The C++ Digital Signal Processing classes

E. Robert Tisdale

May 12, 1999

Abstract

1 Window Functions

$$w_{n-1-k} = w_k \tag{1}$$

1.1 Blackman

`<Type>Vector w = <Type>Blackman(n);`

$$\begin{aligned}
w_k &= 0.42 - 0.5 \cdot \cos\left(\frac{2\pi k}{n-1}\right) + 0.08 \cdot \cos\left(\frac{4\pi k}{n-1}\right) \\
&= 0.5\left(1 - \cos\left(\frac{2\pi k}{n-1}\right)\right) - 0.08\left(1 - \cos\left(\frac{4\pi k}{n-1}\right)\right) \\
&= 1 - \cos^2\left(\frac{\pi k}{n-1}\right) - 0.16\left(1 - \cos^2\left(\frac{2\pi k}{n-1}\right)\right) \\
&= \sin^2\left(\frac{\pi k}{n-1}\right) - 0.16\sin^2\left(\frac{2\pi k}{n-1}\right) \\
&= \sin^2\left(\frac{\pi k}{n-1}\right) - 0.64\sin^2\left(\frac{\pi k}{n-1}\right)\left(1 - \sin^2\left(\frac{\pi k}{n-1}\right)\right) \\
&= 0.36\sin^2\left(\frac{\pi k}{n-1}\right) + 0.64\sin^4\left(\frac{\pi k}{n-1}\right) \\
&= \left(\frac{4}{5}\right)^2 \sin^2\left(\frac{\pi k}{n-1}\right)\left(\left(\frac{3}{4}\right)^2 + \sin^2\left(\frac{\pi k}{n-1}\right)\right)
\end{aligned} \tag{2}$$

1.2 Dolph-Chebyshev

`<Type>Vector w = <Type>Chebyshev(n, ripple);`

$$\delta_p = 10^{-\mathrm{ripple}/20} \tag{3}$$

$$\tau_p = \frac{1+\delta_p}{\delta_p} \tag{4}$$

$$\sigma_p = \cosh\left(\frac{\cosh^{-1}(\tau_p)}{n-1}\right) \tag{5}$$

$$\delta_f = \frac{1}{\pi}\cdot\cos^{-1}\left(1/\sigma_p\right) \tag{6}$$

$$\begin{aligned} x_0 &= \frac{3-\cos(2\pi\delta_f)}{1+\cos(2\pi\delta_f)} \\ &= \frac{3-(2\cos^2(\pi\delta_f)-1)}{1+(2\cos^2(\pi\delta_f)-1)} \\ &= \frac{2-\cos^2(\pi\delta_f)}{\cos^2(\pi\delta_f)} \\ &= 2\sigma_p^2-1 \end{aligned} \tag{7}$$

$$\begin{aligned} x_k &= \frac{x_0+1}{2}\cos(2\pi k/n)+\frac{x_0-1}{2} \\ &= \frac{x_0+1}{2}\big(2\cos^2(\pi k/n)-1\big)+\frac{x_0-1}{2} \\ &= (x_0+1)\cos^2(\pi k/n)-1 \\ &= 2\sigma_p^2\cdot\cos^2(\pi k/n)-1 \geq -1 \end{aligned} \tag{8}$$

$$x_{n-k} = x_k \tag{9}$$

$$x_{n/2} = -1 \tag{10}$$

$$r_k = \begin{cases} \cos\left(\frac{n-1}{2}\cos^{-1}(x_k)\right) & \text{if} \quad |x_k| \leq 1 \\ \\ \cosh\left(\frac{n-1}{2}\cosh^{-1}(x_k)\right) & \text{if} \quad 1 < |x_k| \end{cases} \tag{11}$$

$$r_{n/2} = 0 \tag{12}$$

$$t_k = r_k \tag{13}$$

if n is odd.

$$t_k = \begin{cases} +r_k e^{i\pi k/n} & \forall k \mid 0 \leq k < n/2 \\ \\ -r_k e^{i\pi k/n} & \forall k \mid n/2 \leq k < n \end{cases} \tag{14}$$

if n even.

$$t_{n-k} = t_k^* \tag{15}$$

$$t_{n/2} = 0 \tag{16}$$

$$w_j = \sum_{k=0}^{n-1} t_k e^{i2\pi jk/n} \tag{17}$$

$$w_j \leftarrow \begin{cases} w_{j+(n+1)/2 \bmod n} & \text{if } n \text{ is odd} \\ \\ w_{j+n/2 \bmod n} & \text{if } n \text{ is even} \end{cases} \tag{18}$$

$\forall j \mid 0 \leq j < n$.

