Clio / LspCAD kombo tutorial - Sidst rettet af Henrik Høegh - Sidst rettet 24-10-2010 Download as PDF

I denne artikel vil jeg forsøge at gennemgå forløbet man skal igennem med Clio og LspCAD for at være klar til at simulere delefiltre med de to stykker software. Jeg vil tage udgangspunkt i en alm simpel 2 vejs højttaler.

Jeg vil i eksemplet her tage udgangspunkt i Clio 7.3 og LspCAD 6.37. De to stykker software med tilhørende hardware kan købes her :

- Clio hos Audiomatica
- LspCAD hos IJData

De har distributører i hele Europa, og det er nemt at finde forhandlere.

Det tages som udgangspunkt at man kender begge programmers brugerflade. En introduktion til begge vil komme senere i nye artikler.

Måleopstilling:
Det er vigtig at vi får dels frekvens og fase informationer så præcist med som mulig, da det danner grundlag for hvor præcis vores simulering bliver. Og fordi LspCAD vil have fasen som minimumsfase, mister vi alt i målingerne der har informationer om afstand. LspCAD tror derfor vores enheder er 100% tidskorrigeret, hvilket de ikke er. Vi skal derfor også finde det akustiske delay enhederne imellem.

For at finde et så præcis delay som mulig, placere vi mikrofonen midt imellem enhederne. På den måde ændres afstanden imellem enhederne til mikrofonen mindst, hvis mikrofonen flyttes endnu længere væk.

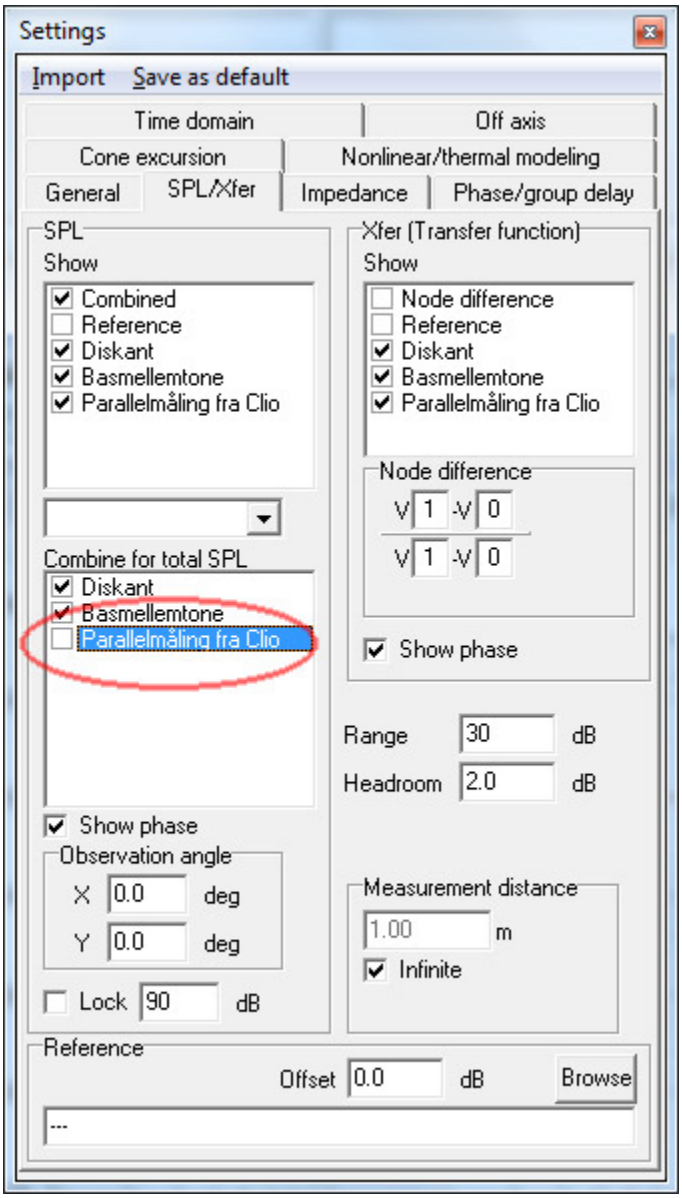
Alle frekvensmålinger her foretages via MLS og skal være gated. Vi bruger samme start og stop gate-tider på vores målinger, og alle frekvensmålinger eksporteres med minimumsfase. Det gøres ved at højre klikke på faseknappen og vælge minimumsfase inden man eksportere.

