Appendix F
Nonlinear Optimization

Our goal is to compute component values so as to achieve or at least approximate a Butterworth response. The process consists of (1) converting components into a transfer function, (2) scaling this model to have a gain of 1 as $S \to \infty$, (3) taking the difference between this and the desired response (squared), (4) integrating the square error over all frequencies, and (5) minimizing this objective function with respect to the unknown components.

Transfer Function

A lumped parameter model consists of nodes connected by components. A floating nodal admittance matrix, A, is generated by assigning indexes (row/column) to each node. The admittance for a capacitor is SC, for a resistor is $1/R$ and for an inductor is $1/SL$. For a component connecting nodes i and j, one would add its admittance to A_{ii} and A_{jj} and subtract it from A_{ij} and A_{ji}. When complete, each element of the matrix will contain a polynomial in S.

To compute the transfer function, one removes the rows and columns for the output node and the ground and computes the determinant to get the numerator, and one removes the rows and columns for the input node and the ground and computes the determinant to get the denominator. (I developed GK reduction specifically for this problem. I also check for common roots using Euclid's Algorithm[1, p. 245].)

One must multiply by S to get sound volume at distance, at which point the numerator and denominator should have polynomials of the same order. To force the gain to be 1 as $S \to \infty$, the coefficients for the highest power in each should be equal.

Square Error Objective Function

Let $F(S)$ represent the difference between the target (Butterworth) and the model, where $F(S) \to 0$ as $S \to \infty$. Since we be integrating for $S = i\omega$ where $i = \sqrt{-1}$, we can use $F(S)F(-S)$ for the square error and perform the integration,

$$J = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(i\omega)F(-i\omega)\, d\omega \qquad (F1)$$

using residues.

From Cauchy's Residue Theorem[2, p. 678][3, p.155] we know that

$$\oint g(x)\, dx = 2\pi i \sum_k R_k \qquad (F2)$$

where the R_k are the residues of poles of $g(x)$ within the closed integral (counter clockwise). In other words

$$g(x) = \sum_k \frac{R_k}{x - P_k} \qquad (F3)$$

where the P_k are the corresponding poles. One can compute residues for simple poles using

$$R_k = \lim_{x \to P_k} (x - P_k)\, g(x) \quad . \qquad (F4)$$

Applying this to $(F1)$ using $S = i\omega$ we get

$$J = \sum_k \lim_{S \to P_k} (S - P_k)\, F(S)F(-S) \qquad (F5)$$

where the P_k are the poles of $F(S)$, since the poles of $F(-S)$ are in the right half plane. So to compute J we must find the roots to the polynomial in the denominator of $F(S)$.

Optimization

Considering the complexity in simply computing the objective function, I decided not to compute gradients (directly). Instead I accumulate parameter values and objective functions until I have more than enough to get a solution, then compute both the gradient and the Hessian using least squares.

Specifically, I perform a line search for each component separately, returning 2 or 3 new sets of parameter values each time. This is only enough to compute the gradient and the main diagonal of the Hessian. For n parameters one needs at least $n(m+1) - m(m-1)/2$ sets of values to solve for the gradient and m diagonals of the Hessian. I add another diagonal whenever the number of values increases to twice the minimum required.

Given a direction and distance from the current best set of parameter values, if the new set is better, the next set uses twice the distance. If not, the next set is taken in the opposite direction (same distance). Three iterations of curve fitting are performed, assuming the new values are inside the current range. This new value replaces one of the current end points. All three sets of parameters are added to the list, assuming they differ from the previous best.

Create an array $\mathbf{u}$ containing all of the unknowns in the gradient $\mathbf{g}$ and Hessian H and reformulate the matrix equation

$$\Delta J(k) \approx \mathbf{g}'\Delta\mathbf{x}(k) + \frac{1}{2}\,\Delta x'(k) H \Delta x(k) \qquad (F6)$$

into the form

$$\Delta J(k) \approx \mathbf{v}'(k)\mathbf{u} \qquad (F7)$$

where $\mathbf{v}(k)$ will contain elements from $\Delta\mathbf{x}(k)$ and $\Delta\mathbf{x}(k)\Delta\mathbf{x}(k)'$. It helps to remember that

$$\Delta x'(k) H \Delta x(k) = \mathrm{Tr}(H \Delta\mathbf{x}(k)\Delta\mathbf{x}'(k)) \quad .$$

The least square solution is then given by

$$\mathbf{u} = \left(\sum_k \mathbf{v}(k)\mathbf{v}'(k)\right)^{-1} \left(\sum_k \mathbf{v}(k)\Delta J(k)\right) \quad . \qquad (F8)$$

I use a combination of steepest descent and Newton, similar to Marquardt-Levenberg[4]. It generally performs better than straight Newton.

Of particular importance is which parameters one allows to vary. If the solution is not unique, then at best it will converge to whatever solution is closest. For example, one can fix R_a and let L_a vary, or fix L_a and let R_a vary, but not vary both at the same time.

I have never had much luck fixing the cutoff frequency of the target filter. Of all the parameters, the target cutoff frequency seems to have the greatest effect on the objective function and consequently should be the first parameter optimized. Fortunately, one can also get several closed form solutions by equivalencing terms of the denominator polynomials. This is narrowed down (or expanded) to three frequencies, and three iteration of curve fitting are performed as with the line search.

References

[1] Bruce Schneider, **Applied Cryptography**, ISBN 0-471-11709-9.

[2] Erwin Kreysig, **Advanced Engineering Mathematics**, LCCN 66-28748.

[3] Ruel V. Churchill, **Complex Variables and Applications**, LCCN 59-15046.

[4] Donald Marquardt, *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, **SIAM Journal on Applied Mathematics**, vol. 11, no. 2, pp. 431-441, 1963. doi:10.1137/0111030.