1.3 Hamming

```
<Type>Vector w = <Type>Hamming(n);
```

$$\begin{aligned} w_k &= 0.54 - 0.46\cos\left(\frac{2\pi k}{n-1}\right) \\ &= 0.54 - 0.46\left(2\cos^2\left(\frac{\pi k}{n-1}\right) - 1\right) \\ &= 1 - 0.92\cos^2\left(\frac{\pi k}{n-1}\right) \\ &= 0.08 + 0.92\sin^2\left(\frac{\pi k}{n-1}\right) \end{aligned} \tag{19}$$

1.4 Hanning

```
<Type>Vector w = <Type>Hanning(n);
```

$$\begin{aligned} w_k &= \frac{1}{2}\left(1 - \cos\left(\frac{2\pi(k+1)}{n+1}\right)\right) \\ &= \frac{1}{2}\left(1 - \left(2\cos^2\left(\frac{\pi(k+1)}{n+1}\right) - 1\right)\right) \\ &= \sin^2\left(\frac{\pi(k+1)}{n+1}\right) \end{aligned} \tag{20}$$

1.5 Kaiser

```
<Type>Vector w = <Type>Kaiser(n, beta);
```

$$\begin{aligned} w_k &= \frac{I_0\left(\beta\sqrt{1 - \left(\frac{2k-(n-1)}{n-1}\right)^2}\right)}{I_0(\beta)} \\ &= \frac{I_0\left(\frac{\beta}{n-1}\sqrt{(n-1)^2 - (2k-(n-1))^2}\right)}{I_0(\beta)} \\ &= \frac{I_0\left(\frac{2\beta}{n-1}\sqrt{k(n-1-k)}\right)}{I_0(\beta)} \end{aligned} \tag{21}$$

2 Discrete Fourier Transforms

2.1 1 Dimensional DFTs

2.1.1 Return by Reference (In Place)

Direct	Inverse	
`<Type>1DCCDDFT(n, i)`	`<Type>1DCCIDFT(n, i)`	Complex to Complex
`<Type>1DRCDDFT(n, i)`	`<Type>1DRCIDFT(n, i)`	Real to Complex
`<Type>1DCRDDFT(n, i)`	`<Type>1DCRIDFT(n, i)`	Complex to Real

2.1.2 Return by Value (Out of Place)

Direct	Inverse	
`<Type>1DCDDFTC(n, i)`	`<Type>1DCIDFTC(n, i)`	Complex from Complex
`<Type>1DCDDFTR(n, i)`	`<Type>1DCIDFTR(n, i)`	Complex from Real
`<Type>1DRDDFTC(n, i)`	`<Type>1DRIDFTC(n, i)`	Real from Complex

2.1.3 Resize

`resize(n, i)`

2.2 2 Dimensional DFTs

2.2.1 Return by Reference (In Place)

Direct	Inverse	
`<Type>2DCCDDFT(m, n, i)`	`<Type>2DCCIDFT(m, n, i)`	Complex to Complex
`<Type>2DRCDDFT(m, n, i)`	`<Type>2DRCIDFT(m, n, i)`	Real to Complex
`<Type>2DCRDDFT(m, n, i)`	`<Type>2DCRIDFT(m, n, i)`	Complex to Real

2.2.2 Return by Value (Out of Place)

Direct	Inverse	
`<Type>2DCDDFTC(m, n, i)`	`<Type>2DCIDFTC(m, n, i)`	Complex from Complex
`<Type>2DCDDFTR(m, n, i)`	`<Type>2DCIDFTR(m, n, i)`	Complex from Real
`<Type>2DRDDFTC(m, n, i)`	`<Type>2DRIDFTC(m, n, i)`	Real from Complex

2.2.3 Resize

`resize(m, n, i)`

2.3 3 Dimensional DFTs

2.3.1 Return by Reference (In Place)

Direct	Inverse	
`<Type>3DCCDDFT(l, m, n, i)`	`<Type>3DCCIDFT(l, m, n, i)`	Complex to Complex
`<Type>3DRCDDFT(l, m, n, i)`	`<Type>3DRCIDFT(l, m, n, i)`	Real to Complex
`<Type>3DCRDDFT(l, m, n, i)`	`<Type>3DCRIDFT(l, m, n, i)`	Complex to Real

2.3.2 Return by Value (Out of Place)

Direct	Inverse	
`<Type>3DCDDFTC(l, m, n, i)`	`<Type>3DCIDFTC(l, m, n, i)`	Complex from Complex
`<Type>3DCDDFTR(l, m, n, i)`	`<Type>3DCIDFTR(l, m, n, i)`	Complex from Real
`<Type>3DRDDFTC(l, m, n, i)`	`<Type>3DRIDFTC(l, m, n, i)`	Real from Complex

2.3.3 Resize

`resize(l, m, n, i)`