Det akustiske delay

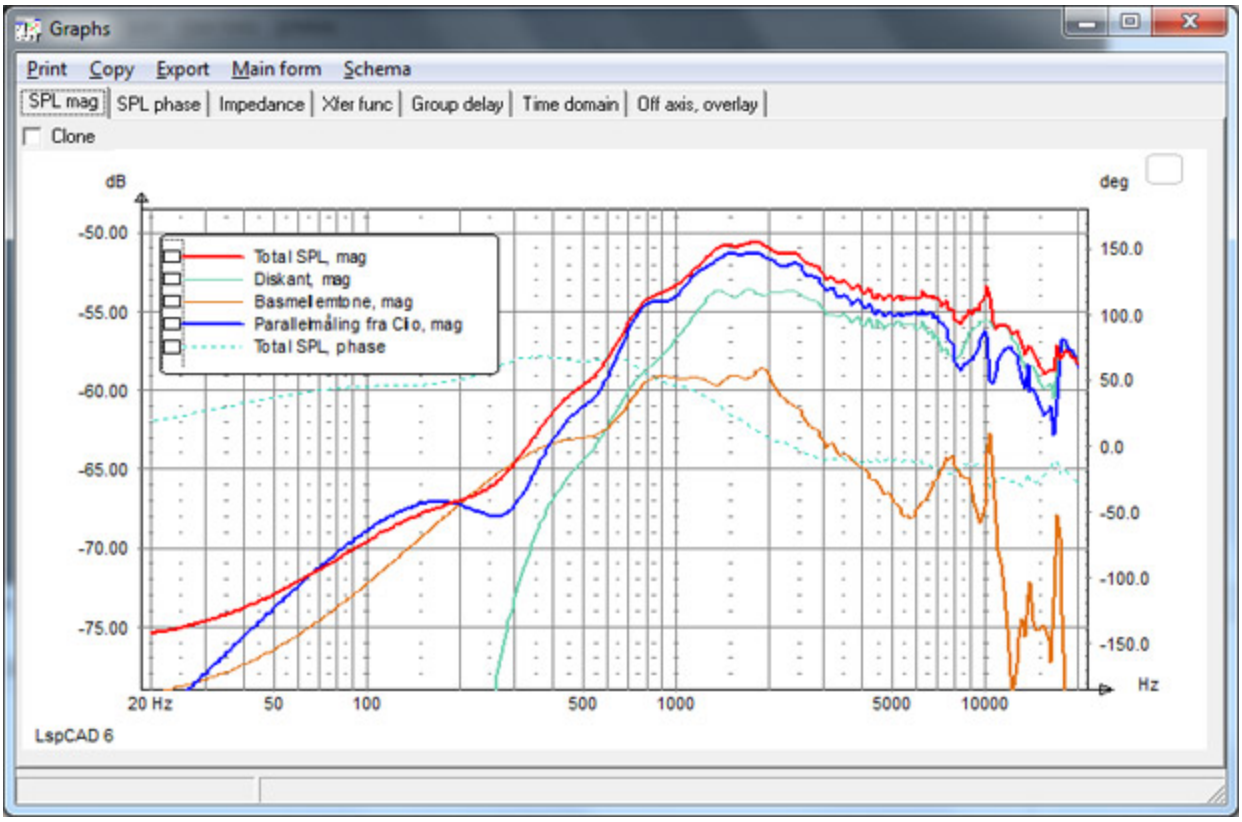
Når vi skal finde det akustiske delay, gør vi det ved at foretage 3 frekvensmålinger uden at flytte mikrofonen. Vi placere mikrofonen så langt fra højttaleren som mulig. Vi skal dog stadig have mulighed for at få valide målinger ned til et stykke under 1000Hz, så det er lidt et kompromis imellem afstand og muligheder.

