

A Time-Domain Approach to Determining Inverse FIR Filters

Frank M. Candocia and Angelica M. Diaz
Image Understanding Laboratory
Department of Electrical and Computer Engineering
Florida International University, Miami, FL. 33199

Abstract

Given an FIR filter, this paper addresses a time-domain means of arriving at its inverse filter in FIR form. To this end, the original FIR filter should lack frequency nulls so that an inverse filter of reasonable support size can be established. With this approach we provide an alternative to the more commonly employed frequency design methods and also provide insight to the significance of the time domain operations being performed. A detailed description of the approach is provided and a designed inverse filter is used for the purposes of image deconvolution.

Keywords: inverse filter, FIR filter, correlation matrix, deconvolution, image deconvolution.

1. Introduction

The designing of inverse filters is useful for applications such as system identification [1] and deconvolution [2]. Such designs are usually performed in the frequency domain by way of Fourier or z-domain analysis [3] with the general inverse filter form being an autoregressive moving average (ARMA) model [4]. In this paper we present a time-domain, i.e. 1-D, design procedure for finite impulse response (FIR) inverse filters. This design is motivated by image deconvolution applications where very fast and efficient deconvolution (real-time) processing is necessary. That is, it is known that spatially invariant FIR filtering, also known as moving average (MA) filtering [4], via the fast Fourier transform is the most efficient form of deconvolution processing. Specifically, the MA filter must be the inverse of the blurring filter so that when convolved with the blurred image, a clean restored image results.

In order for the FIR inverse filter to be effective, the blurring function must be well-behaved, i.e., it must not have any frequency nulls in its spectral representation. If frequency nulls are present in the blurring filter, it cannot be expected that the MA filter will have reasonably small support size and also well-approximate

the inverse filter – thus compromising the quality and effectiveness of the deconvolution.

One must also keep in mind that, though the FIR inverse filter has the advantage of efficient implementation and the ability to completely undo the blurring, any post-blurring noise that corrupted the signal will necessarily be amplified when such deconvolution filtering is performed. This side effect should be considered prior to direct deconvolution with the FIR inverse filter. In particular, it has been noted that deconvolving images (blurred with well behaved filters) that originally exhibited blurred signal to noise ratios (BSNRs) > 40 dB typically results in little or no visually perceptible noise amplification. For BSNRs less than this, the Wiener filter is probably a better direct filtering approach [5]. It essentially trades off deconvolution accuracy with noise amplification resulting in a less noisy but more smooth restored image.

In the remainder of this paper, we detail our time domain approach to FIR inverse filter realization. The 1-D approach presented is extendable to the 2-D domain of images but at greater complexity. As such, our 2-D image filters will be separable and thus realized from the 1-D approach described in this paper. The approach is based on a finite extent (limited data) convolution model discussed in section 2. Section 3 will detail the inverse FIR filter design and section 4 will examine an image deconvolution example prior to concluding.

2. Finite Extent Discrete Convolution

Because our application domain is that of images, we define the finite extent convolution model over a limited number of samples of an observed image y . This image of $N_1 \times N_2$ rows and columns is defined as the *valid* portion of the convolution between defined samples of an original image x and an FIR filter h . That is, the finite extent definition is one where all observed points in y result from the filter's weighted sum on the original samples of x . This model is given by

$$y[n_1, n_2] = \sum_{k_1=n_1-K_1}^{n_1+K_1} \sum_{k_2=n_2-K_2}^{n_2+K_2} h[n_1 - k_1, n_2 - k_2] x[k_1, k_2] \quad (1)$$

Figure 1 graphically illustrates the two dimensional convolution between the finite extent x and h that results in the y of eqn. (1). Note that x has support size of $(N_1+M_1-1) \times (N_2+M_2-1)$ samples, M_1 and M_2 are the support size (number of rows and columns) of h , and y has support size $N_1 \times N_2$. Formally, y is defined over support region $n_1=0, \dots, N_1-1$ and $n_2=0, \dots, N_2-1$. Also,

$$K_i = (M_i - 1) / 2 \quad (2)$$

represents half the support length of the filter h along dimension i where $i=1,2$. Without loss of generality, we will assume M_i is an odd number for this presentation. Since $h[m_1, m_2]$ is defined for $m_1=-K_1, \dots, K_1$ and $m_2=-K_2, \dots, K_2$ and is considered zero outside this support region, $x[k_1, k_2]$ is thus defined over samples $k_1=-K_1, \dots, N_1+K_1-1$ and $k_2=-K_2, \dots, N_2+K_2-1$.

