdel av impedansene, senker egenfrekvensene i rommet. I modellen i FreeFem++ tas dette derimot ikke høyde for. Som vi har sett av figur 5.7 kan det at FreeFem++ ikke inkluderer denne senkingen faktisk kompensere for at FreeFem++ heller ikke forskyver egenfrekvensene tilstrekkelig oppover ved positive imaginærkomponenter i impedansene. Med negative imaginærkomponenter av impedansene vil problemet derimot forverres.

En annen problematikk er at med samme komplekse impedans i FreeFem++ og COMSOL får dempingen langt større effekt i COMSOL-modellen. Hva dette skyldes vites ikke sikkert, men årsaken kan være at impedansverdiene i FreeFem++ er implementert som impedanser for normalt lydinnfall. Det kan for så vidt ikke utelukkes at det er gjort feil ved implementasjon av ligningene beskrevet i kapittel 2.6 i FreeFem++-skriptet.

Når det gjelder tidsbruk er 3D-modellen i FreeFem++ åpenbart overlegen COMSOL. Tidsbesparelsen på rundt 80-90 % er meget god i seg selv, og er dessuten oppnådd med et FreeFem++-skript som kun benytter en prosessorkjerne. COMSOL bruker derimot alle kjernene det har tilgjengelig. Dersom en skulle skrive om 3D-skriptet i FreeFem++ til optimalisert flerkjernekode burde en kunne oppnå en ytterligere, vesentlig tidsbesparelse.

6.2 Vurdering av kilde- og 3D-modellen i FreeFem++

En åpenbar fordel med kildemodellen i forhold til 3D-modellen er at førstnevnte ikke gjør forenklingen ved å til dels se bort fra realdelen av veggimpedansene.

Dessverre har kildemodellen på flere områder vist seg ikke å gi spesielt gode resultater.

For det første har vi fenomenet vist i figur 5.13, der det ser ut som kilden er i motfase med selve moden som eksiteres. Dette utgjør størsteparten av forskjellen mellom de to plottene, effekten av selve dempingen på veggen ved $y = 0$ i nevnte plot ser faktisk ut til å være svært liten.

For det andre har vi dipdene i transferfunksjonen, vist i figur 5.14, som til dels ligger ved romresonansene. Disse skulle åpenbart ikke vært til stede, og årsaken til dem er ukjent. Det kan likevel tenkes at også dette skyldes direktelyd fra kilden, som til tider ser ut til å være i motfase med eksiterte moder.

6.3 Brukergrensesnitt

I utviklingen av simuleringsverktøyet i FreeFem++ er brukergrensesnittet ikke vektlagt i særlig grad. Løsningen som er valgt her, med at romparametre som hjørnekoordinater og impedansparametre enten skrives inn i dialogbokser eller hardkodes i selve skriptet, er ikke spesielt god. Bruk av dialogbokser for dette er ikke problematisk i seg selv, men ettersom alle parametre må skrives inn på nytt hver gang simuleringen kjøres blir dette meget tungvint.

Hardkoding av parameterne gjør gjentakelse av en simulering langt enklere, men en får da problemet med at brukeren må endre på selve kildekoden. For brukere som ikke er vant til programmering kan dette være vanskelig, eller i det minste svært uvant. Det er dessuten en risiko for at brukere kan gjøre skade på skriptet, ved for eksempel å glemme et semikolon eller å feilaktig slette deler av koden. Som i all programmering skal det ikke mer enn en liten syntaksfeil til for å hindre et program i å kjøre. FreeFem++ har heller ingen god angrefunksjon, hvilket naturligvis gjør risikoen større.

Det største problemet med brukergrensesnittet er nok likevel følgene av endring i antall hjørner, som i alle tilfeller må hardkodes. Her må blant annet kodelinjene for mesh-generering og grensebetingelsene i ligningsformuleringene skrives om, noe som kan være utfordrende for en bruker som ikke kjenner programmet og syntaksen godt. Sannsynligheten for å innføre syntaksfeil her er forholdsvis stor.

6.4 Visualisering

Til slutt kan visualiseringen av resultatene nevnes kort. Bildet vist i figur 5.12 viser tydelig modeformene i horisontalplan og z-retning ved en bestemt frekvens,

her 135 Hz. I en mer forseggjort visualiseringsmetode kan det tenkes at en for eksempel kan holde musepekeren over et punkt på transferfunksjonen, og at de tilhørende modeformene umiddelbart plottes. I tillegg bør naturligvis en endelig visualiseringsmetode ha leselige tall på akser og fargekoder i modeplottene.

7 Konklusjon

Målet med denne oppgaven har vært å lage et effektivt simuleringsverktøy for lave frekvenser i romakustikk, basert på FEM. Dette simuleringsverktøyet skulle gi ut rommets egenfrekvenser, modeformer og transferfunksjon mellom kilde og mottaker, opp til 200 Hz. Effektiviteten skulle oppnås ved en oppdeling av det simulerte rommet i separat simulering i 2D, altså gulyplanet, og 1D, altså z-retning. For å muliggjøre dette ble verktøyet begrenset til å kunne brukes på rom formet som rettvinklede prizmer.

Det er i denne oppgaven laget to varianter av et slikt verktøy. Den ene er en relativt enkel modell som simulerer lydfeltet fra en punktkilde, både i 2D og 1D. 2D- og 1D-løsningene kan da ikke kombineres til en fullverdig 3D-løsning. Dette er en svakhet, men modellen har vist seg ellers å ikke fungere tilfredsstillende uansett.

Den andre modellen finner egenfrekvensene i 2D og 1D, for så å kombinere disse for å finne alle rommets egenfrekvenser. Transferfunksjon og modeformer beregnes så ved en lineærkombinasjon av egenfrekvensene. Denne modellen har fungert så godt som perfekt i rom med harde vegger. På andre rom har den dessverre to svakheter:

Den første er at modellen får økende feil ved lavere imaginærdel i impedansene. Resultatene modellen gir er funnet å være på grensen av pålitelig med imaginærdelen av impedansene $\text{Im}(z) \sim 10000 \text{ kg}/(\text{m}^2\text{s})$ på alle flater. Med kun én lavimpedant flate er modellen derimot funnet å gi relativt godt resultat med en impedans så lav som $z = (500 + 1000j) \text{ kg}/(\text{m}^2\text{s})$ dersom alle andre flater er harde. Dette er essensielt for praktisk bruk av verktøyet, da en gjerne plasserer absorberer kun på en liten andel av rommets totale overflateareal.

Modellens andre svakhet er at absorpsjon på rommets flater undervurderes av modellen, slik at simuleringene gir ut for høyt lydnivå ved rommets egenfrekvenser. I tillegg neglisjeres absorpsjonens innvirkning på egenfrekvensene. Ettersom dempingen har sett ut til å senke egenfrekvensene noe resulterer denne neglisjeringen i at de beregnede egenfrekvensene blir kunstig høye.

Oppdelingen av det opprinnelige 3D-problemet i en 2D- og en 1D-komponent har resultert i en vesentlig regnekraftmessig forenkling. Sammenlignet med COMSOL, som simulerer direkte i 3D, er tidsbesparelsen rundt 80-90 %. Oppdelingen har dessuten et videre potensiale dersom flerkjerneteknikk implementeres i det utviklede simuleringsverktøyet. FreeFem++-skriptet som er skrevet i denne oppgaven utnytter kun én prosessorkjerne.

Brukergrensesnittet i verktøyet er meget svakt. Selv implementasjonen med dialogbokser krever hardkoding dersom en ønsker å simulere et rom med et annet antall hjørner. For mindre programmeringskyndige brukere anbefales det derfor å bruke et eksisterende skript skrevet for minimum det antall hjørner som trengs, heller enn å endre på skriptene.

Visualiseringen av simuleringsresultatene gjort her fungerer godt, med unntak av markeringer på akser og fargekoder i modeplottene. Ettersom visualiseringen gjøres i det kostbare programmet MATLAB betraktes den likevel kun som et forslag.

Så lenge en simulerer rom med absorbenter kun på en liten andel av rommets totale overflateareal og husker på at resonansene er for kraftige i dempede rom, ser simuleringsverktøyet altså ut til å gi gode resultater. Med tanke på tidsbesparelsen i forhold til full 3D-simulering virker dette derfor å være en lovende teknikk for beregning av lavfrekvente lydforhold.

8 Videre arbeid

Testingen av simuleringsverktøyet er langt fra fullført, og det som er gjort i denne oppgaven gir kun en indikasjon på begrensningene og mulighetene verktøyet har. Blant annet bør sammenligninger med måledata fra eksisterende rom gjøres, slik at en får testet hvor godt implementasjonen av frekvensavhengige impedanser fungerer. En kan da også undersøke om COMSOL faktisk gir korrekt resultat, slik det er antatt i denne oppgaven.


Dersom mer omfattende arbeider skal gjøres anbefales det å implementere simuleringskonseptet studert i denne oppgaven i et dedikert program basert på flerkjerneteknikk. Et slikt program kan implementere grafisk brukergrensesnitt både for beskrivelse av rommet og ved bestemmelse av impedansparametre. Videre kan valideringen av moder muligens gjøres automatisk ved å sjekke at alle modene er ortogonale [6, s. 286]. Visualiseringen av resultatene bør også kunne gjøres i samme program.

Referanser

- [1] Frédéric Hecht. Freefem++doc, 2013. URL <http://www.freefem.org/ff++/ftp/freefem++doc.pdf>. Besøkt 28.4.2013.
- [2] Lawrence E. Kinsler, Austin R. Frey, Alan B. Coppens, og James V. Sanders. *Fundamentals of acoustics*. Wiley, New York, 4th edition, 2000. Lawrence E. Kinsler ... [et al.] ill.
- [3] Erwin Kreyszig. *Advanced engineering mathematics*. Wiley, Hoboken, N.J., 9th edition, 2006. Erwin Kreyszig ill. 1. utg. 1962.
- [4] Ulf R. Kristiansen og Erlend M. Viggren. Computational methods in acoustics, 2010. URL <http://www.iet.ntnu.no/courses/ttt12/compendium.pdf>. Besøkt 13.4.2013.
- [5] Heinrich Kuttruff. *Room acoustics*. Spon Press, London, 5th edition, 2009. Heinrich Kuttruff ill.
- [6] Allan D. Pierce. *Acoustics : an introduction to its physical principles and applications*. Acoustical Society of America through the American Institute of Physics, Woodbury, N.Y., 1989. lc89080362 Allan D. Pierce ill.
- [7] Jens Holger Rindel. An impedance model for estimating the complex pressure reflection factor, 2011. Forum Acusticum 2011, Aalborg, Danmark.
- [8] Wikipedia. List of fem software packages, 2013. URL http://en.wikipedia.org/wiki/List_of_finite_element_software_packages. Besøkt 20.2.2013.