Vi måler diskantens frekvensgang, derefter basmellemtonens. Og så måler vi begge enheders frekvensgang. Vi har nu en måling af hver, samt summeringen som vi ved er rigtig da det er målt.

I LspCAD importere vi nu de tre målinger og under Settings -> SPL/Xfer slår vi den målte summation fra så den ikke bruges til at regne summeringen ud. Altså så den ikke bidrager til den simulerede summation.

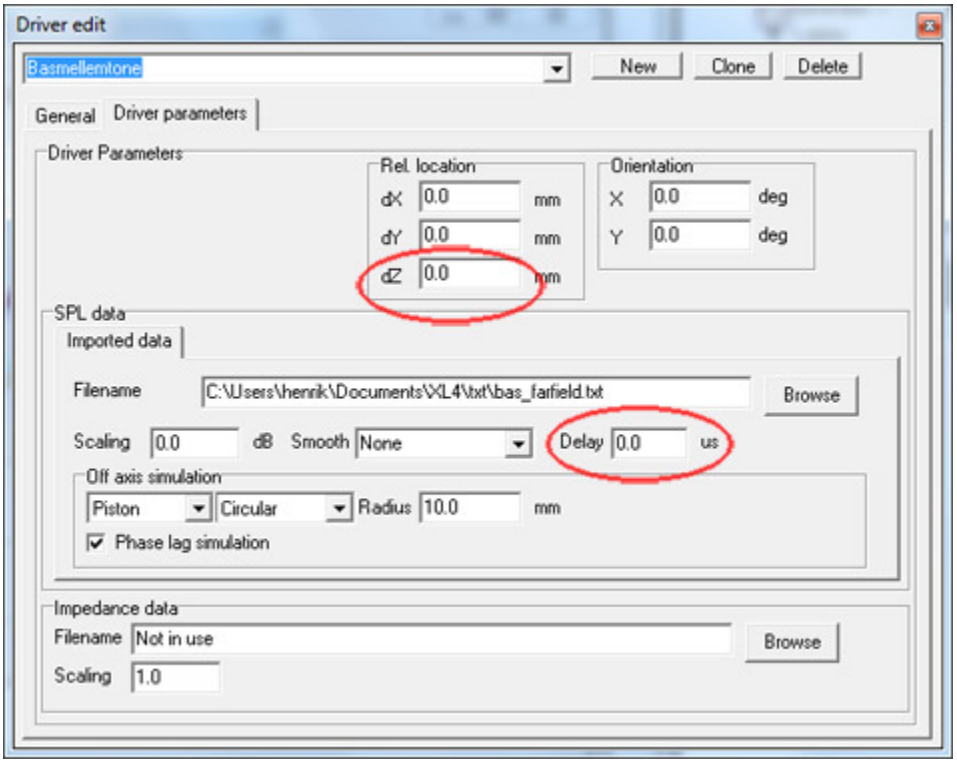


Vi har nu 4 frekvenskurver i LspCAD. Et for diskanten, en for basmellemtonen, en for den målte summation og en simuleret summation. Den simulerede og målte summation burde ligge oven i hinanden hvis alt hvad været rigtig, men det gør det ikke (rød og blå).

