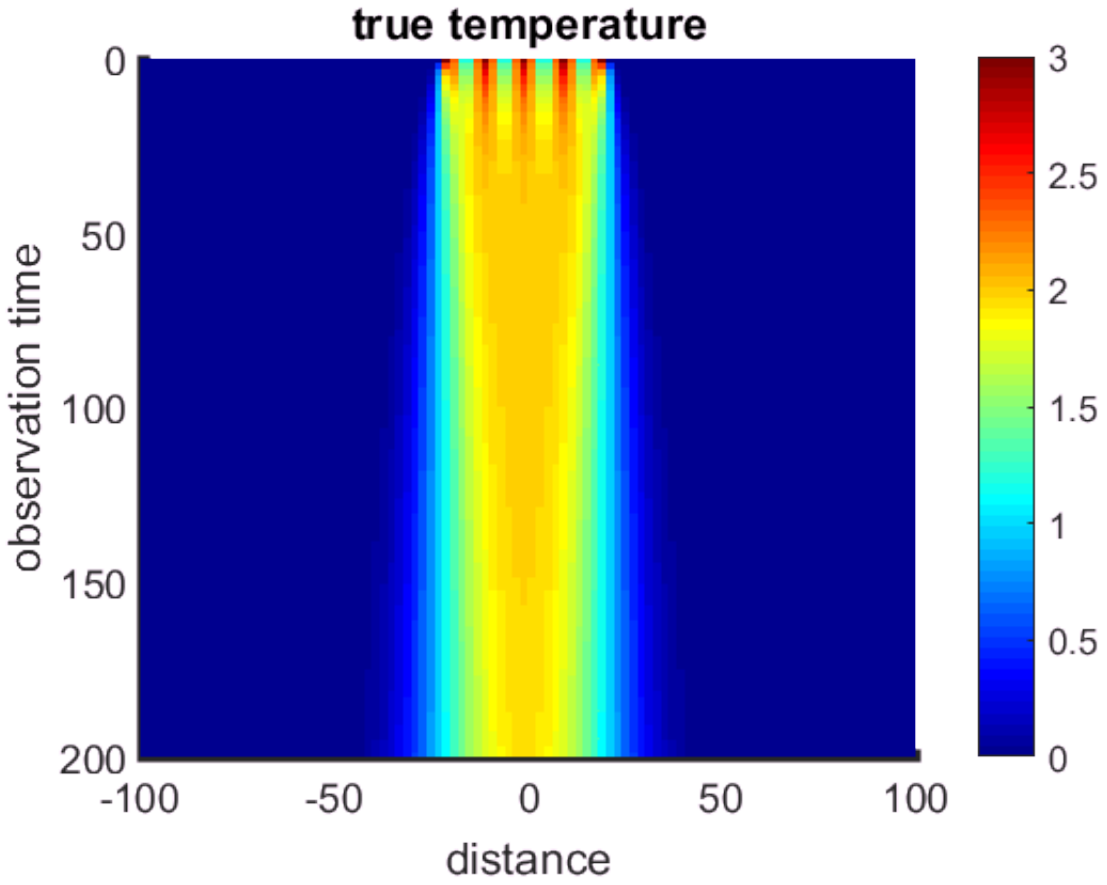
```
% gda12_06
% Supports Figures 12.8, 12.9 and 12.10
% temperature inversion problem
%
% The data are a temperature profile made at
% a time tobs>=0.  The model parameters are the
% initial temperature profile (at time t0=0).
% The problem is solved many times, for
% different tobs's, so that one can examine
% how the ability to reconstruct the initial
% temperature degrades as the data are measured
% later and later in time.

% Thus, time in the plot is the time of observation,
% tobs. In all cases, the quantity being determined is
% the inital temperature distributon at the inital time,
% t0=0.

