

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék

Deák Gergely

**Szervómotorok akusztikai vizsgálata**

Önálló laboratórium I  
Msc

külső Konzulens

Kimpián Tibor,   
Thyssenkrupp Components Technology Hungary Kft.

Belső Konzulens

Orosz György

BUDAPEST, 2021

Feladatkiírás:

* A Thyssenkrupp kormányrendszerek fejlesztésével foglalkozik, ahol a kormányzás rásegítése állandómágneses szinkronmotorral történik. Ezeknek a motoroknak a zaja bejutva az utastérbe zavaró, így ennek a zajnak a csökkentése érdekében szükséges a motor viselkedésének mélyebb megértése. A motor "zajának" jelentős hányada a légrésben keletkező elektromágneses erők hatására jön létre és a motor mechanikai felépítése által módosítva jut be a korányrendszerbe, majd onnan az autó karosszériáján keresztül az utastérbe. Ebben a témában a motor mechanikai viselkedésének megértése a cél egy egyszerűsített koncentrált paraméteres modell segítségével, ami a motor állórészén és házán megjelenő hullámterjedés viselkedését írja le jól.

Tartalom

[1 Bevezetés 4](#_Toc87366933)

[2 Optimalizálás 5](#_Toc87366934)

[2.1 Optimalizálás alapok: 5](#_Toc87366935)

[2.2 Négyzetes hiba 5](#_Toc87366936)

[2.3 A rezgések terjedése 6](#_Toc87366937)

[2.4 A zaj csökkentése 7](#_Toc87366938)

[2.5 A lengő rendszer modellezése 7](#_Toc87366939)

[3 A tervezés részletes leírása, a választott megoldások indoklása 9](#_Toc87366940)

[3.1 A matematikai modell felállítása 9](#_Toc87366941)

[3.2 Rendszeregyenlet 10](#_Toc87366942)

[3.3 A GUI megtervezése 11](#_Toc87366943)

[4 A megtervezett műszaki alkotás értékelése, továbbfejlesztési lehetőségek 12](#_Toc87366944)

[5 Irodalomjegyzék 13](#_Toc87366945)

[6 Mellékletek (dokumentációk: kapcsolási rajz, huzalozási rajz, beültetési rajz, forrás file-ok, kész program, installálási és használati útmutatók stb.) 14](#_Toc87366946)

[6.1 Modusmátrixgenerátor 14](#_Toc87366947)

[6.2 Elmozdulásszámítás 15](#_Toc87366948)

# Bevezetés

Az önálló laboromat a ThyssenKrupp közreműködésével írtam, a kiírt feladat szervomotorok akusztikai vizsgálata volt.

Az elektromos autók terjedésével a zajkibocsátás csökken, így olyan zajok is előtérbe kerülnek, amiket eddig a belső égésű motorok hangja elnyomott, ide tartozik a kormányzásrásegítés zaja is.

Az egyre nagyobb teljesítményű autók és az ergonómiai szempontok megkövetelik a személyautókban a szervó rásegítés beépítését. A hidraulikus rásegítés helyett az alacsonyabb fogyasztás és egyszerűbb felépítés érdekében egyre elterjedtebb az elektromos szervó-motorok alkalmazása.

A motorokon végzett rezgés mérések spektrális elemzésével lehetőségünk van azonosítani az egyes zajkomponensek forrását, melyet fel tudunk használni a motorok későbbi tervezési lépéseinél.

Az előző dolgozatomban megvizsgáltam a rezgéstan alapjait, a rezgések terjedésének fizikáját és a modális elemzés alapjait. Ezeknek az ismereteknek a segítségével készítettem egy olyan programot, amivel a hullámok terjedését pontszerű tömegek között modellezhetjük. [6]

A továbbiakban ennek a programnak a felhasználásával készítek egy olyan optimalizáló algoritmust, amivel az ANSYS bemenő paramétereit lehet finomhangolni.

# Optimalizálás

Az optimalizálás az egy olyan f: A -> R függvény, ami az A halmazból a valós számokba képez.

Keresünk: egy x0 elemet A-ban úgy, hogy f(x0) <= f(x) minden x A-beli elemre „minimalizáció” vagy úgy, hogy f(x0) >= f(f) minden x A-beli elemre „maximalizáció”. [a]

Egy ilyen megfogalmazást optimalizálási feladatnak vagy matematikai programozási feladatnak neveznek. A kifejezés nem áll közvetlen kapcsolatban a számítógépek programozásával; a lineáris programozás közelebb áll hozzá, és ennek egy altípusát jelöli. Sok valós és elméleti probléma modellezhető így, például a fizikában vagy a gépi látásban az energia minimalizálása. [a]