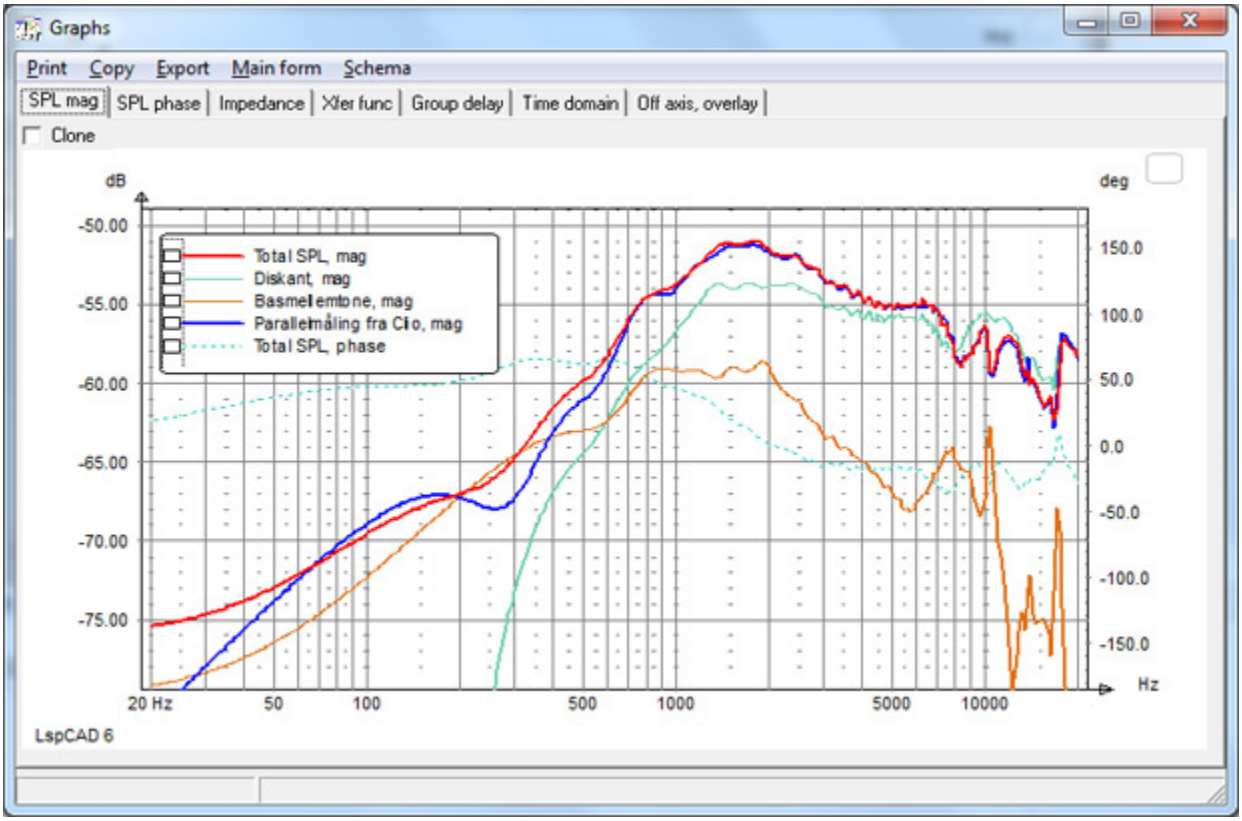


Vi mangler det akustiske delay. Det finder vi nu.

Vi går ind i settings for basmellemtonen ved at trykke på den i simulate mode. Vi finder det felt der hedder Delay og bruger PIL-OP på tastaturet til at øge delayet trinvis. Man kan også benytte feltet dZ istedet hvis man ønsker.



Vi kan se at for hver gang vi øger delayet, ændres den simulerede summation, og kommer tættere på den målte. Når de ligger oven i hinanden, har vi fundet delayet.



