Covariance of Layered Models
By Bill Menke, November 7, 2018

Summary: I develop a method for converting a layered half-space model described by layer property and interface positions (which together constitute the model parameters **m**) into one where the layer property is sampled at equal increments of depth (the depth-series **Y**). The procedure has a variety of applications, including propagating covariance from **m** to **Y**.

The model is a stratified half-space with a top surface at $x = 0$ and extending to $x \rightarrow +\infty$. The layer property (say, resistivity) of each layer is given by r_i and the bottom of each layer is at depth $x = h_i$ (except that the last layer is the half-space and has no bottom). Thus, the model is defined by $2N - 1$ model parameters $\mathbf{m} = [r_1, r_2, \cdots, r_N, h_1, h_2, \cdots, h_{N-1}]^{\mathrm{T}}$. We assume that an estimate of **m** and its covariance $\mathbf{C}_m$ are available.

A layered model can be thought of being composed of a sum of step functions of the form $rH(x-h)$ and $rH(h-x)$ where $H(.)$ is the Heaviside step function, r is an amplitude and h is an offset. We construct a smoother version of the model by:

$$\text{replacing } rH(x-h) \text{ with } \tfrac{1}{2}r[1 + \text{erf}\{a(x-h)\}]$$

and

$$\text{replacing } rH(h-x) \text{ with } \tfrac{1}{2}r[1 - \text{erf}\{a(h-x)\}]$$

Her erf(.) is the error function. The parameter a controls the smoothness of the transition between layers. The smooth model with has continuous Fréchet derivatives:

$$\frac{d}{dr}\tfrac{1}{2}r[1 + \text{erf}\{a(x-h)\}] = \tfrac{1}{2}[1 + \text{erf}\{a(x-h)\}]$$

$$\frac{d}{dh}\tfrac{1}{2}r[1 + \text{erf}\{a(x-h)\}] = -\frac{2ar}{\sqrt{\pi}}\exp(-z^2)$$

$$\frac{d}{dr}\tfrac{1}{2}r[1 - \text{erf}\{a(h-x)\}] = \tfrac{1}{2}[1 - \text{erf}\{a(h-x)\}]$$

$$\frac{d}{dh}\tfrac{1}{2}r[1 - \text{erf}\{a(h-x)\}] = \frac{2ar}{\sqrt{\pi}}\exp(-z^2)$$

The attached code converts a layered model specified by **m** into a depth-series $Y_n = Y(x_n)$ with equally spaced values $x_n = (n-1)\Delta x$. An exemplary model:

$$N = 4 \quad \text{and} \quad \mathbf{r} = [2.0, 3.5, 2.5, 1.0]^{\mathrm{T}} \quad \text{and} \quad \mathrm{h} = [10, 25, 30]^{\mathrm{T}}$$

is shown in Figure 1. The code also computes the derivatives dY/dr_i and dY/dh_i (the correctness of which I have verified by comparison to a first-difference calculation (Figure 2 and 3).

Taylor's Theorem can be used to linearize $\mathbf{Y}(\mathbf{m})$ about a reference value $\mathbf{m}_0$:

$$\mathbf{Y} \approx \mathbf{Y}|_{\mathbf{m}_0} + \mathbf{M}\Delta\mathbf{m} \quad \text{with} \quad \Delta\mathbf{m} = \mathbf{m} - \mathbf{m}_0 \quad \text{and} \quad \mathbf{M} = \left.\frac{\partial \mathbf{Y}}{\partial \mathbf{m}}\right|_{\mathbf{m}_0}$$

and to perform approximate error propagation:

$$\mathbf{C}_Y = \mathbf{M}\mathbf{C}_m\mathbf{M}^{\mathrm{T}}$$

The matrix **M,** which is constructed by the code, has the form:

$$\mathbf{M} = \left[\frac{d\mathbf{Y}}{dr_1}, \cdots, \frac{d\mathbf{Y}}{dr_N}, \frac{d\mathbf{Y}}{dh_1}, \cdots, \frac{d\mathbf{Y}}{dh_N}\right]_{\mathbf{m}_0}$$

An example with:

$$\mathbf{C}_m \equiv \begin{bmatrix} \mathbf{C}_r & \mathbf{C}_{rh} \\ \mathbf{C}_{rh} & \mathbf{C}_h \end{bmatrix} \quad \text{with} \quad \mathbf{C}_r = \sigma_r^2\mathbf{I} \quad \text{and} \quad \mathbf{C}_h = \sigma_h^2\mathbf{I} \quad \text{and} \quad \mathbf{C}_{rh} = 0$$

is shown in Figure 4.

