Table of Contents

test_quadphase

```
% demostrate quadtree compression on wrapped phase
% lap11jul09 removed coherence references
% lap11jul09 added code to skip over signature NX & NY in .qls file
% lap14jul09 added code to test quadesahp reconstructor
% 20150312 try different example
% 20150420 try example from Laguna del Maule
```

initialize

```
close all;
clear all;
giphtpath;
printfun = 'printpdf';
format compact;
nf = 0;

srcname = '../src/pha2qls.c';
exeext  = mexext;
exename = strrep(srcname,'.c',sprintf('.%s',exeext(4:end)))
```

```
Environment variable GIPHT_HOME is set to /Users/feigl/gipht
exename =
../src/pha2qls.maci64
```

select input file

name of input file containing wrapped phase; number of columns in each interferogram; number of lines in each interferogram sphnam = 'pha_11176_21540_ort.pha'; nrows = 1230; ncols = 1420; % Iceland surge sphnam = '../IN/psp_5565_10575_ort_121x81.pha'; nrows = 81; ncols = 121; % Fawnskin laguna del maule sphnam = '/Users/feigl/Documents/data/ALOS/T113/6450fbs/In5602_21035.old/psm_5602_21035_ort.pha'; % nrows = 2000; ncols = 1200; % Laguna del Maule phabig = read_pha(sphnam,1200); phasmall=phabig(1100:1499,300:699); imagesc(phasmall);axis equal write_pha('psm_5602_21035_ort_400x400.pha',phasmall);

```
sphnam = 'psm_5602_21035_ort_400x400.pha'; nrows =          400; ncols
 =          400; % Laguna del Maule
```

build output file names

name of output file containing wrapped phase, after filtering by quad-treee resampling

```
qphnam = regexprep(sphnam,'p(\w*)_','q$1_','once');
% names of output files with gradients
grxnam =
 regexprep(regexprep(sphnam,'p(\w*)_','grx_'),'.pha','.i2','once');
grynam =
 regexprep(regexprep(sphnam,'p(\w*)_','gry_'),'.pha','.i2','once');
% name of output file containing quad tree list
qlsnam = regexprep(sphnam,'.pha','.qls');
```

set thresholds

Limit for misfit by 1-parameter model

```
%ithresh = 16;
ithresh = 127;  %
% Max for misfit by 3-parameter model
maxcmd = 8; % use 3-parameter model (ramp)
%maxcmd = 16; % use 3-parameter model (ramp)
% maxcmd = 255; % use 1-parameter model (mean)
minpix  = 4;
maxpix  = 1000;
% name of executable: not yet used
%pha2qlsname = exename;
pha2qlsname = '';
```

run the program

```
npatches =
 pha2qls(sphnam,ncols,nrows,qphnam,grxnam,grynam,qlsnam,ithresh,minpix,maxcmd,maxp
```

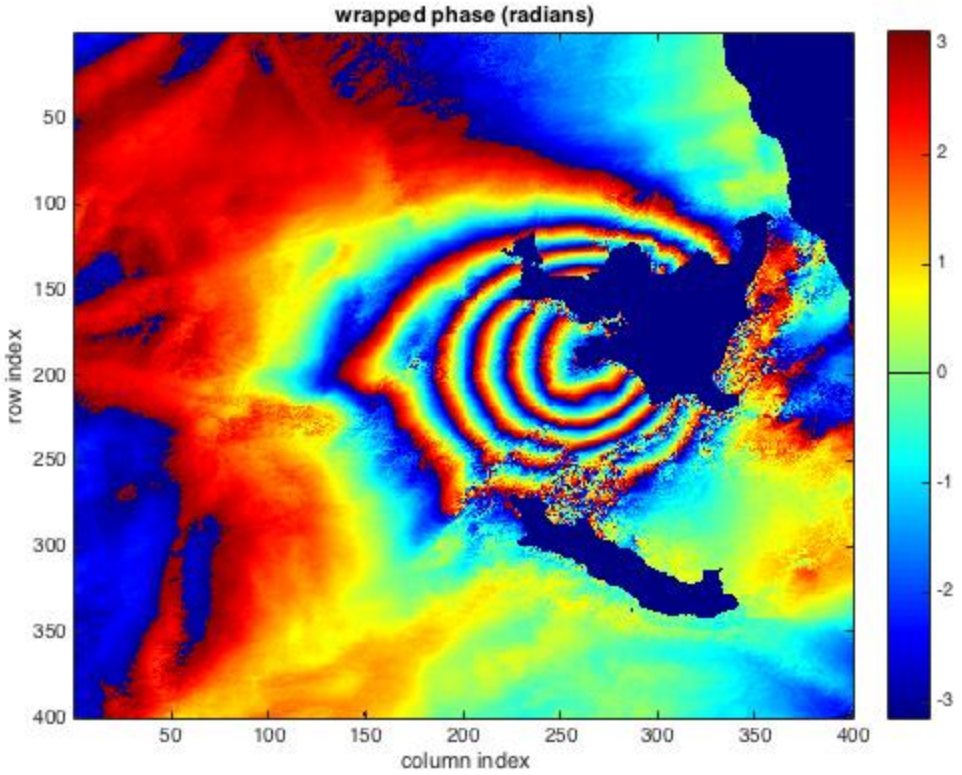
```
Starting pha2qls with command line:
/Users/feigl/gipht/src/pha2qls.maci64 psm_5602_21035_ort_400x400.pha
 400 400 -V -P qsm_5602_21035_ort_400x400.pha -X grx_400x400.i2 -Y
 gry_400x400.i2 -L 127 -N 4 -M 8 -Q 1000
pha2qls successful.
```

read images from the output files

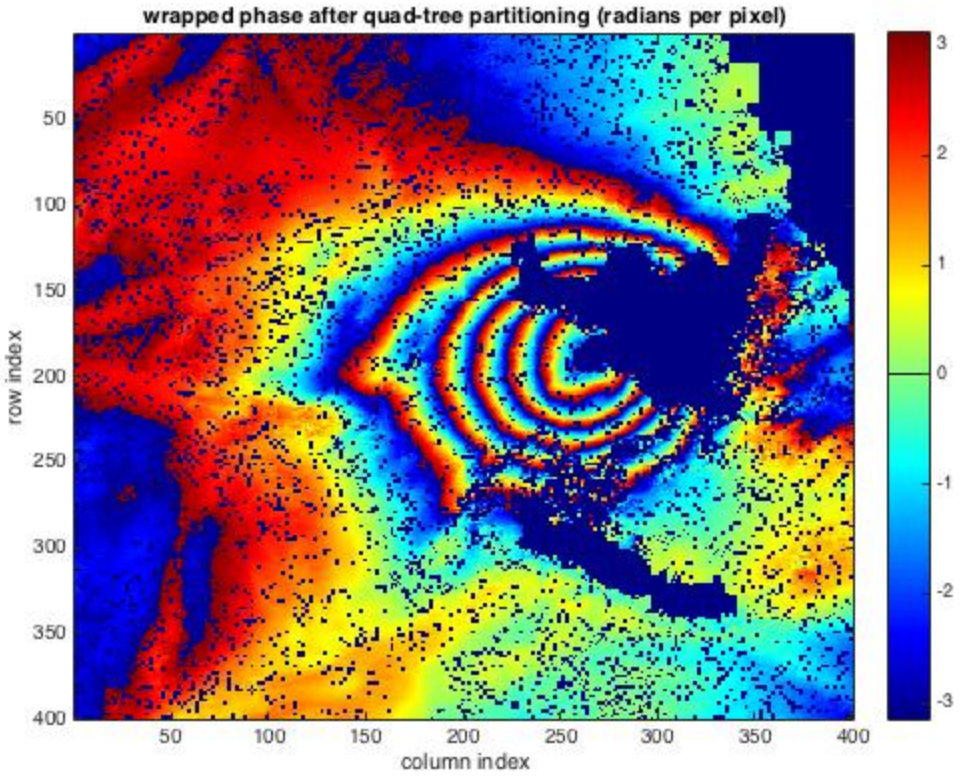
```
pbyte  = read_pha(sphnam,ncols); % original phase file
qbyte  = read_pha(qphnam,ncols); % after quad tree resampling
p  =2.0*pi*double(pbyte)/256.;   % convert to radians
q  =2.0*pi*double(qbyte)/256.;   % convert to radians
p(pbyte==0) = NaN;               % convert zeros to missing values,
 coded as NaN
```

```
q(qbyte==0) = NaN;
% p(pbyte==-128) = NaN;
% q(qbyte==-128) = NaN;
% p(pbyte== 127) = NaN;
% q(qbyte== 127) = NaN;

nf=nf+1;h(nf)=figure;imagesc(p);colorbar;cmapblackzero;
title('wrapped phase (radians)');
xlabel('column index');ylabel('row index');
feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));

nf=nf+1;h(nf)=figure;imagesc(q);colorbar;cmapblackzero;
title('wrapped phase after quad-tree partitioning (radians per
 pixel)')
xlabel('column index');ylabel('row index');
feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
```

```
Opened psm_5602_21035_ort_400x400.pha
Reshaping to 400 rows pha 400 columns = 160000 bytes as signed 1-byte
 integers
Read 400 rows pha 400 columns = 160000 bytes as signed 1-byte integers
Opened qsm_5602_21035_ort_400x400.pha
Reshaping to 400 rows pha 400 columns = 160000 bytes as signed 1-byte
 integers
Read 400 rows pha 400 columns = 160000 bytes as signed 1-byte integers
```



