

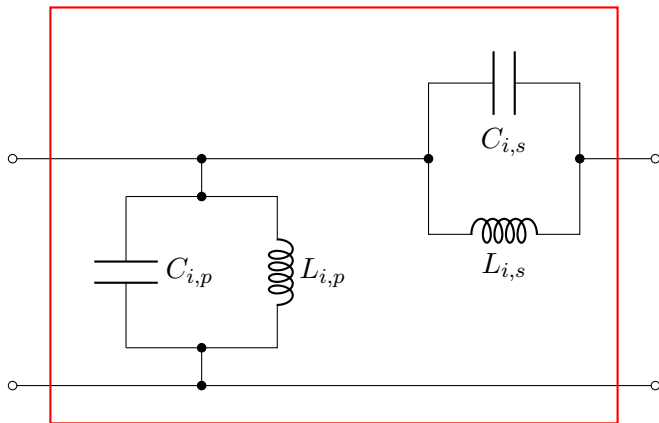
Koncentrált paraméterű RF szűrő optimalizációja aktív tanulással

Pintér Bálint, Szilágyi Gábor

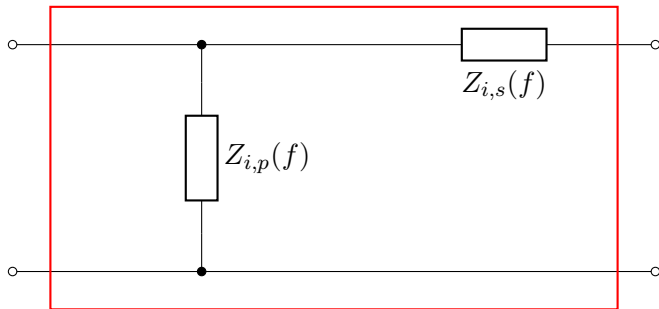
BME VIK

2023. május 30.

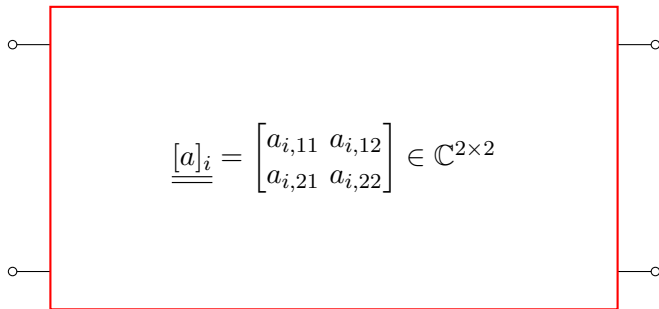
Az optimalizálandó hálózat egysége



Az egységálózat helyettesítőképe



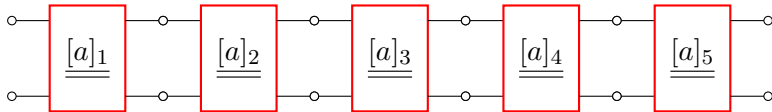
Az egységálózat ABCD paraméterei ($[a]$ -mátrixa)



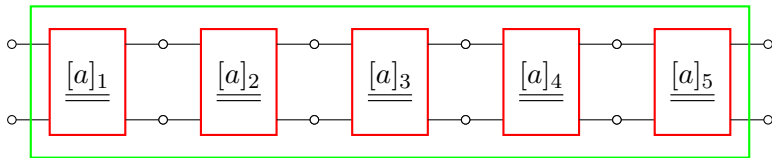
A diagram of a two-port network block. It consists of a large red rectangle. On the left side, there are two small circles representing input terminals, each with a short horizontal line extending to the left. On the right side, there are two small circles representing output terminals, each with a short horizontal line extending to the right. Inside the rectangle, the following equation is written:

$$\underline{\underline{[a]}}_i = \begin{bmatrix} a_{i,11} & a_{i,12} \\ a_{i,21} & a_{i,22} \end{bmatrix} \in \mathbb{C}^{2 \times 2}$$

Az optimalizálandó hálózat



Az optimalizálandó hálózat



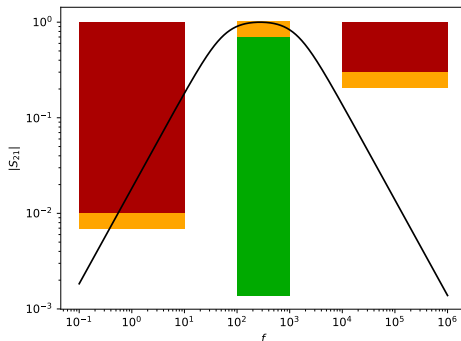
$$\underline{\underline{[a]}} = \prod_i \underline{\underline{[a]_i}} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Az egész hálózat $|S_{21}(f)|$ paraméterére vonatkozik specifikáció.