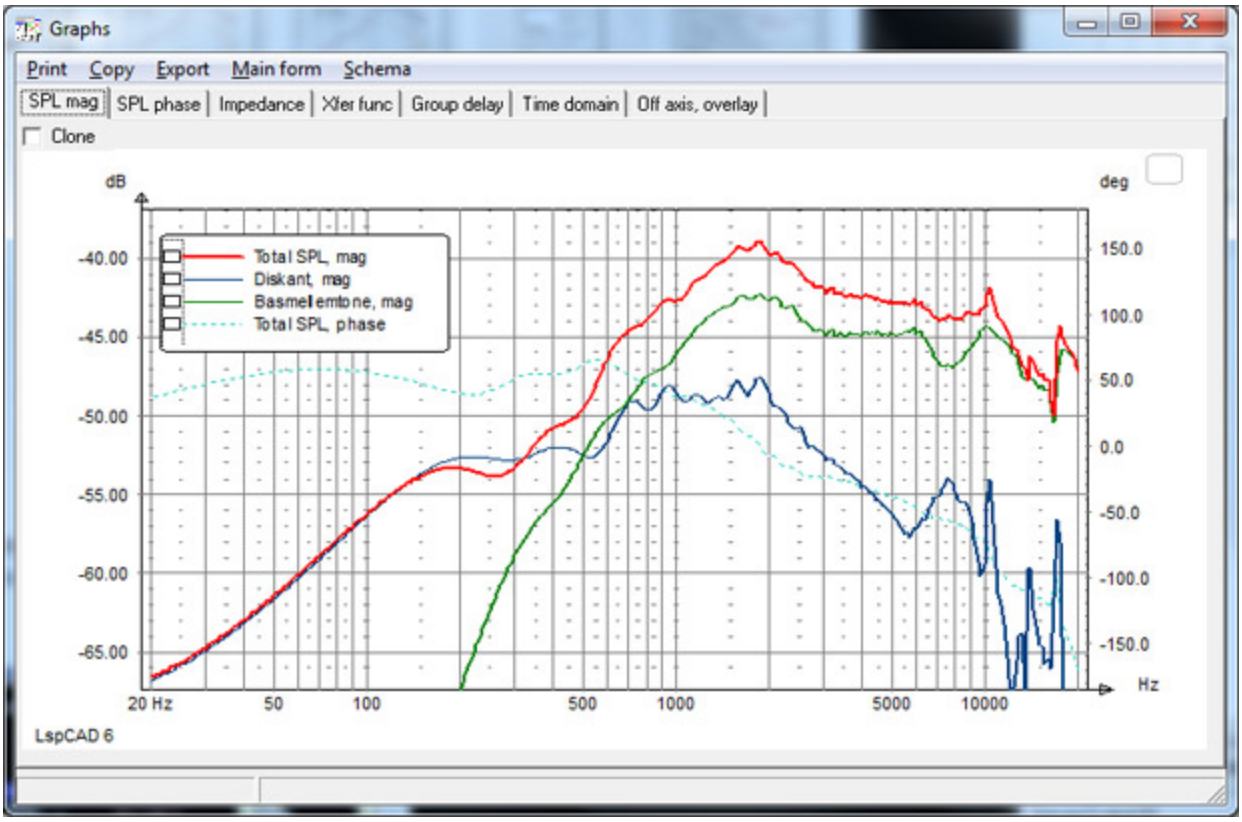
De gode målinger
Vi skal nu have lavet de frekvensmålinger i Clio der skal dannen grundlag for vores simulering i LspCAD. Da vi har fundet det akustiske delay, er der ingen grund til at måle alt for langt fra højttaleren. Vi skal dog gerne længere væk end højttalerens bredde for at få hele baffeltabet med. Løfter vi højttaleren lidt op fra gulvet, hjælper det somregel med at måle dybere endnu.

Vi laver en frekvensmåling af hver enhed, hvor vi placere mikrofonen ud for hver enhed under deres måling. Derved får vi nogle gode målinger som er valide langt ned i frekvens. Herhjemme kan jeg måle ned til ca. 175 Hz uden at få den første reflektion med.

Vores frekvensmålinger gemmes og eksporteres til txt filer. Inden skal vi dog huske at skifte til minimumsfase.

Bagefter måler vi impedansen på hver enhed via en simpel sinus måling. Husk og hæv opløsningen i Clio.

Nu laver vi et nyt projekt i LspCAD hvor vi importere de to enheders frekvens og impedansmålinger. På basmellemtonen indtaster vi også det akustiske delay vi fandt tidligere.



Vi er nu klar til at begynde vores filtersimulering.