Second, note that if e and f are finitely-supported sequences with respective P-periodizations e_P and f_P, then

$$\begin{aligned}\sum_{k'=0}^{P-1} \overline{e_P(k'+2n)}\, f_P(k'+2m) \;&= \\ &= \sum_{k'=0}^{P-1}\sum_j\sum_i \overline{e(k'+2n+jP)}\, f(k'+2m+iP) \\ &= \sum_{k'=0}^{P-1}\sum_j\sum_l \overline{e(k'+jP+2n)}\, f(k'+jP+2(m+lP')),\end{aligned}$$

after substituting $i \leftarrow l+j$. The sums over $0 \le k' < P$ and $j \in \mathbf{Z}$ combine into a sum over all integers $k \in \mathbf{Z}$, and the l and k sums may be interchanged, giving

$$\sum_{k'=0}^{P-1} \overline{e_P(k'+2n)}\, f_P(k'+2m) = \sum_l\sum_k \overline{e(k+2n)}\, f(k+2(m+lP')).$$

If $e=f=h$ and h is self-orthonormal, then the inner sum over k is $\delta(n-(m+lP'))$. Thus the outer sum over l is $\delta_{P'}(n-m)$, which is 1 if and only if $n \equiv m \pmod{P'}$; otherwise it is zero. The same holds if $e=f=g$ is self-orthonormal. If $e=h$ and $f=g$ are independent, the inner sum over k is always zero, so the total is zero. This establishes periodic independence and self-orthonormality.

Finally,

$$\begin{aligned}\sum_{k'=0}^{P'-1} \overline{f_P(2k'+n)}\, f_P(2k'+m) \;&= \\ &= \sum_{k'=0}^{P'-1}\sum_j\sum_i \overline{f(2k'+n+jP)}\, f(2k'+m+iP) \\ &= \sum_{k'=0}^{P'-1}\sum_j\sum_i \overline{f(2(k'+jP')+n)}\, f(2(k'+jP')+m+(i-j)P),\end{aligned}$$

so substituting $i \leftarrow l+j$ and $k' \leftarrow k-jP'$ and combining the k' and j summations into one makes this

$$\sum_k\sum_l \overline{f(2k+n)}\, f(2k+m+lP).$$

The l and k sums may be interchanged. The cases $f \leftarrow h$ and $f \leftarrow g$ give

$$\sum_{k'=0}^{P'-1} \overline{h_P(2k+n)}\, h_P(2k+m) \quad = \quad \sum_l\sum_k \overline{h(2k+n)}\, h(2k+m+lP);$$

$$\sum_{k'=0}^{P'-1} \overline{g_P(2k+n)}\, g_P(2k+m) \quad = \quad \sum_l \sum_k \overline{g(2k+n)}\, g(2k+m+lP).$$

If h and g satisfy the completeness condition, adding these together gives $\sum_l \delta(n-(m+lP)) = \delta_P(n-m)$, proving periodic completeness. □

17. **Solution:** The following is a Standard C implementation. We begin by implementing the inverse filter transform:

Contents of `ipcqfilt.c`

```
int mod(int x, int M) {   /* x%M for M>1 and any x */
  if(x<0) x-= x*M; /* x-x*modulus>0 equals x mod M */
  return x%M;
}
void ipcqfilter(float out[], const float in[], int q) {
  int n2, k2;
  for(k2=0; k2<q; k2++) {
    out[2*k2]=out[2*k2+1]=0;
    for(n2=0;n2<L/2;n2++) {
      out[2*k2] += h[2*n2]*in[mod(k2-n2, q)];
      out[2*k2] += g[2*n2]*in[mod(k2-n2, q) + q];
      out[2*k2+1] += h[2*n2+1]*in[mod(k2-n2, q)];
      out[2*k2+1] += g[2*n2+1]*in[mod(k2-n2, q) + q];
    }
  }
}
```

Notice that these functions will work with filters of any even length L.

Next, we implement the inverse to Mallat's periodic discrete wavelet transform on $N = 2^J K$ samples, generalizing `ipdwt0()`:

Reconstruction from Mallat's Periodic Wavelet Expansion

```
ipdwt( u[], N, J, h[], g[], L ):
[0] If J>0, then do [1-] to [4]
[1]    Compute ipdwt(u[], N/2, J-1, h[], g[], L)
[2]    Allocate temp[0]=0,...,temp[N-1]=0
[3]    Compute ipcqfilter(temp[], u[], N/2, h[], g[], L)
[4]    For i=0 to N-1, let u[i] = temp[i]
```

For practical reasons, we should place the allocation and deallocation of `temp[]` as close as possible to the filter transform. This frees unneeded memory for the recursive function call. In Standard C, this becomes: