ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850

APPLICATION NOTE NO. 371

April 2007

A 2D WAVELET LIFTING EXAMPLE

1. INTRODUCTION

In the previous application note (AN-370), we looked at the basic notions of wavelet lifting applied to one dimensional (1D) signals. Here we will look at a program and examples where the similar procedure is applied to two dimensions (2D). The methods and results will be seen to be similar to the 1D case.

2. THE LIFTING STRUCTURE

In AN-370, a 1D signal was decomposed into even and odd samples. Here a 2D image is decomposed into two lattices, each containing half the pixels, in the so-called "quincunx" lattice (See AN-369 on 2D sampling). The notation with the delay (z^{-1}) and downsamplers (↓2) is meant to represent this decomposition. Likewise the upsamplers (↑2) and advance (z) on the right side represent the interleaving of the two quincunx components into a single image. Otherwise, the only difference is that the filterings (convolutions) for the P and U operations are 2D.

We will begin by more-or-less just presenting a Matlab program and showing a sequence of example images, and will discuss the results afterward.

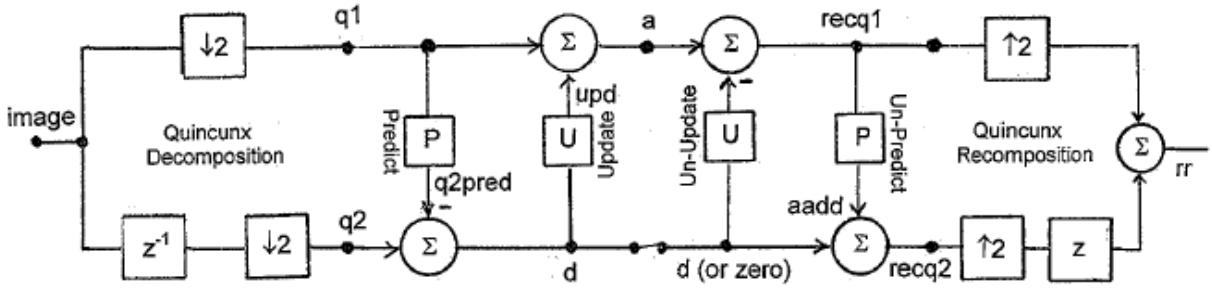


Fig. 1 The basic lifting scheme for 2 Dimensions

3. THE PROGRAM

```
% imagelft.m

% Standard "lena" image
load('lena');
lena=double(lena);

% Alternative "impulse" image
zer=zeros(256,256);
imp=zeros(256,256);
imp(5,4)=255;
% lena=imp;   % comment out to use impulse testing

%   QUINCUNX q1
q=[1 0;0 1];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q1s=q;
% OFFSET QUINCUNX q2
q=[0 1;1 0];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q=[q q;q q];
q2s=q;

% "DOWNSAMPLED"
q1=lena.*q1s;
q2=lena.*q2s;
```

```
% DISPLAY ORIGINAL AND TWO DOWNSAMPLE
figure(1)
colormap('gray')
imagesc(lena)
axis('square')
title('Original Lena 256x256')
figure(2)
colormap('gray')
imagesc(q1)
axis('square')
title('Quincunx Sampled Version 1, q1')
figure(3)
colormap('gray')
imagesc(q2)
axis('square')
title('Quincunx Sampled Version 2, q2')

% TO RESTORE "BRIGHTNESS," HOLD HORIZONTALLY - JUST FOR DISPLAY
q1hold=conv2(q1,[1 1],'same');
figure(4)
colormap('gray')
imagesc(q1hold)
axis('square')
title('Ver 1 Held')

%  LINEAR INTERPOLATION OF q1, CALL lenalin - FOR COMPARISON
linmask=[0 1/4 0;1/4 1 1/4;0 1/4 0]
lenalin=conv2(q1,linmask,'same');
figure(5)
colormap('gray')
imagesc(lenalin)
axis('square')
title('q1 Linear Interp')

% BEGIN LIFTING - PREDICT THE q2 SAMPLES
predmask=[0 1/4 0;1/4 0  1/4;0 1/4 0]
q2pred=conv2(q1,predmask,'same');
figure(6)
colormap('gray')
imagesc(q2pred)
axis('square')
title('Predicted q2')
```

```
%  PREDICTION ERROR d
d=q2-q2pred;
figure(7)
colormap('gray')
image(d)
axis('square')
title('Prediction Error d')

%  COMPUTE UPDATE FROM d
updatem=predmask/2
upd=conv2(d,updatem,'same');
figure(8)
colormap('gray')
image(upd)
axis('square')
title('Update')

%  COMPUTE a, UPDATED q1
a=upd+q1;
figure(9)
colormap('gray')
imagesc(a)
axis('square')
title('a')

% BEGIN SYNTHESIS, UPDATE a TO GET RECOVERED recq1
recq1=-conv2(d,updatem,'same')+a;
figure(10)
colormap('gray')
imagesc(recq1)
axis('square')
title('a Unupdated = Recovered q1')

% NOW PREDICT recq2, COMPONENT TO BE ADDED TO d
aadd=conv2(recq1,predmask,'same');
figure(11)
colormap('gray')
imagesc(aadd)
axis('square')
title('Predicted Component from recq1')
```