A HMS-vurdering

NTNU		Utarbeidet av		Nummer		Dato	
HMS		HMS-avd.		HMSRV2601		22.03.2011	
		Godkjent av		Side		Erstatet	
		Rektor		01.12.2006			



Kartlegging av risikofylt aktivitet

Enhet: _____

Linjeleder: (ansv. veileder, faglig ansvarlig) _____

Deftakere ved kartleggingen (m/ funksjon): _____
(Ansv. veileder, student, evt. medveileder, evt. andre m. kompetanse)

Kort beskrivelse av hovedaktivitet/hovedprosess: _____

Signaturer: *[Signature]* _____

Utvikling av simuleringverktøy Dato: _____
Faglærer Ulf R Kristiansen

13.06.2013

Ulf R Kristiansen (faglærer), Truls V Klami (student)
 Masteroppgave ved Truls V Klami: Utvikling av simuleringverktøy

[Signature]

ID nr.	Aktivitet/prosess	Ansvarlig	Eksisterende dokumentasjon	Eksisterende sikringsiltak	Lov, forskrift o.l.	Kommentar
1	<i>Ingen risikofylt aktivitet</i>					
2						
3						
4						
5						
6						

NTNU	NTNU	Utarbeidet av	Nummer	Dato
		HMS-avd.	HMSRV2603	04.02.2011
HMS KS		Godkjent av	Side	Erstatet
		Rektor		09.02.2010

Risikovurdering



Enhet: _____ **Utvikling av simuleringsverktøyt Dato:** 13.06.2013

Linjeleder: (ansv. veileder, faglig ansvarlig) _____ **Faglærer Ulf R Kristiansen**

Deftakere ved risikovurderingen (m/ funksjon): _____ **Ulf R Kristiansen (faglærer), Truls V Klami (student)**

(Ansv. veileder, student, evt. medveiledere, evt. andre m. kompetanse) **Masteroppgave ved Truls V Klami. Utvikling av simuleringsverktøy**

Risikovurderingen gjelder hovedaktivitet: _____ **Utvikling av simuleringsverktøy**

Signaturer: *Ulf R Kristiansen* _____ *Truls V Klami*

ID nr.	Aktivitet/prosess fra kartleggingsskjemaet	Mulig uønsket hendelse	Vurdering av sannsynlighet (1-5)	Vurdering av konsekvens				Risiko-verdi (menneske)	Kommentarer/status Forslag til tiltak
				Menneske (A-E)	Ytre miljø (A-E)	Øk./materieill (A-E)	Ørr-dømme (A-E)		
1	<i>Ingen risikofyllt aktivitet</i>								
2									
3									
4									
5									
6									

B Brukermanual

For å bruke simuleringsverktøyet må programmet FreeFem++-cs lastes ned og installeres. Programmet lastes ned fra <https://www.ljll.math.upmc.fr/lehyaric/ffcs/index.htm>.

Kort fortalt virker simuleringsverktøyet ved at et skript kjøres i FreeFem++-cs. Dette er selve simuleringen. Dette gir ut en transferfunksjon mellom en kilde og en mottaker, en liste over egenfrekvenser i rommet og en mappe med .jpg-bilder av modeplot. Transferfunksjonen og modeplottene visualiseres så ved kjøring av et MATLAB-skript.

Simuleringsverktøyet brukes ved at skriptfilen, altså enten `eigen-original.edp` eller `eigen-ui.edp`, åpnes i FreeFem++-cs. En får da skriptet i øvre venstre del av FreeFem++-cs-vinduet, og kan endre på skriptet her. Dersom `eigen-original.edp` velges må brukeren hardkode alle romparametre i dette skriptet, velges `eigen-ui.edp` skrives disse i stedet inn gjennom dialogbokser. Skriptet med dialogbokser er enklere i bruk, men er meget tungvint å bruke dersom simuleringer med mange felles parametre skal gjøres etter hverandre. Til sist må det også nevnes at dersom antall vegger i rommet skal endres, default er 4, må dette hardkodes. Hvordan dette gjøres forklares både her og i skriptene.

Syntaks i skriptene er tilnærmet lik C++-syntaks. Skriptene er vesentlig enklere å forstå dersom en har en viss kjennskap til C++.

B.1 Bruk av `eigen-original.edp`

B.1.1 Bestemmelse av parametre

Alle parametre som skal endres av brukeren er samlet først i skriptet. Merk at alle tallverdier er i SI-enheter. Skriptet er i utgangspunktet satt opp for et rektangulært rom med for det meste harde vegger, og kilde og mottaker i motsatte hjørner.

Det brukeren skal gjøre er å erstatte tallverdiene som står på parameterne med de korrekte tallverdier for situasjonen som skal simuleres. Dersom antall hjørner skal endres bør brukeren konsultere kapittel B.3.

Først bestemmes x-, y- og z-koordinatene til henholdsvis kilde og mottaker. Disse variablene er kalt `sourcecx` osv.

Deretter bestemmes takhøyden, kalt `h`.

Deretter bestemmes x- og y-koordinatene til hvert hjørne i rommets horisontalplan. Disse variablene er kalt `xcorner(0)` osv. Merk her at `FreeFem++-cs`, i likhet med `C++`, begynner indeksering i 0, slik at vi i et rom med fire hjørner i skriptet har hjørne 0, 1, 2 og 3.

Deretter bestemmes resistans-, masse- og luftgap-parameterne for rommets tak, gulv og vegger. For en innføring i disse parameterne vises det til Rindel [7]. For gulv og tak kalles parameterne `Rgulv`, `mgulv` osv., for veggene har vi `R(0)`, `m(0)` osv.

Dette var romparameterne, deretter kan simuleringsparameterne bestemmes. Disse er godt forklart i skriptet, og trenger ingen grundig omtalelse her. Disse parameterne trenger normalt heller ikke å endres, med unntak av `saveLoc`.

Denne tekststrengen angir hvilken mappe på maskinen resultatene skal lagres i, og må spesifiseres med full adresse. Det er veldig viktig at denne adressen er gyldig, og at mappen for resultater eksisterer. `FreeFem++-cs` kan ikke lage mapper. En må her også lage to mapper i den aktuelle mappen, der bildene av modeplot skal lagres. Disse må kalles `eigenplots2D` og `eigenplots1D`.

Dersom visualiseringen i MATLAB skal benyttes, beskrevet i kapittel B.4, anbefales det å sette resultatmappen til å være samme mappe som MATLAB-skriptet ligger i. En slipper da å skrive om MATLAB-koden.

B.1.2 Kjøring av skript

Når parameterne er endret til riktige verdier kan skriptet kjøres. Underveis i kjøringen plotter programmet moder i horisontalplanet. Disse må brukeren selv avgjøre om er gyldige eller ikke, samt om de registreres dobbelt. For hvert plot kommer det opp en dialogboks, og informasjon om hva som skal gjøres står i tekstfeltet nederst i `FreeFem++-cs`-vinduet. Kort fortalt skal en trykke 0 dersom moden ikke skal godkjennes, og 1 dersom den skal godkjennes.

Resultater genereres så automatisk, og visualisering i MATLAB kan gjøres så snart simuleringen er ferdig.

B.2 Bruk av `eigen-ui.edp`

Ved bruk av dette skriptet settes alle parametre ved kjøring. Unntaket er som tidligere nevnt bestemmelse av antall hjørner, dersom dette tallet skal endres vises det til kapittel B.3.

Under kjøring av skriptet blir alle nødvendige parametre hentet inn gjennom dialogbokser. Beskrivelse av hva som skal skrives inn står i tekstfeltet nederst i FreeFem++-cs-vinduet. Der det samtidig bes om flere parametre, slik som x- og y-koordinater for et hjørne, skal disse skrives inn med mellomrom mellom. En kan også skrive inn ett tall av gangen, og trykke enter for hvert tall, men dette er naturligvis mer tungvint.

Når det bes om en tekststreng med mappeadresse for resultatene gjelder de samme regler som ved `eigen-original.edp`. Også her må de to undermappene `eigenplots2D` og `eigenplots1D` eksistere på forhånd.

Evaluering av modene gjøres som i `eigen-original.edp`.

B.3 Endring av antall hjørner

Når antall hjørner skal endres er det enkelte kodelinjer som må endres, strykes eller tilføyes.

Først settes verdien på parameteren `corners`, som er nesten øverst i skriptet.

I `eigen-original.edp` må en, hvis en skal øke antall hjørner, så skrive på ekstra linjer der x- og y-koordinatene for hvert hjørne bestemmes. Her må en da skrive for eksempel:

```
xcorner(4) = 1;  
ycorner(4) = 3;
```

osv.

Det samme må gjøres på resistans-, masse- og luftsjikt-parameterne, altså skrive for eksempel:

```
R(4) = 2513;  
m(4) = 2.139;  
d(4) = 1e10;
```

osv.

Merk at disse modifikasjonene ikke gjøres i `eigen-ui.edp`.

I begge skript må man så legge til en eller flere linjer på border-genereringen. Dette er plassert like etter all brukerinput i skriptene, og er også markert. En må ganske enkelt skrive helt tilsvarende linjer for hvert ekstra hjørne man vil ha, i

tillegg til å endre indekseringen i den siste, allerede eksisterende linjen. Der det står

```
border b3(t=0,1) {x=xcorner(3)+(xcorner(0)-xcorner(3))*t; y=ycorner(3)+(ycorner(0)-ycorner(3))*t;}
```

skal denne linjen, dersom en legger til ett hjørne, erstattes med

```
border b3(t=0,1) {x=xcorner(3)+(xcorner(4)-xcorner(3))*t; y=ycorner(3)+(ycorner(4)-ycorner(3))*t;}
border b4(t=0,1) {x=xcorner(4)+(xcorner(0)-xcorner(4))*t; y=ycorner(4)+(ycorner(0)-ycorner(4))*t;}
```

Deretter må linjen

```
mesh Thm=buildmesh(b0(ceil(l(0)*ms)) + b1(ceil(l(1)*ms)) + b2(ceil(l(2)*ms)) + b3(ceil(l(3)*ms)));
```

endres i begge skript. Denne står omlag 1/4 ned i skriptet, og beskrivelse av hvordan denne skal endres står i skriptene. Dersom antall hjørner er 5, skal denne linjen bli:

```
mesh Thm=buildmesh(b0(ceil(l(0)*ms)) + b1(ceil(l(1)*ms)) + b2(ceil(l(2)*ms)) + b3(ceil(l(3)*ms)) + b4(ceil(l(4)*ms)));
```

Til slutt må grensebetingelsene i selve ligningsformuleringen endres. Dette er omtrent midt i skriptet, og også her står en beskrivelse i skriptet om hva som skal endres. Dersom antall hjørner er 5, erstattes linjen

```
+int1d(Thm,b3)(N*c*sqrt(sigma)*rho*pN/imag(z(3)));
```

med

```
+int1d(Thm,b3)(N*c*sqrt(sigma)*rho*pN/imag(z(3)))
+int1d(Thm,b3)(N*c*sqrt(sigma)*rho*pN/imag(z(4)));
```

Når alt dette er endret på, kan skriptet kjøres for det nye antallet hjørner.

B.4 Visualisering

For visualisering av resultatene i MATLAB trenger en ikke gjøre annet enn å kjøre visualiseringsskriptet, som altså bør være plassert i samme mappe som simuleringsresultatene. Skriptet plotter da transferfunksjon, og modeplot i horisontalplan og z-retning for en og en, og økende, frekvens. For å gå til neste frekvens med modeplot, må en trykke en tast. Noen mulighet til å gå tilbake i transferfunksjonen igjen finnes dessverre ikke.

B.5 Annet

For ytterligere info for programmet FreeFem++-cs vises det til programmets nettsider, <https://www.ljll.math.upmc.fr/lehyaric/ffcs/index.htm>.

For ytterligere brukerstøtte og andre spørsmål om simuleringsverktøyet kan utvikler kontaktes.

