```
% Midterm 1
% Problem 1, part a

function [x, k, Cm, X2] = LMLSQ(h, var0, J, ep)

% INPUTS:
% h    = Functional form of our inverse problem - data (n x 1 symbolic)
% var0 = Initial guess of variables you want to solve for (symbolic terms
%        in h). Note, must be in alphebetical order. (m x 1)
% J    = Jacobian matrix of h (n x m). Note, Jacobian must also be
%        calcluated s.t. partial derivatives in comumns 1,2...m are in same
%        order as var0.
% ep   = Convergence criteria (scalar)

% OUTPUTS:
% x    = vector containing final estimation of variables
% k    = number of iterations needed to converge on final estimations
% Cm   = model covariance matrix
% X2   = Chi-squared value




% Set generic parameters
v     = 0.5;         % Dampening factor
l0    = 1;           % Lamda (dampening)

% Find number of variables to solve for
lmvars = symvar(h);
nv     = length(lmvars);
nd     = length(J);
I      = eye(nv);

% Set number of iterations
% Note: Will only iterate until convergence test is passed
nints = 100;

dea = []; dca = [];
% Iterate
for k = 1:nints
    % find initial values
    if k == 1
        for j = 1:nv
            varstr = char(lmvars(j));
            evalc([varstr '=' num2str(var0(j))]);
        end
        l = l0;
        fi = eval(subs(h));
        ri = (fi'*fi);
        ci = 0;
    end

    % calculate Jacobian
    Ji = eval(subs(J));
    % calculate residual function
    fi = eval(subs(h));

    % calculate convergence test
    cn = 2.*Ji'*fi;
```

```
    dc = mean(abs(cn-ci));
    de = mean(ep.*(1+abs(cn)));
    dca = [dca; dc]; dea = [dea; de];
    if dc < de
        break % stops the for-loop if convergence test is passed
    end

    % calculate Hessian
    Hi  = (Ji'*Ji)+(I.*l);
    % calculate incremental change in model
    dm = -inv(Hi)*Ji'*fi;

    % update each of the model parameters using dm
    for j = 1:nv
        % find name of symbolic for each element of dm
        varstr  = char(lmvars(j));
        % add incremental value dm to each symbolic
        evalc([varstr '=' varstr '+' num2str(dm(j))]);
    end

    % calculate residual value
    rn = (fi'*fi);
    % compare residual value to last residual value
    rs = ri-rn;
    if rs < 0
        % if current guess is worse, increase lamda
        l = l/v;
    elseif rs >= 0
        % if current guess is better, decrease lamda
        l = l*v;
    end

    % update residual and cost function
    ri = rn;
    ci = cn;

end

Cm = inv(Ji'*Ji);
x  = eval(lmvars);
X2 = ri;
```

```
Not enough input arguments.

Error in LMLSQ (line 29)
lmvars = symvar(h);
```

Published with MATLAB® R2017a