EE368B – Image and Video Compression

Solution to Homework Set #1

Prepared by Markus Flierl

1 Lossless, memoryless encoding of images

The marginal probability can be determined by a histogram, i.e., by calculating the relative frequency of occurrence of each intensity value. Figure 1 shows the probabilities for the images *airfield, boats, bridge, harbour,* and *peppers.*

The lower bound for the smallest average code word length using a variable length code is given by the entropy of an image. The entropy can be calculated using the formula on lecture slide **Lossless Compression no. 10**, the probability of a certain gray-level-value is approximated by the relative frequency of its occurrence in the picture. For encoding each individual image separately, the values in Table 1 are obtained.

Image	airfield	boats	bridge	harbour	peppers
Entropy [bits/pixel]	7.1206	7.0333	5.7056	6.7575	7.5925

Table 1: Smallest average code word length for each image if they are encoded by an individual VLC table for each image.

If the same set of codewords is used for all images, the minimum average codeword length, i.e. $-log_2(p)$, is calculated by using the statistics of all the images together. The actual codeword length for each separate image can be found by weighting this with the probability of occurrence of the respective intensity value and summing over all possible intensity values. The results are given in Table 2. The overall minimum average code word length is 7.5129 bits/pixel.

Image	airfield	boats	bridge	harbour	peppers
Entropy [bits/pixel]	7.8362	7.4766	7.1167	7.2392	7.8959

Table 2: Smallest average code word length for each image if they are encoded with one VLC table.

The bottom right plot in Figure 1 shows the probabilities when estimating the histogram of all images together.