clear all;

% independent spartial variable x
Nx = 100;
xmin = -100;
xmax = 100;
Dx = (xmax-xmin)/(Nx-1);
x = xmin + Dx*[0:Nx-1]';

% independent temporal variable t
Nt = 100;
tmin = 0;
tmax = 200;
Dt = (tmax-tmin)/(Nt-1);
t = tmin + Dt*[0:Nt-1]';

% time/space matrices
tt = t*ones(1,Nx);
xx = ones(Nt,1)*x';

% model parameters, one at each distance
s = 1.0;
mtrue = zeros(Nx,1);
L=5;
mtrue(4*Nx/10:6*Nx/10)=(2+cos( 2*pi*([4*Nx/10:6*Nx/10]'-Nx/2)/L ));

% the true data as a function of observation time and distance
% are constructed by evaluating an analytic solution
% to a heat flow equation
dtxtrue = zeros(Nt,Nx);
for i=[1:Nx]
    % data kernel, Gij, gives temperature at xi due to source
    % at xj for a time, t0
    t0=t(i);
    if( t0==0 )
        G=eye(Nx,Nx);
    else
        erfA = erf( (x*ones(1,Nx)-ones(Nx,1)*(x-Dx/2)') / sqrt(t0) );
        erfB = erf( (x*ones(1,Nx)-ones(Nx,1)*(x+Dx/2)') / sqrt(t0) );
        G = 0.5*(erfA-erfB);
    end
```

```
    dtxtrue(i,:) = (G*mtrue)';
end

% plot the true data
figure(1);
clf;
colormap('jet');
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, tmin, tmax] );
axis ij,
imagesc( [xmin, xmax], [tmin, tmax], dtxtrue);
ylabel('observation time');
xlabel('distance');
title('true temperature');
colorbar();
```

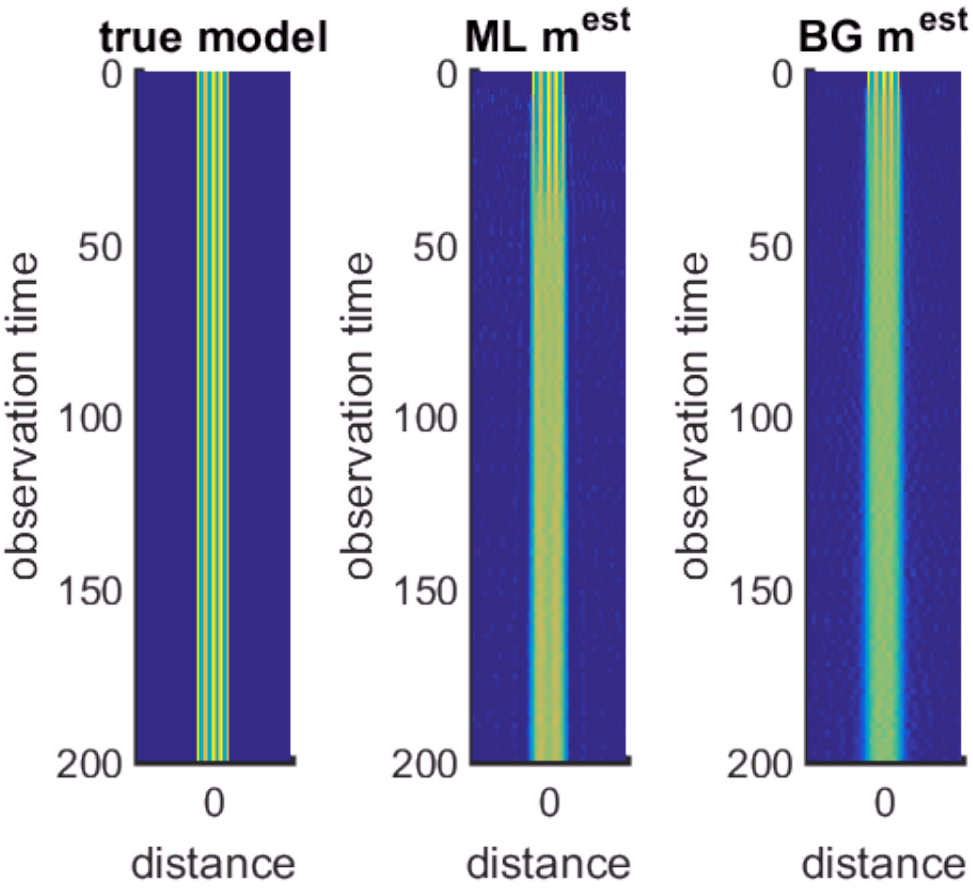



```
% Figure 12.8 (B) Temporal evolution of the temperature T(x, t) of 100 adjacent slabs. The
% initial temperature distribution of the slabs, T(x, t = 0), is taken to be the model
% parameter vector, m. It is nonzero only for slabs near |x| ≤ 20. The temperature, T(x, t = 0
% at subsequent times, t, can be computed from the initial temperature distribution, since the
% data kernel can be calculated from the physics of heat transport. Note that the band of hot
% temperatures widens with increasing time, and that fine scale temperature fluctuations are
% preferentially attenuated. MatLab script gda12_06.