It is very convenient to express the finite extent convolution of eqn. (1) in matrix/vector form. This equivalent form is

$$\mathbf{y} = \mathbf{H} \mathbf{x} \quad (3)$$

where $\mathbf{x} = \text{vec}(x[k_1, k_2])$ and $\mathbf{y} = \text{vec}(y[n_1, n_2])$ and vec is the vectorizing operator that stacks the columns of a matrix one on top of the other such that a vector results [6].

The form of (3) can be used to represent either the 2-D or 1-D finite extent convolution model. As our filter development will be in 1-D from this point forward, the finite extent convolution model in this case is given by

$$y[n] = \sum_{k=n-K}^{n+K} h[n-k] x[k] \quad (4)$$

where $n=0, \dots, N-1$ and $k=-K, \dots, N+K-1$. It is thus clear that y is a sequence of N samples $n=0, \dots, N-1$, h is a sequence of M samples and x is a sequence of $N+M-1$ samples. In the matrix/vector form of eqn. (3), eqn. (4) is written in the form of eqn. (6) such that

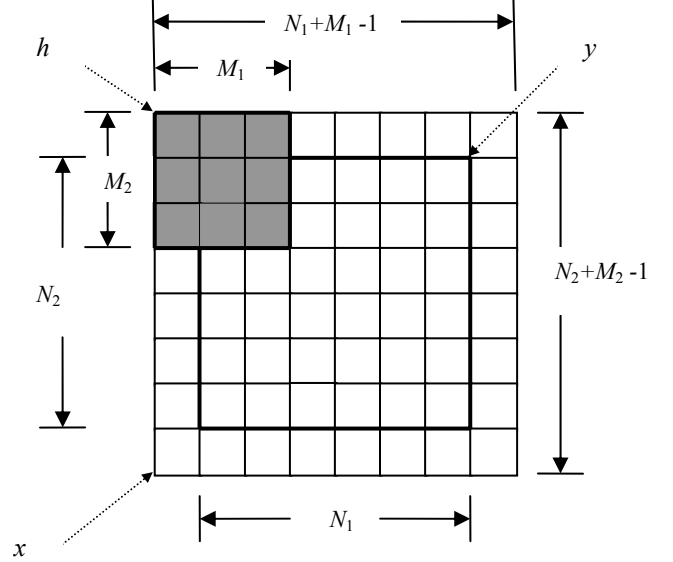


Figure 1. Graphical illustration of the two dimensional finite extent convolution model. The finite extent x is represented by the big square and the “sliding window” h is represented by the grayed-in region. Notice that our finite extent convolution of “valid” pixels results in the smaller and bold square region that is y . In this example, we have used $M_1 = M_2 = 3$, $N_1 = N_2 = 6$ and this results in an x size of $N_1 + M_1 - 1 = N_2 + M_2 - 1 = 8$

$$\mathbf{H} = \left\{ \begin{array}{ll} h[n-k] & -K \leq n-k \leq K \\ 0 & \text{otherwise} \end{array} \right\} \Bigg|_{\substack{n=0, \dots, N-1 \\ k=-K, \dots, N+K-1}} \quad (5)$$

We have used the notation in eqn. (5) as a compact form of expressing the elements of a matrix. That is, $\mathbf{Z} = \{z[n_1, n_2]\}_{n_1=0, \dots, N_1-1; n_2=0, \dots, N_2-1}$ would indicate that \mathbf{Z} is an $N_1 \times N_2$ matrix where element $[n_1, n_2]$ in this matrix comes from value $z[n_1, n_2]$. In eqn. (5), we see that, using eqn. (2), \mathbf{H} is an $N \times (N+M-1)$ matrix whose $[n, k]$ element comes from the condition in the curly braces $\{\}$. Also note that the first column index into matrix \mathbf{H} in eqn. (5) is $k=-K$ for the purpose of populating this matrix but that arrays are usually indexed either starting from 0 or 1 when accessed in program code.