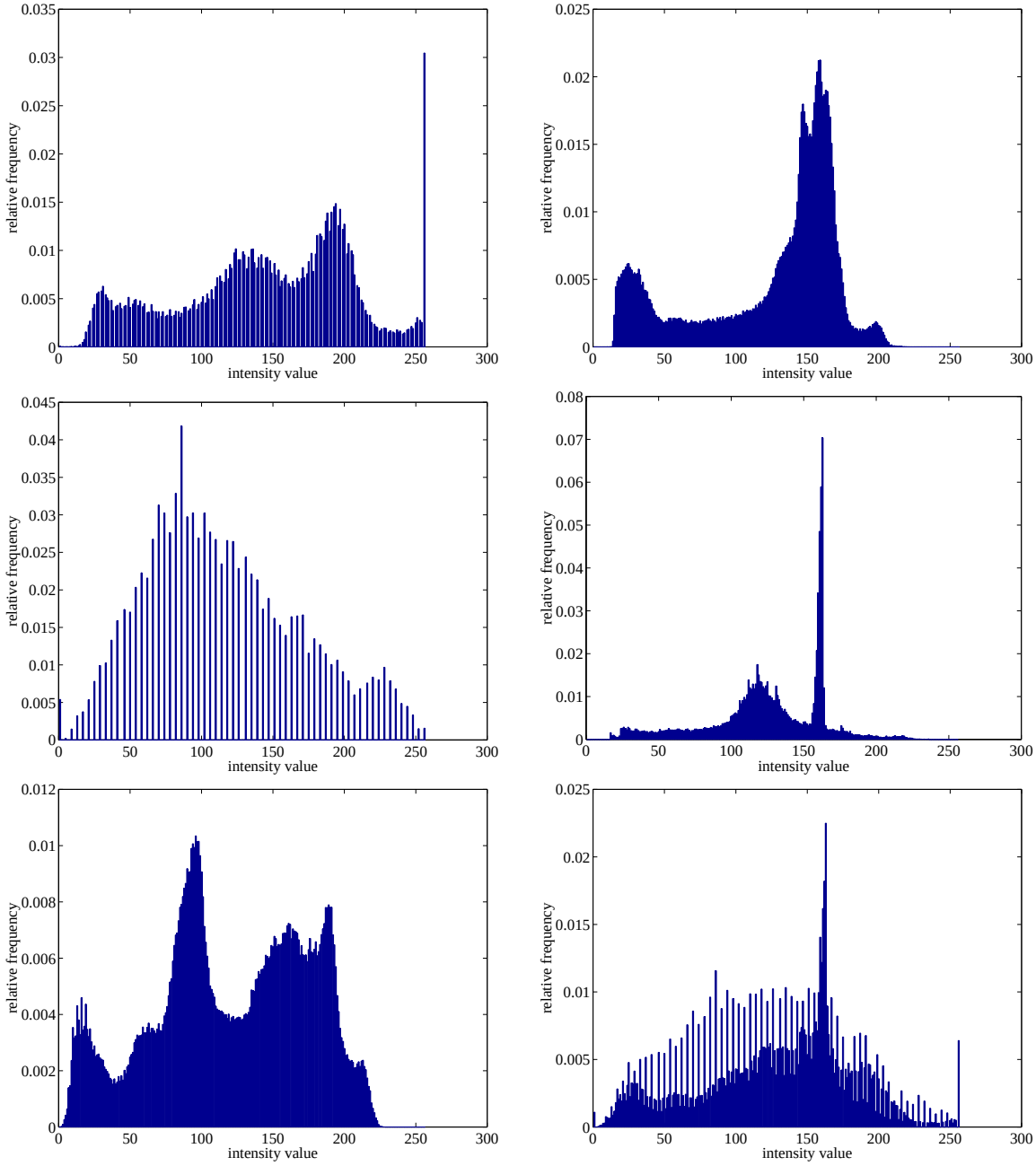

Figure 1: Marginal probabilities for the images *airfield, boats, bridge, harbour,* and *peppers*. The last plot shows the marginal probabilities for all images.

2 Lossless encoding of pixel pairs

The idea of this problem is to compare the intensity values of a pixel and its left neighbor and count the number occurrences of the resulting intensity pairs. To get a very general impression of the images statistics, we look at the pairs (p_1, p_2), then (p_2, p_3), (p_3, p_4) etc., although for the real transmission we would encode (p_1, p_2), (p_3, p_4), etc.

Figure 2 depicts the joint histogram of two horizontally adjacent pixels for the images *airfield, boats, bridge, harbour,* and *peppers.*

Table 3 contains the values for encoding each individual image separately. As those values are for the encoding of two pixels, they have to be divided by two to compare them to the values found for memoryless encoding as investigated in Problem 1.

Image	airfield	boats	bridge	harbour	peppers
Entropy [bits/pixel-pair]	12.3339	11.7790	9.6037	11.3800	12.4931
Entropy [bits/pixel]	6.1670	5.8895	4.8018	5.6900	6.2466
$\Delta H_{\text{memoryless}}$ [bits/pixel]	0.9536	1.1438	0.9038	1.0675	1.3459

Table 3: Encoding of pixel pairs: smallest average code word length for each image if they are encoded by an individual VLC table for each image.

If the same set of codewords is used for all images, the minimum average codeword length, i.e. $-\log_2(p)$ is calculated by using the statistics of all the images together. The actual codeword length for each separate image can be found by weighting this with the probability of occurrence of the respective intensity value and summing over all possible intensity values. The values are given in Table 4.

Image	airfield	boats	bridge	harbour	peppers
Entropy [bits/pixel]	6.7996	6.2643	5.8209	6.1100	6.5946
$\Delta H_{\text{memoryless}}$ [bits/pixel]	1.0366	1.2123	1.2958	1.1292	1.3013

Table 4: Encoding of pixel pairs: smallest average code word length for each image if they are encoded with one VLC table.

The entropy of all the images together is 6.3179 bits/pixel. We gain over memoryless encoding 1.1950 bits/pixel. The overall histogram is depicted in the bottom left plot of Figure 2.