test_quadphase14_01.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:18:49 feigl



test_quadphase14_02.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:18:50 feigl

read the list of compressed output

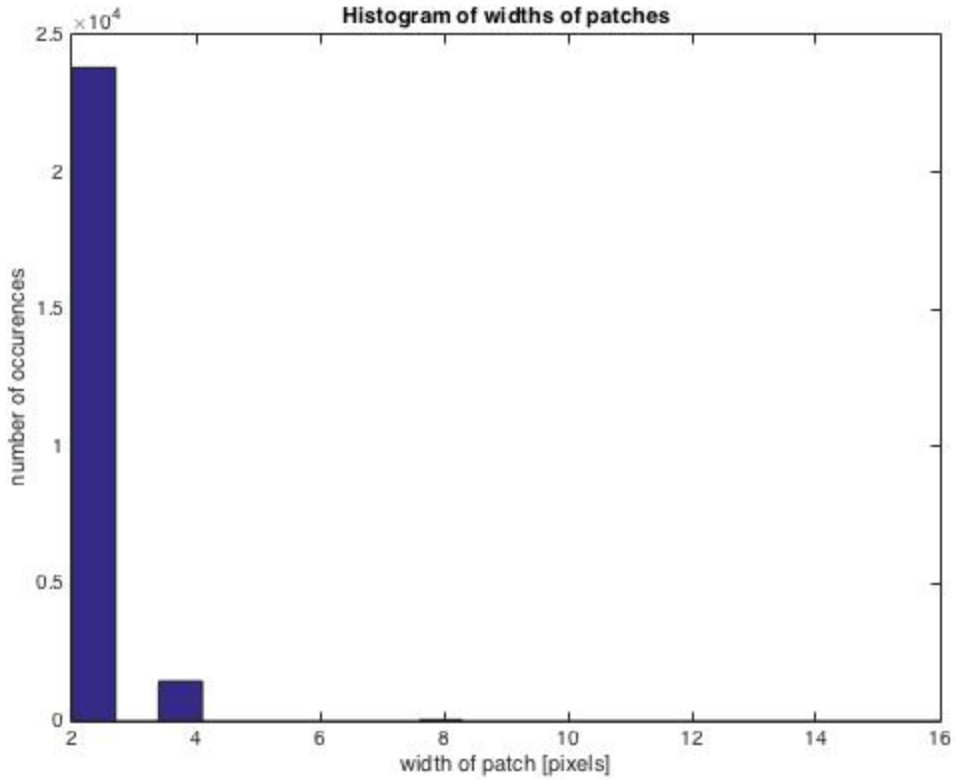
```
%[im,jm,qpv,nok,nnull,i1,i2,j1,j2,tr]=textread('qha_11176_21540_ort.qls','%d
%d%d%d%d%d%d%d%d%d%d','headerlines',1);
%qlist = read_i2('psp_11176_21540_ort.qls',6);
%qlist = read_i2('pha_11176_21540_ort.qls',6);
qlist = read_i2(qlsnam,6);
npatch = numel(qlist(:,1))-1

% Expand .phalist into .pha file
whos qlist
% nx=qlist(1,3); % Number of Cols
% ny=qlist(1,4); % Number of Rows
% if ( ncols ~= nx || nrows ~= ny )
%     disp('Warning nx & nx dont match qls file header');
% end
if typecast(qlist(1,3:4),'int32') ~= ncols  ||
 typecast(qlist(1,5:6),'int32') ~= nrows
    error(sprintf('Number of columns (%d %d) or rows (%d %d)
 incorrect.\n'...
        ,typecast(qlist(1,3:4),'int32'),ncols...
        ,typecast(qlist(1,5:6),'int32'),nrows));
end
```

```
Opened psm_5602_21035_ort_400x400.qls. Read 2 x 25304 rows x 6 columns
 = 303648 bytes WITHOUT FLIPPING as int16
```

```
npatch =
       25303
  Name           Size              Bytes  Class    Attributes

  qlist      25304x6             303648  int16
```

reconstructed images

```
r  = zeros(nrows,ncols);
r0 = zeros(nrows,ncols);
r1 = zeros(nrows,ncols);
r2 = zeros(nrows,ncols);

% find indices to pixels
qi1=double(qlist(2:end,1));  % Index to col of first pixel in patch
qj1=double(qlist(2:end,2));  % Index to row of first pixel in patch
qkw=double(qlist(2:end,3));% width of square quad

% scale to radians
qpv=2*pi*double(qlist(2:end,4))/256./256.;  % phase value
grx=2*pi*double(qlist(2:end,5))/256./256.;  % X-ward gradient of phase
 value coded as 256^2 DN = 1 cycle per pixel
gry=2*pi*double(qlist(2:end,6))/256./256.;  % Y-ward gradient of phase
 value coded as 256^2 DN = 1 cycle per pixel

% consider distribution of pixels
nf=nf+1;h(nf)=figure;
hist(qkw,20);
title('Histogram of widths of patches');
xlabel('width of patch [pixels]');
ylabel('number of occurences');
feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
```



test_quadphase14_03.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:18:51 feigl