$$S_{21} = \frac{2}{A + B/Z_0 + C \cdot Z_0 + D}$$

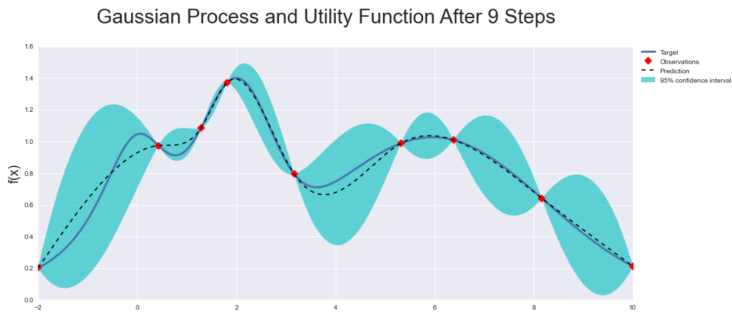
A teljesítendő specifikáció

- ▶ Záró sáv
- ▶ Áteresztő sáv
- ▶ Átmeneti sáv (fehér)
- ▶ Extra büntetett régió



Bayesi optimalizálás

- ▶ Specifikált probléma, költségfüggvény \rightarrow optimális megoldás
- ▶ Nagy szabadsági fok, kiértékelés költséges, bonyolult, időigényes lehet
- ▶ Bayesi optimalizálás, megoldásához: már létező könyvtár (Bayesian Optimization lib, ami a scikit-learn-ön alapul)



Bayesian Optimization könyvtár

- ▶ Példányosítás, inputok: költségfüggvény, határok (maximalizálás)
- ▶ *maximize()* függvény, inputok: kezdeti random pontok, iterációszám
- ▶ Ebben: *utility* függvény definiálás, majd iteratív megoldás

Bayesian Optimization könyvtár

- ▶ Példányosítás, inputok: költségfüggvény, határok (maximalizálás)
- ▶ *maximize()* függvény, inputok: kezdeti random pontok, iterációszám
- ▶ Ebben: *utility* függvény definiálás, majd iteratív megoldás

Minden iterációs lépésben:

- ▶ Paraméterfrissítés: *update_param()* tagfüggvény
- ▶ frissített *utility*-val javaslat (*suggest()*) egy új kiértékelési helyre, *x_probe*-ra
- ▶ *probe()* kiértékeli *x_probe* helyen

Iteráció lefut: maximalizált pont a paramétertérben.

Paraméterfrissítés

Utility függvény *update_param* tagfüggvénye:

- ▶ Feltárási stratégia: UCB, EI, POI
alapbeállítás: UCB (Upper Confidence Bounds method)
- ▶ κ, χ hiperparaméterek
- ▶ κ_{decay} -vel szorozódik (csökken) κ minden $\kappa_{decay-delay}$ utáni iterációban

Új kiértékelési hely javaslat

$Suggest()$ javaslata = $argmax()$ -ot néz egy akvizíciós függvényre, acq_max -ra

- ▶ Input: *utility* függvény
- ▶ Első lépés: mintavételezés 10^5 pontra, random, egyenletes eloszlással
- ▶ Második lépés: L-BFGS-B optimalizációs metódus (minden mintavételezési pontra)
- ▶ Visszatér az legnagyobb, ez lesz: x_probe

L-BFGS-B optimalizáció

- ▶ L-BFGS-B solver: *scipy.optimizer* könyvtárból, minimalizálási feladatokra \rightarrow módosítani kell *acq_max*-ot
- ▶ Másodrendű optimalizációs algoritmus, Kvázi-Newton módszer
- ▶ L-BFGS algoritmus (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) kiterjesztése korlátok kezelésével
- ▶ Másodrendű deriváltat közelít, ahol azt közvetlenül nem lehet kiszámolni
- ▶ Nem kell kiszámolni a Hesse-mátrixot (csak $\underline{\underline{H}}^{-1}$ -t közelíti)
- ▶ Limitált memória: csak az elmúlt m lépés koordináta- és gradiensvektorát tárolja el.

Egy talált megoldás