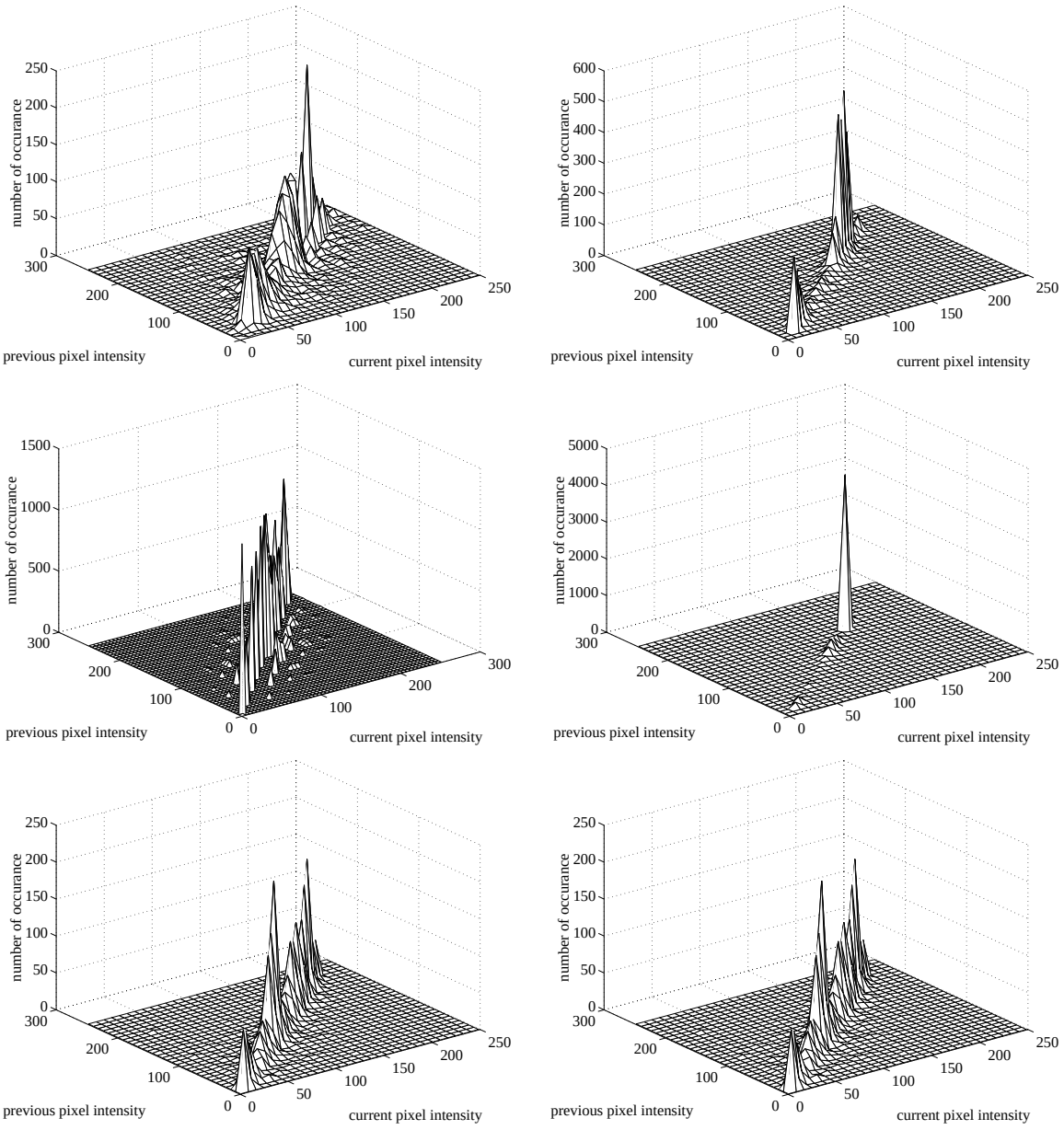

Figure 2: Joint histograms for the images *airfield, boats, bridge, harbour,* and *peppers.* The last plot shows the joint histogram for all images.