Test reconstruction two ways, then compare

for kk=0:0

```
for kk = 0:3
% for kk = 3:3
    switch kk
        case 0
            titlestr = sprintf('after reconstruction by PHA2QLS
 (radians)');
        case 1
            titlestr = sprintf('after reconstruction by MATLAB
 (radians)');
        case 2
            titlestr = sprintf('after reconstruction by QLS2PHA
 (radians)');
        case 3
            titlestr = sprintf('deviations between two reconstructions
 (radians)');
        otherwise
            error(sprintf('unknown kk = %d\n',kk));
    end

    if kk == 0
        % reconstruction is from PHA2QLS
        r = q;
```

```
    elseif kk == 1
        % reconstruction is from Matlab
        r1=zeros(nrows,ncols);
        grxpatch = zeros(nrows,ncols);

        qi2=qi1+qkw-1;   % column index of last pixel in patch
        qj2=qj1+qkw-1;   % row index of last pixel in patch

        % Round to nearest integer
        % qim=round(double(qi1+qi2)/2.0);  % column index of middle of
 patch
        % qjm=round(double(qj1+qj2)/2.0);  % row    index of middle of
 patch
        % 20140106 Do not round to nearest integer
        qim=double(qi1+qi2)/2.0;  % column index of middle of patch
        qjm=double(qj1+qj2)/2.0;  % row    index of middle of patch

        % convert indices from C to Fortran convention
        qim = qim+1; qjm = qjm+1;
        qi1 = qi1+1; qi2 = qi2+1;
        qj1 = qj1+1; qj2 = qj2+1;


        fprintf(1,'Extrema of qi1 %d %d\n',min(qi1),max(qi1));
        fprintf(1,'Extrema of qj1 %d %d\n',min(qj1),max(qj1));

        fprintf(1,'Extrema of qi2 %d %d\n',min(qi2),max(qi2));
        fprintf(1,'Extrema of qj2 %d %d\n',min(qj2),max(qj2));

        fprintf(1,'Extrema of qkw %d %d\n',min(qkw),max(qkw));

      % 2011-MAR-24 - GREAT BIG BUG - NOW fixed in pha2qls3.c
        qqp=2*pi*double(qlist(2:end,4))/256./256.;  % phase value
        qgx=2*pi*double(qlist(2:end,5))/256./256.;  % X-ward gradient
 of phase value coded as 256^2 DN = 1 cycle per pixel
        qgy=2*pi*double(qlist(2:end,6))/256./256.;  % Y-ward gradient
 of phase value coded as 256^2 DN = 1 cycle per pixel

%        for k=1:npatch;
%            for i=i1(k):i2(k)
%                for j=j1(k):j2(k)
%                   r1(j,i) = qpv(k) + (i-imid(k))*grx(k) + (j-
jmid(k))*gry(k);
%                   grxpatch(j,i) = grx(k);
%                end
%            end
%        end

%            for(j=j1;j<=j2;j++) { // outer loop is rows NY
%            for(i=i1;i<=i2;i++) { // inner loop is cols NX
%                k=i+j*nx;  // index
%                /* Development code to watch for exceeding array
 bounds */
%                if( k >= npix ){
```

```
%                       fprintf(stdout, "Error: Exceeding %d array
 bounds at i1=%d j1=%d nwidth=%d\n", npix, i1, j1, nwidth);
%                       exit(-1);
%                   }
%
%                   /* debug code to watch for overwriting */
%                   if( debug == 1 ) {
%                       tv = pout[k];
%                       if( tv != 0 ){
%                           printf("Warning: Overwriting previous patch
 at i1=%d j1=%d nwidth=%d\n", i1, j1, nwidth);
%                       }
%                   }
%                   r = pv + (double)(i-xmid)*gx + (double)(j-
ymid)*gy; /* value in cycles */
%                   r = 256.0 * r; /* 2011-JUL-18 */
%                   //r = 256.0 * (r/256.0 - rint(r/256.0));
%                   if ( debug == 1){
%                       if(r > 127 || r < -128) {
%                           printf("Warning: Overflow at patch at i1=%d
 j1=%d nwidth=%d\n", i1, j1, nwidth);
%                       }
%                   }
%                   pout[k] = (signed char)rint(r); /* value in DN such
 that 256 DN = 1 cycle */
%                   if( debug == 1 ) fprintf(stdout, "%3d ", pout[k]);

nf=nf+1;h(nf)=figure;
hold on; axis ij;axis equal;
for k=1:npatch;
    for i=qi1(k):qi2(k) % index over columns
        for j=qj1(k):qj2(k) % index over rows
            %rphase=qqp(k);
            %r = pv + (double)(i-xmid)*gx + (double)(j-ymid)*gy; /*
            %value in radians
            if i > ncols || j > nrows || i < 1 || j < 1
                i
                ncols
                j
                nrows
                error('Problem with dimension');
            else
                if abs(qqp(k)) > 0
                    rphase = qqp(k) + double(i-qim(k))*qgx(k) +
 double(j-qjm(k))*qgy(k);
                else
                    rphase = 0;
                end
                r1(j,i) = rwrapm2(rphase);
                if qkw(k) > 4
                   plot([qi1(k) qi2(k) qi2(k) qi1(k) qi1(k)],[qj1(k)
 qj1(k) qj2(k) qj2(k) qj1(k)],'k-');
                end
            end
```

```
        end
    end
end
%r1 = rwrapm(r1);
r  = r1;
xlabel('column index I');
ylabel('row index J');
feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));


    elseif kk == 2
        % reconstruction is from C version of the reconstructor
        srcname = '../src/qls2pha.c';
        exeext  = mexext;
        exename = strrep(srcname,'.c',sprintf('.%s',exeext(4:end)))

        %[ssx,srx] = unix('../src/qls2pha.maci64
 psp_11176_21540_ort.qls -o trx.pha -d 1');
        %[ssx,srx] = unix('../src/qls2pha.a64 psp_11176_21540_ort.qls
 -o rsp_11176_21540_ort.pha');
        % PSP is after filtering according to power spectral filtering
        % Werner et al.
        %[ssx,srx] = unix('../src/qls2pha.maci64
 psp_11176_21540_ort.qls -o rsp_11176_21540_ort.pha');
        % Instead use unfiltered version, directly from Diapason
        %cmd2 = sprintf('%s %s\n',exename,'pha_11176_21540_ort.qls -o
 rha_11176_21540_ort.pha -d1')
        % reconstructed
        rphnam = regexprep(sphnam,'p??_','rsp_');

        cmd2 = sprintf('%s %s -o %s -d1\n',exename,qlsnam, rphnam)
        [ssx,srx] = unix(cmd2);

        if ( ssx ~= 0 )
            error(sprintf('FAILURE of quadphase reconstruction program
\n====Reason====\n%s==============\n',srx));
        end


  %r2=2.0*pi*double(read_pha('rsp_11176_21540_ort.pha',ncols))/256.0;

  %r2=2.0*pi*double(read_pha('rha_11176_21540_ort.pha',ncols))/256.0;
        r2=2.0*pi*double(read_pha(rphnam,ncols))/256.0;
        r =r2;
    end

    disp('dimensions of r'); size(r)
    disp('dimensions of q'); size(q)

    if kk < 3
        nf=nf+1;h(nf)=figure;
        clim = [-pi,pi];
        imagesc(r,clim);
        colorbar;colormap('jet');cmapblackzero;
```

```
        title(titlestr);
        xlabel('column index');ylabel('row index');
        feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
    end

    %     if kk == 1
    %         nf=nf+1;h(nf)=figure;
    %
imagesc(grxpatch);colormap('jet');colorbar;cmapblackzero;
    %         title('X phase gradient from QLS list (radians per
pixel)')
    %         xlabel('column index');ylabel('row index');
    %         title(titlestr);
    %         feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
    %     end

    % reconstruction deviations in phase
    switch kk
        case 0
            e=rarcm(p,q); % angular deviation
            iempty=find(abs(p)<2.0*pi/256.0);
            e(iempty)=NaN;
            iempty=find(abs(q)<2.0*pi/256.0);
            e(iempty)=NaN;
        case {1,2}
            e=rarcm(r,q);
            iempty=find(abs(r)<2.0*pi/256.0);
            e(iempty)=NaN;
            iempty=find(abs(q)<2.0*pi/256.0);
            e(iempty)=NaN;
        case 3
            % compare PHA2QLS reconstruction to QLS2PHA reconstruction
            % e = rarcm(r0,r2);
            % iempty=find(abs(r0)<2.0*pi/256.0);
            % compare Matlab reconstruction to QLS2PHA reconstruction
            e = rarcm(r1,r2);
            iempty=find(abs(r1)<2.0*pi/256.0);
            e(iempty)=NaN;
            iempty=find(abs(r2)<2.0*pi/256.0);
            e(iempty)=NaN;
        otherwise
            error(sprintf('Unknown kk = %d\n',kk));
    end

    fprintf(1,'Extrema of r %10.4f %10.4f\n',min(min(r)),max(max(r)));
    fprintf(1,'Extrema of e %10.4f %10.4f\n',min(min(e)),max(max(e)));

    % map NaN to zero for imagesc
    e0 = e;
    iempty = find(isfinite(e) == 0);
    nempty = numel(iempty)
    e0(iempty) = 0;

    % find values near zero
```