% the true data as a function of time and distance
dtx = zeros(Nt,Nx);
for i=[1:Nx]
    % data kernel, Gij, gives temperature at xi due to source
```

```
    % at xj for a time, t0
    t0=t(i);
    if( t0==0 )
        G=eye(Nx,Nx);
    else
        erfA = erf( (x*ones(1,Nx)-ones(Nx,1)*(x-Dx/2)') / sqrt(t0) );
        erfB = erf( (x*ones(1,Nx)-ones(Nx,1)*(x+Dx/2)') / sqrt(t0) );
        G = 0.5*(erfA-erfB);
    end
    dtxtrue(i,:) = (G*mtrue)';
end

% true model, replicated at all times
mtxtrue = ones(Nt,1)*mtrue';

% plot the true model
figure(2);
clf;

subplot(1,3,1);
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, tmin, tmax] );
axis ij,
c1 = min(min(mtxtrue));
c2 = max(max(mtxtrue));
caxis( [c1, c2] );
image( [xmin, xmax], [tmin, tmax], mtxtrue, 'CDataMapping', 'scaled');
ylabel('observation time');
xlabel('distance');
title('true model');

R1at = 10;
R2at = 40;

% damped minimum length case
% estimate the model using the data each time
mtxest = zeros(Nt,Nx);
for i=[1:Nt]
    % true data
    dtrue=dtxtrue(i,:)';

    % add noise
    sd = 0.01;
    dobs = dtrue + random('Normal',0,sd,Nx,1);

    % compute data kernel
    t0=t(i);
    if( t0==0 )
        G=eye(Nx,Nx);
    else
        erfA = erf( (x*ones(1,Nx)-ones(Nx,1)*(x-Dx/2)') / sqrt(t0) );
        erfB = erf( (x*ones(1,Nx)-ones(Nx,1)*(x+Dx/2)') / sqrt(t0) );
        G = 0.5*(erfA-erfB);
    end

    % damped minimum length solution
    epsilon=1.0e-3;
    mtxest(i,:) = (G'*((G*G'+epsilon*eye(Nx,Nx))\dobs))';
```

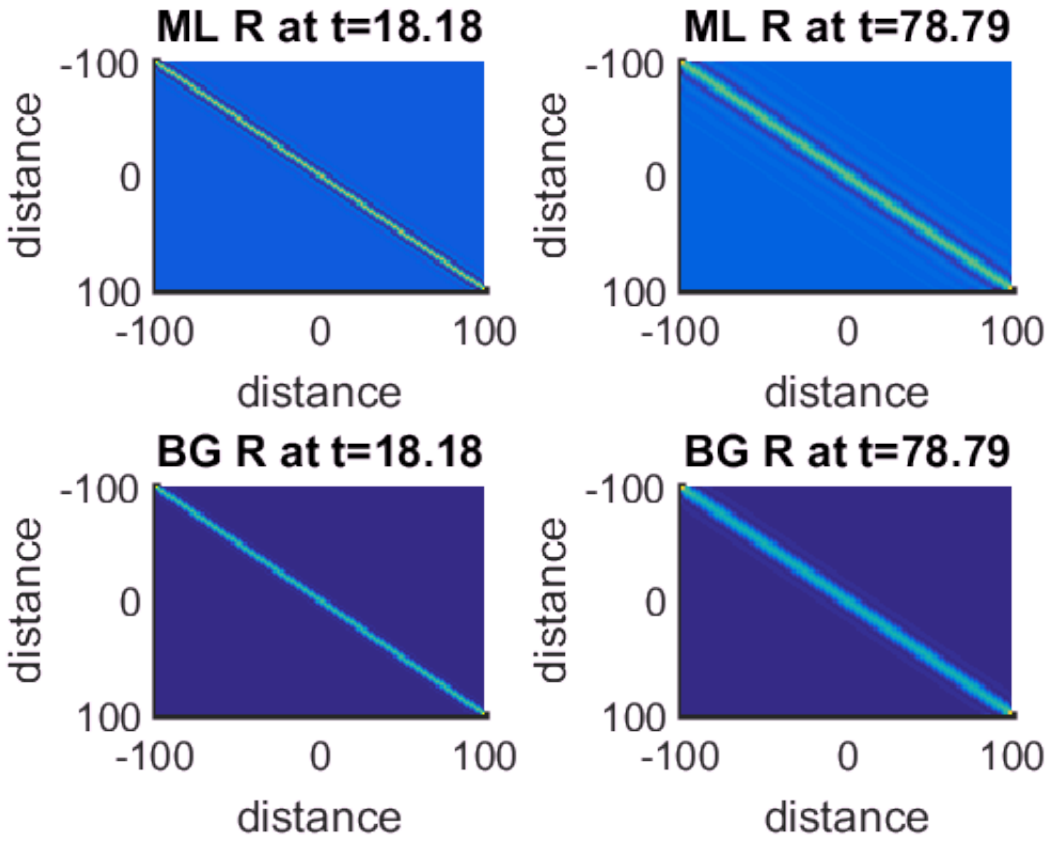
```
    % save resolving kernels
    if( i==R1at )
        R1ML = G'*inv((G*G'+epsilon*eye(Nx,Nx)))*G;
    elseif (i==R2at)
        R2ML = G'*inv((G*G'+epsilon*eye(Nx,Nx)))*G;
    end
end

% plot ML estimated solution
subplot(1,3,2);
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, tmin, tmax] );
axis ij,
caxis( [c1, c2] );
image( [xmin, xmax], [tmin, tmax], mtxest, 'CDataMapping', 'scaled');
ylabel('observation time');
xlabel('distance');
title('ML m^{est}');

% Backus-Gilbert case
% estimate the model using the data each time
mtxest = zeros(Nt,Nx);
for i=[1:Nt]
    % true data
    dtrue=dtxtrue(i,:)';

    % add noise
    sd = 0.01;
    dobs = dtrue + random('Normal',0,sd,Nx,1);

    % compute data kernel
    t0=t(i);
    if( t0==0 )
        G=eye(Nx,Nx);
    else
        erfA = erf( (x*ones(1,Nx)-ones(Nx,1)*(x-Dx/2)') / sqrt(t0) );
        erfB = erf( (x*ones(1,Nx)-ones(Nx,1)*(x+Dx/2)') / sqrt(t0) );
        G = 0.5*(erfA-erfB);
    end

    % construct Backus-Gilbert generalized inverse
    GMG = zeros(Nx,Nx);
    u = G*ones(Nx,1);
    for k = [1:Nx]
        S = G * diag(([1:Nx]-k).^2) * G';
        epsilon=1e-3;