$$\mathbf{y} = \underbrace{\begin{pmatrix} y[0] \\ y[1] \\ \vdots \\ y[N-1] \end{pmatrix}}_{N \text{ samples}} = \underbrace{\begin{pmatrix} h[+K] & \dots & h[-K] & 0 & \dots & 0 \\ 0 & h[+K] & \dots & h[-K] & \dots & 0 \\ \vdots & & \ddots & & \ddots & \\ 0 & \dots & 0 & h[+K] & \dots & h[-K] \end{pmatrix}}_{N \times (N+M-1) \text{ samples}} \underbrace{\begin{pmatrix} x[-K] \\ x[-K+1] \\ \vdots \\ x[N+K-1] \end{pmatrix}}_{N+M-1 \text{ samples}}$$

$$= \mathbf{H} \cdot \mathbf{x} \quad (6)$$

This is not a problem as array indexing and the coordinate origin that these indices represent are left to the users' discretion. For the purposes of 2-D filtering as in eqn. (1) – and as alluded to earlier – we will assume that h is a separable filter such that

$$h[m_1, m_2] = h_1[m_1]h_2[m_2]^T \quad (7)$$

and two subsequent 1-D filtering operations via eqn. (4) using h_1 and h_2 yields the equivalent of eqn. (1) with h in eqn. (7). In this case, the \mathbf{H} of eqn. (3) for the 2-D filtering would be

$$\mathbf{H} = \mathbf{H}_2^T \otimes \mathbf{H}_1 \quad (8)$$

where “ \otimes ” is the matrix kronecker product [6] and \mathbf{H}_1 and \mathbf{H}_2 are filled in using h_1 and h_2 , respectively, according to eqn. (5).

3. Determining the Inverse Filter

In this section we will determine the inverse filter by finding a solution to x in our finite extent convolution model of eqn. (6). Recall that in the previous section, we defined the finite extent convolution of eqn. (1) in matrix/vector form as $\mathbf{y}=\mathbf{H}\mathbf{x}$. Because \mathbf{H} in eqn. (5) is $N \times (N+M-1)$ in size, it is necessarily a rank-deficient matrix. This means there are less rows than columns in eqn. (3) which translates to having less equations than unknowns (in \mathbf{x}). There is clearly no unique solution for such a case. Nonetheless, one of the useful unique solutions to eqn. (3) is the minimum-norm one given in the least squares (LS) sense. Specifically, the LS minimum norm solution to eqn. (3), i.e. \mathbf{x}_{ls} , is known to be [7]

$$\mathbf{x}_{ls} = \mathbf{H}^T (\mathbf{H}\mathbf{H}^T)^{-1} \mathbf{y} \quad (9)$$

Notice that multiplying both sides of eqn. (9) by \mathbf{H} results in

$$\mathbf{H} \mathbf{x}_{ls} = \mathbf{H}\mathbf{H}^T (\mathbf{H}\mathbf{H}^T)^{-1} \mathbf{y} = \mathbf{y} \quad (10)$$

Since $\mathbf{H}\mathbf{H}^T (\mathbf{H}\mathbf{H}^T)^{-1} = \mathbf{I}$ (assuming $\mathbf{H}\mathbf{H}^T$ is invertible), this implies that the inverse filter must be

$$\mathbf{H}_{inv} = \mathbf{H}^T (\mathbf{H}\mathbf{H}^T)^{-1} \quad (11)$$

when expressed in matrix form. Generally, in all practical FIR filtering applications, \mathbf{H} is full row rank thus the inverse of $\mathbf{H}\mathbf{H}^T$ exists. This does not mean that \mathbf{x}_{ls} will be the perfectly recovered image that we seek if we performed deconvolution. It is only the minimum norm (energy) solution that satisfies the convolution

relation of eqn. (3). In fact, the quality of the solution will depend on the characteristics of the filter h . For now, we are interested in determining the inverse FIR filter h_{inv} to which \mathbf{H}_{inv} is the matrix counterpart. In this way, we can perform a deconvolution by convolving y with the inverse filter, i.e. $\hat{x} = h_{inv} * y = h_{inv} * h * x = x$ if indeed h_{inv} is the true inverse filter to h such that $h_{inv} * h = \delta$ with δ representing the unit delta sequence. Note that the hat ‘ $\hat{\cdot}$ ’ over x is used to indicate estimate.