Tipikusan A egy Rn tér részhalmaza, amit egyenletekkel és egyenlőtlenségekkel határoznak meg. Az A halmaz azokból a pontokból áll, amelyek megoldják az így megadott egyenlőtlenség-rendszert. Ezeket a pontokat megengedett megoldásoknak is hívják. [a]

Az f függvényt többféleképpen is nevezik, például minimalizáció esetén költségfüggvénynek, energiafüggvénynek, energia funkcionálnak, indirekt hasznossági függvénynek, vagy maximalizáció esetén hasznossági függvénynek. A feladatnak megfelelően a minimalizáló vagy maximalizáló megengedett megoldást optimális megoldásnak hívják. [a]

Konvenció szerint a standard alakot minimalizációként fogalmazzák meg. Az általános feladatban, ahol a megengedett pontok halmaza vagy a függvény nem konvex, több helyi minimum is adódhat. Az x\* pont helyi minimum, ha van egy δ > 0, hogy minden x pontra, amire : ‖ x − x ∗ ‖ ≤ δ ; {\displaystyle \|\mathbf {x} -\mathbf {x} ^{\*}\|\leq \delta ;\,} {\displaystyle \|\mathbf {x} -\mathbf {x} ^{\*}\|\leq \delta ;\,} teljesül [a]

f ( x ∗ ) ≤ f ( x )

Szavakkal: akkor helyi minimum, ha van egy környezete, ahol a függvény értéke nem vesz fel kisebb értéket, mint az adott pontban. [a]

A nem konvex feladatot megoldó algoritmusok nem tudnak különbséget tenni a helyi és a globális minimum között, ezért megoldásként helyi minimumot adnak. Az alkalmazott matematika és a numerikus analízis egy egész ága, a globális optimalizáció foglalkozik olyan determinisztikus algoritmusok kifejlesztésével, amelyek véges időben konvergálnak a globális minimumhoz. [a]

## Optimalizálás alapok:

## Négyzetes hiba

A legegyszerűbb optimalizálási módszer, ha két függvény értékeinek négyzetes különbségeit vesszük és összeadjuk a teljes értelmezési tartományban, így kapunk egy skalármennyiséget, amivel minősíteni tudjuk a két függvény kapcsolatát. Ennek a skalármennyiségnek a minimumát keressük a paramétertérben. Első próbára az átviteli függvények valós részét helyettesítettem be a képletbe és annak a minimumát kerestem.

Ahol e – a hibák vektora E pedig a hibák összege, ezt az E-t kell minimalizálni.

Paraméterek:

visszacsatolt = 1;

j = sqrt(-1);

m = 1;

k = 1;

c = 1;

Ms = 18;

force = 1;

force\_pos = 1;

omegakezdo = 0.1;

Nomega = 10000;

Kiertekeles = 1.5;

A hiba számításhoz használt function:

E\_real\_np =@(x) sum((real(FRF\_matrix(:,n))-real(elmozdulasszamitas\_optimum(m, x(1), x(2), force, force\_pos, Ms, omegakezdo, Nomega, Kiertekeles, visszacsatolt,p))).^2) ;

A hiba függvény megjelenítése: Ahhoz, hogy a Matlabban meg tudjam jeleníteni az ezcontour vagy surf függvényeket kellett volna használnom, ha vectoros számítással akartam volna megjeleníteni, de akkor át keleltt volna írnom az elmozdulászámítás függfényt is, mert az jelenleg nem tudja kezelni a vectoros bemenetet k-ra és c-re, ezért inkább egy loopban végig iteráltam a k és a c paramétereket az eredményeket pedig a paraméterek számának megfelelő nagyságú mátrixba mentettem. Ez a megoldás lassabban futott le, de arra ez is megfelelt, hogy lássam a hibának hol van a minimuma.

%% surface generator

% legyenek az iterációs változók

% s és d

Nk = 100; % rugómerevség

Nc = 100; % csillapítási tényező

k\_end = 5;

c\_end = 2;

k\_start = 0.1;

c\_start = 0.01;

alteration\_k = (k\_end-k\_start)/Nk;

alteration\_c = (c\_end-c\_start)/Nc;

E\_real\_plot = zeros(Nk,Nc);

ki = 0;

ci = 0;

f = waitbar(0,'Please wait...');

for x = k\_start:alteration\_k:k\_end

ki=ki+1;

ci = 0;

for y = c\_start:alteration\_c:c\_end

ci=ci+1;

E\_real\_plot(ki,ci) = E\_real\_np([x y]);

end

status = (ki\*ci)/(Nk\*Nc);

waitbar(status ,f ,'Please wait...');

pause(0.01)

end

close(f)

Chart, surface chart

Description automatically generated

. ábra: Hiba megjelenítése

Következtetés:

A négyzetes hibára való optimalizálás nem vezetett sikerrel mert a függvénynek nincs minimuma. A matlab fminsearch() functionja egy 10^8 nagyságrendű hibaminimumot adott vissza minimumértékként, amit legjobb kedvemben sem neveznék kis hibának. :D

## FRAC hibakeresés

A másik módszer, amivel a hibát minimalizálni tudom a FRAC (Frequency Response Assurance Criterion) függvény. A frac egy két vektor közötti lineáris kapcsolatot ír le a két vektor által bezárt szög cosinusával. Ha az érték 1 a két vektor párhuzamos a korreláció mértéke 1, ha a frac értéke 0 a korreláció 0.

