```
% gda11_05

% backprojection in a a simple tomography example
% The data are "travel times", that is integrals
% of m(x,y) along straight line rays between sources
% and receivers on the edges of a square region.
% Supports Figure 11.7

clear all;

clear G epsilon;
global G epsilon;

figure(1);
clf;
figure('pos',[10, 10, 900, 300]);

% true model m(x,y) is read from a file
Sbig=double(imread('../data/magma.tif','tif'));
Sbig = -(Sbig-128);
[Nybig, Nxbig] = size(Sbig);

% decimate the model
idec=4;
Nx=floor(Nxbig/idec);
Ny=floor(Nybig/idec);
S=Sbig(1:idec:Nybig-1, 1:idec:Nxbig-1);

% independent variable x
xmin = 0;
xmax =  Nx;
dx=(xmax-xmin)/(Nx-1);
x = xmin+dx*[0:Nx-1]';
Dx = xmax-xmin;

% independent variable y
ymin = 0;
ymax = Ny;
dy=(ymax-ymin)/(Ny-1);
y = ymin+dy*[0:Ny-1]';
Dy = ymax-ymin;

% rearrage model into vector
% order of model parameters in vector: index = (ix-1)*Ny + iy
mtrue = zeros(Nx*Ny,1);
rowindex = zeros(Nx*Ny,1);
colindex = zeros(Nx*Ny,1);
for ix = [1:Nx]
for iy = [1:Ny]
    q = (ix-1)*Ny + iy;
    mtrue(q)=S(ix,iy);
    rowindex(q)=ix;
    colindex(q)=iy;
end
end

% plot true model
subplot(1,3,1);
set(gca, 'YDir', 'reverse' );
```

```
set(gca, 'LineWidth', 3 );
set(gca, 'FontSize', 14 );
colormap('jet');
hold on;
axis( [xmin, xmax, ymin, ymax] );
caxis( [-128, 128] );
image( [xmin, xmax], [ymin, ymax], S, 'CDataMapping', 'scaled');
hold on;
xlabel('y');
ylabel('x');
title( 'true model' );

% define sources and receivers on the edges of the box
Nside = 25;
Ns = 4*Nside;
xs = zeros(Ns, 2);
% side 1
xs(1:Nside,1)=xmin+dx/10;
xs(1:Nside,2)=ymin+Dy*[1:Nside]'/(Nside+1);
% side 2
xs(Nside+1:2*Nside,1)=xmax-dx/10;
xs(Nside+1:2*Nside,2)=ymin+Dy*[1:Nside]'/(Nside+1);
% side 3
xs(2*Nside+1:3*Nside,1)=xmin+Dx*[1:Nside]'/(Nside+1);
xs(2*Nside+1:3*Nside,2)=ymin+dy/10;
% side 4
xs(3*Nside+1:4*Nside,1)=xmin+Dx*[1:Nside]'/(Nside+1);
xs(3*Nside+1:4*Nside,2)=ymax-dy/10;

% source receiver pairs
N = Ns*(Ns-1)/2;
ray = zeros(N,4);
k=0;
for i = [1:Ns]
    for j = [i+1:Ns]
        R = sqrt( (xs(i,1)-xs(j,1))^2 + (xs(i,2)-xs(j,2))^2 );
        if (R<2) % exclude really short rays
            continue;
        end
        k = k+1;
        ray(k,1) = xs(i,1);
        ray(k,2) = xs(i,2);
        ray(k,3) = xs(j,1);
        ray(k,4) = xs(j,2);
    end
end
N=k;

% build data kernel
% order of model parameters in vector: index = (ix-1)*Ny + iy
M = Nx*Ny;
G=spalloc(N,M,2*N*Nx);
Nr = 2000;
bins=[1:Nx*Ny];
for k = [1:N]
    x1 = ray(k,1);
    y1 = ray(k,2);
    x2 = ray(k,3);
    y2 = ray(k,4);
    r = sqrt( (x2-x1)^2 + (y2-y1)^2 );
    dr = r/Nr;
```

```
% I use a sloppy way of computing the length of the ray
% in each pixel.  I subdivide the ray into Nr pieces, and
% assign each piece to exactly one pixel, the one its
% closest to

if( 0 ) % slow but sure way
    for i = [1:Nr]
        x = x1 + (x2-x1)*i/Nr;
        y = y1 + (y2-y1)*i/Nr;
        ix = 1+floor( (Nx-1)*(x-xmin)/Dx );
        iy = 1+floor( (Ny-1)*(y-ymin)/Dy );
        q = (ix-1)*Ny + iy;
        G(k,q) = G(k,q) + dr;
    end
else % faster way, or so we hope
    % calculate the array indices of all the ray pieces
    xv = x1 + (x2-x1)*[1:Nr]'/Nr;
    yv = y1 + (y2-y1)*[1:Nr]'/Nr;
    ixv = 1+floor( (Nx-1)*(xv-xmin)/Dx );
    iyv = 1+floor( (Ny-1)*(yv-ymin)/Dy );
    qv = (ixv-1)*Ny + iyv;
    % now count of the ray segments in each pixel of the
    % image, and use the count to increment the appropriate
    % element of G.  The use of the hist() function to do
    % the counting is a bit weird, but it seems to work
    count=hist(qv,bins);
    icount = find( count~=0 );
    G(k,icount) = G(k,icount) + count(icount)*dr;
end
end

% compute true travel times
d = G*mtrue;

% inversion of data by damped least squares
epsilon = 1.0e-2*max(max(abs(G)));
mest1=bicg(@dlsfun,G'*d,1e-5,3*M);
```

```
bicg converged at iteration 135 to a solution with relative residual 9.7e-06.