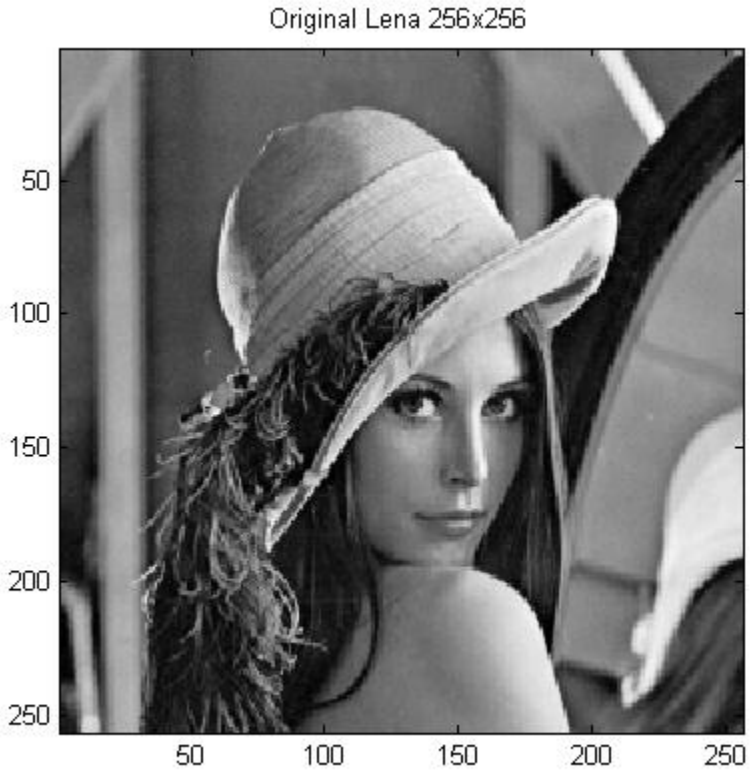
```
% RECOVERED q2
recq2=d+aadd;
figure(12)
colormap('gray')
imagesc(recq2)
axis('square')
title('Recovered q2')

r=recq1+recq2;
figure(13)
colormap('gray')
imagesc(r)
axis('square')
title('Recovered by Lifting')

% line below for impulse testing
% r(1:9,1:9)


% NOW LOOK AT CASE WHERE d=0

d=0*d;

%a=imp;
%d=zer;


% BEGIN SYNTHESIS, UPDATE a TO GET RECOVERED recq1
recq1=-conv2(d,updatem,'same')+a;


% NOW PREDICT recq2, COMPONENT TO BE ADDED TO d
aadd=conv2(recq1,predmask,'same');
% RECOVERED q2
recq2=d+aadd;

rr=recq1+recq2;
figure(14)
colormap('gray')
imagesc(rr)
axis('square')
title('Recovered by Lifting With d=0')

% line below for impulse testing
 rr(1:10,1: 10)
```