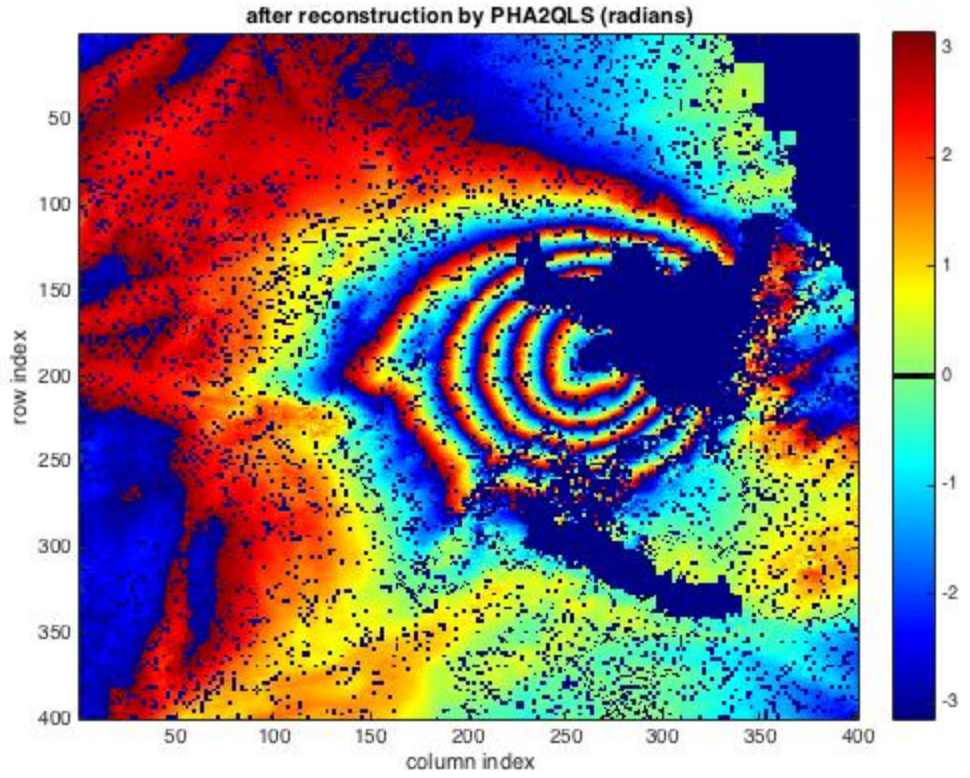
```
    izero = find(abs(e0)<=2.0*pi/256.0);
    nzero = numel(izero)
    e0(izero) = 0;

    % find values above DN threshold
    ibad2=find(abs(e)>2.0*pi/256.0);


    % map of deviations
    nf=nf+1;h(nf)=figure;
    clim = [0,pi];
    imagesc(e0,clim);

    colormap('jet');colorbar;cmapblackzero;
    %brighten(0.7);
    %imagesc(histeq(e0));colorbar;
    xlabel('column index');ylabel('row index');
    title(strcat('Nonzero Deviations:',titlestr));
    feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));


    switch kk
        case 0
            %fprintf(1,'Skipping histogram for case kk = %d\n',kk);
            res = colvec(rwrapm(p-q)); % wrapped residual
            nf=nf+1;h(nf)=figure;
            %            inonzero = 1:numel(res); % take all points

%                         inonzero = find(abs(res)>0);
            %             inonzero = find(abs(res)>=2.0*pi/256.0);
            %             iok1 = find(abs(p) > 0);
            %             iok2 = find(abs(q) > 0);
            %             inonzero = intersect(iok1,iok2);
%             % eliminate points where quadtree fails
             inonzero = find(abs(q) > 0);
%             % eliminate points where quadtree fails
%             iok0 = find(abs(q) > 0);
% %             % trim serious outliers
%             iok1 = find(res > quantile(res,0.025));
%             iok2 = find(res < quantile(res,0.975));
%             iok1 = find(res > (-127/256)*2*pi);
%             iok2 = find(res < ( 127/256)*2*pi);
%             inonzero = intersect(iok0,iok1);
%             inonzero = intersect(inonzero,iok2);

%             if numel(inonzero) > 1000
%                 nbins = floor(numel(inonzero)/20);
%             else
%                 nbins = 10;
%             end
            nbins = 64;

            hist(colvec(res(inonzero)),nbins);
            xlabel('phase (radians)');
```

```
            ylabel('Number of pixels');
            title(strcat('wrapped residual: ',titlestr));
            feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));

% %             % Quantile-Quantile plot for Von Mises Distribution
%             nf=nf+1;h(nf)=figure;
%             qqplotvonmises(colvec(res(inonzero))/2./pi,titlestr);
% %             feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));

            % Quantile-Quantile plot for normal distribution

 [hqq,phat,chi2gof_h,chi2gof_p,chi2gof_stats]=qqplot(colvec(res(inonzero)),'normal
            nf=nf+1;feval(printfun,sprintf('%s_
%02d.pdf',mfilename,nf),hqq(1));
            nf=nf+1;feval(printfun,sprintf('%s_
%02d.pdf',mfilename,nf),hqq(2));
            nf=nf+1;feval(printfun,sprintf('%s_
%02d.pdf',mfilename,nf),hqq(3));

%             % Quantile-Quantile plot for beta distribution
%             nf=nf+1;
%             res = res/max(max(res));
%             qqplot(colvec(abs(res(inonzero))),'beta');
%             feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
%
% %             % Quantile-Quantile plot for exponential distribution
%             nf=nf+1;
%             qqplot(colvec(abs(res(inonzero))),'exponential');
%             feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));

%             % Quantile-Quantile plot for generalized pareto
%             nf=nf+1;
%             qqplot(colvec(abs(res(inonzero))),'generalized pareto');
%             feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));

        case {1,2}
            nf=nf+1;h(nf)=figure;
            inonzero = find(abs(e0)>0);
            hist(colvec(e0(inonzero)),64);
            xlabel('deviations in phase (radians)');
            ylabel('Number of pixels');
            title(strcat('Nonzero deviations: ',titlestr));
            feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
        case 3
            % histogram of deviations
            nf=nf+1;h(nf)=figure;
            %hist(colvec(e),256);
            hist(colvec(e),64);
            xlabel('phase value (radians)');
            ylabel('Number of pixels');
            title(strcat('Nonzero ',titlestr));
            feval(printfun,sprintf('%s_%02d.pdf',mfilename,nf));
```

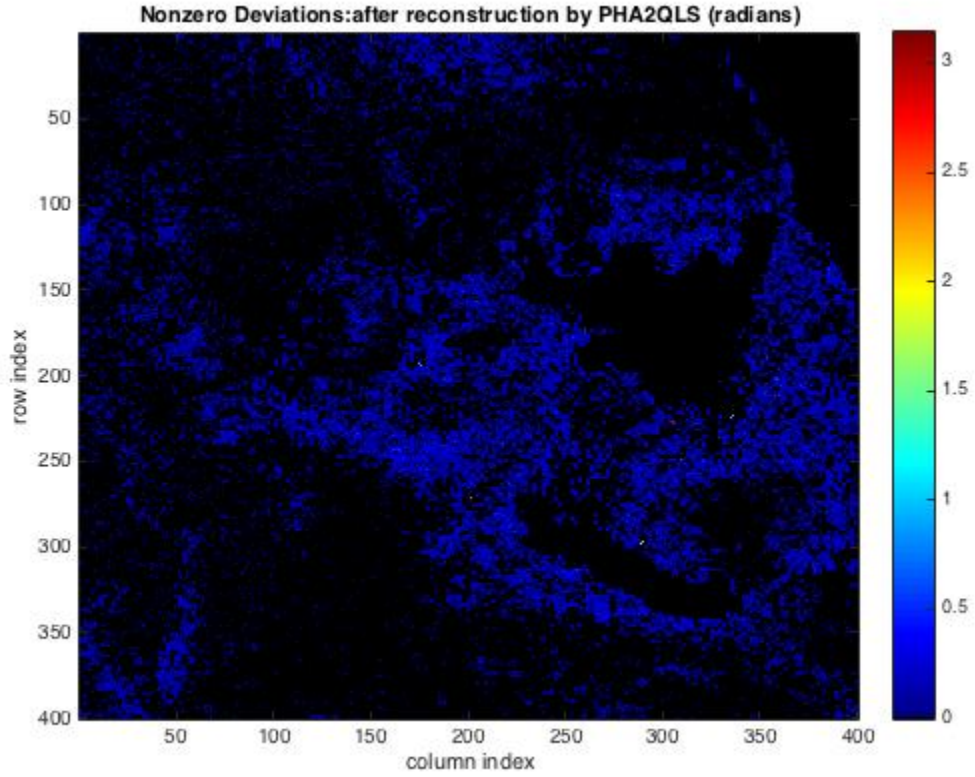
```
            fprintf(1,'maximum deviation in radians          %#12.4e
\n'  ,nanmax(colvec(e)));
            fprintf(1,'number   of deviations above threshold %12d\n'
    ,numel(ibad2));
            fprintf(1,'number   of good pixels               %12d\n'
    ,numel(e));
            fprintf(1,'fraction of pixels     above threshold %#12.4e
\n'  ,numel(ibad2)/numel(e));
        otherwise
            error(sprintf('Unknown case kk = %d\n',kk));
    end

end
```

```
dimensions of r
ans =
   400   400
dimensions of q
ans =
   400   400
Extrema of r    -3.1416     3.1170
Extrema of e     0.0000     3.1416
nempty =
       39210
nzero =
      122015
Testing the null hypothesis that the data are random sample from a
 normal distribution

The Null Hypothesis cannot be rejected at the 5.0 percent significance
 level.
Interpretation: sample is compatible with a normal distribution.
Extrema of qi1 1 399
Extrema of qj1 1 399
Extrema of qi2 2 400
Extrema of qj2 2 400
Extrema of qkw 2 16
dimensions of r
ans =
   400   400
dimensions of q
ans =
   400   400
Extrema of r    -3.1661     3.1170
Extrema of e     0.0000     0.0124
nempty =
       39395
nzero =
      160000
exename =
../src/qls2pha.maci64
cmd2 =
../src/qls2pha.maci64 psm_5602_21035_ort_400x400.qls -o
 psmrsp_5602rsp_21035rsp_ortrsp_400x400.pha -d1
```