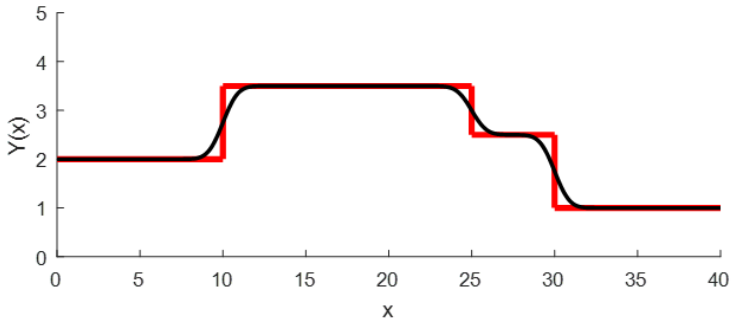

Figure 1. Approximation of a layered model (red curve) with a smooth function based on error functions (black curve).

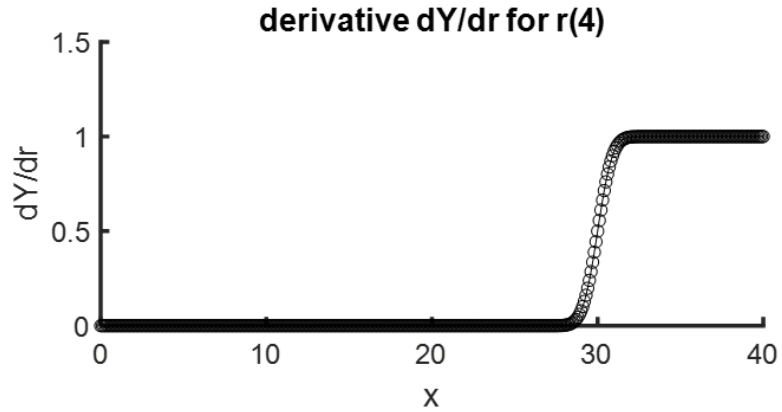

Figure 2. Analytic (solid curve) and numerical (circles) estimates of the derivative dY/dr_4.

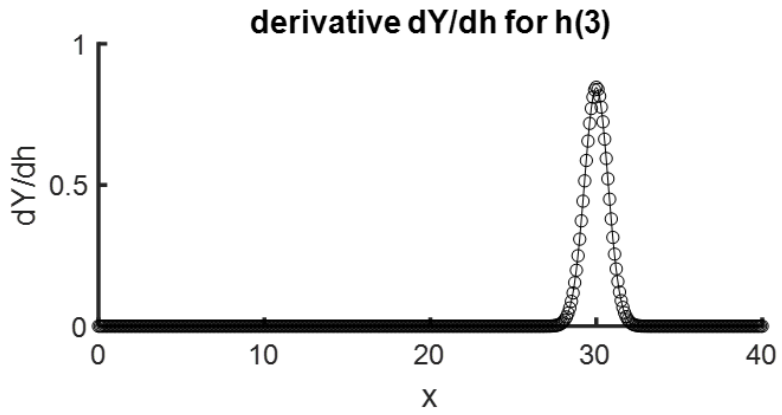

Figure 3. Analytic (solid curve) and numerical (circles) estimates of the derivative dY/dh_3.