In determining the inverse filter, we repeat that $\mathbf{H}\mathbf{x}$ in eqn. (3) is the matrix counterpart to $h * x$ in the finite extent *valid* convolution model thus \mathbf{H} is viewed as the *valid* convolution matrix that performs the operation of eqn. (4). What then is \mathbf{H}^T ? As it turns out, \mathbf{H}^T is the matrix equivalent of a *full* linear correlation filtering. It is essentially the same as the *valid* convolution operation except the filter is not “flipped” and the output resulting from this filtering results in the *full* linear correlation. This is most easily seen in the 1-D case. Specifically, for a filter h of support size M with $m=-K, \dots, K$ as in eqn. (4), the full linear correlation of h with a length N signal x defined for $k=0, \dots, N-1$ would yield an output y of length $N+M-1$ as

$$y[n] = \sum_{k=\max(0, n-K)}^{\min(N-1, n+K)} x[k]h[k-n+K] \quad (12)$$

for $n=-K, \dots, N+K-1$. As an example in 1-D, if $N=4$ such that $\mathbf{x}=[x_0, x_1, x_2, x_3]$ and $K=1$ such that $\mathbf{h}=[h_0, h_1, h_2]$, then $\mathbf{y}=[x_0h_2, x_0h_1+x_1h_2, x_0h_0+x_1h_1+x_2h_2, x_1h_0+x_2h_1+x_3h_2, x_2h_0+x_3h_1, x_3h_0]$ is the full linear correlation. The significance of this is that \mathbf{H} multiplied with any vector yields a *valid* or truncated convolution and \mathbf{H}^T multiplied with any vector yields a full linear correlation. With this in mind, $\mathbf{H}\mathbf{H}^T \equiv \mathbf{R}$ is a truncated correlation matrix. To see this, consider the autocorrelation sequence of a 3-tap filter $h=[h_0, h_1, h_2]$ defined for $n=-1, \dots, 1$ such that

$$\begin{aligned} r &= h \star h \\ &= [h_0, h_1, h_2] \star [h_0, h_1, h_2] \\ &= [h_0h_2, h_0h_1 + h_1h_2, h_0^2 + h_1^2 + h_2^2, h_0h_1 + h_1h_2, h_0h_2] \\ &= [r[-2], r[-1], r[0], r[1], r[2]] \end{aligned} \quad (13)$$

where ‘ \star ’ is being used to denote correlation and where eqn. (12) was used in obtaining r by

$$r[n] = \sum_{k=\max(-K, n-K)}^{\min(K, n+K)} h[k]h[k-n+K] \quad (14)$$

for $n = -2K, \dots, 2K$.

The importance of this is that, since we have argued that the matrix version of the inverse filter is $\mathbf{H}_{inv} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1} = \mathbf{H}^T\mathbf{R}^{-1}$, it will be the manner in which we truncate \mathbf{H} (and hence \mathbf{R}) that affects the accuracy of the inverse filter.

Before proceeding, notice that in the equivalent operation of $\mathbf{H}\mathbf{x} \leftrightarrow h*x$, where \leftrightarrow is being used to indicate equivalence across domains, the filter h appears in every row of \mathbf{H} . Likewise, we would have an equivalence of $\mathbf{R}\mathbf{x} \leftrightarrow r*x$ since the autocorrelation sequence r appears shifted in every row of \mathbf{R} . By noting this and that \mathbf{R}^{-1} must be a symmetric matrix since \mathbf{R} is itself symmetric, then all rows in \mathbf{R}^{-1} must also be shifted versions of the sequence r^{-1} [8]. That, along with the fact that $r*r^{-1} = \delta$ is a property that must hold for all sequences with inverses, we must then have that $\mathbf{R}r^{-1} = \delta$ is one of the conditions for determining our inverse filter. This system of equations is given in eqn. (17).

