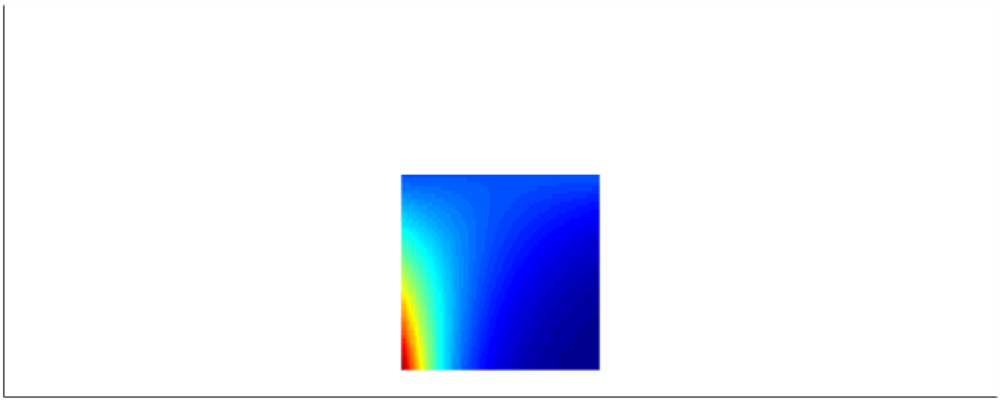
```
% gda06_03
%
% Squeezing example
% The data kernel is a series of decaying exponentials in
% an auxillary variable z and is underdetermined.  Three
% models, all that fit the data to similar accuracy, are
% constructed:  Simple damped least squares; solution
% squeezed shallow ("up"), solution squeezed deep ("dn")

% Note: Becaue the data are noisy with noise that randomly
% varies between runs, you may have to run the scipt a few
% times until the least squares solution (black curve in
% figure 2 (top) is a good approximation to the true
% model (blue curve).

clear all;

% auxilliary variable, z
M=101;
zmin=0;
zmax=10;
Dz=(zmax-zmin)/(M-1);
z=zmin+Dz*[0:M-1]';

% true solution is a boxcar
mtrue = zeros(M,1);
i1 = floor( 5*M/11)+1;
i2 = floor( 7*M/11)+1;
ic = floor(0.5*(i1+i2));
zc = z(ic);
mtrue(i1:i2) = 1.0;

% data kernel is declining exponentials, normalized
% so that they compute a weighted average of the model
N=floor(M/2);
G = zeros(N,M);
c0=0.01;
for i = [1:N];
    c = c0*(i-1);
    v = exp(-c*z);
    vn = sum(v);
    G(i,1:M)=v'/vn;
end

% draw the data kernel
gda_draw(G);
```

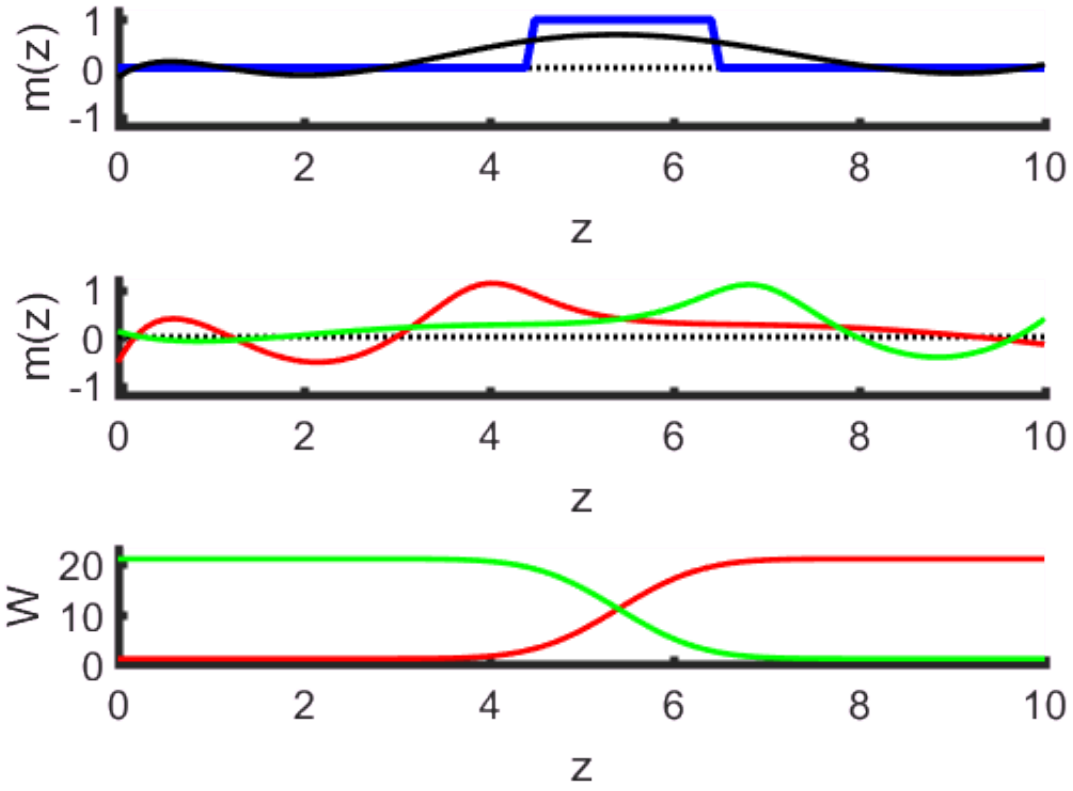


```
% Figure 6.3 Squeezed solutions for a simple inverse problem for model parameters m(z_i), wher
% (A) The data kernel represents smooth averages of the data and has the form G_ij=c_j  exp#(-
% where the c_js are constants.