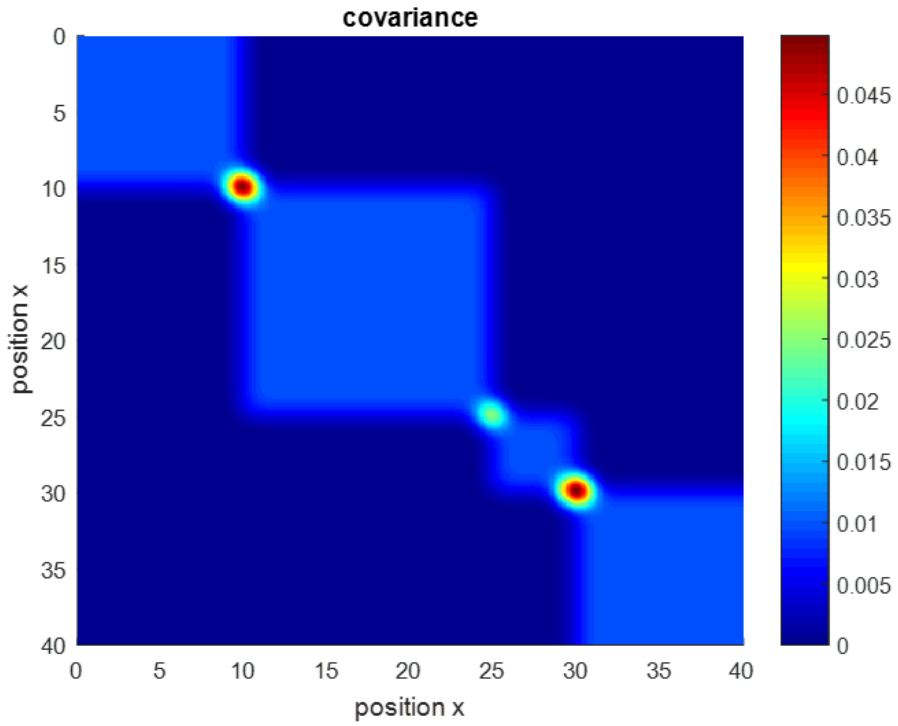

Figure 4. Covariance $\mathbf{C}_Y$ where the vector $Y_i = Y(x_i)$ is for points equally-spaced along the x-axis. The rectangles result from the propagation of the uncertainty in the $r's$, the bullseyes from the h's.

```
clear all;

% steepness of steps in the smoothed layer model
a = 1;

% layered model
N = 4;
h = [10, 25, 30, 10000]'; % position of layer interfaces
% for search reasons, h(N) must be set to a large number
% even those the exact value doesn't matter.
r = [2, 3.5, 2.5, 1]'; % resistivity of layers

% depths at which resistivity is evaluated
Nx = 401;
x = 0.1*[0:Nx-1]';

% the  layered model contains step functions H(x-h).  I
% smooth the layered model by replacing each step function
% H(x-h) by the function 0.5*(1+erf(a*(x-h))), where
% the parameter a controls the sharpness of the steps.

% a note on talking derivates of an error function
% derfz/dz = (2/sqrt(pi))*exp(-z^2)
% z = a*(h-x)=ah-ax   so  dzdh = a
% derfz/dh = derfz/dz dz/dh = a*(2/sqrt(pi))*exp(-z^2)
% z = a*(x-h)=ax-ah   so  dzdh = -a
% derfz/dh = derfz/dz dz/dh = -a*(2/sqrt(pi))*exp(-z^2)

% So, suppose that one starts with model parameters m=(r;h)
% with estimated values (r0, h0) and forms a new set of model
% parameters Y that correponds to resistivity evaluated at
% Nx equally spaced values of x, using the smoothed step
% function method.  The following loop calculates the function
% Y(x,r,h), as well as the derivatives dY/dr and dY/dh

y = zeros(Nx,1);
Y = zeros(Nx,1);
dydr = zeros(Nx,1);
dYdr = zeros(Nx,N);
dydh = zeros(Nx,1);
dYdh = zeros(Nx,N);
c = (2/sqrt(pi));
for i=[1:Nx]
    z = a*(h(1)-x(i));
    y(i) = r(1)*0.5*(1+erf(z));
    Y(i) = Y(i)+y(i);
    dydr(i) = 0.5*(1+erf(z));
    dYdr(i,1) = dYdr(i,1) + dydr(i);
    dydh(i) = a*0.5*r(1)*c*exp(-z^2);
    dYdh(i,1) = dYdh(i,1) + dydh(i);

    for j=[1:N-2]
        z1 = a*(x(i)-h(j));
        z2 = a*(h(j+1)-x(i));
        y(i) = 0.5*r(j+1)*erf(z1) + 0.5*r(j+1)*erf(z2);
        Y(i) = Y(i) + y(i);
        dydr(i) = 0.5*erf(z1) + 0.5*erf(z2);
        dYdr(i,j+1) = dYdr(i,j+1) + dydr(i);
        dydh(i) = -0.5*a*c*r(j+1)*exp(-z1^2) ;
```

```
        dYdh(i,j) = dYdh(i,j) + dydh(i);
        dydh(i) = 0.5*a*c*r(j+1)*exp(-z2^2);
        dYdh(i,j+1) = dYdh(i,j+1) + dydh(i);
    end

    z = a*(x(i)-h(N-1));
    y(i) = 0.5*r(N)*(1+erf(z));
    Y(i) = Y(i) + y(i);
    dydr(i) = 0.5*(1+erf(z));
    dYdr(i,N) = dYdr(i,N) + dydr(i);
    dydh(i) = -0.5*a*c*r(N)*exp(-z^2);
    dYdh(i,N-1) = dYdh(i,N-1) + dydh(i);
end

% plots the original and smoothed layered models
figure(1);
clf;
axis( [0, x(end), 0, 5 ] );
hold on;
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
xlabel('x');
ylabel('Y(x)');