Since we can't deal with an infinitely long system of equations, we truncate the system in eqn. (17) about the origin to a finite number of equations in as many unknowns, yielding $\mathbf{R}_t r_t^{-1} = \delta_t$ where the subscript t denotes truncated versions of these matrices and vectors. For example, if h was a 3-tap filter and r^{-1} was being approximated out to 5 samples, then \mathbf{R}_t would be 5×5 and given by

$$\mathbf{R}_t = \begin{bmatrix} r[0] & r[1] & r[2] & 0 & 0 \\ r[-1] & r[0] & r[1] & r[2] & 0 \\ r[-2] & r[-1] & r[0] & r[1] & r[2] \\ 0 & r[-2] & r[-1] & r[0] & r[1] \\ 0 & 0 & r[-2] & r[-1] & r[0] \end{bmatrix} \quad (18)$$

while the truncated inverse function of r is given by $r_t^{-1} = [r^{-1}[-2], r^{-1}[-1], r^{-1}[0], r^{-1}[1], r^{-1}[2]]^T$ and $\delta_t = [0, 0, 1, 0, 0]^T$. We would simply solve for r_t^{-1} using

$$r_t^{-1} = \mathbf{R}_t^{-1} \delta_t \quad (19)$$

Finally, using our matrix and sequence equivalents, we have that $\mathbf{H}_{inv} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1} = \mathbf{H}^T\mathbf{R}^{-1} \approx \mathbf{H}^T\mathbf{R}_t^{-1} \leftrightarrow h_{inv} = h \star r^{-1} \approx h \star r_t^{-1}$. Thus, the inverse filter is, approximately (due to truncation)

$$h_{inv} = h \star r_t^{-1} \quad (20)$$

where notice that the correlation relation of eqn. (20) necessitates the length of h_{inv} to be greater than or equal to the length of h . In the non approximate case, we have exactly that

$$h_{inv} = h \star r^{-1} \quad (21)$$

and notice that, convolving both sides of eqn. (21) by h , we get $h \star h_{inv} = h \star (h \star r^{-1})$. In the case that h is a symmetric filter, which corresponds to the important case of linear phase filters, then autocorrelation is equivalent to convolution and

$$\begin{aligned} h \star h_{inv} &= h \star (h \star r^{-1}) \\ &= h \star (h \star r^{-1}) = (h \star h) \star r^{-1} \\ &= r \star r^{-1} \\ &= \delta \end{aligned} \quad (22)$$

where we used the properties in eqn. (13), (21) and the commutative properties of the linear operators of convolution and correlation [3].

In summary, the procedure for obtaining the 1-D inverse filter h_{inv} in FIR form from a length M filter h is:

Steps To Determining the Inverse Filter

1. Select the desired inverse filter length Q where $Q \geq M$ and Q is odd.
2. Let $N = Q - M + 1$ where N is the length of r_t^{-1} .
3. If $N \geq M$
 - a. Zero-pad $(N-M)/2$ zeros before and after the length M filter h so that a length N vector can be used in populating the required $N \times (2N-1)$ matrix \mathbf{H} of eqn. (6)
 - b. Let $\mathbf{R}_t = \mathbf{H}\mathbf{H}^T$
- Otherwise,
 - a. Use the length M filter h in populating the required $M \times (2M-1)$ matrix \mathbf{H} of eqn. (6)
 - b. Let $\mathbf{R} = \mathbf{H}\mathbf{H}^T$
 - c. Let \mathbf{R}_t be the center $N \times N$ portion of \mathbf{R} , i.e., $\mathbf{R}_t = \mathbf{R}[a:b, a:b]$ where $a = (M-N)/2 + 1$ and $b = (M+N)/2$ assuming the row and column indexing of matrix \mathbf{R} starts at 1.
4. Establish a length N vector δ_t that is populated with all 0s except the central element which is 1.
5. Determine r_t^{-1} as given in eqn. (19)
6. Determine h_{inv} as given in eqn. (20)

Note that the above steps are for a filter h of length M where M is odd as the finite extent convolution model of eqn. (6) was developed in this manner. The M even case, though not presented here, is developed in a manner similar to that previously discussed for the M odd case.