3 Lossless predictive coding of images

Using an individual code table for every single image and the prediction coefficients given on lecture slide **Lossless Compression no. 32**, Table 5 contains the entropies of the prediction errors for a given picture and prediction method A, B, or C. Predictor A uses pixel S_1 for the prediction, predictor B uses the minimum-variance approach, and predictor C uses the minimum-entropy approach.

For part B and C, the linear predictor

$$\hat{S_0} = \sum_{i=1}^{3} a_i S_i$$

provides non-integer values. But we want to predict a integer-valued signal. Therefore, it is sufficient to quantize the output of the linear predictor before we determine the integer-valued prediction error.

However, there is a little trick for this homework: Since we're not required to encode the signal, it is enough to have the center of histogram bins as an integer value when we calculate the probabilities of prediction errors. This trick works because the image values are already integer-valued.

Image	airfield	boats	bridge	harbour	peppers
H_A [bits/pixel]	5.9883	5.0234	4.4532	5.1158	5.0846
H_B [bits/pixel]	5.8940	4.5119	5.8523	5.2053	5.1111
H_C [bits/pixel]	5.8469	4.5501	5.8149	5.1603	5.0554

Table 5: Smallest average code word length for each prediction error if they are encoded by an individual VLC table for each prediction error.

Using one code table for all images and the prediction coefficients given on lecture slide **Lossless Compression no. 32**, Table 6 contains the entropies of the prediction errors for a given picture, now coded with the common code table, and prediction method A, B, or C.

Image	airfield	boats	bridge	harbour	peppers
H_A [bits/pixel]	6.1896	5.1594	5.5403	5.2626	5.2323
H_B [bits/pixel]	6.0145	4.6936	5.9173	5.2795	5.1909
H_C [bits/pixel]	5.9605	4.7056	5.8777	5.2357	5.1339

Table 6: Smallest average code word length for each prediction error if they are encoded with VLC table for prediction error.

For all images together, the values for the entropy are: 5.4769, 5.4192, and 5.3827 bits/pixel for predictor A, B, and C. We gain over joint encoding of pixels 0.8410, 0.8987, and 0.9352 bits/pixel for predictor A, B, and C.

A Script for Problem 1: Individual Codes

```
%
%    EE368B - Image and Video Compression
%
%    Lossless, memoryless encoding of images

%    Markus Flierl
%    2000-10-03

name1 = 'Images/airfield512x512.tif';
name2 = 'Images/boats512x512.tif';
name3 = 'Images/bridge512x512.tif';
name4 = 'Images/harbour512x512.tif';
name5 = 'Images/peppers512x512.tif';

img = imread(name5, 'tiff');

if(0)
  figure;
  h=imagesc(img);
  colormap('gray');
  axis('equal');
end;

[N,u]=imhist(img,256);
p = N/sum(N);

figure;
bar(u,p);
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('intensity value');
ylabel('relative frequency')

H = averageCodeWordLength(p, -log2(p+eps))
```

B Script for Problem 1: One Code

```
%
%    EE368B - Image and Video Compression
%
%    Lossless, memoryless encoding of images

%    Markus Flierl
%    2000-10-03

name1 = 'Images/airfield512x512.tif';
name2 = 'Images/boats512x512.tif';
name3 = 'Images/bridge512x512.tif';
name4 = 'Images/harbour512x512.tif';
name5 = 'Images/peppers512x512.tif';

noOfBins = 256;
N = zeros(noOfBins, 5);
p = zeros(noOfBins, 5);
sumN = zeros(noOfBins, 1);
for i=1:5
  file = eval(['name' num2str(i)]);
  img = imread(file, 'tiff');
  [N(:,i),u]=imhist(img, noOfBins);
  p(:,i) = N(:,i)/sum(N(:,i));
  sumN = sumN + N(:,i);
```

```
end;

% probability of pixel intensities over all images
pall = sumN/sum(sumN);

H = zeros(1,5);
for i = 1:5
  H(1,i) = averageCodeWordLength(p(:,i), -log2(pall+eps));
end;

disp('Avg. code word length for each image when encoded with one VLC table:');
disp(H);

figure;
bar(u,pall);
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('intensity value');
ylabel('relative frequency')

Hall = averageCodeWordLength(pall, -log2(pall+eps))
```