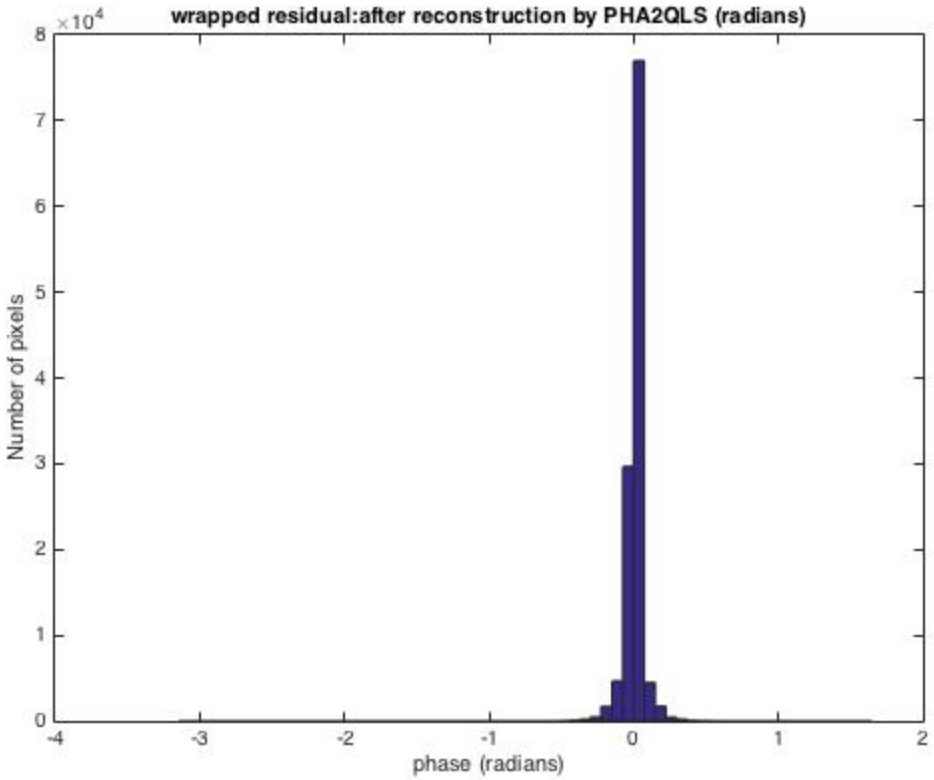
```
Opened psmrsp_5602rsp_21035rsp_ortrsp_400x400.pha
Reshaping to 400 rows pha 400 columns = 160000 bytes as signed 1-byte
 integers
Read 400 rows pha 400 columns = 160000 bytes as signed 1-byte integers
dimensions of r
ans =
   400   400
dimensions of q
ans =
   400   400
Extrema of r    -3.1416     3.1170
Extrema of e     0.0000     0.0245
nempty =
       38683
nzero =
      159973
dimensions of r
ans =
   400   400
dimensions of q
ans =
   400   400
Extrema of r    -3.1416     3.1170
Extrema of e     0.0000     0.0123
nempty =
       39395
nzero =
      160000
maximum deviation in radians             1.2272e-02
number   of deviations above threshold           0
number   of good pixels                     160000
fraction of pixels     above threshold   0.0000e+00
```



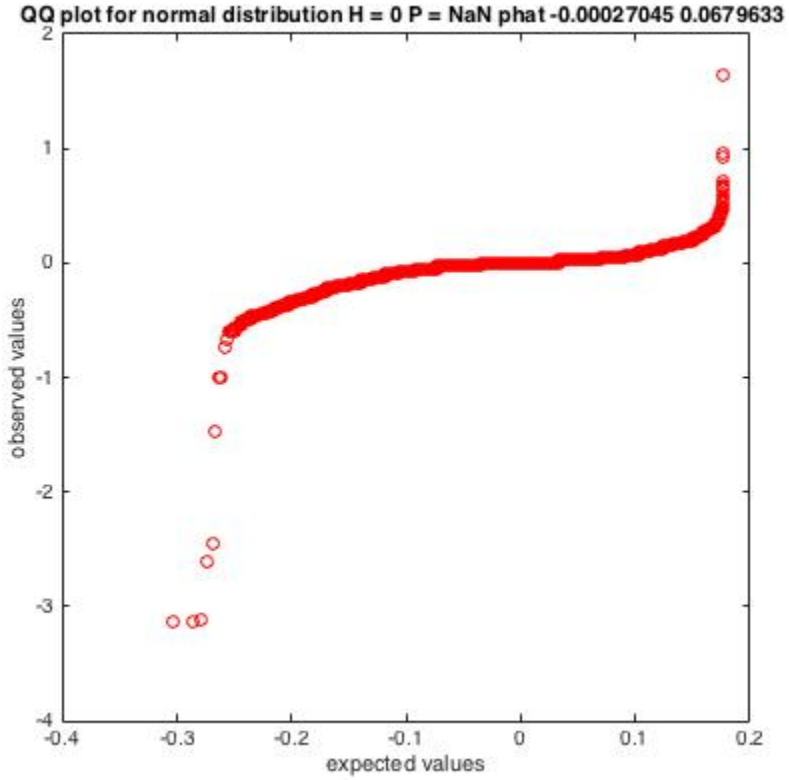
test_quadphase14_04.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:18:52 feigl