Truls V Klami - truls.klami@gmail.com

C MATLAB-kode

C.1 Visualisering av resultater

```
clear all
close all

fidinfo = fopen('info.txt');
temp = fgetl(fidinfo);
antFrekvens = str2double(temp);

temp = fgetl(fidinfo);
bildRes = str2double(temp);

temp = fgetl(fidinfo);
f0 = str2double(temp);

temp = fgetl(fidinfo);
fend = str2double(temp);

fres = (fend-f0)/(antFrekvens-1);
antBild = ceil(antFrekvens/bildRes);

bildRes = bildRes*fres;

fid = fopen('transfer_filID.txt');
for i=1:antFrekvens
    temp = fgetl(fid);
    p(i) = str2double(temp);
    temp = fgetl(fid);
    f(i) = str2double(temp);
end

spl = 20*log10(abs(p)/2e-5);

for i=1:antBild
    temp = int2str(i-1);
    bilde2D = imread(['eigenplots2D\freqnr' temp '.jpg']);
    bilde1D = imread(['eigenplots1D\freqnr' temp '.jpg']);
    eval(['bilde2D' int2str(i) ' = bilde2D;'])
    eval(['bilde1D' int2str(i) ' = bilde1D;'])
end

figur = figure(1);
set(figur, 'Position', [100, 100, 1000, 800])
```

```

for i=1:antBild
    subplot(4,4,1:4)
    hold off
    plot(f,spl)
    set(gca,'FontSize',16)
    xlabel('Frekvens [Hz]')
    ylabel('SPL [dB]')

    y = spl(floor((i-1)*bildRes/fres+1)) + ((i-1)*bildRes/fres+1-
        floor((i-1)*bildRes/fres+1))*spl(ceil((i-1)*bildRes/fres+1));

    x = f0+bildRes*(i-1);
    hold on
    plot(x,y,'o')

    subplot(4,4,[5:7 9:11 13:15])
    eval(['image(bilde2D' int2str(i) ')'])
    axis off
    subplot(4,4,[8 12 16])
    eval(['image(bilde1D' int2str(i) ')'])
    axis off
    pause
end

```

C.2 Kode for beregning av egenfrekvenser i rektangulært rom med harde vegger

```
clc
clear all
n = 5;
lx = 2.4;
ly = 1.5;
lz = 2;
fend = 200;

c = 343;

f2 = zeros(n,n,n);
teller = 1;
for i=0:n
    for j=0:n
        for k=0:n
            f2(i+1,j+1,k+1) = c/2*sqrt((i/lx)^2+(j/ly)^2+(k/lz)^2);
            if f2(i+1,j+1,k+1) < fend
                f(teller) = f2(i+1,j+1,k+1);
                mode = ['(' int2str(i) ', ' int2str(j) ', ' int2str(k)
                    ')'];
                disp(['Frekvens(' int2str(teller) ') = ' num2str(f(
                    teller)) ' - Mode ' mode])
                teller=teller+1;
            end
        end
    end
end
f = sort(f)
```

C.3 cpu-test

```
tic
t = 0:0.01:1000;
y = sin(t);
parfor i=1:10000
    test = sqrt(y);
end
toc
```

D FreeFem++-kode

D.1 Kildemodell

Skriptet har hardkodete romparametre som gitt i kapittel 4.5.

```
bool printResults = true;
string saveLoc = "C:/Users/Truls/Documents/MATLAB/Master"; //Mappe
    for lagring av data som skal behandles av MATLAB

int i, corners;
int maxcorners = 40; //40 er satt som maks antall hjorner, dette kan
    endres
real [int] xcorner(maxcorners), ycorner(maxcorners), l(maxcorners),
    R(maxcorners), m(maxcorners), d(maxcorners);
complex [int] zvegg(20);

corners = 4; //Antall hjorner

//Kildeposisjon:
real sourcex = 0.01;
real sourcey = 0.01;
real sourcez = 0.01;

//Motakerposisjon:
real recx = 2.39;
real recy = 1.49;
real recz = 1.99;

real h = 2; // Takhoyde

//x- og y-koordinater i gulv-planet:
xcorner(0) = 0;
ycorner(0) = 0;

xcorner(1) = 2.4;
ycorner(1) = 0;

xcorner(2) = 2.4;
ycorner(2) = 1.5;

xcorner(3) = 0;
ycorner(3) = 1.5;
//////////
//Med flere hjorner skrives xcorner(4) = ... osv nedover her
```

```

//////////

//Vegg-, gulv- og takimpedanser ved R-, m- og d-parametre:
R(0) = 2513;
m(0) = 2.139;
d(0) = 1e10;

R(1) = 1e10;
m(1) = 1e10;
d(1) = 1e10;

R(2) = 1e10;
m(2) = 1e10;
d(2) = 1e10;

R(3) = 1e10;
m(3) = 1e10;
d(3) = 1e10;

//////////
//Med flere hjorner skrives R(4) = ... osv nedover her
//////////

real Rgulv = 1e10;
real mgulv = 1e10;
real dgulv = 1e10;

real Rtak = 1e10;
real mtak = 1e10;
real dtak = 1e10;

//Simuleringsparametre:
real resTransfer = 0.1; //Omtrentlig oppløsning på transferfunksjon
    i frekvens [Hz]
real resModes = 100; //Omtrentlig oppløsning på modeplott i frekvens
    [Hz]
real f0 = 20; //Startfrekvens
real fend = 200; //Sluttfrekvens

//Slutt på input av data
//
    ////////////////////////////////////////////

real c = 343; //[m/s]
real rho = 1.21; //[kg/m^3]
real f, k2; // k2 = k^2 = (f*2*pi/c)^2
func g = 1;

```

```

for (i=0;i<corners;i++)
{
l(i) = sqrt((xcorner((i+1)%corners)-xcorner(i))^2+(ycorner((i+1)%
corners)-ycorner(i))^2);
}

//1D-rom:
real b = 0.1;

border s1D(t=0,1) {x=b/2+0.01*sin(2*pi*t); y=sourcez+0.01*cos(2*pi*t
);}
border gulv(t=0,1) {x=b*t; y=0;}
border tak(t=0,1) {x=b-b*t; y=h;}
border vegg1(t=0,1) {x=b; y=h*t;}
border vegg2(t=0,1) {x=0; y=h-h*t;}

//2D-rom:
border s2D(t=0,1) {x=sourcex+0.005*sin(2*pi*t); y=sourcey+0.005*cos
(2*pi*t);}
border b0(t=0,1) {x=xcorner(0)+(xcorner(1)-xcorner(0))*t; y=ycorner
(0)+(ycorner(1)-ycorner(0))*t;}
border b1(t=0,1) {x=xcorner(1)+(xcorner(2)-xcorner(1))*t; y=ycorner
(1)+(ycorner(2)-ycorner(1))*t;}
border b2(t=0,1) {x=xcorner(2)+(xcorner(3)-xcorner(2))*t; y=ycorner
(2)+(ycorner(3)-ycorner(2))*t;}
border b3(t=0,1) {x=xcorner(3)+(xcorner(0)-xcorner(3))*t; y=ycorner
(3)+(ycorner(0)-ycorner(3))*t;}
//////////
//Med flere hjorner skrives border b4(t=0,1) ... osv nedover her
//////////

real antit = ceil((fend-f0)/resTransfer)+1;
int modPlot = resModes/resTransfer; //Modulus for a bestemme om
iterasjonen skal plottes

real meshfact = 10; //Antall noder i mesh pr bolgelengde ved hoyeste
frekvens

mesh Th2D=buildmesh(b0(ceil(l(0)*meshfact*fend/c)) + b1(ceil(l(1)*
meshfact*fend/c)) + b2(ceil(l(2)*meshfact*fend/c)) + b3(ceil(l(3)
*meshfact*fend/c)) + s2D(6));
//////////
//Med flere hjorner skrives + b4(ceil(l(4)*meshfact*fend/c)) osv inn
i uttrykket ovenfor
//////////

mesh Th1D=buildmesh(gulv(4) + vegg1(30) + tak(4) + vegg2(30) + s1D
(20));

```

```

int teller;
real [int] transfer2D(antit), transfer1D(antit);
real [int] frekvens(antit);
int plotteller = 0;
////////////////////////////////////
//Start pa den store for-lokka

for (teller=0; teller<antit; teller=teller+1)
{
f = (fend-f0)/(antit-1)*teller+f0;
k2 = (f*2*pi/c)^2;

//Beregning av impedanser:
complex zgulv = Rgulv+1i*(2*pi*f*mgulv - rho*c^2/(2*pi*f*dgulv));
complex ztak = Rtak+1i*(2*pi*f*mtak - rho*c^2/(2*pi*f*dtak));

for (i=0;i<corners;i++)
{
zvegg(i) = R(i)+1i*(2*pi*f*m(i) - rho*c^2/(2*pi*f*d(i)));
}
//Slutt pa beregning av impedanser

//2D-simulering, altsa xy-planet:
fespace Vh2D(Th2D,P2); // P2 betyr kvadratiske basis-funksjoner
Vh2D<complex> pN,N;

solve sound(pN,N)=int2d(Th2D) (pN*N * k2 - dx(pN)*dx(N) - dy(pN)*dy(N)
))
- int1d(Th2D,s2D) (g*N)
- int1d(Th2D,b0) (N*1i*2*pi*f*rho*pN/zvegg(0))
- int1d(Th2D,b1) (N*1i*2*pi*f*rho*pN/zvegg(1))
- int1d(Th2D,b2) (N*1i*2*pi*f*rho*pN/zvegg(2))
- int1d(Th2D,b3) (N*1i*2*pi*f*rho*pN/zvegg(3));

Vh2D p2Dreal = real(pN);
//Slutt pa 2D-simulering

//1D-simulering, z-retning:
fespace Vh1D(Th1D,P2);
Vh1D<complex> pN1D,N1D;
solve sound2(pN1D,N1D)=int2d(Th1D) (pN1D*N1D*k2-dx(pN1D)*dx(N1D) - dy
(pN1D)*dy(N1D))
-int1d(Th1D,s1D) (g*N1D)
-int1d(Th1D,tak) (N1D*1i*2*pi*f*rho*pN1D/ztak)
-int1d(Th1D,gulv) (N1D*1i*2*pi*f*rho*pN1D/zgulv);

Vh1D p1Dreal = real(pN1D);
//Slutt pa 1D-simulering i z-retning

```



```

////////////////////////////////////
//Plotting og eksportering av bilder:

if (teller%modPlot == 0)
{
plotteller++;
string tittelPlot = "f = " + f + " Hz";
string tittelFil = saveLoc + "/bilder/modeplot2D" + teller/modPlot +
    ".jpg";
string tittelFil1D = saveLoc + "/bilder/modeplot1D" + teller/modPlot
    + ".jpg";
if (printResults == true)
{
    plot(p2Dreal, wait=0, ps=tittelFil, fill=1, nbiso=100, value=1,
        cmm=tittelPlot, aspectratio=1);
    plot(p1Dreal, wait=0, ps=tittelFil1D, fill=1, nbiso=100, value
        =1, cmm=tittelPlot, aspectratio=1);
}
}
//Slutt pa plotting
////////////////////////////////////

transfer2D(teller) = p2Dreal(recx, recy);
transfer1D(teller) = p1Dreal(0.1*3/4, recz);
frekvens(teller) = f;

cout << endl << "f: " << f << endl;

} //Slutt pa den store for-lokka

if (printResults == true)
{
{
ofstream ut(saveLoc+"/transferfunksjon2D.txt");
for (i=0; i<antit; i++)
{
ut << transfer2D(i) << endl << frekvens(i) << endl;
}
}
};

{
ofstream ut(saveLoc+"/transferfunksjon1D.txt");
for (i=0; i<antit; i++)
{
ut << transfer1D(i) << endl << frekvens(i) << endl;
}
}
};
}
}