C Function: averageCodeWordLength.m

```
function L = averageCodeWordLength(p, l);
%averageCodeWordLength Determine average code word length
%
%    EE368B - Image and Video Compression
%
%    p : probability of events (vector)
%    l : code word length for each event (vector)

%    Markus Flierl
%    2000-10-03

p = p/sum(p);
L = sum(p.*l);
```

D Script for Problem 2: Individual Codes

```
%
%    EE368B - Image and Video Compression
%
%    Lossless encoding of pixel pairs

%    Markus Flierl
%    2000-10-03

name1 = 'Images/airfield512x512.tif';
name2 = 'Images/boats512x512.tif';
name3 = 'Images/bridge512x512.tif';
name4 = 'Images/harbour512x512.tif';
name5 = 'Images/peppers512x512.tif';

img = imread(name5, 'tiff');

counts = jointhist(img);

figure;
samples = 1:8:256;
```

```
mesh(samples, samples, counts(samples, samples));
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('current pixel intensity');
ylabel('previous pixel intensity');
zlabel('number of occurance');

p = counts./(width*height);

H=sum(sum(-p.*log2(p+eps)))

H/2
```

E Script for Problem 2: One Code

```
%
%    EE368B - Image and Video Compression
%
%    Lossless, memoryless encoding of images

%    Markus Flierl
%    2000-10-03

name1 = 'Images/airfield512x512.tif';
name2 = 'Images/boats512x512.tif';
name3 = 'Images/bridge512x512.tif';
name4 = 'Images/harbour512x512.tif';
name5 = 'Images/peppers512x512.tif';

noOfBins = 256;
N = zeros(noOfBins, noOfBins, 5);
p = zeros(noOfBins, noOfBins, 5);
sumN = zeros(noOfBins, noOfBins);
for i=1:5
  file = eval(['name' num2str(i)]);
  img = imread(file, 'tiff');
  N(:,:,i)=jointhist(img);
  p(:,:,i) = N(:,:,i)/sum(sum(N(:,:,i)));
  sumN = sumN + N(:,:,i);
end;

% probability of pixel intensities over all images
pall = sumN/sum(sum(sumN));

H = zeros(1,5);
for i = 1:5
  H(1,i) = sum(sum(-p(:,:,i).*log2(pall+eps)));
end;

disp(H/2);

figure;
samples = 1:8:256;
mesh(samples, samples, counts(samples, samples));
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('current pixel intensity');
ylabel('previous pixel intensity');
zlabel('number of occurance');

Hall = sum(sum(-pall.*log2(pall+eps)))/2
```

F Function: jointhist.m

```
function [N, bins1, bins2] = jointhist(img);
%JOINTHIST joint histogram of neighboring pixels in image
%
%    EE368B - Image and Video Compression
%
%    joint histogram of neighboring pixels in image
%
%    img : image
%    bins1 : bin value for current pixel
%    bins2 : bin value for previous pixel

%    Markus Flierl
%    2000-10-03

dim    = size(img);
width  = dim(2);
height = dim(1);

N = zeros(256,256);
for(i=1:height)
  for(j=2:1:width)
    pel1 = double(img(i,j));
    pel2 = double(img(i,j-1));
    N(pel1+1,pel2+1) = N(pel1+1,pel2+1)+1;
  end;
end;

bins1 = 1:256;
bins2 = 1:256;
```