plot([0,h(1)]',[r(1),r(1)]','r-', 'LineWidth',3);
for i=[1:N-2]
    plot([h(i),h(i)]',[r(i),r(i+1)]','r-', 'LineWidth',3);
    plot([h(i),h(i+1)]',[r(i+1),r(i+1)]','r-', 'LineWidth',3);
end
plot([h(N-1),h(N-1)]',[r(N-1),r(N)]','r-', 'LineWidth',3);
plot([h(N-1),x(end)]',[r(N),r(N)]','r-', 'LineWidth',3);
plot(x,Y,'k-','LineWidth',2);

% This part checks the derivative dYdr for r(K)
% The value of K must ve manually varied, 1<=K<=N
% (which I've done; the derivatives are OK)
K = 4;
Dr = 0.01;
r2 = r; r2(K)=r2(K)+Dr;
y = zeros(Nx,1);
Y2 = zeros(Nx,1);
for i=[1:Nx]
    z = a*(h(1)-x(i));
    y(i) = r2(1)*0.5*(1+erf(z));
    Y2(i) = Y2(i)+y(i);
    for j=[1:N-2]
        z1 = a*(x(i)-h(j));
        z2 = a*(h(j+1)-x(i));
        y(i) = 0.5*r2(j+1)*erf(z1) + 0.5*r2(j+1)*erf(z2);
        Y2(i) = Y2(i) + y(i);
    end
    z = a*(x(i)-h(N-1));
    y(i) = 0.5*r2(N)*(1+erf(z));
    Y2(i) = Y2(i) + y(i);
end
dYdra = (Y2-Y)/Dr;
figure(2);
clf;
hold on;
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
ylabel('dY/dr');
xlabel('x');
title(sprintf('derivative dY/dr for r(%d)',K));
```

```
plot(x,dYdr(:,K),'k-');
plot(x,dYdra,'ko');

% This part checks the derivatives dYdh for h(K)
% the value of K must ve manually varied, 1<=K<N
% (which I've done; the derivatives are OK)
K=3;
Dh = 0.01;
h2 = h; h2(K)=h2(K)+Dh;
y = zeros(Nx,1);
Y2 = zeros(Nx,1);
for i=[1:Nx]
    z = a*(h2(1)-x(i));
    y(i) = r(1)*0.5*(1+erf(z));
    Y2(i) = Y2(i)+y(i);
    for j=[1:N-2]
        z1 = a*(x(i)-h2(j));
        z2 = a*(h2(j+1)-x(i));
        y(i) = 0.5*r(j+1)*erf(z1) + 0.5*r(j+1)*erf(z2);
        Y2(i) = Y2(i) + y(i);
    end
    z = a*(x(i)-h2(N-1));
    y(i) = 0.5*r(N)*(1+erf(z));
    Y2(i) = Y2(i) + y(i);
end
dYdha = (Y2-Y)/Dh;
figure(3);
clf;
hold on;
set(gca,'LineWidth',2);
set(gca,'FontSize',14);
ylabel('dY/dh');
xlabel('x');
title(sprintf('derivative dY/dh for h(%d)',K));
plot(x,dYdh(:,K),'k-');
plot(x,dYdha,'ko');

% To first order, we can write:
% Y(h,r,z) = Y(h0,r0,x) + [ [dY/dr]; DY/dh] ] [Dr; Dh]
%
% Let Dm = [Dr; Dh] and M = [ [dY/dr]; DY/dh] ]
%
% Then standard error propagation gives
% covY = m * covm * M'
% where covm is the covariance matrix of [r; m]
M = [dYdr, dYdh(:,1:N-1)];

% assume uncorelated resistivities with uniform variance
sigmar = 0.1;
% and uncorelated layer positions with uniform variance
sigmah = 0.25;
% and no covariance between layer positiona nd resistivity, so
covm = diag( [(sigmar^2)*ones(N,1); (sigmah^2)*ones(N-1,1)] );

covY = M * covm * M';

figure(4);
clf;
hold on;
title('covariance');
colormap('jet');
axis ij
axis( [0, x(end), 0, x(end) ] );
```

```
imagesc([0,x(end)], [0,x(end)], covY);
colorbar;
xlabel('position x');
ylabel('position x');
```