```
function [handles] = createPolyFit(order, curve, interval, dSa)
% Order must be defined in GUI, either 3 or 4, remember str2num
% Remember that curve must bew defined by user via file
%
% order - order of fitted polynomial
% curve - Pairs of Sa and MAF
% interval - lowest and highest number of Sa for fitting
% dSa - spacing of Sa vector

order = str2num(order);

% Not sure how hazard curve is loaded in, assuming that each will be a row
% in a loaded in vector, change if necessary
LoadedSa = curve(:,1);
LoadedMAF = curve(:,2);
npoints = length(LoadedSa);

% Arbitrary Fitting Bounds
Sa_lowerbound = .1;
Sa_upperbound = 2.5;

% Use loaded and MAF to interpolate
Sa_logspace = logspace(log10(Sa_lowerbound), log10(Sa_upperbound), 1000);
MAF_logspace = exp(interp1(log(LoadedSa), log(LoadedMAF), log(Sa_logspace)));

% Sa for fitted curve, can be different interval than the logspace. Use
% to find MAF values for range that we want to integrate over
Sa_fitted = interval(1):dSa:interval(2);

if order == 4
    [p, ~] = polyfit(log(Sa_logspace), log(MAF_logspace), 4);

    regularpoly_SSR = exp(p(5)+p(4)*log(Sa_logspace)+p(3)*log(Sa_logspace).^2+ ...
        p(2)*log(Sa_logspace).^3+p(1)*log(Sa_logspace).^4);

    regularpoly = exp(p(5)+p(4)*log(Sa_fitted)+p(3)*log(Sa_fitted).^2+ ...
        p(2)*log(Sa_fitted).^3+p(1)*log(Sa_fitted).^4);

    SST = sum((MAF_logspace - mean(MAF_logspace)).^2);
    SSR = sum((MAF_logspace - regularpoly_SSR).^2);
    R_squared = 1-SSR/SST;

    % Alter to reflect 4th degree of freedom
    derivpoly = abs((p(4)+2*p(3)*log(Sa_fitted)+3*p(2)*log(Sa_fitted) ...
        .^2+4*p(1)*log(Sa_fitted).^3)./Sa_fitted.*exp(p(5)+p(4)*log(Sa_fitted)+p ↙
(3)...
        *log(Sa_fitted).^2+p(2)*log(Sa_fitted).^3+p(1)*log(Sa_fitted).^4));

    str1=['Curve: ',num2str(p(5)),' + ',num2str(p(4)),'x + ',num2str(p(3)),...
```

```
        'x^2 + ',num2str(p(2)), 'x^3 + ', num2str(p(1)),...
        newline 'R^2 = ', num2str(R_squared)];

elseif order == 3
    [p, ~] = polyfit(log(Sa_logspace),log(MAF_logspace), 3);

    regularpoly_SSR = exp(p(4)+p(3)*log(Sa_logspace)+p(2)*log(Sa_logspace).^2+...
        p(1)*log(Sa_logspace).^3);

    regularpoly = exp(p(4)+p(3)*log(Sa_fitted)+p(2)*log(Sa_fitted).^2+...
        p(1)*log(Sa_fitted).^3);

    SST = sum((MAF_logspace - mean(MAF_logspace)).^2);
    SSR = sum((MAF_logspace - regularpoly_SSR).^2);
    R_squared = 1-SSR/SST;

    derivpoly = abs((p(3)+2*p(2)*log(Sa_fitted)+3*p(1)*log(Sa_fitted)...
        .^2)./Sa_fitted.*exp(p(4)+p(3)*log(Sa_fitted)+p(2)*...
        log(Sa_fitted).^2+p(1)*log(Sa_fitted).^3));

    str1=['Curve: ',num2str(p(4)),' + ',num2str(p(3)),'x + ',num2str(p(2)),...
        'x^2 + ',num2str(p(1)), 'x^3 ' newline 'R^2 = ', num2str(R_squared)];

end


handles.hazardDerivative = [Sa_fitted; derivpoly];
handles.hazardCurve =  [Sa_fitted; regularpoly];

figure
loglog(LoadedSa, LoadedMAF, Sa_fitted, regularpoly)
title('Seismic Hazard Curve')
xlabel('Sa')
ylabel('Mean Annual Frequency')
legend('Hazard Curve','Polyfit')
% xlim([Sa_fitted(1),Sa_fitted(end)])
set(gca, ...
  'Box'         , 'off'     , ...
  'TickDir'     , 'out'     , ...
  'TickLength'  , [.02 .02] , ...
  'XMinorTick'  , 'on'      , ...
  'YMinorTick'  , 'on');
t=text(.015, min(regularpoly), str1);
t.FontSize = 8;
t.BackgroundColor = 'w';
t.EdgeColor = 'k';
grid on
end
```