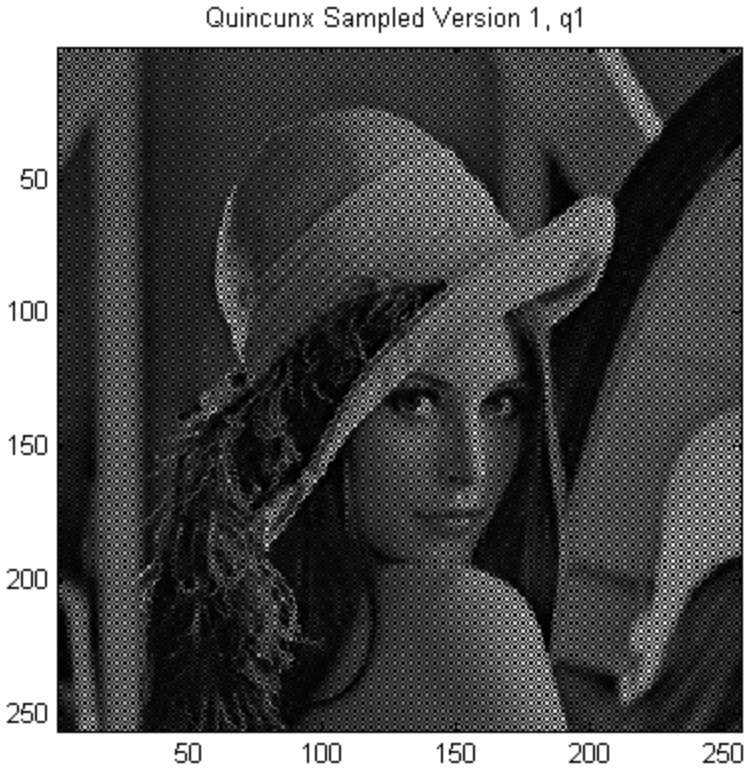
4. IMAGES

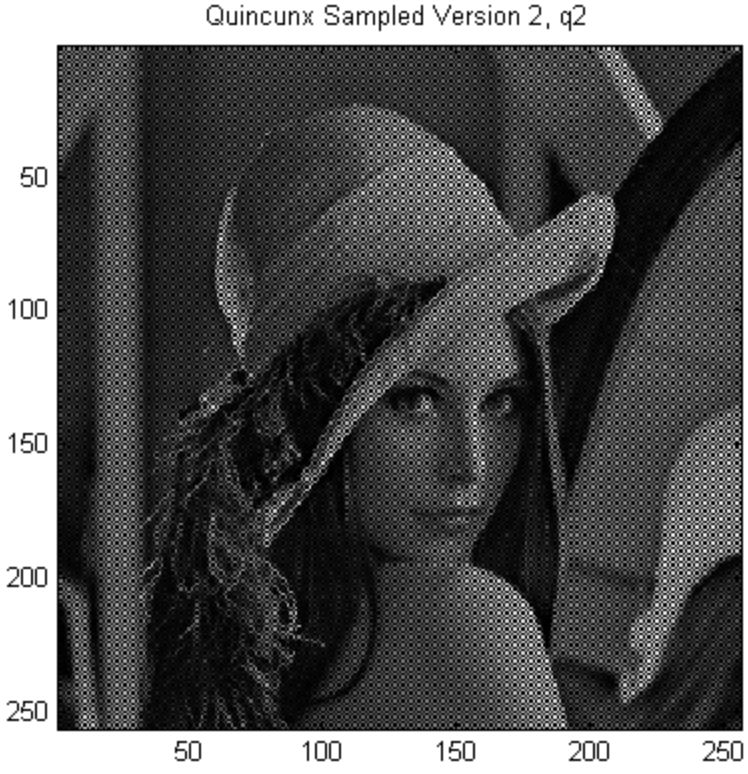
We begin with the original "Lena" image (Fig. 1 of program) that is a standard in many image processing studies. The image has an interesting history which some readers may want to look up. This version is 256 x 256 pixels.

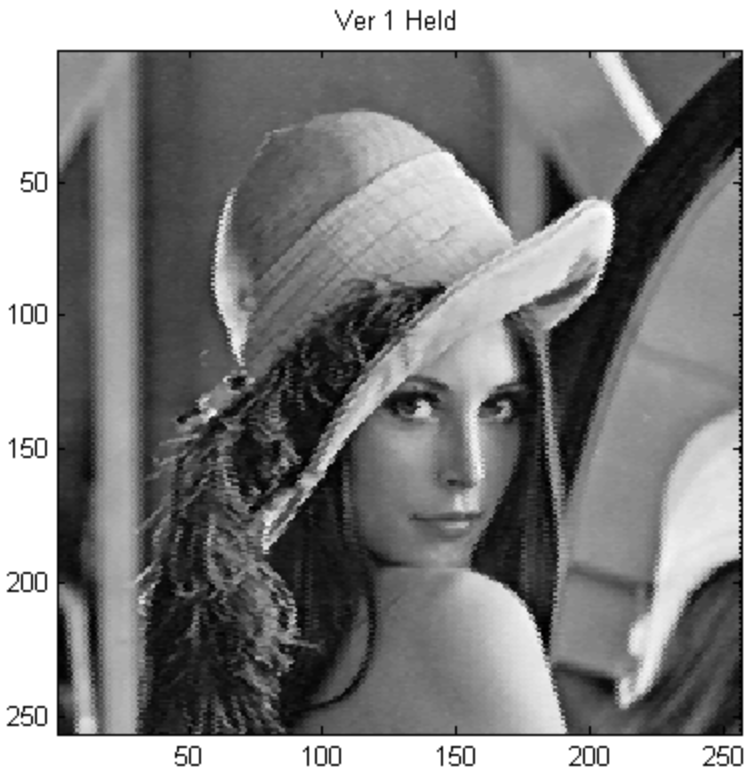
The two images on page 7 (Figures 2 and 3 of program) are the two quincunx samplings. We immediately see that the images are first of all dark, and secondly, details are lost. Both are a consequence of the fact that half the samples have been set to zero. We could if we wished actually remove half the samples which we know to be all zeros. Since we know exactly which pixels have been forced to zero, we would not need to store them or transmit them in practice.

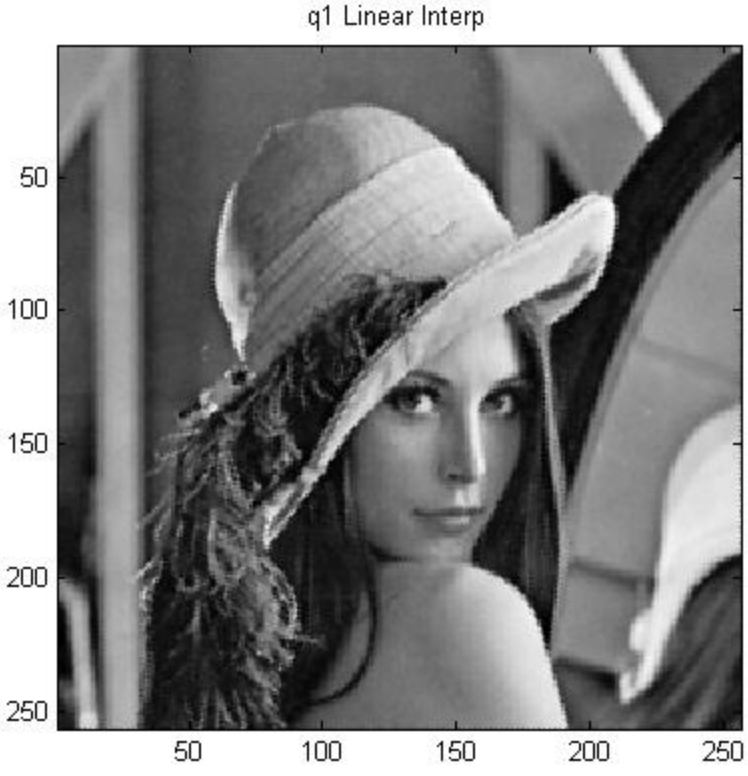
Since this darkness is a distraction, for comparison (not part of the lifting), we show the top image on page 8 (Fig. 4 of program), which is a "held" version of the q1 sampling. That is, every pixel that was set to zero is replaced with the pixel that is horizontally to its left. Now the image is essentially as bright as the original, but lacks definition, and the horizontal blurring is evident. The bottom image on page 8 (Fig. 5 of the program) shows the linear interpolated version of the q1 sampling. This will be a fundamental comparison image. This we obtain by convolving the q1 sampling with a