4. Experimental Results

To experimentally verify the results of our inverse filter realization, we select a well-behaved, i.e. no frequency nulls, $M=5$ tap FIR filter $h = [0.05, 0.2, 0.4, 0.2, 0.05]$. This filter is symmetric thus its discrete time Fourier transform (DTFT) is real and the filter is also linear phase. Using the approach developed in section 3, we selected odd filter lengths (values of Q) of increasing size and determined h_{inv} for each. As Q increased, the approximation accuracy increasingly improved. In Figures 2 and 3, we report results for the computed h_{inv} of length $Q=17$. In Fig. 2, the resulting 17 tap inverse filter is plotted. In Fig. 3, the result of the linear convolution between h and h_{inv} is given. One can notice that the result is very close to the ideal delta sequence that should result. In Fig. 4, we have plotted the reciprocal of the DTFT of the 5 tap filter h , i.e., $1/H(\omega)$ (solid curve). Superimposed on this plot is the DTFT of the 17-tap FIR inverse filter that was computed, i.e. $H_{inv}(\omega)$ (dashed curve). It is evident that there is very good agreement between these two curves.

To test the inverse filter with an image, we created a 2-D separable 5×5 filter using the 5-tap filter just described. We used the standard 256×256 sized ‘Lena’ image depicted in Fig. 5a and filtered it using the finite extent convolution model of eqn. (1). The valid filtered portion resulted in an image of size 252×252 that is depicted in Fig. 5b. Then, we created a 2-D separable inverse filter of size 17×17 using the 17-tap inverse filter of Fig. 2. Again, using the convolution model of eqn. (1), we convolved the blurred image in Fig. 5b with the inverse filter. This resulted in a *valid* image of size 236×236 that is depicted in Fig. 5c. Clearly, all of the details of the original image of Fig. 5a have been preserved in Fig. 5c thus illustrating the effectiveness of removing the blurring evident in Fig. 5b.

We stress that no noise was added to the blurred image prior to performing our inverse filtering. That is, the point of this example was to demonstrate the accuracy of the inverse filtering. It is clear that direct inverse filtering a noisy image will amplify the noise in the signal. For relatively high SNRs, this is not problematic as the noise will not be visually perceptible in the restored image. For low SNRs, direct inverse filtering is typically considered unacceptable and methods that incorporate noise constraints and/or that make use of a priori image models are deemed more appropriate, albeit at a substantially increased computational cost. Ultimately, the inverse filter’s ability to undo blurring from a well-behaved filter must be weighed against any noise amplification that may result from its use.

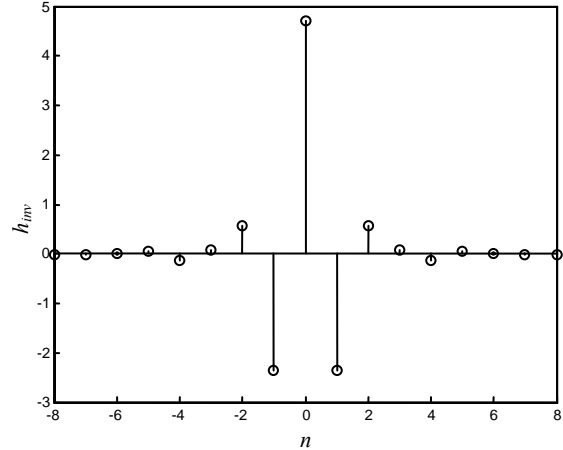


Figure 2. The inverse filter $h_{inv}[n]$ corresponding to the blurring filter $h=[0.05, 0.2, 0.4, 0.2, 0.05]$. It was approximated out to 17 samples using the time domain approach of section 3.

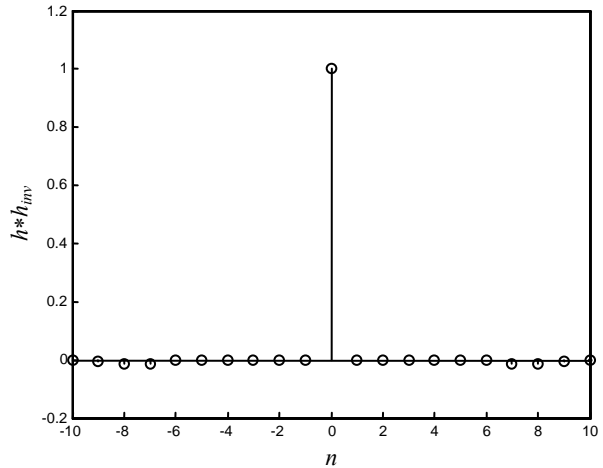


Figure 3. Convolution of the original 5 sample filter $h[n]$ with its approximated 17 sample inverse $h_{inv}[n]$. The near delta function response clearly demonstrates the accuracy of the $h_{inv}[n]$.