```

D.2 3D-modell

Skriptet har hardkodede romparametre som gitt i kapittel 4.5.

```
int corners, i, j;
real [int] x1(100), y1(100); //100 er satt som maks antall hjorner,
    dette kan endres

////////////////////////////////////
//Input av data:
//Romdata:

//Kilde- og mottaker-koordinater:
real sourcex = 0.01;
real sourcey = 0.01;
real sourcez = 0.01;

real recx = 2.39;
real recy = 1.49;
real recz = 1.99;

real h = 2; //Takhoyde

corners = 4; //Antall hjorner

real [int] xcorner(corners), ycorner(corners), R(corners), m(corners
    ), d(corners);

//x- og y-koordinater i gulv-planet:
xcorner(0) = 0;
ycorner(0) = 0;

xcorner(1) = 2.4;
ycorner(1) = 0;

xcorner(2) = 2.4;
ycorner(2) = 1.5;

xcorner(3) = 0;
ycorner(3) = 1.5;
//Skriv inn xcorner(4) = ...; osv nedover her, hvis corners > 4

//Resistans-, masse- og luftgap-parametre for impedans:
real Rgulv = 1e10;
real mgulv = 1e10;
real dgulv = 1e10;
```

```

real Rtak = 1e10;
real mtak = 1e10;
real dtak = 1e10;

R(0) = 2513;
m(0) = 2.139;
d(0) = 1e10;

R(1) = 1e10;
m(1) = 1e10;
d(1) = 1e10;

R(2) = 1e10;
m(2) = 1e10;
d(2) = 1e10;

R(3) = 1e10;
m(3) = 1e10;
d(3) = 1e10;
//Skriv inn R(4) = ...; osv nedover her, hvis corners > 4

//Slutt pa input av romdata
////////////////////////////////////

//Input av simuleringsparametre ved beregning av egenfrekvenser og –
moder:

real f0 = 20;          //Startfrekvens [Hz], bor settes lavt nok til a
fa med laveste egenfrekvens
real fend = 300;      //Sluttfrekvens [Hz], bor settes hoyere enn
fendtransfer
real antit = 2800;    //Antall iterasjoner, oppløsning blir ca. (fend-
f0)/antit
real overlapp = 1.4;  //Stort overlapp øker sannsynlighet for a fa
med alle moder, men flere moder registreres dobbelt.

string saveLoc = "C:/Users/Truls/Documents/MATLAB/Master"; //Mappe
for lagring av data som skal behandles av MATLAB
//Husk a opprette mapper der data skal lagres, det skjer dessverre
ikke automatisk

//Input av parametre for generering av transferfunksjon og modeplot:
real f0transfer = 20; //Nedre grense for beregning av
transferfunksjon.
real fendtransfer = 200; //ovre grense for beregning av
transferfunksjon, bor vaere vesentlig mindre enn fend.
real antittransfer = 1801;
real plotres = 10; //Forhold mellom oppløsning i transferfunksjon
og modeplot. Ma vaere heltall større enn 0.

```

```

//Slutt pa input av data
////////////////////////////////////

real c = 343; //[m/s]
real rho = 1.21;

//2D-rom:
border b0(t=0,1) {x=xcorner(0)+(xcorner(1)-xcorner(0))*t; y=ycorner
(0)+(ycorner(1)-ycorner(0))*t;}
border b1(t=0,1) {x=xcorner(1)+(xcorner(2)-xcorner(1))*t; y=ycorner
(1)+(ycorner(2)-ycorner(1))*t;}
border b2(t=0,1) {x=xcorner(2)+(xcorner(3)-xcorner(2))*t; y=ycorner
(2)+(ycorner(3)-ycorner(2))*t;}
border b3(t=0,1) {x=xcorner(3)+(xcorner(0)-xcorner(3))*t; y=ycorner
(3)+(ycorner(0)-ycorner(3))*t;}
////////////////////////////////////
//Med flere hjorner skrives border b4(t=0,1) ... osv nedover her
////////////////////////////////////

//1D-romdata:
real b = 0.1;
border gulv(t=0,1) {x=b*t; y=0;}
border tak(t=0,1) {x=b-b*t; y=h;}
border vegg1(t=0,1) {x=b; y=h*t;}
border vegg2(t=0,1) {x=0; y=h-h*t;}

// Beregning av romvolum
real [int] l(corners); // lengde pa hver vegg
for (i=0;i<corners;i++)
{
l(i) = sqrt((xcorner((i+1)%corners)-xcorner(i))^2+(ycorner((i+1)%
corners)-ycorner(i))^2);
}
//Formel s. 46 i Rottmann
real gulvareal;
for (i=0; i<(corners-1); i++){
gulvareal += 0.5*(xcorner(i)*(ycorner((i+1)%corners)-ycorner((i
-1+corners)%corners)));
}
gulvareal = abs(gulvareal);
real volum = gulvareal*h;
// Slutt pa volumberegning

//Genererer mesh:
real meshfact = 10; //Angir antall noder pr bolgelengde
real ms = meshfact*fend/c; //ms star for mesh size
////////////////////////////////////

```

```

mesh Thm=buildmesh(b0(ceil(l(0)*ms)) + b1(ceil(l(1)*ms)) + b2(ceil(l
  (2)*ms)) + b3(ceil(l(3)*ms)));
// Skriv inn i linjen over, for siste parentes, + b4(ceil(l(4)*ms))
  osv, dersom flere hjorner benyttes
//////////
mesh Thm1D = buildmesh(gulv(ceil(b*ms)) + vegg1(ceil(h*ms)) + tak(
  ceil(b*ms)) + vegg2(ceil(h*ms)));

complex [int] z(corners);
complex zgulv, ztak;

fespace Vhm(Thm,P2);
Vhm<complex> pN,N;

fespace Vhm1D(Thm1D,P2);
Vhm1D<complex> pN1D, N1D;

Vhm [int] psi(100); //Maks antall egenmoder i 2D satt til 100
real [int] modedefrekvens(100);
Vhm1D [int] psi1D(50); //Maks antall egenmoder i 1D satt til 50
real [int] modedefrekvens1D(50);

real modeteller = 0; //Teller for modenummerering
modedefrekvens = 0;

real modeteller1D = 0;
modedefrekvens1D = 0;

real res = (fend-f0)/(antit-1); //Frekvensopplosning

real [int] transfer(antit);
real [int] frekvens(antit);

int nev=10; //Antall egenverdier som beregnes i hver iterasjon
real[int] ef(nev); //ef er egenfrekvens
real[int] ef1D(nev);

real teller = 0; //Teller for store for-lokke
real f = f0;

string tittelPlot, tittelFil;

//////////
// Start store for-lokke:
for (teller=0; teller<antit; teller=teller+1)
{
if (antit > 1)
f = (fend-f0)/(antit-1)*teller+f0;

```

```

//Beregning av impedanseverdier:
complex zgulv = Rgulv+1i*(2*pi*f*mgulv - rho*c^2/(2*pi*f*dgulv));
if (imag(zgulv) == 0)
    zgulv += 1e-10i;
complex ztak = Rtak+1i*(2*pi*f*mtak - rho*c^2/(2*pi*f*dtak));
if (imag(ztak) == 0)
    ztak += 1e-10i;
for (i=0;i<corners;i++){
    z(i) = R(i)+1i*(2*pi*f*m(i) - rho*c^2/(2*pi*f*d(i)));
    if (imag(z(i)) == 0)
        z(i) += 1e-10i;
}

real sigma = (2*pi*f/c)^2; //Sigma er verdien som egenverdier skal
    beregnes i naerheten av
//2D-simulering:
// OP = A - sigma B ; // the shifted matrix
varf op(pN,N)= int2d(Thm) (dx(pN)*dx(N) + dy(pN)*dy(N) - sigma* pN*N)
    +int1d(Thm,b0) (N*c*sqrt(sigma)*rho*pN/imag(z(0)))
    +int1d(Thm,b1) (N*c*sqrt(sigma)*rho*pN/imag(z(1)))
    +int1d(Thm,b2) (N*c*sqrt(sigma)*rho*pN/imag(z(2)))
    +int1d(Thm,b3) (N*c*sqrt(sigma)*rho*pN/imag(z(3)));
//////////
//Med flere enn 4 hjorner skrives nye linjer med z(4), z(5) osv. inn
    her
//////////

varf b(pN,N) = int2d(Thm) (pN*N);

matrix OP= op(Vhm,Vhm, solver=Crout, factorize=1);
matrix B= b(Vhm,Vhm, solver=CG, eps=1e-20);
real[int] ev(nev); //Vektor for egenverdiene
Vhm[int] eV(nev); //Vektor for egenmodene
int k=EigenValue(OP,B, sym=true, sigma=sigma, value=ev, vector=eV, tol=1e
    -10,maxit=0,ncv=0);

cout<<nev<<" egenfrekvenser:"<<endl;

//Prosedyre for a velge ut egenverdiene i naerheten av
    senterfrekvensen f_teller
for (i=0;i<nev;i++){
    ef(i) = sqrt(ev(i))*c/(2*pi);
    if (abs(ef(i)-f) <= res/2*overlapp){//*overlapp for a fa med
        alle moder, da de flytter pa seg for hver iterasjon
        real temp = ef(i);
        string tittelPlot, tittelFil;
        cout<<ef(i)<<endl;
        modfrekvens(modeteller) = temp;
        psi[modeteller] = eV[i];
    }
}

```

```

        modeteller = modeteller+1; //kan ikke bruke ++ pga
        modeteller ikke er int
    }
}
//Slutt pa 2D-simulering

//1D-simulering:
varf op1D(pN1D,N1D)= int2d(Thm1D) (dx(pN1D)*dx(N1D) + dy(pN1D)*dy(N1D)
    ) - sigma* pN1D*N1D
    +int1d(Thm1D,gulv) (N1D*c*sqrt(sigma)*rho*pN1D/imag(zgulv))
    +int1d(Thm1D,tak) (N1D*c*sqrt(sigma)*rho*pN1D/imag(ztak));

varf b1D(pN1D,N1D) = int2d(Thm1D) (pN1D*N1D);

matrix OP1D= op1D(Vhm1D,Vhm1D,solver=Crout,factorize=1);
matrix B1D= b1D(Vhm1D,Vhm1D,solver=CG,eps=1e-20);
real[int] ev1D(nev);
Vhm1D[int] eV1D(nev);
int k1D=EigenValue(OP1D,B1D,sym=true,sigma=sigma,value=ev1D,vector=
    eV1D,tol=1e-10,maxit=0,ncv=0);

for(i=0; i<nev; i++){
    ef1D(i) = sqrt(ev1D(i))*c/(2*pi);
    if (abs(ef1D(i)-f) <= res/2*overlapp){
        real temp = ef1D(i);
        modedefrekvens1D(modeteller1D) = temp;
        psi1D[modeteller1D] = eV1D[i];
        modeteller1D = modeteller1D+1; //kan ikke bruke ++ pga
        modeteller ikke er int
    }
}
//Slutt pa 1D-simulering

frekvens(teller) = f;

cout << endl << "Frekvens = " << f << " Hz" << endl;

////////////////////////////////////
} //Slutt pa den store for-lokka
////////////////////////////////////

// Evaluering av modene, vurder hvilke som er unike og gyldige:
int antgyldig = 0;
for (i=0; i<modeteller; i++){
    int temp;
    tittelPlot = "f = " + modedefrekvens(i) + " Hz";
    //Justerer fortegn, gjøres her for a lette evalueringen, selv om det
    da blir flere fortegnjusteringer enn nødvendig.