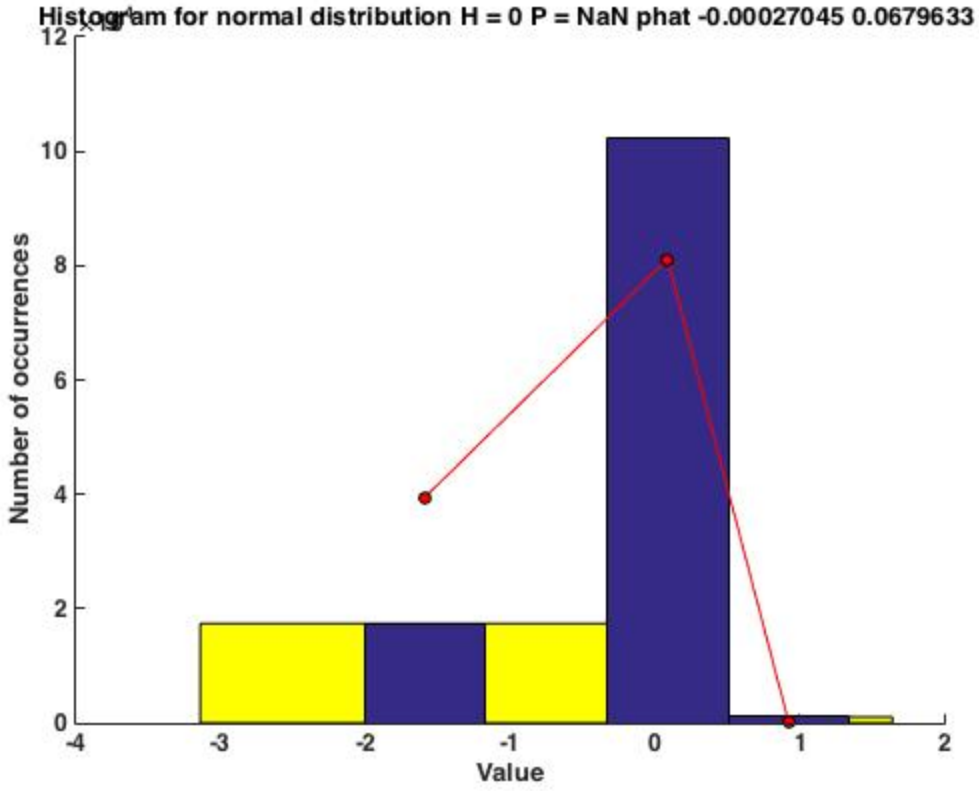


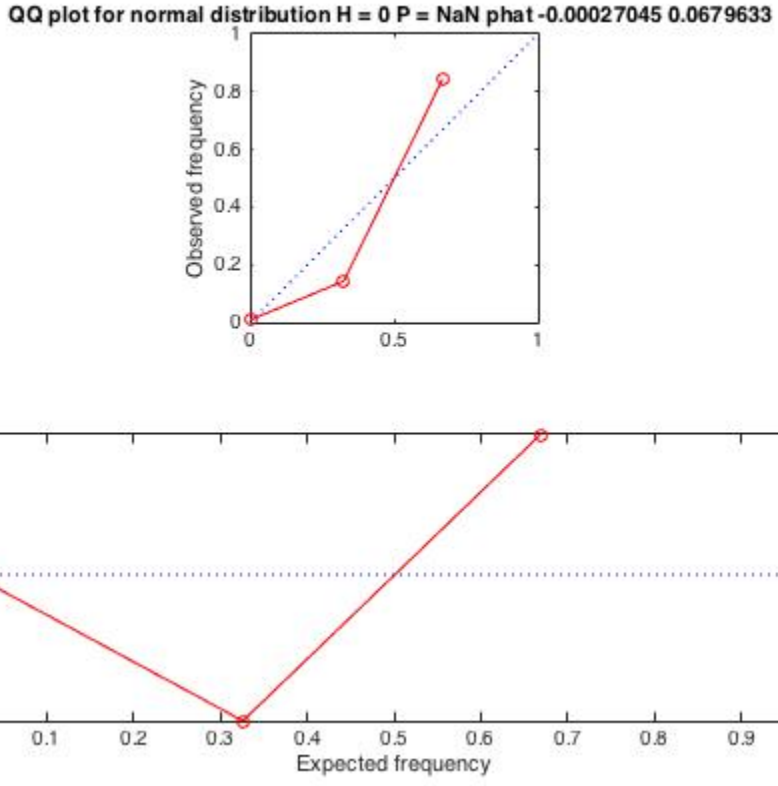
test_quadphase14_05.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:18:53 feigl



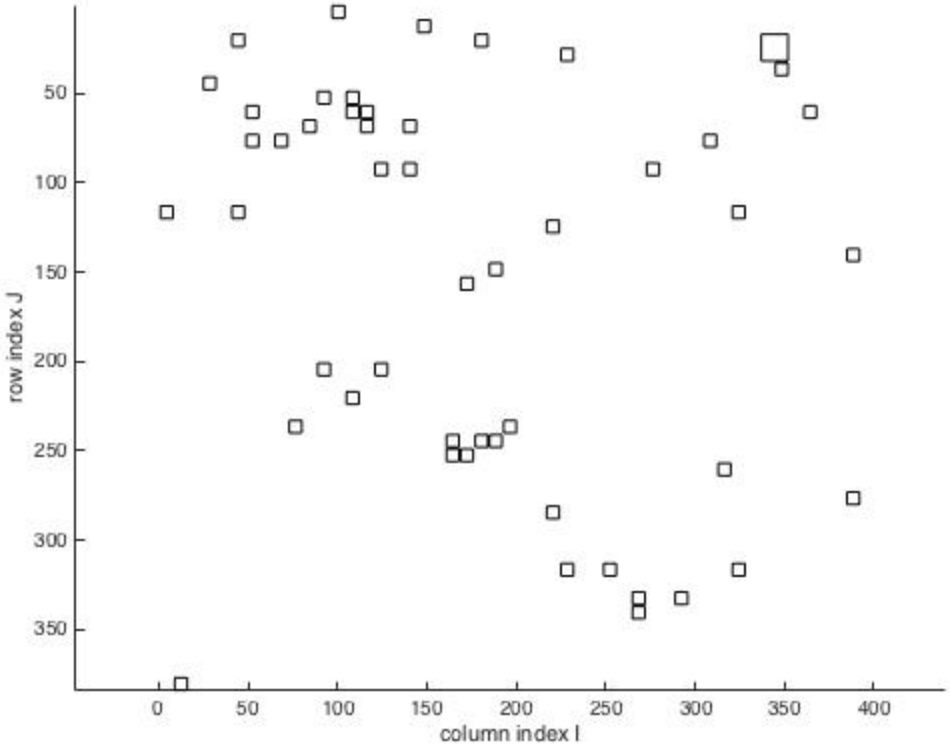
test_quadphase14_06.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:18:54 feigl