5. Conclusion

A time-domain approach for determining the 1-D FIR inverse filter of an original FIR filter has been presented. The method was based on a finite extent convolution model. Using the minimum norm least squares solution for the convolution model’s underdetermined system of equations, we showed that the inverse filter is determined by a convolution between the original filter h and the inverse of its autocorrelation sequence. The process of determining the inverse filter was succinctly summarized in a simple 6 step procedure. In general, the larger the size of the inverse filter, the more accurate the inverse filter is.

Given that the inverse filter is in FIR form, the original filter h should not have any frequency nulls. If it does, any reasonable length FIR filter will only result in a gross approximation to the true inverse of the original filter. In these instances, an AR model is a better choice to model the inverse filter.

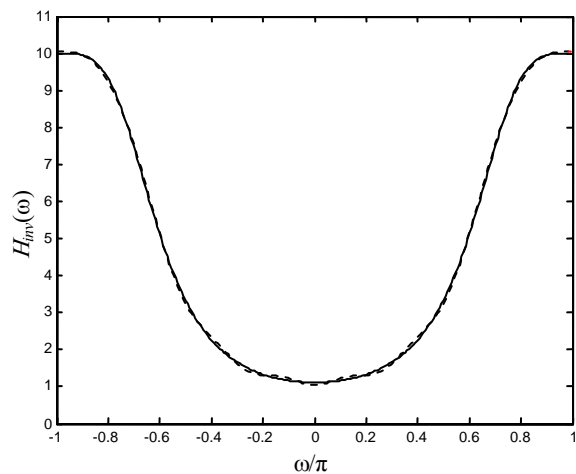


Figure 4. Spectral illustration of the approximation accuracy of $h_{inv}[n]$. We computed $H(\omega)$, the DTFT of the 5 sample filter $h[n]$, and $H_{inv}(\omega)$, the DTFT of the 17 sample filter $h_{inv}[n]$, and have plotted $1/H(\omega)$ (solid curve) and $H_{inv}(\omega)$ (dashed curve). There is excellent agreement between them.

6. Acknowledgement

This work was supported in part by NSF grant ECS-0508218 and by UNCFSP/DOD under grant W911NF-04-1-0045.

7. References

- [1] J. Juang, Applied System Identification, Upsaddle River, NJ: Prentice Hall, 1993.
- [2] A.K. Katsaggelos, ed., Digital Image Restoration, New York: Springer-Verlag, 1991.
- [3] A.V. Oppenheim and R.W. Schaffer, Discrete-Time Signal Processing, Englewood Cliffs, NJ: Prentice Hall, 1989.
- [4] P. Stoica and R. Moses, Spectral Analysis of Signals, Upsaddle River, NJ: Prentice Hall, 2005.
- [5] A.K. Jain, Fundamentals of Digital Image Processing, Upsaddle River, NJ: Prentice Hall, 1988.
- [6] A. Graham, Kronecker Products and Matrix Calculus: with Applications, New York: Halsted Press, 1981.
- [7] T.K. Moon and W.C. Stirling, Mathematical Methods and Algorithms for Signal Processing, Prentice Hall, 1999.
- [8] G.H. Golub and C.F. Van Loan, Matrix Computations, Baltimore: John Hopkins University Press, 2nd Ed., 1989.



Figure 5. Results of deconvolution with a separable inverse filter based on the filter shown in Fig. 2. (a) Original 'Lena' image, (b) valid portion of the FIR filtered 'Lena' image, (c) valid portion of the image in 'b' FIR filtered with the separable inverse filter. Given our finite extent convolution model, the deconvolution (without noise) is excellent but necessarily results in an image of reduced support size.