```

```

psi[i] = psi[i]*psi[i](sourcecx, sourcecy)/abs(psi[i](sourcecx,
sourcecy));
plot(psi[i],wait=0,fill=1,value=1,nbiso=30, cmm=tittelPlot);
cout << endl << "Trykk 0 hvis moden allerede er plottet en gang
tidligere eller hvis trykket har samme fortegn i hele rommet.
Trykk 1 dersom plottet er unikt og gyldig" << endl;
temp = 1; //for eventuell bypass av godkjenning av mode
cin >> temp; //Kommenteres ut for bypass av godkjenning av moder
, i sa fall godkjennes alt
if (temp == 0){
cout << "Mode ikke godkjent." << endl;
modedefrekvens(i) = -1; //Ikke-gyldige egenfrekvenser far
frekvensen -1.
}
else{
antgyldig++;
cout << "modedefrekvens(" << i << ") = " << modedefrekvens(i) <<
" Hz godkjent." << endl;
}
}
int antgyldig1D = 0;
real diff = modedefrekvens1D(0); //Sikrer at 0Hz-losninger registreres
som ugyldige
for (i=0; i<modeteller1D; i++){
int temp;
tittelPlot = "f = " + modedefrekvens1D(i) + " Hz";
plot(psi1D[i],wait=0,fill=1,value=1,nbiso=30, cmm=tittelPlot);
if (i>0)
diff = modedefrekvens1D(i)-modedefrekvens1D(i-1);
if (diff < (c/(4*h))){//20}{ //1/4 bolgelengde valgt som
minimumavstand mellom moder i z-retning
modedefrekvens1D(i) = -1;
}
else{
antgyldig1D++;
}
}
// Slutt pa modeevaluering

//Lager vektorer med de gyldige egenfrekvensene og gyldige psi, inkl
psi0, altsa konstant trykk:
Vhm [int] psigyldig (antgyldig+1);
psigyldig[0] = sqrt(1/gulvareal); // Normalisering, \cite[s. 287]{
pierce}
real [int] modfreggyldig(antgyldig);
int tempteller = 0;

for (i=0; i<modeteller; i++){
if (modedefrekvens(i) > 0){

```



```

        modfreqgyldig(tempteller) = modefrekvens(i);
        psigyldig[tempteller+1] = psi[i];
        tempteller++;
    }
}

Vhm1D [int] psigyldig1D (antgyldig1D+1);
psigyldig1D[0] = sqrt(1/(h*b));
real [int] modfreqgyldig1D(antgyldig1D);
tempteller = 0;
for (i=0; i<modeteller1D; i++){
    if (modefrekvens1D(i) > 0){
        modfreqgyldig1D(tempteller) = modefrekvens1D(i);
        psigyldig1D[tempteller+1] = psilD[i]*psilD[i] (b/2, sourcez) /
            abs(psilD[i] (b/2, sourcez)); //Gir positivt fortegn ved
            kilden.
        tempteller++;
    }
}

//Lager matrise med kombinerte 3D-egenfrekvenser, pa matriseformen
    matrise(antall 2D-frekvenser, antall 1D-frekvenser):
real [int,int] modfrekvens3D (antgyldig+1, antgyldig1D+1);
modfrekvens3D(0,0) = 0;
for (i=0; i<antgyldig; i++){
    modfrekvens3D(i+1,0) = modfreqgyldig(i);
}
for (j=0; j<antgyldig1D; j++){
    modfrekvens3D(0,j+1) = modfreqgyldig1D(j);
}

for (i=0; i<antgyldig; i++){
    for (j=0; j<antgyldig1D; j++){
        modfrekvens3D(i+1,j+1) = sqrt(modfreqgyldig(i)^2 +
            modfreqgyldig1D(j)^2);
    }
}
//Matrise med egenfrekvenser fullfort

int antgyldig3D = (antgyldig+1)*(antgyldig1D+1)-1;

//Beregning av matrise med dempingskonstanter:
real [int,int] dempmatrise (antgyldig+1, antgyldig1D+1);
real A; //Absorpsjonsareal
real felleskomp;
int n;
dempmatrise(0,0) = 10^30;
for (i=0; i<antgyldig+1; i++){
    for (j=0; j<antgyldig1D+1; j++){

```

```

A = 0;
cout << endl << "modfrekvens3D(" << i << ", " << j << ") =
" << modfrekvens3D(i, j);
zgulv = Rgulv+1i*(2*pi*modfrekvens3D(i, j)*mgulv - rho*c
^2/(2*pi*(modfrekvens3D(i, j)+1e-30)*dgulv)); //+1e-30
er for a unnga 0-divisjon
ztak = Rtak+1i*(2*pi*modfrekvens3D(i, j)*mtak - rho*c
^2/(2*pi*(modfrekvens3D(i, j)+1e-30)*dtak));
for (n=0; n<corners; n++){
z(n) = R(n)+1i*(2*pi*modfrekvens3D(i, j)*m(n) - rho*c
^2/(2*pi*(modfrekvens3D(i, j)+1e-30)*d(n)));
felleskomp = (real(z(n)))^2 + (imag(z(n)))^2 + (rho*
c)^2;
A += h*1(n)*(1-(felleskomp-2*real(z(n))*rho*c)/(
felleskomp+2*real(z(n))*rho*c));
}
felleskomp = (real(zgulv))^2 + (imag(zgulv))^2 + (rho*c)
^2;
A += gulfvareal*(1-(felleskomp-2*real(zgulv)*rho*c)/(
felleskomp+2*real(zgulv)*rho*c));
felleskomp = (real(ztak))^2 + (imag(ztak))^2 + (rho*c)
^2;
A += gulfvareal*(1-(felleskomp-2*real(ztak)*rho*c)/(
felleskomp+2*real(ztak)*rho*c));
dempmatrise(i, j) = c*A/(8*volum);
}
}
// Dempingsmatrise ferdig

real [int] demp(antgyldig3D);

for (i=0; i<antgyldig; i++){
cout << "Egenfrekvens " << i << " = " << modfreqgyldig(i) << "
Hz" << endl;
}

for (i=0; i<modeteller1D; i++){
cout << "Egenfrekvens1D " << i << " = " << modefrekvens1D(i) <<
" Hz" << endl;
}

Vhm psikomb;
Vhm1D psikomb1D;
int plotTeller = 0;
real [int] frekvensut (antittransfer);
complex [int, int] coefs (antgyldig+1, antgyldig1D+1);
coefs(0, 0) = 0;
complex [int] finalp (antittransfer);
real sf; //solution frequency

```

```

real An; //psi(kilde) * psi(mottaker)
f0 = f0transfer;
fend = fendtransfer;

real [int] coefskomb2D (antgyldig+1);
coefskomb2D = 0;
real [int] coefskomb1D (antgyldig1D+1);
coefskomb1D = 0;

for (n=0; n<antittransfer; n++){
    psikomb = 0;
    psikomb1D = 0;
    coefs = 0;
    coefskomb2D = 0;
    coefskomb1D = 0;
    finalp(n) = 0;
    sf = f0+(fend-f0)/(antittransfer-1)*n;
    for (i=0; i<antgyldig+1; i++){
        for (j=0; j<antgyldig1D+1; j++){
            An = psigyldig[i](sourcex, sourcey)*psigyldig[i](recx
                , recy)*psigyldig1D[j](b/2, sourcez)*psigyldig1D[j]
                (b/2, recz);
            coefs(i, j) = An/((2*pi*sf)^2 - (2*pi*modfrekvens3D(i
                , j))^2 - 2i*dempmatrise(i, j)*2*pi*modfrekvens3D(i
                , j));
            finalp(n) += coefs(i, j);
            if (n%plotres==0){
                coefskomb2D(i) += abs(coefs(i, j));
                coefskomb1D(j) += abs(coefs(i, j));
            }
        }
    }
    if (n%plotres==0){
        for (i=0; i<antgyldig+1; i++){
            psikomb = psikomb + coefskomb2D(i)*psigyldig[i];
        }
        for (j=0; j<antgyldig1D+1; j++){
            psikomb1D = psikomb1D + coefskomb1D(j)*psigyldig1D[j];
        }
        tittelPlot = "f = " + sf + " Hz";
        tittelFil = saveLoc + "/eigenplots2D/freqnr" + plotTeller +
            ".jpg";
        plot(psikomb, wait=0, ps=tittelFil, fill=1, value=1, nbiso=100,
            cmm=tittelPlot);
        tittelFil = saveLoc + "/eigenplots1D/freqnr" + plotTeller +
            ".jpg";
        plot(psikomb1D, wait=0, ps=tittelFil, fill=1, value=1, nbiso=100,
            cmm=tittelPlot);
        plotTeller++;
    }
}

```

```

    }
    frekvensut(n) = sf;
}

tittelFil = saveLoc + "/transferfunction.txt";
{
ofstream p(tittelFil);
for (i=0; i<antittransfer; i++){
    p << abs(finalp(i)) << endl << frekvensut(i) << endl;
}
}

tittelFil = saveLoc + "/eigenfreq.txt";
{
ofstream ef(tittelFil);
for (i=0; i<antgyldig+1; i++){
    for (j=0; j<antgyldig1D+1; j++){
        ef << " | " << modfrekvens3D(i, j);
    }
    ef << " |" << endl;
}
}

cout << endl << "antgyldig2D: " << antgyldig << endl << "antgyldig1D
: " << antgyldig1D << endl;
cout << endl << "Modedefrekvens og dempematrise i (0,0): " <<
    modfrekvens3D(0,0) << " — " << dempmatrise(0,0) << endl;

tittelFil = saveLoc + "/info.txt";
{
ofstream info(tittelFil);
info << antittransfer << endl << plotres << endl << f0 << endl <<
    fend << endl;
}

```

D.3 3D-modell med brukerinntut gjennom dialogbokser

```
int corners, i, j;
real [int] x1(100), y1(100); //100 er satt som maks antall hjorner,
    dette kan endres

////////////////////////////////////
//Input av data:

//Kilde- og mottaker-koordinater:
real sourcex, sourcey, sourcez, recx, recy, recz;

real h = 2; //Takhoyde

corners = 4; //Antall hjorner

real [int] xcorner(corners), ycorner(corners), R(corners), m(corners
    ), d(corners);

//x- og y-koordinater og impedansparametre i gulv-planet:
for (i=0;i<corners;i++){
    cout << "Enter x- then y-coordinate of corner " << i+1 <<": ";
    cin >> xcorner(i) >> ycorner(i);
    cout << "(" << xcorner(i) << ", " << ycorner(i) << ")" << endl;

    cout << "Enter R, m and d-parameters for wall " << i+1 << ": ";
    cin >> R(i) >> m(i) >> d(i);
    cout << "R = " << R(i) << ", m = " << m(i) << ", d = " << d(i)
        << endl << endl;
}

//Romhoyde:
cout << "Enter room height: ";
cin >> h;
cout << "Room height h = " << h << endl << endl;
//Kilde:
cout << "Enter x-, then y- then z-coordinate of the sound source: ";
cin >> sourcex >> sourcey >> sourcez;
cout << "Source position is: (" << sourcex << ", " << sourcey << ",
    " << sourcez << ")" << endl;
//Mottaker:
cout << "Enter x-, then y- then z-coordinate of the receiver: ";
cin >> recx >> recy >> recz;
cout << "Receiver position is: (" << recx << ", " << recy << ", " <<
    recz << ")" << endl << endl;
```

```

real Rgulv, mgulv, dgulv, Rtak, mtak, dtak;
//Gulvimpedans:
cout << "Enter R, m and d-parameters for the floor: ";
cin >> Rgulv >> mgulv >> dgulv;
cout << "R = " << Rgulv << ", m = " << mgulv << ", d = " << dgulv <<
    endl;
//Takimpedans:
cout << "Enter R, m and d-parameters for the ceiling: ";
cin >> Rtak >> mtak >> dtak;
cout << "R = " << Rtak << ", m = " << mtak << ", d = " << dtak <<
    endl << endl;

//Slutt pa input av romdata
////////////////////////////////////

//Input av simuleringssparametre ved beregning av egenfrekvenser og -
moder:
real f0, fend;
cout << "Enter lower and upper frequency limits for the calculation
    of eigenfrequencies: ";
cin >> f0 >> fend;
cout << endl << "Eigenfrequencies calculated for f = [" << f0 << ",
    " << fend << "]" << endl << endl;

real antit = (fend-f0)*10+1; //Antall iterasjoner, opplosning blir
    ca. (fend-f0)/antit
real overlapp = 1.4; //Stort overlapp oker sannsynlighet for a fa
    med alle moder, men flere moder registreres dobbelt.

//Input av parametre for generering av transferfunksjon og modeplot:
real f0transfer, fendtransfer;
cout << "Enter lower and upper frequency limits for the
    transferfunction.";
cout << endl << "Set these well within the limits of calculated
    eigenfrequencies";
cin >> f0transfer >> fendtransfer;
cout << endl << "Transferfunction calculated for f = [" <<
    f0transfer << ", " << fendtransfer << "]" << endl << endl;

string saveLoc;

cout << "Enter full folder address (\\"C:/Users/.../FreeFemsimulation
    \") for simulation results to be stored: ";
cout << endl << "Remember to enter an existing address, as FreeFem++
    cannot create folders.";
cout << endl << "If you don't, the simulation results will not be
    stored.";
cin >> saveLoc;
cout << endl << "Saving files in: " << saveLoc << endl << endl;

