

# Analog Filter Transformations v1.0

Dante Vilorio, B.S. Electrical Engineering – dviloria1919@gmail.com

*Abstract:*

*This paper covers frequency transformations of lowpass analog prototype filters and the bilinear transform. This paper presents a method of calculating the frequency transformation of a bandpass filter quickly and a method of calculating the bilinear transform quickly.*

Note: this paper is not peer reviewed and mistakes may be present.

## 1 LOWPASS ANALOG PROTOTYPE FREQUENCY TRANSFORMATIONS

---

Let  $H(S)$  be a lowpass filter of order  $N$  with a cutoff frequency of 1 rad/s:

$$H(S) = \frac{\sum_{i=0}^N b_i S^{N-i}}{\sum_{j=0}^N a_j S^{N-j}}$$

### 1.1 LOWPASS ANALOG PROTOTYPE TO LOWPASS

$H(S)$  can be transformed into a lowpass filter  $H_{LPF}(S)$  of order  $N$  with cutoff frequency  $\omega_n$  via:

$$H_{LPF}(S) = \frac{\sum_{i=0}^N b_i \omega_n^i S^{N-i}}{\sum_{j=0}^N a_j \omega_n^j S^{N-j}}$$

This is accomplished by using the substitution  $S = \frac{S}{\omega_n}$  in  $H(S)$ .

### 1.2 LOWPASS ANALOG PROTOTYPE TO HIGHPASS

$H(S)$  can be transformed into a highpass filter  $H_{HPF}(S)$  of order  $N$  with cutoff frequency  $\omega_n$  via:

$$H_{HPF}(S) = \frac{\sum_{i=0}^N b_i \omega_n^{-i} S^i}{\sum_{j=0}^N a_j \omega_n^{-j} S^j}$$

This is accomplished by using the substitution  $S = \frac{\omega_n}{S}$  in  $H(S)$ .

### 1.3 LOWPASS ANALOG PROTOTYPE TO BANDPASS

$H(S)$  can be transformed into a bandpass filter  $H_{BPF}(S)$  of order  $2N$  with geometric center frequency  $\omega_n$  and quality factor  $Q$  via:

$$H_{BPF}(S) = \frac{\sum_{i=0}^N b_i [\omega_n S]^i [Q(S^2 + \omega_n^2)]^{N-i}}{\sum_{j=0}^N a_j [\omega_n S]^j [Q(S^2 + \omega_n^2)]^{N-j}}$$

This is accomplished by using the substitution  $S = Q(\frac{S}{\omega_n} + \frac{\omega_n}{S})$  in  $H(S)$ .

$H_{BPF}(S)$  can alternatively be designed with upper and lower cutoff frequencies  $\omega_1$  and  $\omega_2$  by designing for a geometric center frequency  $\omega_n$  and quality factor  $Q$  given by:

$$\omega_n = \sqrt{\omega_1 \omega_2}$$

$$Q = \frac{\omega_n}{\omega_1 - \omega_2}$$

### 1.3.1 Fast implementation of lowpass analog prototype to bandpass

First we will rewrite  $H_{BPF}(S)$  in matrix form:

$$A = [a_0 \cdots a_N] \text{ and is a } 1 \times (N+1) \text{ matrix}$$

$$B = [b_0 \cdots b_N] \text{ and is a } 1 \times (N+1) \text{ matrix}$$

$$Qq = \begin{bmatrix} Q^0 & & \\ & \ddots & \\ & & Q^{-N} \end{bmatrix} \text{ and is a } (N+1) \times (N+1) \text{ diagonal matrix}$$

$$P = \begin{bmatrix} p_{N,0,0} & \cdots & p_{N,0,N} \\ \vdots & \ddots & \vdots \\ p_{N,N,0} & \cdots & p_{N,N,N} \end{bmatrix} \text{ and is a } (N+1) \times (N+1) \text{ matrix}$$

$$Ss = \begin{bmatrix} (S/\omega_n)^N \\ \vdots \\ (S/\omega_n)^0 \end{bmatrix} \text{ and is a } (N+1) \times 1 \text{ matrix}$$

$$A(S) = \frac{AQqPSs}{BQqPSs}$$

The matrices  $A, B, Qq$ , and  $Ss$  are self-explanatory. The matrix  $P$  holds the coefficients of the polynomial  $P_{N,n} = S^n [S^2 + 1]^{N-n}$  represented as  $p_{N,n,s}$  with  $N, n$  coming from  $P_{N,n}$  and  $s$  meaning that this is the coefficient of the term  $S^{N-s}$ .