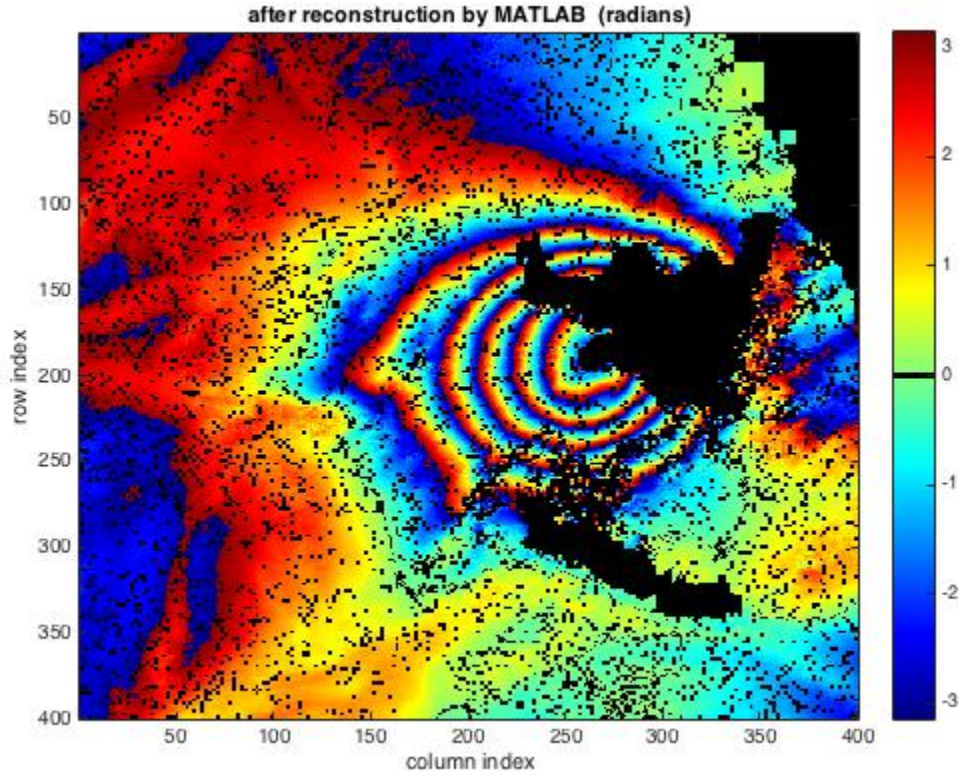




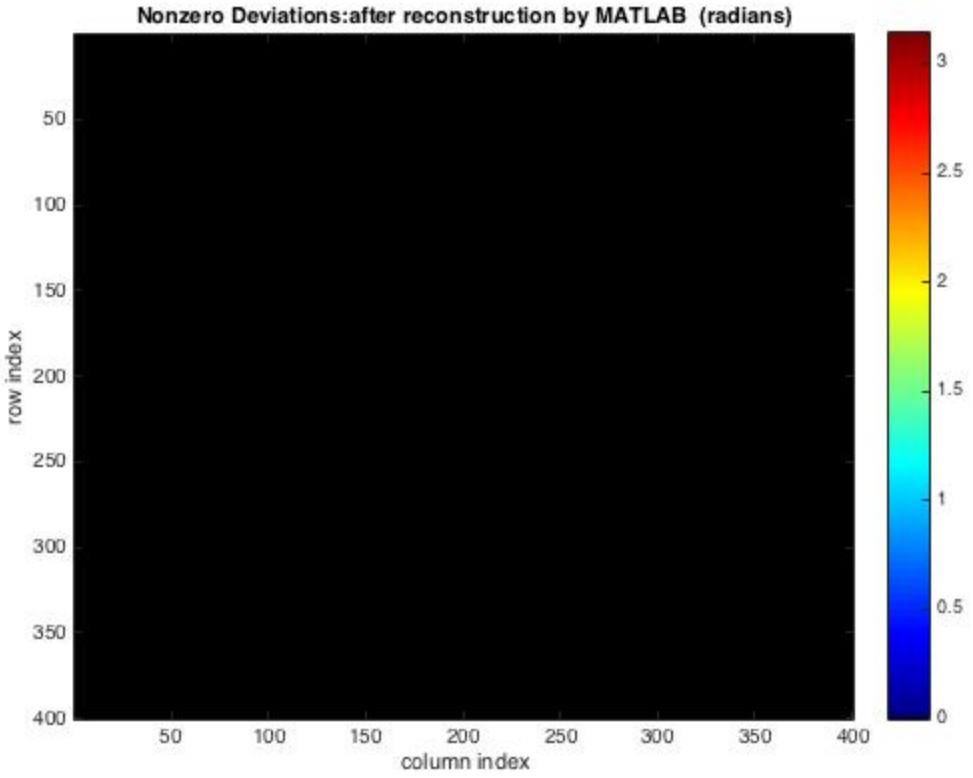
test_quadphase14_09.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:31 feigl



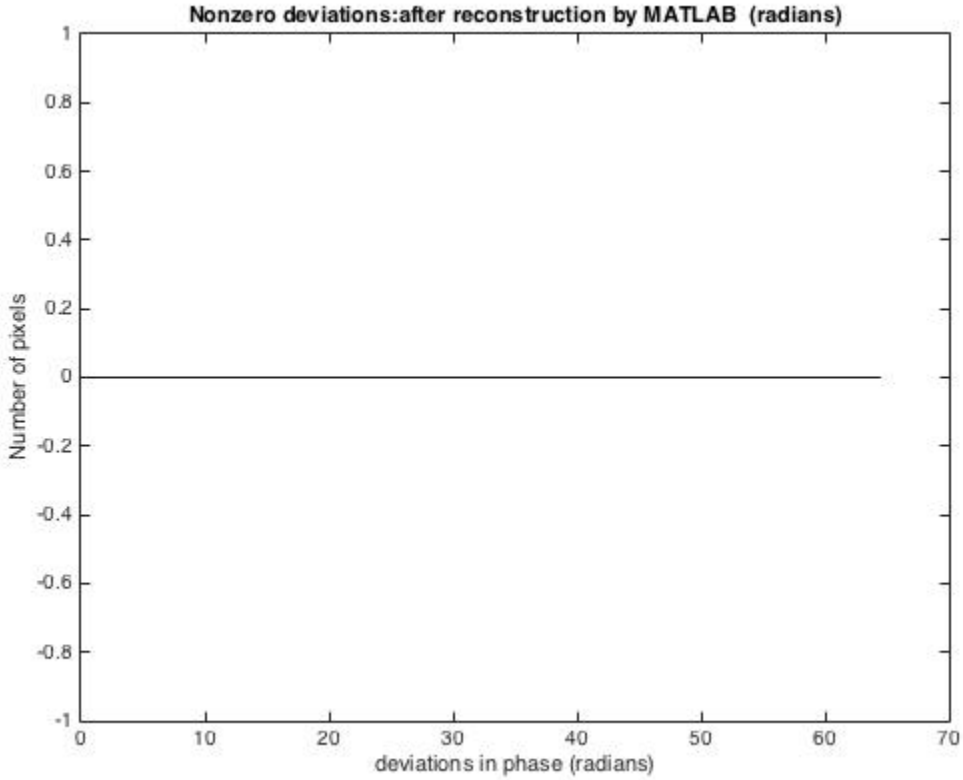
test_quadphase14_10.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:36 feigl



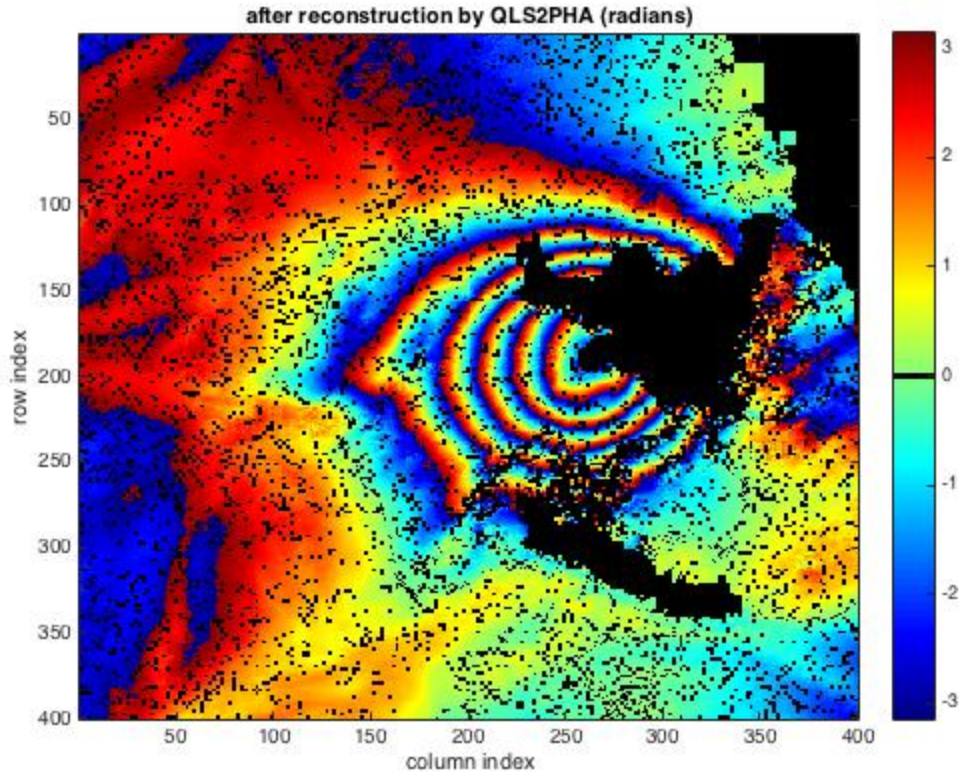
test_quadphase14_11.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:39 feigl



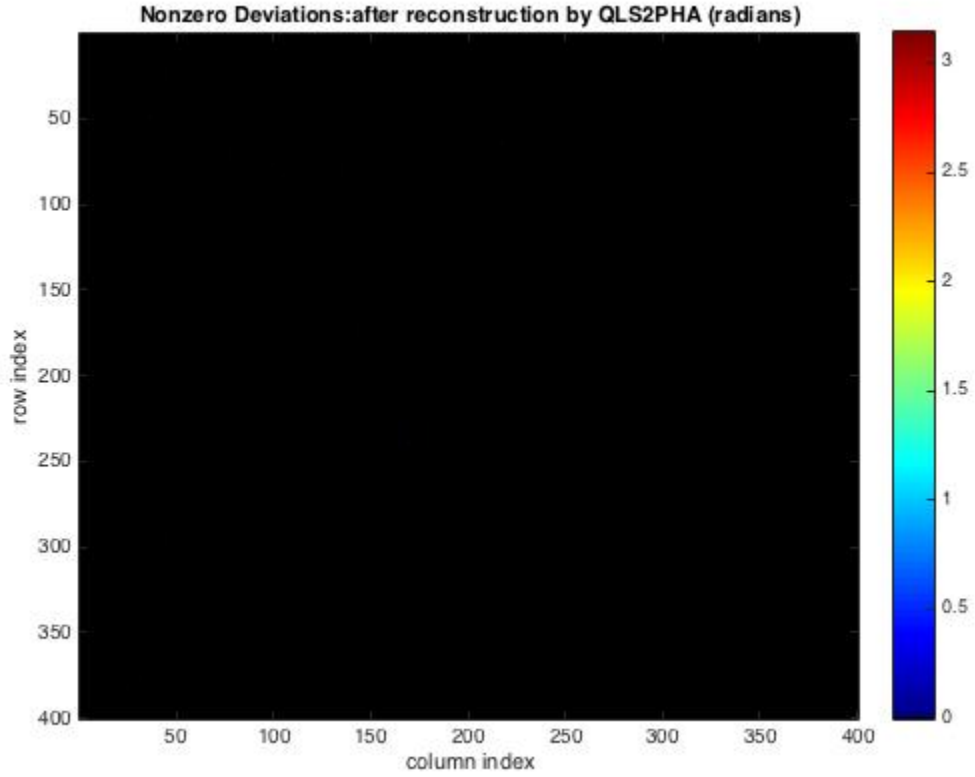
test_quadphase14_12.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:40 feigl