```

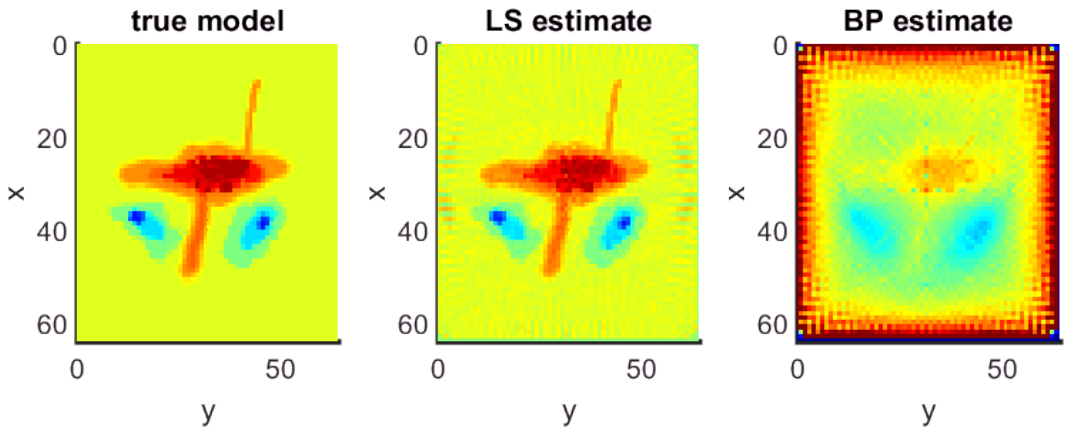
```
% inversion of data by back projection
% normalize each row of the data kernel to unity
% with compensatory normalization of the data
rowsum = sum(G,2);
rowsumr = 1./rowsum;
rowsumr(find(~isfinite(rowsumr)))=0;
dp = d .* rowsumr;
GP = spalloc(N,M,nnz(G));
% can this be done w/o a for loop?
for i=[1:M]
     GP(:,i) = G(:,i) .* rowsumr;
end
mest2=GP'*dp;

% rearrange m back into image
Sest1 = zeros(Nx,Ny);
Sest2 = zeros(Nx,Ny);
for k = [1:M]
    ix=rowindex(k);
```

```
    iy=colindex(k);
    Sest1(ix,iy)=mest1(k);
    Sest2(ix,iy)=mest2(k);
end

% plot estimated image (damped least squares)
subplot(1,3,2);
set(gca, 'YDir', 'reverse' );
hold on;
set(gca, 'LineWidth', 3 );
set(gca, 'FontSize', 14 );
colormap('jet');
axis( [xmin, xmax, ymin, ymax] );
caxis( [-128, 128] );
image( [xmin, xmax], [ymin, ymax], Sest1, 'CDataMapping', 'scaled');
xlabel('y');
ylabel('x');
title( 'LS estimate' );

% plot estimated image (back projection)
subplot(1,3,3);
set(gca, 'YDir', 'reverse' );
hold on;
set(gca, 'LineWidth', 3 );
set(gca, 'FontSize', 14 );
colormap('jet');
axis( [xmin, xmax, ymin, ymax] );
zmin = min(min(Sest2));
zmax = max(max(Sest2));
DZ = zmax-zmin;
caxis( [1.2*zmin, 0.18*zmax] ); % color scale fudged here
image( [xmin, xmax], [ymin, ymax], Sest2, 'CDataMapping', 'scaled');
xlabel('y');
ylabel('x');
title( 'BP estimate' );
```




```
% Figure 11.7 Example of backprojection. (A) True two-dimensional model, for which traveltimes
% associated with a dense and well-distributed set of rays are measured. (B) Estimated model,
% using damped least squares. (C) Estimated model, using backprojection. MatLab script gda11_0
```