Quincunx Sampled Version 1, q1
50
100
150
200
250
50
100
150
200
250


Quincunx Sampled Version 2, q2
50
100
150
200
250
50
100
150
200
250


Ver 1 Held
50
100
150
200
250
50
100
150
200
250


q1 Linear Interp
50
100
150
200
250
50
100
150
200
250

matrix:

$$\begin{matrix} 0 & 1/4 & 0 \\ 1/4 & 1 & 1/4 \\ 0 & 1/4 & 0 \end{matrix}$$

This means that every original sample is retained exactly while the samples that were set to zero are here replaced by 1/4 of each of the four nearest neighbors. In contrast, the "matrix" for the hold was:

$$\begin{matrix} 1 & 1 \end{matrix}$$

With the linear interpolation, we regain the brightness in a much smoother fashion. We have not added any information of course, but the result is less jagged. In fact, it is a bit difficult to imagine a "simple" restoration that could do as well. It's not bad – just less sharp than the original. We want to see if lifting does as well.

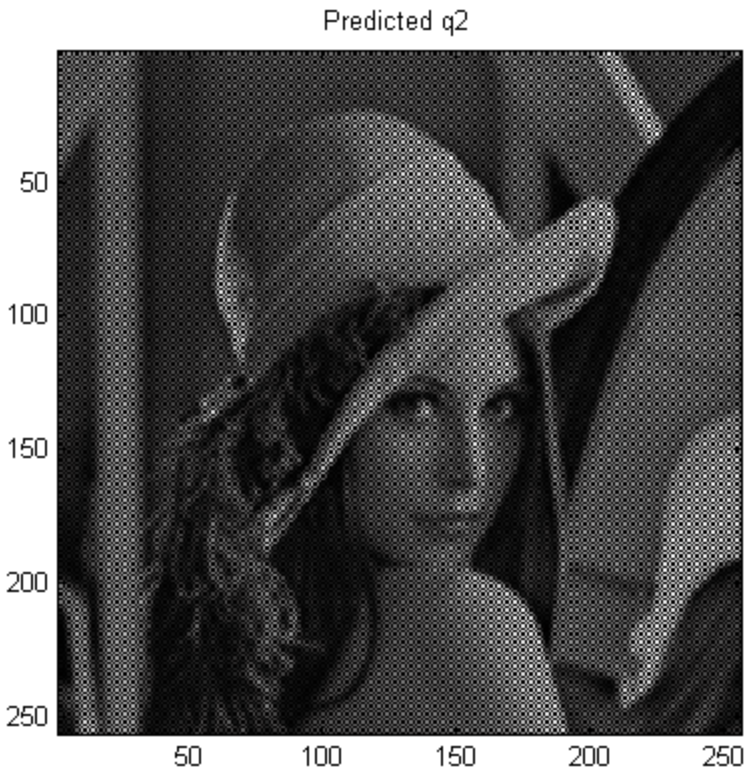
The lifting procedure is illustrated beginning with the top image on page 10, which is the predicted version of the quincunx sampling q2, predicted from q1. This is Fig. 6 of the program, and note that it was obtained by convolving q1 with the matrix (P):

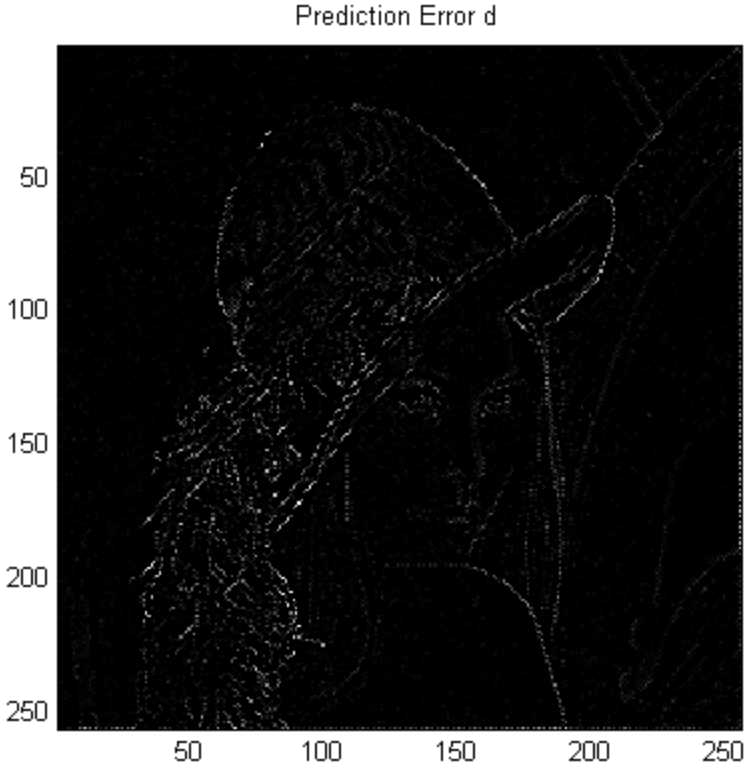
$$\begin{matrix} 0 & 1/4 & 0 \\ 1/4 & 0 & 1/4 \\ 0 & 1/4 & 0 \end{matrix}$$