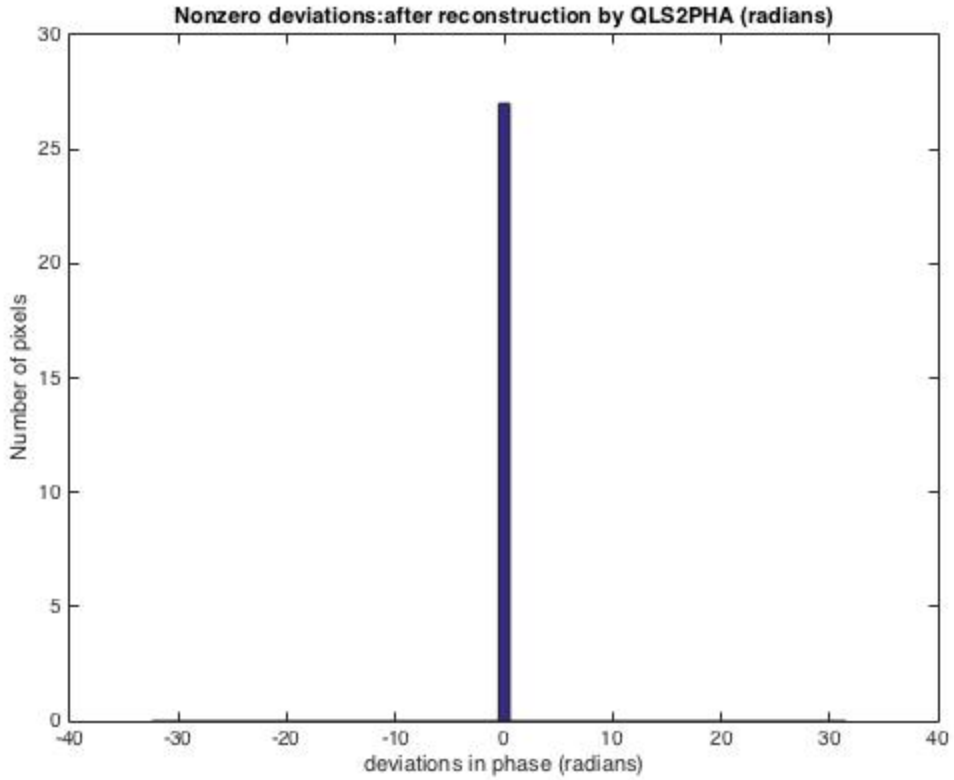
test_quadphase14_13.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:40 feigl



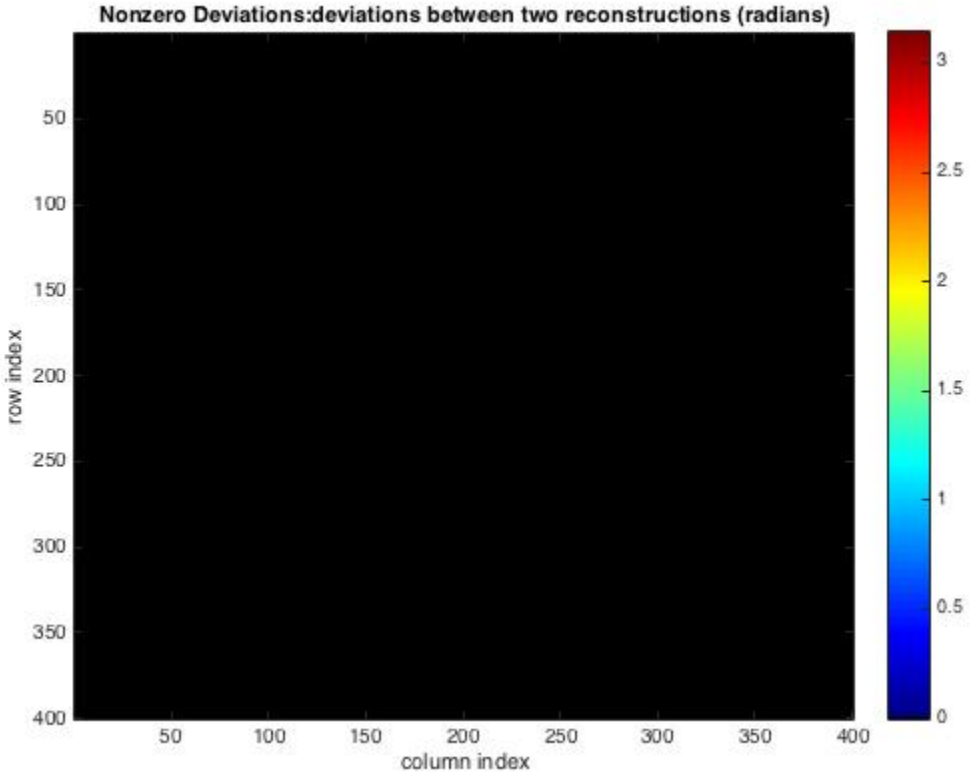
test_quadphase14_14.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:41 feigl



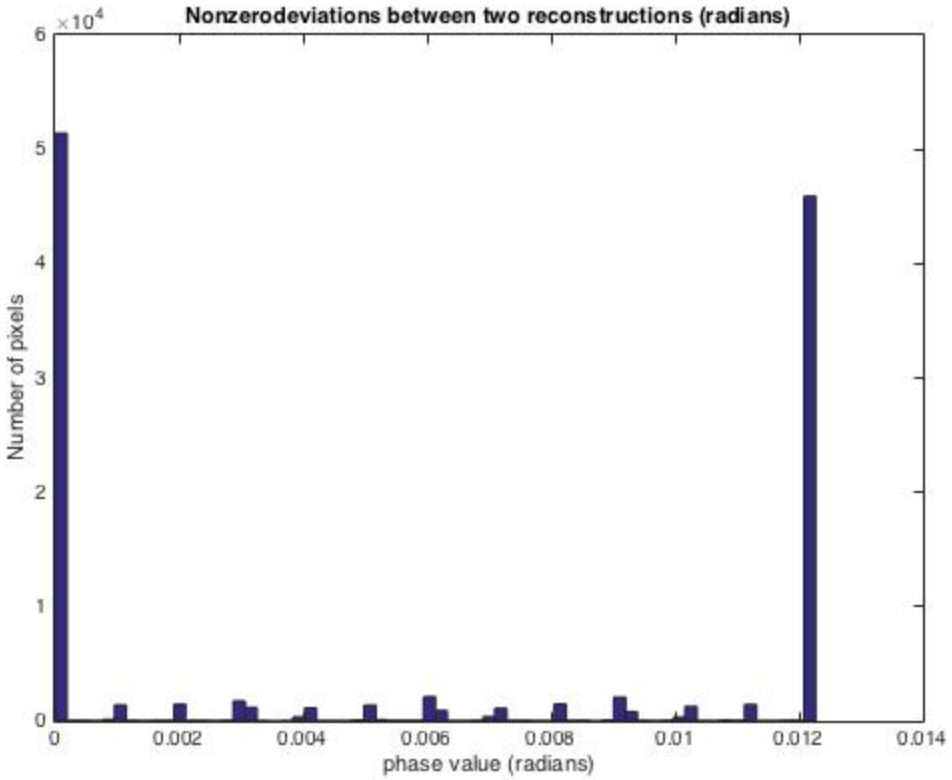
test_quadphase14_15.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:42 feigl



test_quadphase14_16.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:42 feigl



test_quadphase14_17.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:43 feigl



test_quadphase14_18.pdf /Users/feigl/gipht/demoQ 2015-04-20 18:19:44 feigl

Published with MATLAB® R2015a