```

```

//Slutt pa input av data
////////////////////////////////////

real antittransfer = (fendtransfer-f0transfer)*10+1;
real plotres = 10; //Forhold mellom opplosning i transferfunksjon og
modeplot. Ma vaere heltall større enn 0.

real c = 343; //[m/s]
real rho = 1.21;

//2D-rom:
border b0(t=0,1) {x=xcorner(0)+(xcorner(1)-xcorner(0))*t; y=ycorner
(0)+(ycorner(1)-ycorner(0))*t;}
border b1(t=0,1) {x=xcorner(1)+(xcorner(2)-xcorner(1))*t; y=ycorner
(1)+(ycorner(2)-ycorner(1))*t;}
border b2(t=0,1) {x=xcorner(2)+(xcorner(3)-xcorner(2))*t; y=ycorner
(2)+(ycorner(3)-ycorner(2))*t;}
border b3(t=0,1) {x=xcorner(3)+(xcorner(0)-xcorner(3))*t; y=ycorner
(3)+(ycorner(0)-ycorner(3))*t;}
////////////////////////////////////
//Med flere hjørner skrives border b4(t=0,1) ... osv nedover her
////////////////////////////////////

//1D-romdata:
real b = 0.1;
border gulv(t=0,1) {x=b*t; y=0;}
border tak(t=0,1) {x=b-b*t; y=h;}
border vegg1(t=0,1) {x=b; y=h*t;}
border vegg2(t=0,1) {x=0; y=h-h*t;}

// Beregning av romvolum
real [int] l(corners); // lengde pa hver vegg
for (i=0;i<corners;i++)
{
l(i) = sqrt((xcorner((i+1)%corners)-xcorner(i))^2+(ycorner((i+1)%
corners)-ycorner(i))^2);
}
//Formel s. 46 i Rottmann
real golvareal;
for (i=0; i<(corners-1); i++){
golvareal += 0.5*(xcorner(i)*(ycorner((i+1)%corners)-ycorner((i
-1+corners)%corners)));
}
golvareal = abs(golvareal);
real volum = golvareal*h;
// Slutt pa volumberegning

//Genererer mesh:

```

```

real meshfact = 10; //Angir antall noder pr bolgelengde
real ms = meshfact*fend/c; //ms star for mesh size
//////////
mesh Thm=buildmesh(b0(ceil(l(0)*ms)) + b1(ceil(l(1)*ms)) + b2(ceil(l
(2)*ms)) + b3(ceil(l(3)*ms)));
// Skriv inn i linjen over, for siste parentes, + b4(ceil(l(4)*ms))
osv, dersom flere hjorner benyttes
//////////
mesh Thm1D = buildmesh(gulv(ceil(b*ms)) + vegg1(ceil(h*ms)) + tak(
ceil(b*ms)) + vegg2(ceil(h*ms)));

complex [int] z(corners);
complex zgulv, ztak;

fespace Vhm(Thm,P2);
Vhm<complex> pN,N;

fespace Vhm1D(Thm1D,P2);
Vhm1D<complex> pN1D, N1D;

Vhm [int] psi(100); //Maks antall egenmoder i 2D satt til 100
real [int] modedefrekvens(100);
Vhm1D [int] psi1D(50); //Maks antall egenmoder i 1D satt til 50
real [int] modedefrekvens1D(50);

real modeteller = 0; //Teller for modenummerering
modedefrekvens = 0;

real modeteller1D = 0;
modedefrekvens1D = 0;

real res = (fend-f0)/(antit-1); //Frekvensopplosning

real [int] transfer(antit);
real [int] frekvens(antit);

int nev=10; //Antall egenverdier som beregnes i hver iterasjon
real[int] ef(nev); //ef er egenfrekvens
real[int] ef1D(nev);

real teller = 0; //Teller for store for-lokke
real f = f0;

string tittelPlot, tittelFil;

//////////
// Start store for-lokke:
for (teller=0; teller<antit; teller=teller+1)
{

```



```

if (antit > 1)
f = (fend-f0)/(antit-1)*teller+f0;

//Beregning av impedanseverdier:
complex zgulv = Rgulv+1i*(2*pi*f*mgulv - rho*c^2/(2*pi*f*dgulv));
if (imag(zgulv) == 0)
    zgulv += 1e-10i;
complex ztak = Rtak+1i*(2*pi*f*mtak - rho*c^2/(2*pi*f*dtak));
if (imag(ztak) == 0)
    ztak += 1e-10i;
for (i=0;i<corners;i++){
    z(i) = R(i)+1i*(2*pi*f*m(i) - rho*c^2/(2*pi*f*d(i)));
    if (imag(z(i)) == 0)
        z(i) += 1e-10i;
}

cout << endl << "ztak = " << ztak << endl;

real sigma = (2*pi*f/c)^2; //Sigma er verdien som egenverdier skal
    beregnes i naerheten av
//2D-simulering:
// OP = A - sigma B ; // the shifted matrix
varf op(pN,N)= int2d(Thm) (dx(pN)*dx(N) + dy(pN)*dy(N) - sigma* pN*N)
    +int1d(Thm,b0) (N*c*sqrt(sigma)*rho*pN/imag(z(0)))
    +int1d(Thm,b1) (N*c*sqrt(sigma)*rho*pN/imag(z(1)))
    +int1d(Thm,b2) (N*c*sqrt(sigma)*rho*pN/imag(z(2)))
    +int1d(Thm,b3) (N*c*sqrt(sigma)*rho*pN/imag(z(3)));
//////////
//Med flere enn 4 hjorner skrives tilsvarende linjer med z(4), z(5)
    osv. inn her
//////////

varf b(pN,N) = int2d(Thm) (pN*N);

matrix OP= op(Vhm,Vhm,solver=Crout,factorize=1);
matrix B= b(Vhm,Vhm,solver=CG,eps=1e-20);
real[int] ev(nev); //Vektor for egenverdiene
Vhm[int] eV(nev); //Vektor for egenmodene
int k=EigenValue(OP,B,sym=true,sigma=sigma,value=ev,vector=eV,tol=1e
    -10,maxit=0,ncv=0);

cout<<nev<<" egenfrekvenser:"<<endl;

//Prosedyre for a velge ut egenverdiene i naerheten av
    senterfrekvensen fteller
for(i=0;i<nev;i++){
    ef(i) = sqrt(ev(i))*c/(2*pi);
    if (abs(ef(i)-f) <= res/2*overlapp){//*overlapp for a fa med
        alle moder, da de flytter pa seg for hver iterasjon
    }
}

```

```

        real temp = ef(i);
        string tittelPlot, tittelFil;
        cout<<ef(i)<<endl;
        modeteller = modeteller+1; //kan ikke bruke ++ pga
            modeteller ikke er int
    }
}
//Slutt pa 2D-simulering

//1D-simulering:
varf op1D(pN1D,N1D) = int2d(Thm1D) (dx(pN1D)*dx(N1D) + dy(pN1D)*dy(N1D)
    ) - sigma* pN1D*N1D
    +int1d(Thm1D,gulv) (N1D*c*sqrt(sigma)*rho*pN1D/imag(zgulv))
    +int1d(Thm1D,tak) (N1D*c*sqrt(sigma)*rho*pN1D/imag(ztak));

varf b1D(pN1D,N1D) = int2d(Thm1D) (pN1D*N1D);

matrix OP1D= op1D(Vhm1D,Vhm1D,solver=Crout,factorize=1);
matrix B1D= b1D(Vhm1D,Vhm1D,solver=CG,eps=1e-20);
real[int] ev1D(nev);
Vhm1D[int] eV1D(nev);
int k1D=EigenValue(OP1D,B1D,sym=true,sigma=sigma,value=ev1D,vector=
    eV1D,tol=1e-10,maxit=0,ncv=0);

for(i=0; i<nev; i++){
    ef1D(i) = sqrt(ev1D(i))*c/(2*pi);
    if (abs(ef1D(i)-f) <= res/2*overlapp){
        real temp = ef1D(i);
        modeteller1D = modeteller1D+1; //kan ikke bruke ++ pga
            modeteller ikke er int
    }
}
//Slutt pa 1D-simulering

frekvens(teller) = f;

cout << endl << "Frekvens = " << f << " Hz" << endl;

//////////
} //Slutt pa den store for-lokka
//////////

// Evaluering av modene, vurder hvilke som er unike og gyldige:
int antgyldig = 0;
for (i=0; i<modeteller; i++){

```

```

    int temp;
    tittelPlot = "f = " + modefrekvens(i) + " Hz";
    //Justerer fortegn, gjøres her for å lette evalueringen, selv om det
    da blir flere fortegnjusteringer enn nødvendig.
    psi[i] = psi[i]*psi[i](sourceX, sourceY)/abs(psi[i](sourceX,
        sourceY));
    plot(psi[i], wait=0, fill=1, value=1, nbiso=30, cmm=tittelPlot);
    cout << endl << "Trykk 0 hvis moden allerede er plottet en gang
        tidligere eller hvis trykket har samme fortegn i hele rommet.
        Trykk 1 dersom plottet er unikt og gyldig" << endl;
    temp = 1; //for eventuell bypass av godkjenning av mode
    cin >> temp; //Kommenteres ut for bypass av godkjenning av moder
    , i så fall godkjennes alt
    if (temp == 0){
        cout << "Mode ikke godkjent." << endl;
        modefrekvens(i) = -1; //Ikke-gyldige egenfrekvenser får
            frekvensen -1.
    }
    else{
        antgyldig++;
        cout << "modefrekvens(" << i << ") = " << modefrekvens(i) <<
            " Hz godkjent." << endl;
    }
}
int antgyldig1D = 0;
real diff = modefrekvens1D(0); //Sikrer at 0Hz-losninger registreres
som ugyldige
for (i=0; i<modeteller1D; i++){
    int temp;
    tittelPlot = "f = " + modefrekvens1D(i) + " Hz";
    plot(psi1D[i], wait=0, fill=1, value=1, nbiso=30, cmm=tittelPlot);
    if (i>0)
        diff = modefrekvens1D(i)-modefrekvens1D(i-1);
    if (diff < (c/(4*h))){//20}{ //1/4 bolgelengde valgt som
        minimumavstand mellom moder i z-retning
        modefrekvens1D(i) = -1;
    }
    else{
        antgyldig1D++;
    }
}
// Slutt på modeevaluering

//Lager vektorer med de gyldige egenfrekvensene og gyldige psi, inkl
    psi0, altså konstant trykk:
Vhm [int] psigyldig (antgyldig+1);
psigyldig[0] = sqrt(1/gulvareal); // Normalisering, \cite[s. 287]{
    pierce}
real [int] modfreqgyldig(antgyldig);