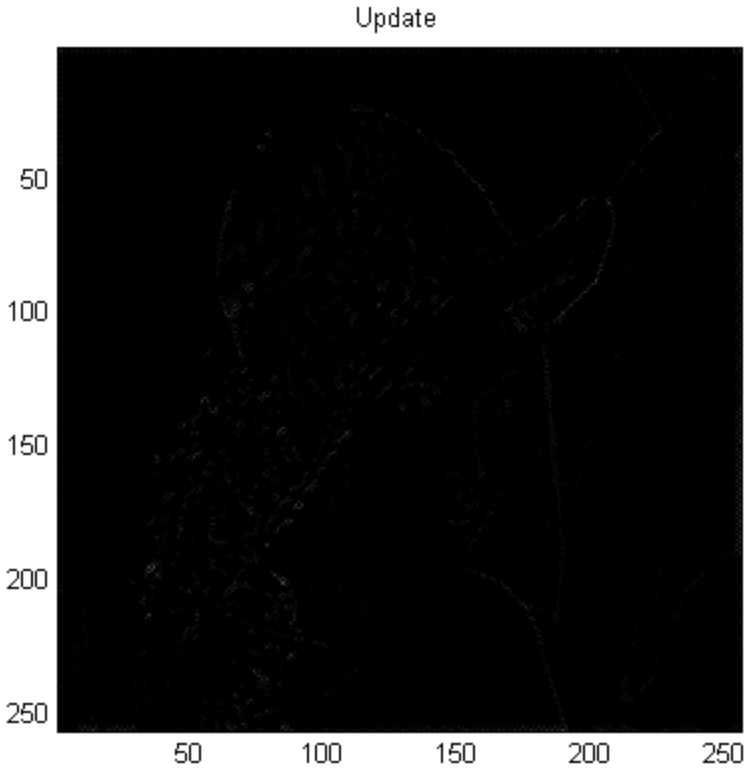
which is the same as the linear interpolation matrix with the important exception that the center element is zero instead of one.

What we are doing here is completely analogous to the one dimensional case where odd samples were predicted to be the average of the even samples. We can compare the predicted q2 with the actual q2, and they look quite similar, the predicted version being slightly less sharp. This is pretty much what we would expect. But what is the difference? This we see in the bottom image on page 10 (Fig. 7 of the program) which is the prediction error. As we might expect, we are subtracting two quite similar images, and the difference is small, black most everywhere; except of course at the "edges," the places where the image has sudden changes of intensity – as a particular example, the edges of the hat.

The scheme of Fig. 1 shows that the error d has two ways to go. It may or may not be transmitted by the switch for the reconstruction, but it is always used to modify q1 into the image a. In general, we suppose that if d is small enough (as it seems to be here) it might be ignored, so that we could get by with only a, half the data. This is the same reduction factor (1/2) t hat we get if we just keep q1 and linearly interpolate it.