Each coefficient  $p_{N,n,s}$  can be calculated according to:

$$p_{N,n,s} = \binom{N-n}{\frac{s-n}{2}} * 0^{(n+s)\%2} \text{ with } 0^0 = 1$$

## 2 ANALOG TO DIGITAL FILTER TRANSFORMATION

---

Let  $A(S)$  be an analog filter of order  $N$ :

$$A(S) = \frac{\sum_{i=0}^N b_i S^{N-i}}{\sum_{j=0}^N a_j S^{N-j}}$$

## 2.1 THE BILINEAR TRANSFORM

$A(S)$  can be transformed to a digital filter  $D(Z)$  of order  $N$  using the bilinear transform implemented via:

$$D(Z) = \frac{\sum_{i=0}^N b_i [2f_s]^i [Z^{-1} + 1]^i [Z^{-1} - 1]^{N-i}}{\sum_{j=0}^N a_j [2f_s]^j [Z^{-1} + 1]^j [Z^{-1} - 1]^{N-j}}$$

This is accomplished by using the substitution  $S = 2f_s \frac{Z^{-1}-1}{Z^{-1}+1}$  in  $A(S)$ .

### 2.1.1 Fast implementation of the bilinear transform

We will first start by rewriting  $D(Z)$  in matrix form:

$A = [a_0 \cdots a_N]$  and is a  $1 \times (N+1)$  matrix

$B = [b_0 \cdots b_N]$  and is a  $1 \times (N+1)$  matrix

$F_s = \begin{bmatrix} (2f_s)^0 & & \\ & \ddots & \\ & & (2f_s)^N \end{bmatrix}$  and is a  $(N+1) \times (N+1)$  diagonal matrix

$P = \begin{bmatrix} p_{N,0,0} & \cdots & p_{N,0,N} \\ \vdots & \ddots & \vdots \\ p_{N,N,0} & \cdots & p_{N,N,N} \end{bmatrix}$  and is a  $(N+1) \times (N+1)$  matrix

$ZZ = \begin{bmatrix} Z^0 \\ \vdots \\ Z^{-N} \end{bmatrix}$  and is a  $(N+1) \times 1$  matrix

$$D(Z) = \frac{AF_s P Z Z}{BF_s P Z Z}$$

The matrices  $A, B, F_s$ , and  $ZZ$  are self-explanatory. The matrix  $P$  holds the coefficients of the polynomial  $P_{N,n} = [Z^{-1} + 1]^n [Z^{-1} - 1]^{N-n}$  represented as  $p_{N,n,z}$  with  $N, n$  coming from  $P_{N,n}$  and  $z$  meaning that this is the coefficient of the term  $Z^{-z}$ .

There is a way to recursively calculate this matrix. The first column  $p_{N,n,0}$  is entirely ones, and the first row can be calculated as:

$$p_{N,0,z} = \binom{N}{z} * (-1)^z$$

An observation, valid only for  $n \geq 1$  and  $z \geq 1$ , was made that allows for the rest of the coefficients to be calculated:

$$p_{N,n,z} = p_{N,n-1,z} + p_{N,n-1,z-1} + p_{N,n,z-1}$$

There is no formal proof of this observation, but one can use a computer to show that this is true for the first couple hundred  $N$ .

This method is quite quick, but measurements showed that it is better (for repeat transforms of the same order) to cache/precompute the results of the matrix  $P$  instead of using the recursive method every time.

## 2.2 FREQUENCY WARPING

For  $A(S)$  to have a frequency characteristic  $\omega_n$  in the digital domain,  $A(S)$  must be designed to have frequency characteristic  $\omega_{n,warped}$  in the analog domain where  $\omega_{n,warped}$  is:

$$\omega_{n,warped} = 2f_s \tan\left(\frac{\pi\omega_n}{f_s}\right)$$

## 3 REFERENCES

---

[1] [https://en.wikipedia.org/wiki/Bilinear\\_transform](https://en.wikipedia.org/wiki/Bilinear_transform)

[2] [https://en.wikipedia.org/wiki/Prototype\\_filter](https://en.wikipedia.org/wiki/Prototype_filter)