        S = S+epsilon*eye(Nx);
        uSinv = u'/S;
        GMG(k,:) = uSinv / (uSinv*u);
    end

    % estimate solution
    mtxest(i,:) = (GMG*dobs)';

    % save resolving kernels
    if( i==R1at )
        R1BG = GMG*G;
    elseif (i==R2at)
        R2BG = GMG*G;
```

```
    end

end

% plot BG estimated solution
subplot(1,3,3);
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, tmin, tmax] );
axis ij,
caxis( [c1, c2] );
image( [xmin, xmax], [tmin, tmax], mtxest, 'CDataMapping', 'scaled');
ylabel('observation time');
xlabel('distance');
title('BG m^{est}');
```




```
% Figure 12.9 Model for the temperature distribution problem. (A) The true model is the initia
% temperature distribution, T(x, t = 0). While this function is not a function of time, it is
% displayed on the (x, t) image for comparison purposes. (B) Minimum length (ML) estimate of
% the model, T(x, t = 0), for a data set consisting of observations at all distances at a sing
% time t > 0. (C) Corresponding Backus-Gilbert (BG) estimate. In both the ML and BG cases, the
% ability of the data to resolve fine details declines with time, with the BG case declining
% fastest. MatLab script gda12_06.

% plot resolution kernels
figure(3);
clf;

% plot ML resolution at time t1
subplot(2,2,1);
```

```
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, xmin, xmax] );
axis ij,
imagesc( [xmin, xmax], [xmin, xmax], R1ML);
ylabel('distance');
xlabel('distance');
title(sprintf('ML R at t=%.2f',t(R1at)));

% plot ML resolution at time t1
subplot(2,2,2);
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, xmin, xmax] );
axis ij,
imagesc( [xmin, xmax], [xmin, xmax], R2ML);
ylabel('distance');
xlabel('distance');
title(sprintf('ML R at t=%.2f',t(R2at)));

% plot BG resolution at time t1
subplot(2,2,3);
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, xmin, xmax] );
axis ij,
imagesc( [xmin, xmax], [xmin, xmax], R1BG);
ylabel('distance');
xlabel('distance');
title(sprintf('BG R at t=%.2f',t(R1at)));

% plot BG resolution at time t2
subplot(2,2,4);
set( gca, 'LineWidth', 3);
set( gca, 'FontSize', 14);
hold on;
axis( [xmin, xmax, xmin, xmax] );
axis ij,
imagesc( [xmin, xmax], [xmin, xmax], R2BG);
ylabel('distance');
xlabel('distance');
title(sprintf('BG R at t=%.2f',t(R2at)));
```




```
% Figure 12.10 Model resolution matrices for the temperature distribution problem. (A) Minimum
% length (ML) solution for data at time t = 10. (B) ML solution for data at time t = 40. (C)
% Backus-Gilbert (BG) solution for data at time t = 10. (D) BG solution for data at time t = 4
% Note that the BG resolution matrix has smaller sidelobes than the corresponding ML case. Mat
% script gda12_06.
```