Predicted q2
50
100
150
200
250
50
100
150
200
250

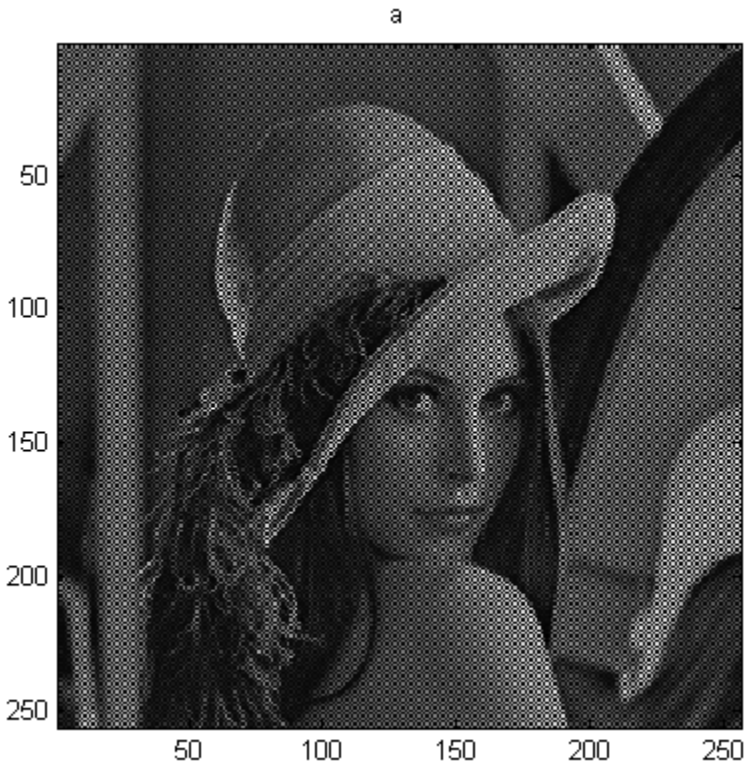

Prediction Error d
50
100
150
200
250
50
100
150
200
250

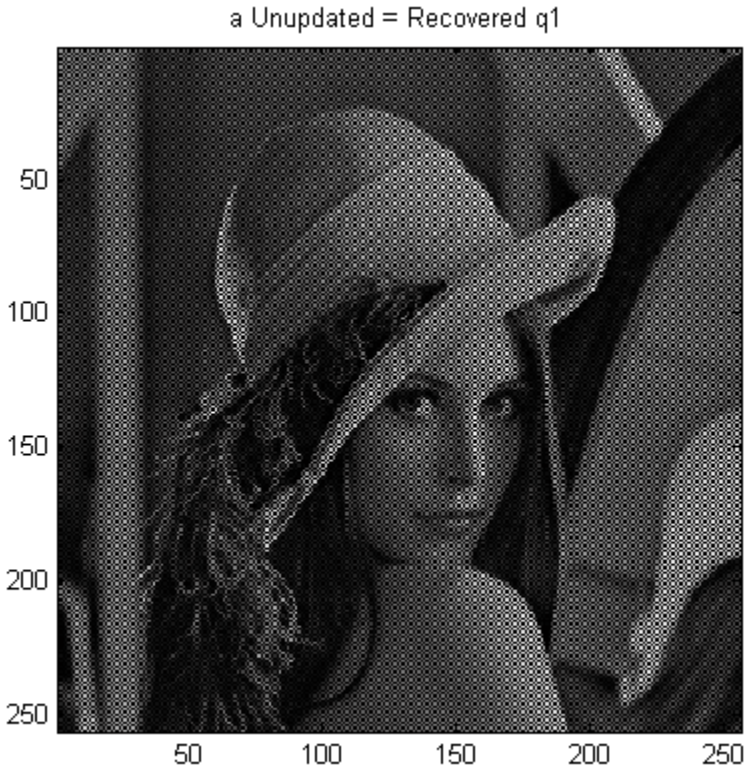


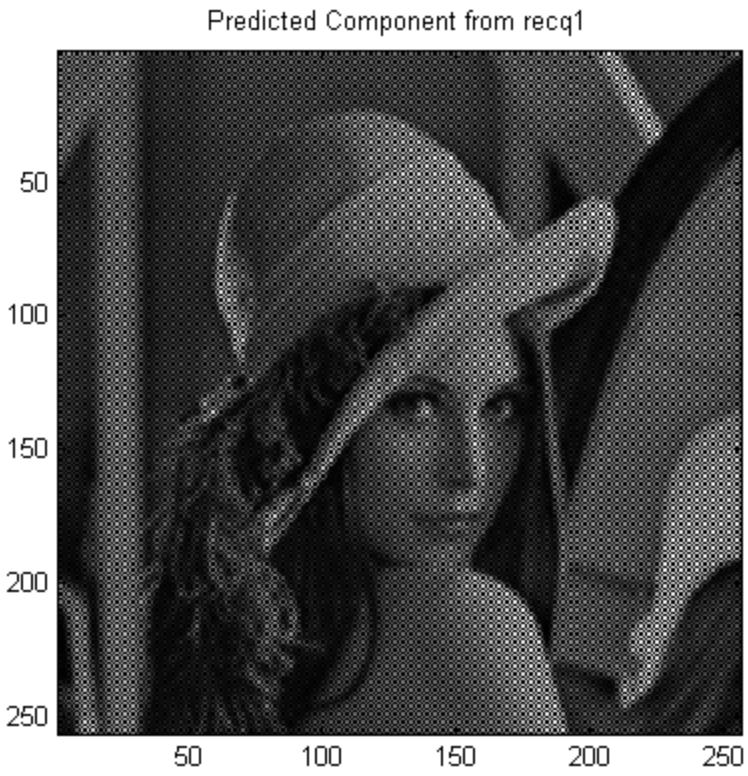
In order to actually obtain the image a, we need to go through the “updating” operation, and here we choose U=P/2, as in the 1D case. Not surprisingly, this update of something that is already dark is something even darker (figure above on this page). This update is added to q1 to give a (top figure on page 12), which looks very much like q1 itself.