Text, letter

Description automatically generated

Graphical user interface, chart, surface chart

Description automatically generated

. ábra: Korreláció

Látszik, hogy a két átmeneti függvény közötti legnagyobb korreláció is 1.6x10^-4 körüli érték k[0.1 5] és c[0.01 2] intervallumon.

Mivel a FRAC akkor jó, ha a korreláció 1, ezért a függvénynek nem a minimumát, hanem a maximumát kell keresni. Ezt úgy oldottam meg, hogy az előző függvény inverzét vettem. Ez a következő ererdményt adta.

fmin =

3.7123 -0.0017

e\_min =

1.3660e-13

Chart, box and whisker chart

Description automatically generated

. ábra: A korreláció inverze

Itt valamit lehet elrontottam, de ez alapján se tűnik úgy, hogy a két függvény korrelálna egymással. A továbbiakban megnézem, a Fracot nem az egész értéktartományban vizsgálom, hanem feldarabolom és egy ablakkal végigmegyek rajta az eredményeket pedig mivel 0 és 1 közötti számot ad, összeszorzom egymással így kapok egy az egészet leíró számot.

fmax =

1.6954 0.2174

e\_max =

3.1179e+04

## A zaj csökkentése

## A lengő rendszer modellezése

# A tervezés részletes leírása, a választott megoldások indoklása

## A matematikai modell felállítása

## Rendszeregyenlet

## A GUI megtervezése

# A megtervezett műszaki alkotás értékelése, továbbfejlesztési lehetőségek

# Irodalomjegyzék

# Mellékletek (dokumentációk: kapcsolási rajz, huzalozási rajz, beültetési rajz, forrás file-ok, kész program, installálási és használati útmutatók stb.)

## Modusmátrixgenerátor

function [M, K, C, FI, OMEGA2] = modusmatrixgenerator(m, k, c, Ms, visszacsatolt)

%m - tömeg

%c - csillapítás

%k - rugómerevség

%M - tömegmátrix

%C - csillapításmátrix

%K - rugómerevségmátrix

%Ms - tömegek száam

mm = m\*ones(1,Ms);

cm = c\*ones(1,Ms);

km = k\*ones(1,Ms);

M = eye(Ms).\*m;

K = zeros(Ms,Ms);

C = zeros(Ms,Ms);

I2 = [1 -1; -1 1];

for num = 1:Ms-1

Ksub = km(num)\*I2;

K(num:num+1,num:num+1) = K(num:num+1,num:num+1) + Ksub;

Csub = cm(num)\*I2;

C(num:num+1,num:num+1) = C(num:num+1,num:num+1) + Csub;

end

if visszacsatolt == 1

K(Ms,1) = K(Ms,1) -km(Ms);

K(1,Ms) = K(1,Ms) -km(Ms);

K(1,1) = K(1,1) + km(Ms);

K(Ms,Ms) = K(Ms,Ms) + km(Ms);

C(Ms,1) = C(Ms,1) -cm(Ms);

C(1,Ms) = C(1,Ms) -cm(Ms);

C(1,1) = C(1,1) + cm(Ms);

C(Ms,Ms) = C(Ms,Ms) + cm(Ms);

end

%FI - 1-re normált sajátvektor - Fi\_n -ek módusalakok

%OMEGA2 - sajátértékek - omega\_n\_negyzetek

[FI,OMEGA2] = eig(M\K);

end

## Elmozdulásszámítás

function [U, ALFA, omega, OMEGA] = elmozdulasszamitas(C, FI, OMEGA2, f, fnum, omegakezdo, Nomega, Kiertekeles, Ms)

j = sqrt(-1);

F = zeros(Ms,1);

F(fnum) = f;

KSZIOMEGAx2 = diag(FI.'\*C\*FI);

OMEGA = diag(sqrt(OMEGA2));

%omegakezdo = 0;

%Nomega = 1000;

%Kiertekeles = 1;

omega = linspace(omegakezdo,OMEGA(end)\*Kiertekeles,Nomega).';

ALFA = zeros(Nomega,Ms);

U = zeros(Ms,Nomega);

for n = 1:Ms

ALFA(:,n) = FI(:,n).'\*F./(OMEGA(n)^2+j\*omega\*KSZIOMEGAx2(n)-omega.^2);

U = U + FI(:,n).\*ALFA(:,n).';

end

end