```

```

int tempteller = 0;

for (i=0; i<modeteller; i++){
    if (modedefrekvens(i) > 0){
        moddefreggyldig(tempteller) = modedefrekvens(i);
        psigyldig[tempteller+1] = psi[i];
        tempteller++;
    }
}

Vhm1D [int] psigyldig1D (antgyldig1D+1);
psigyldig1D[0] = sqrt(1/(h*b));
real [int] moddefreggyldig1D(antgyldig1D);
tempteller = 0;
for (i=0; i<modeteller1D; i++){
    if (modedefrekvens1D(i) > 0){
        moddefreggyldig1D(tempteller) = modedefrekvens1D(i);
        psigyldig1D[tempteller+1] = psi1D[i]*psi1D[i] (b/2, sourcez)/
            abs(psi1D[i] (b/2, sourcez)); //Gir positivt fortegn ved
            kilden.
        tempteller++;
    }
}

//Lager matrise med kombinerte 3D-egenfrekvenser, pa matriseformen
    matrise(antall 2D-frekvenser, antall 1D-frekvenser):
real [int,int] moddefrekvens3D (antgyldig+1, antgyldig1D+1);
moddefrekvens3D(0,0) = 0;
for (i=0; i<antgyldig; i++){
    moddefrekvens3D(i+1,0) = moddefreggyldig(i);
}
for (j=0; j<antgyldig1D; j++){
    moddefrekvens3D(0,j+1) = moddefreggyldig1D(j);
}

for (i=0; i<antgyldig; i++){
    for (j=0; j<antgyldig1D; j++){
        moddefrekvens3D(i+1,j+1) = sqrt(moddefreggyldig(i)^2 +
            moddefreggyldig1D(j)^2);
    }
}
//Matrise med egenfrekvenser fullfort

int antgyldig3D = (antgyldig+1)*(antgyldig1D+1)-1;

//Beregning av matrise med dempingskonstanter:
real [int,int] dempmatrise (antgyldig+1, antgyldig1D+1);
real A; //Absorpsjonsareal
real felleskomp;

```

```

int n;
dempmatrise(0,0) = 10^30;
for (i=0; i<antgyldig+1; i++){
    for (j=0; j<antgyldig1D+1; j++){
        A = 0;
        cout << endl << "modfrekvens3D(" << i << ", " << j << ") = "
            << modfrekvens3D(i,j);
        zgulv = Rgulv+1i*(2*pi*modfrekvens3D(i,j)*mgulv - rho*c
            ^2/(2*pi*(modfrekvens3D(i,j)+1e-30)*dgulv)); //+1e-30
            er for a unnga 0-divisjon
        ztak = Rtak+1i*(2*pi*modfrekvens3D(i,j)*mtak - rho*c
            ^2/(2*pi*(modfrekvens3D(i,j)+1e-30)*dtak));
        for (n=0; n<corners; n++){
            z(n) = R(n)+1i*(2*pi*modfrekvens3D(i,j)*m(n) - rho*c
                ^2/(2*pi*(modfrekvens3D(i,j)+1e-30)*d(n)));
            felleskomp = (real(z(n)))^2 + (imag(z(n)))^2 + (rho*c)
                ^2;
            A += h*1(n)*(1-(felleskomp-2*real(z(n))*rho*c)/(
                felleskomp+2*real(z(n))*rho*c));
        }
        felleskomp = (real(zgulv))^2 + (imag(zgulv))^2 + (rho*c)
            ^2;
        A += gulvareal*(1-(felleskomp-2*real(zgulv))*rho*c)/(
            felleskomp+2*real(zgulv)*rho*c));
        felleskomp = (real(ztak))^2 + (imag(ztak))^2 + (rho*c)
            ^2;
        A += gulvareal*(1-(felleskomp-2*real(ztak))*rho*c)/(
            felleskomp+2*real(ztak)*rho*c));
        dempmatrise(i,j) = c*A/(8*volum);
    }
}
// Dampingsmatrise ferdig

real [int] demp(antgyldig3D);

for (i=0; i<antgyldig; i++){
    cout << "Egenfrekvens " << i << " = " << modfreqgyldig(i) << "
        Hz" << endl;
}

for (i=0; i<modeteller1D; i++){
    cout << "Egenfrekvens1D " << i << " = " << modefrekvens1D(i) <<
        " Hz" << endl;
}

Vhm psikomb;
Vhm1D psikomb1D;
int plotTeller = 0;
real [int] frekvensut (antittransfer);

```

```

complex [int,int] coefs (antgyldig+1,antgyldig1D+1);
coefs(0,0) = 0;
complex [int] finalp (antittransfer);
real sf; //solution frequency
real An; //psi(kilde) * psi(mottaker)
f0 = f0transfer;
fend = fendtransfer;

real [int] coefskomb2D (antgyldig+1);
coefskomb2D = 0;
real [int] coefskomb1D (antgyldig1D+1);
coefskomb1D = 0;

for (n=0; n<antittransfer; n++){
    psikomb = 0;
    psikomb1D = 0;
    coefs = 0;
    coefskomb2D = 0;
    coefskomb1D = 0;
    finalp(n) = 0;
    sf = f0+(fend-f0)/(antittransfer-1)*n;
    for (i=0; i<antgyldig+1; i++){
        for (j=0; j<antgyldig1D+1; j++){
            An = psigyldig[i](sourcex,sourcey)*psigyldig[i](recx
                ,recy)*psigyldig1D[j](b/2,sourcez)*psigyldig1D[j]
                (b/2,recz);
            coefs(i,j) = An/((2*pi*sf)^2 - (2*pi*modfrekvens3D(i
                ,j))^2 - 2i*dempmatrise(i,j)*2*pi*modfrekvens3D(i
                ,j));
            finalp(n) += coefs(i,j);
            if (n%plotres==0){
                coefskomb2D(i) += abs(coefs(i,j));
                coefskomb1D(j) += abs(coefs(i,j));
            }
        }
    }
    if (n%plotres==0){
        for (i=0; i<antgyldig+1; i++){
            psikomb = psikomb + coefskomb2D(i)*psigyldig[i];
        }
        for (j=0; j<antgyldig1D+1; j++){
            psikomb1D = psikomb1D + coefskomb1D(j)*psigyldig1D[j];
        }
        tittelPlot = "f = " + sf + " Hz";
        tittelFil = saveLoc + "/eigenplots2D/freqnr" + plotTeller +
            ".jpg";
        plot(psikomb,wait=0,ps=tittelFil,fill=1,value=1,nbiso=100,
            cmm=tittelPlot);
    }
}

```

```

        tittelFil = saveLoc + "/eigenplots1D/freqnr" + plotTeller +
            ".jpg";
        plot(psikomb1D,wait=0,ps=tittelFil,fill=1,value=1,nbiso=100,
            cmm=tittelPlot);
        plotTeller++;
    }
    frekvensut(n) = sf;
}

tittelFil = saveLoc + "/transferfunksjon.txt";
{
ofstream p(tittelFil);
for (i=0; i<antittransfer; i++){
    p << abs(finalp(i)) << endl << frekvensut(i) << endl;
}
}

cout << endl << "antgyldig2D: " << antgyldig << endl << "antgyldig1D
: " << antgyldig1D << endl;
cout << endl << "Modedefrekvens og dempematrise i (0,0): " <<
    modfrekvens3D(0,0) << " — " << dempmatrise(0,0) << endl;

tittelFil = saveLoc + "/info.txt";
{
ofstream info(tittelFil);
info << antittransfer << endl << plotres << endl << f0 << endl <<
    fend << endl;
}

```

E COMSOL-simulering

COMSOL-simuleringen av hardt rom, eksportert til MATLAB-kode

```
function out = model
%
% Harde vegger – til vedlegg.m
%
% Model exported on Jun 12 2013, 13:06 by COMSOL 4.2.0.150.

import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');

model.modelPath('D:\klami\Cmsol-filer masteroppgave\Til rapport');

model.modelNode.create('mod1');

model.geom.create('geom1', 3);

model.mesh.create('mesh1', 'geom1');

model.physics.create('acpr', 'PressureAcoustics', 'geom1');

model.study.create('std1');
model.study('std1').feature.create('freq', 'Frequency');

model.geom('geom1').feature.create('blk1', 'Block');
model.geom('geom1').feature('blk1').setIndex('size', '2.4', 0);
model.geom('geom1').feature('blk1').setIndex('size', '1.5', 1);
model.geom('geom1').feature('blk1').setIndex('size', '2', 2);
model.geom('geom1').runAll;
model.geom('geom1').run('blk1');
model.geom('geom1').feature.create('pt1', 'Point');
model.geom('geom1').feature('pt1').setIndex('p', '0.01', 0);
model.geom('geom1').feature('pt1').setIndex('p', '0.01', 1);
model.geom('geom1').feature('pt1').setIndex('p', '0.01', 2);
model.geom('geom1').runAll;

model.view('view1').set('transparency', 'on');

model.geom('geom1').run('pt1');
model.geom('geom1').feature.create('pt2', 'Point');
model.geom('geom1').feature('pt2').setIndex('p', '2.39', 0);
model.geom('geom1').feature('pt2').setIndex('p', '1.49', 1);
```



```

model.geom('geom1').feature('pt2').setIndex('p', '1.99', 2);
model.geom('geom1').runAll;
model.geom('geom1').run;

model.study('std1').feature('freq').set('geomselection', 'geom1');
model.study('std1').feature('freq').set('physselection', 'acpr');
model.study('std1').feature('freq').set('plist', 'range(20,1,200)');

model.physics('acpr').feature.create('pps1', 'PowerPointSource', 0);
model.physics('acpr').feature('pps1').selection.set([5]);
model.physics('acpr').feature('pps1').set('P', 1, '1');

model.study('std1').feature('freq').set('geomselection', 'geom1');
model.study('std1').feature('freq').set('physselection', 'acpr');

model.sol.create('soll1');
model.sol('soll1').study('std1');
model.sol('soll1').feature.create('st1', 'StudyStep');
model.sol('soll1').feature('st1').set('study', 'std1');
model.sol('soll1').feature('st1').set('studystep', 'freq');
model.sol('soll1').feature.create('v1', 'Variables');
model.sol('soll1').feature.create('s1', 'Stationary');
model.sol('soll1').feature('s1').feature.create('p1', 'Parametric');
model.sol('soll1').feature('s1').feature.remove('pDef');
model.sol('soll1').feature('s1').feature('p1').set('pname', 'freq');
model.sol('soll1').feature('s1').feature('p1').set('plist', 'range
(20,1,200)');
model.sol('soll1').feature('s1').feature('p1').set('plot', 'off');
model.sol('soll1').feature('s1').feature('p1').set('probesel', 'all')
;
model.sol('soll1').feature('s1').feature('p1').set('probes', {});
model.sol('soll1').feature('s1').feature('p1').set('control', 'freq')
;
model.sol('soll1').feature('s1').set('control', 'freq');
model.sol('soll1').feature('s1').feature('aDef').set('complexfun',
true);
model.sol('soll1').feature('s1').feature.create('fc1', 'FullyCoupled'
);
model.sol('soll1').feature('s1').feature.remove('fcDef');
model.sol('soll1').attach('std1');

model.result.create('pg1', 3);
model.result('pg1').set('data', 'dset1');
model.result('pg1').feature.create('surf1', 'Surface');
model.result('pg1').feature('surf1').set('expr', {'acpr.p-t'});
model.result('pg1').name('Acoustic Pressure (acpr)');
model.result.create('pg2', 3);
model.result('pg2').set('data', 'dset1');
model.result('pg2').feature.create('surf1', 'Surface');