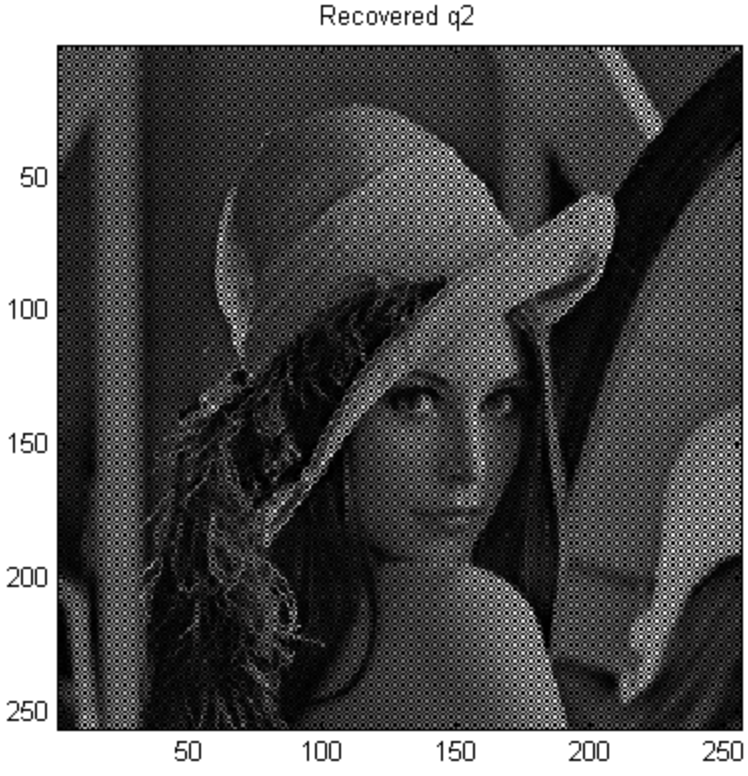
Now we are ready to look at the reconstruction (right side) of Fig. 1, and it is going to make a difference whether we block d by opening the switch, or pass it through. Keeping d, we “un-update” and recover q1 (recq1 – bottom figure of page 12), and “un-predict” req1 to give a recovered q2 as recq2. These are added to give back, exactly, the original (top image on page 14 is same as original image on page 6). This perfect recovery will work in general, regardless of our choice of P and U.

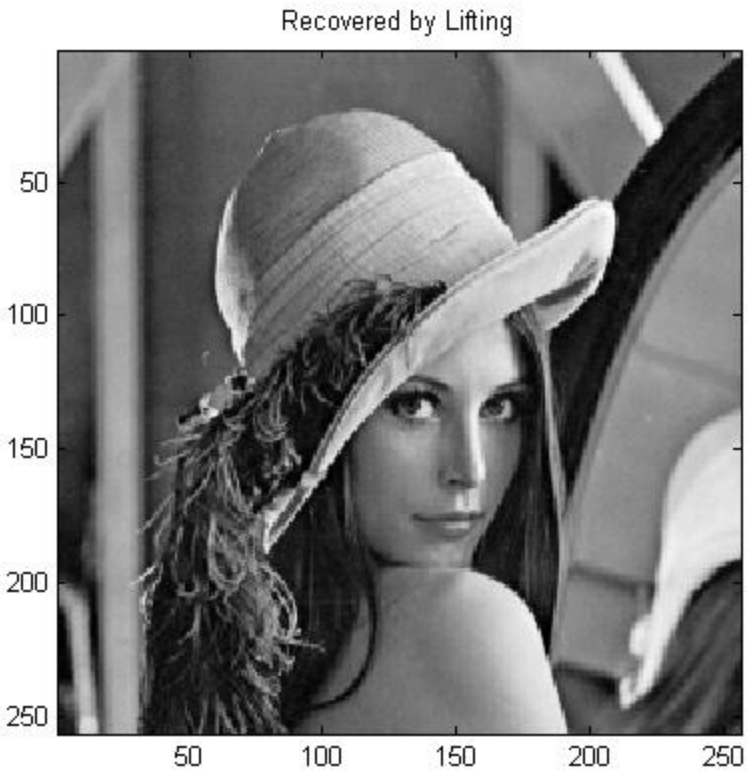
So nothing of great significance has been demonstrated so far, except that our deconstruction and successful reconstruction efforts indicate we must of gotten things “lined up” properly. We have to show this first, and we have. So we next need to look at the actual interesting case where we set d to zero, after it has been used to update to a, and before we do the reconstruction (I.e., open the switch).