G Script for Problem 3: Individual Codes

```
%
%    EE368B - Image and Video Compression
%
%    Lossless predictive coding of images

%    Markus Flierl
%    2000-10-03

name1 = 'Images/airfield512x512.tif';
name2 = 'Images/boats512x512.tif';
name3 = 'Images/bridge512x512.tif';
name4 = 'Images/harbour512x512.tif';
name5 = 'Images/peppers512x512.tif';

img = imread(name5, 'tiff');

[N1, N2, N3, e1, e2, e3] = prederrorhist(img);

p1=N1/sum(N1);
p2=N2/sum(N2);
p3=N3/sum(N3);

H(1)=averageCodeWordLength(p1, -log2(p1+eps));
H(2)=averageCodeWordLength(p2, -log2(p2+eps));
H(3)=averageCodeWordLength(p3, -log2(p3+eps));

disp(H);
```

H Script for Problem 3: One Code

```
%
%    EE368B - Image and Video Compression
%
%    Lossless predictive coding of images

%    Markus Flierl
%    2000-10-03

name1 = 'Images/airfield512x512.tif';
name2 = 'Images/boats512x512.tif';
name3 = 'Images/bridge512x512.tif';
name4 = 'Images/harbour512x512.tif';
name5 = 'Images/peppers512x512.tif';

noOfBins = 1021;
N1 = zeros(noOfBins, 5);
N2 = zeros(noOfBins, 5);
N3 = zeros(noOfBins, 5);
sumN1 = zeros(noOfBins, 1);
sumN2 = zeros(noOfBins, 1);
sumN3 = zeros(noOfBins, 1);
p1 = zeros(noOfBins, 5);
p2 = zeros(noOfBins, 5);
p3 = zeros(noOfBins, 5);
for i=1:5
  file = eval(['name' num2str(i)]);
  img = imread(file, 'tiff');
  [N1(:,i), N2(:,i), N3(:,i), e1, e2, e3] = prederrorhist(img);

  p1(:,i) = N1(:,i)/sum(N1(:,i));
  p2(:,i) = N2(:,i)/sum(N2(:,i));
  p3(:,i) = N3(:,i)/sum(N3(:,i));

  sumN1 = sumN1 + N1(:,i);
  sumN2 = sumN2 + N2(:,i);
  sumN3 = sumN3 + N3(:,i);
end;

% probability of pixel intensities over all images
pall1 = sumN1/sum(sumN1);
pall2 = sumN2/sum(sumN2);
pall3 = sumN3/sum(sumN3);

H = zeros(3,5);
for i = 1:5
  H(1,i)=averageCodeWordLength(p1(:,i), -log2(pall1+eps));
  H(2,i)=averageCodeWordLength(p2(:,i), -log2(pall2+eps));
  H(3,i)=averageCodeWordLength(p3(:,i), -log2(pall3+eps));
end;

disp(H(1,:));
disp(H(2,:));
disp(H(3,:));

mean(H')'
```

I Function: prederrorhist.m

```
function [N1, N2, N3, e1, e2, e3] = prederrorhist(img);
%PREDERRORHIST histogram for prediction error
%
```

```
%    EE368B - Image and Video Compression
%
%    histogram for prediction error
%
%    img : image
%    e1, e2, e2 : prediction error images
%    N1, N2, N3 : number of occurrances of prediction error values

%    Markus Flierl
%    2000-10-03

dim    = size(img);
width  = dim(2);
height = dim(1);

kern1=[0 0; 1 0];
kern2=[-5/8 3/4; 7/8 0];
kern3=[-1/2 5/8; 7/8 0];

pred1=filter2(kern1, img, 'valid');
pred2=filter2(kern2, img, 'valid');
pred3=filter2(kern3, img, 'valid');

pred1 = round(pred1);
pred2 = round(pred2);
pred3 = round(pred3);

e1   = pred1 - double(img(2:height,2:width));
e2   = pred2 - double(img(2:height,2:width));
e3   = pred3 - double(img(2:height,2:width));

[y1,x1]=hist(e1(:),-510:1:510);
[y2,x2]=hist(e2(:),-510:1:510);
[y3,x3]=hist(e3(:),-510:1:510);

N1 = y1(:);
N2 = y2(:);
N3 = y3(:);
```