```

```

model.result('pg2').feature('surfl').set('expr', {'acpr.Lp'});
model.result('pg2').name('Sound Pressure Level (acpr)');
model.result.create('pg3', 3);
model.result('pg3').set('data', 'dset1');
model.result('pg3').feature.create('isol', 'Isosurface');
model.result('pg3').feature('isol').set('expr', {'acpr.p-t'});
model.result('pg3').feature('isol').set('number', '10');
model.result('pg3').name('Acoustic Pressure, Isosurfaces (acpr)');
model.result('pg1').run;

model.material.create('mat1');
model.material.remove('mat1');
model.material.create('mat1');
model.material('mat1').name('Air');
model.material('mat1').set('family', 'air');
model.material('mat1').propertyGroup('def').set('relpermeability', '1');
model.material('mat1').propertyGroup('def').set('relpermittivity', '1');
model.material('mat1').propertyGroup('def').set('dynamicviscosity', 'eta(T[1/K]) [Pa*s]');
model.material('mat1').propertyGroup('def').set('ratioofspecificeat', '1.4');
model.material('mat1').propertyGroup('def').set('electricconductivity', '0[S/m]');
model.material('mat1').propertyGroup('def').set('heatcapacity', 'Cp(T[1/K]) [J/(kg*K)]');
model.material('mat1').propertyGroup('def').set('density', 'rho(pA[1/Pa],T[1/K]) [kg/m^3]');
model.material('mat1').propertyGroup('def').set('thermalconductivity', 'k(T[1/K]) [W/(m*K)]');
model.material('mat1').propertyGroup('def').set('soundspeed', 'cs(T[1/K]) [m/s]');
model.material('mat1').propertyGroup('def').func.create('eta', 'Piecewise');
model.material('mat1').propertyGroup('def').func('eta').set('funcname', 'eta');
model.material('mat1').propertyGroup('def').func('eta').set('arg', 'T');
model.material('mat1').propertyGroup('def').func('eta').set('extrap', 'constant');
model.material('mat1').propertyGroup('def').func('eta').set('pieces', {'200.0' '1600.0' '-8.38278E-7+8.35717342E-8*T^1-7.69429583E-11*T^2+4.6437266E-14*T^3-1.06585607E-17*T^4'});
model.material('mat1').propertyGroup('def').func.create('Cp', 'Piecewise');
model.material('mat1').propertyGroup('def').func('Cp').set('funcname', 'Cp');

```

```

model.material('mat1').propertyGroup('def').func('Cp').set('arg', 'T
');
model.material('mat1').propertyGroup('def').func('Cp').set('extrap',
'constant');
model.material('mat1').propertyGroup('def').func('Cp').set('pieces',
{'200.0' '1600.0' '1047.63657-0.372589265*T^1+9.45304214E-4*T
^2-6.02409443E-7*T^3+1.2858961E-10*T^4'});
model.material('mat1').propertyGroup('def').func.create('rho', '
Analytic');
model.material('mat1').propertyGroup('def').func('rho').set('
funcname', 'rho');
model.material('mat1').propertyGroup('def').func('rho').set('args',
{'pA' 'T'});
model.material('mat1').propertyGroup('def').func('rho').set('expr',
'pA*0.02897/8.314/T');
model.material('mat1').propertyGroup('def').func('rho').set('
dermethod', 'manual');
model.material('mat1').propertyGroup('def').func('rho').set('argders
', {'pA' 'd(pA*0.02897/8.314/T,pA)'; 'T' 'd(pA*0.02897/8.314/T,T)
'});
model.material('mat1').propertyGroup('def').func.create('k', '
Piecewise');
model.material('mat1').propertyGroup('def').func('k').set('funcname'
, 'k');
model.material('mat1').propertyGroup('def').func('k').set('arg', 'T
');
model.material('mat1').propertyGroup('def').func('k').set('extrap',
'constant');
model.material('mat1').propertyGroup('def').func('k').set('pieces',
{'200.0' '1600.0' '-0.00227583562+1.15480022E-4*T^1-7.90252856E
-8*T^2+4.11702505E-11*T^3-7.43864331E-15*T^4'});
model.material('mat1').propertyGroup('def').func.create('cs', '
Analytic');
model.material('mat1').propertyGroup('def').func('cs').set('funcname
', 'cs');
model.material('mat1').propertyGroup('def').func('cs').set('args', {
'T'});
model.material('mat1').propertyGroup('def').func('cs').set('expr', '
sqrt(1.4*287*T)');
model.material('mat1').propertyGroup('def').func('cs').set('
dermethod', 'manual');
model.material('mat1').propertyGroup('def').func('cs').set('argders'
, {'T' 'd(sqrt(1.4*287*T),T)'});
model.material('mat1').propertyGroup('def').addInput('temperature');
model.material('mat1').propertyGroup('def').addInput('pressure');
model.material('mat1').set('family', 'air');

model.sol('sol1').study('std1');
model.sol('sol1').feature.remove('s1');

```

```

model.sol('sol1').feature.remove('v1');
model.sol('sol1').feature.remove('st1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature('st1').set('study', 'std1');
model.sol('sol1').feature('st1').set('studystep', 'freq');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature.create('s1', 'Stationary');
model.sol('sol1').feature('s1').feature.create('p1', 'Parametric');
model.sol('sol1').feature('s1').feature.remove('pDef');
model.sol('sol1').feature('s1').feature('p1').set('pname', 'freq');
model.sol('sol1').feature('s1').feature('p1').set('plist', 'range
(20,1,200)');
model.sol('sol1').feature('s1').feature('p1').set('plot', 'off');
model.sol('sol1').feature('s1').feature('p1').set('plotgroup', 'pg1'
);
model.sol('sol1').feature('s1').feature('p1').set('probesel', 'all')
;
model.sol('sol1').feature('s1').feature('p1').set('probes', {});
model.sol('sol1').feature('s1').feature('p1').set('control', 'freq')
;
model.sol('sol1').feature('s1').set('control', 'freq');
model.sol('sol1').feature('s1').feature('aDef').set('complexfun',
true);
model.sol('sol1').feature('s1').feature.create('fc1', 'FullyCoupled'
);
model.sol('sol1').feature('s1').feature.remove('fcDef');
model.sol('sol1').attach('std1');
model.sol('sol1').runAll;

model.result('pg1').run;
model.result.create('pg4', 'PlotGroup1D');
model.result('pg4').run;
model.result('pg4').feature.create('ptgr1', 'PointGraph');
model.result('pg4').feature('ptgr1').selection.set([6]);
model.result('pg4').run;
model.result('pg4').run;
model.result('pg4').feature('ptgr1').set('expr', 'acpr.Lp');
model.result('pg4').feature('ptgr1').set('descr', 'Sound pressure
level');
model.result('pg4').run;

model.study('std1').feature('freq').set('geomselection', 'geom1');
model.study('std1').feature('freq').set('physselection', 'acpr');
model.study('std1').feature('freq').set('plist', 'range(20,0.1,200)'
);

model.sol('sol1').study('std1');
model.sol('sol1').feature.remove('s1');
model.sol('sol1').feature.remove('v1');

```

```

model.sol('soll1').feature.remove('st1');
model.sol('soll1').feature.create('st1', 'StudyStep');
model.sol('soll1').feature('st1').set('study', 'std1');
model.sol('soll1').feature('st1').set('studystep', 'freq');
model.sol('soll1').feature.create('v1', 'Variables');
model.sol('soll1').feature.create('s1', 'Stationary');
model.sol('soll1').feature('s1').feature.create('p1', 'Parametric');
model.sol('soll1').feature('s1').feature.remove('pDef');
model.sol('soll1').feature('s1').feature('p1').set('pname', 'freq');
model.sol('soll1').feature('s1').feature('p1').set('plist', 'range
(20,0.1,200)');
model.sol('soll1').feature('s1').feature('p1').set('plot', 'off');
model.sol('soll1').feature('s1').feature('p1').set('plotgroup', 'pg1'
);
model.sol('soll1').feature('s1').feature('p1').set('probesel', 'all'
);
model.sol('soll1').feature('s1').feature('p1').set('probes', {});
model.sol('soll1').feature('s1').feature('p1').set('control', 'freq'
);
model.sol('soll1').feature('s1').set('control', 'freq');
model.sol('soll1').feature('s1').feature('aDef').set('complexfun',
true);
model.sol('soll1').feature('s1').feature.create('fc1', 'FullyCoupled'
);
model.sol('soll1').feature('s1').feature.remove('fcDef');
model.sol('soll1').attach('std1');
model.sol('soll1').runAll;

model.result('pg1').run;
model.result('pg4').run;

model.name('Harde vegger.mph');

model.result('pg4').run;

model.study('std1').feature('freq').set('geomselection', 'geom1');
model.study('std1').feature('freq').set('physselection', 'acpr');

model.result.numerical.create('pev1', 'EvalPoint');
model.result.numerical('pev1').selection.set([6]);
model.result.numerical('pev1').set('expr', 'acpr.Lp');
model.result.numerical('pev1').set('descr', 'Sound pressure level');
model.result.table.create('tbl1', 'Table');
model.result.table('tbl1').comments('Point Evaluation 1 (acpr.Lp)');
model.result.numerical('pev1').set('table', 'tbl1');
model.result.numerical('pev1').setResult;

model.name('Harde vegger.mph');

```

```

model.physics('acpr').feature.create('imp1', 'Impedance', 2);
model.physics('acpr').feature('imp1').selection.set([6]);
model.physics('acpr').feature('imp1').set('Zi', 1, '500+1000i');

model.study('std1').feature('freq').set('geomselection', 'geom1');
model.study('std1').feature('freq').set('physselection', 'acpr');
model.study('std1').feature('freq').set('plist', 'range(20,0.2,200)'
);

model.sol('sol1').study('std1');
model.sol('sol1').feature.remove('s1');
model.sol('sol1').feature.remove('v1');
model.sol('sol1').feature.remove('st1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature('st1').set('study', 'std1');
model.sol('sol1').feature('st1').set('studystep', 'freq');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature.create('s1', 'Stationary');
model.sol('sol1').feature('s1').feature.create('p1', 'Parametric');
model.sol('sol1').feature('s1').feature.remove('pDef');
model.sol('sol1').feature('s1').feature('p1').set('pname', 'freq');
model.sol('sol1').feature('s1').feature('p1').set('plist', 'range
(20,0.2,200)');
model.sol('sol1').feature('s1').feature('p1').set('plot', 'off');
model.sol('sol1').feature('s1').feature('p1').set('plotgroup', 'pg1'
);
model.sol('sol1').feature('s1').feature('p1').set('probesel', 'all')
;
model.sol('sol1').feature('s1').feature('p1').set('probes', {});
model.sol('sol1').feature('s1').feature('p1').set('control', 'freq')
;
model.sol('sol1').feature('s1').set('control', 'freq');
model.sol('sol1').feature('s1').feature('aDef').set('complexfun',
true);
model.sol('sol1').feature('s1').feature.create('fc1', 'FullyCoupled'
);
model.sol('sol1').feature('s1').feature.remove('fcDef');
model.sol('sol1').attach('std1');
model.sol('sol1').runAll;

model.result('pg1').run;
model.result('pg4').run;
model.result.table.remove('tbl1');
model.result.table.create('tbl1', 'Table');
model.result.table('tbl1').comments('Point Evaluation 1 (acpr.Lp)');
model.result.numerical('pev1').set('table', 'tbl1');
model.result.numerical('pev1').setResult;

model.name('Harde vegger.mph');

```

```
model.result('pg4').run;
```

```
out = model;
```