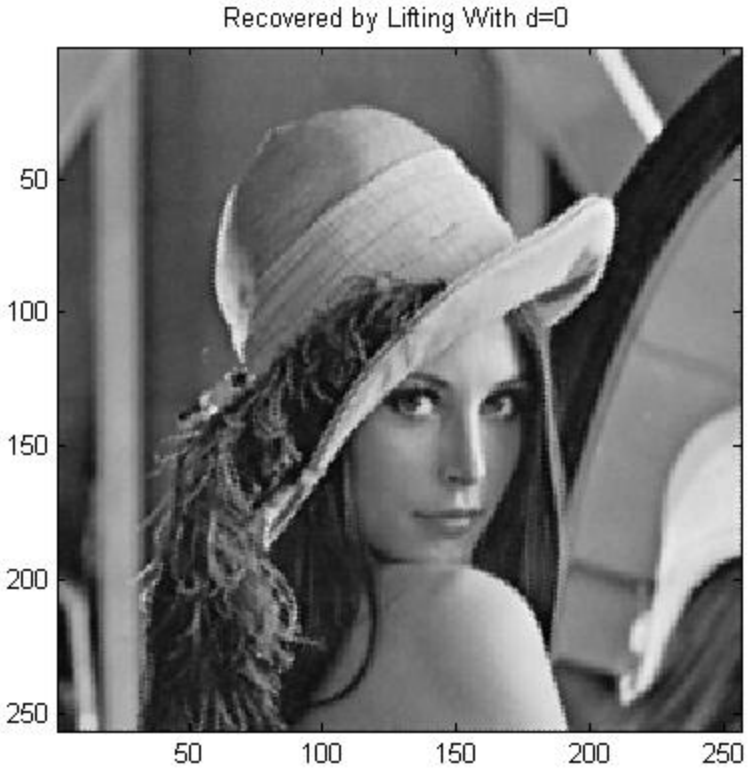

a
50
100
150
200
250
50
100
150
200
250


a Unupdated = Recovered q1
50
100
150
200
250
50
100
150
200
250


Predicted Component from recq1
50
100
150
200
250
50
100
150
200
250


Recovered q2
50
100
150
200
250
50
100
150
200
250


Recovered by Lifting
50
100
150
200
250
50
100
150
200
250


Recovered by Lifting With d=0
50
100
150
200
250
50
100
150
200
250

The result is seen in the bottom figure on page 14, which is visibly less sharp than the full recovery (= original) just above it. Below we will show that lifiting with d=0 is equivalent here to the linear interpolation of a.

So the interesting question is: Is the linear interpolation of a better than the linear interpolation of q1? That is, compare the image at the bottom of page 8 to the image at the bottom of page 14. This is fairly hard to judge, but many people will have a slight preference for the lifting case. A lot depends on viewing distance, whether the images are viewed on a computer screen (and what type of screen) or printed out (and what type of printer.

5. A TOY PROGRAM FOR "IMPULSE RESPONSE" TESTING

Below is a simplified version of the lifting program which we want to use to look at a very small, hand constructed image, which we can set up to input images that are 2D impulses:

```
function it1(x1,x2,dmult)

x=zeros(8,8);
x(x1,x2)=1
s1=[1 0 1 0 ;
    0 1 0 1 ;
    1 0 1 0 ;
    0 1 0 1];
s1=[s1 s1;s1 s1];
s2=abs(s1-1);
q1=x.*s1;
q2=x.*s2;
P=[0 1/4 0;1/4 0 1/4;0 1/4 0];
U=P/2;
q2pred=conv2(q1,P,'same');
d=q2-q2pred;

upd=conv2(d,U,'same');
a=upd+q1;
d=d*dmult;
recq1=-conv2(d,U,'same')+a;
aadd=conv2(recq1,P,'same');
recq2=d+aadd;

r=recq1+recq2;

%ratr=rats(r)
r128=r*128
```

We can arrange for an "impulse" to appear as an input image x either on the q1 quincunx or on the q2 quincunx

```
xq1 =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

```
xq2 =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

If we perform the lifting operation, maintaining d through the reconstruction, the output (impulse response) will of course be the same as the input. On the other hand, we can set d=0 and find the outputs. The response to xq1 is

```
     0     0     0    -1     0     0     0     0
     0     0    -3    -4    -3     0     0     0
     0    -3    -8    23    -8    -3     0     0
    -1    -4    23   112    23    -4    -1     0
     0    -3    -8    23    -8    -3     0     0
     0     0    -3    -4    -3     0     0     0
     0     0     0    -1     0     0     0     0
     0     0     0     0     0     0     0     0
```

and that to xq2 is

```
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     4     0     0     0     0
     0     0     8    16     8     0     0     0
     0     4    16    16    16     4     0     0
     0     0     8    16     8     0     0     0
     0     0     0     4     0     0     0     0
     0     0     0     0     0     0     0     0
```

Here the outputs are all multiplied by 128 for convenience. Thus the response to xq1 is 112/128 = 7/8 at the center, while the response to xq2 is 16/128 = 1/8 at the center, so the lifting scheme as employed here responds much more to the q1 points than it does to the q2 points. The system is not spatially-invariant (equivalent to time-invariance). Note further that the response to xq1 is 23/128 at the points that are above/below and left/right relative to the center, and no more than 8/128=1/16 anywhere else. Accordingly, the impulse response to xq1 resembles the linear interpolation result to a fair degree.

The linear interpolation result would be:

```
  0     0     0     0     0     0     0     0
  0     0     0     0     0     0     0     0
  0     0     0    32     0     0     0     0
  0     0    32   128    32     0     0     0
  0     0     0    32     0     0     0     0
  0     0     0     0     0     0     0     0
  0     0     0     0     0     0     0     0
  0     0     0     0     0     0     0     0
```

This is true - we could have just typed the values - but what is curious here is the way in which this was computed. What we did was set a=x (overriding whatever was there), with d=0. Thus the impulse response from the point a to the output is linear interpolation. This is, on the one hand a surprise perhaps, but is clearly just a result of our choice of the prediction operator P.

There are two consequence of this observation to be made. The first is that our choice of P was logical but somewhat arbitrary. If we had to come up with another choice, perhaps just to see what happens, what would we try? Our answer here is to say that we would look at a different interpolation operation.

Perhaps the more fundamental observation is that above we compared our lifting result (bottom of page 14) to linear interpolation of the q1 samples (bottom of page 8). In the lifting scheme (setting d=0, the interesting case otherwise we have no data reduction), it is the lattice a that is being linear interpolated. That is, we let the q2 lattice modify q1 slightly, giving us a, and then we linearly interpolate a. Thus the a image may be thought of as an “improved” version of q1, which gives a superior output following linear interpolation.