% create synthetic data by adding random noise to the
% true data
sd=0.0001;
dtrue = G*mtrue;
dobs = dtrue + random('Normal',0,sd,N,1);

% damped least squares
epsilon2 = 1e-9;
W = eye(M,M);
mest0 = (G'*G + epsilon2*W)\(G'*dobs);
e = dobs - G*mest0;
E0 = (e'*e); % error
Ed = (dobs'*dobs); % energy in data, for comparison

% squeeze up; the weights are low to the left
% of z=zc and high to the right of it. The error
% function erf(z) ramps up smoothly from -1 for
% z<<0 to +1 for z>>0, so I can use a scaled
% and shifted version of it to achieve weights
% that smoothly ramp up/down from one constant
% value to another at the point z0.  (Any
% "sigmoidal" shaped function would have sufficed;
% Matlab's sigmf() would be fine, too.

epsilon2 = 5e-9;
n=2;
A=1;
B=10;
Wup = diag((B+1)+B*erf((z-zc)/A)); % weights grow with z
mest_up = (G'*G + epsilon2*Wup)\(G'*dobs);
e = dobs - G*mest_up;
Eup = e'*e;

% squeeze down
epsilon2 = 5e-9;
Wdn = diag((B+1)-B*erf((z-zc)/A)); % weights decline with z
mest_dn = (G'*G + epsilon2*Wdn)\(G'*dobs);
e = dobs - G*mest_dn;
Edn = e'*e;
```

```
% I have adusted all the coefficients by trial and error
% so that E0/Ed is about 10-6 (a very good fit)
% Eup/E0 and Edn/E0 are both about 1.05 (almost as good)
fprintf('Normalized error %.2e ratio: up %.3f dn %.3f\n', E0/Ed, Eup/E0, Edn/E0 );
```

```
Normalized error 4.24e-07 ratio: up 1.081 dn 1.077
```

```
% plot true and unsqueezed models
figure(2);
clf;
subplot(3,1,1);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [zmin, zmax, -1.2, 1.2] );
plot( [zmin, zmax]', [0, 0]', 'k:', 'LineWidth', 2 );
plot( z, mtrue, 'b-', 'LineWidth', 3 );
plot( z, mest0, 'k-', 'LineWidth', 2 );
xlabel('z');
ylabel('m(z)');

% plot unsqueezed and squeezed models
subplot(3,1,2);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [zmin, zmax, -1.2, 1.2] );
plot( [zmin, zmax]', [0, 0]', 'k:', 'LineWidth', 2 );
plot( z, mest_up, 'r-', 'LineWidth', 2 );
plot( z, mest_dn, 'g-', 'LineWidth', 2 );
xlabel('z');
ylabel('m(z)');

% plot weights
subplot(3,1,3);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
w = diag(Wup);
axis( [zmin, zmax, 0, 1.1*max(w)] );
plot( z, w, 'r-', 'LineWidth', 2 );
w = diag(Wdn);
plot( z, w, 'g-', 'LineWidth', 2 );
xlabel('z');
ylabel('W');
```




```
% Figure 6.3 Squeezed solutions for a simple inverse problem for model parameters m(z_i), wher
% (B) The true model (blue curve) is a boxcar centered at z_0=5.4.  The ordinary damped
% least squares solution (black curve) is smoother that the boxcar, but has negligible error.
% The shallow solution (red curve) and deep solution (green curve) also have negligible error,
% at shallow and deep depths, respectively, than the ordinary damped least squares solution. T
% indicate that the data are sufficient to constrain the central part of the solution to the 4
% (D) The two weight functions (green and red) preferentially penalize shallow and deep struct
```