lsr.m

The function *lsr.m* is a standalone matlab script that was written to perform least squares regression. Matlab has built in functions *lscov.m* for linear regression and *nlinfit.m* for nonlinear regression. *lsr.m* is intended to parallel the syntax of nlinfit, with the additional functionality to:

- perform total least squares
- perform linear least squares
- automatically detect linearity of the modelfun using numerical Hessian
- input analytical Jacobians
- perform χ^2 Goodness of fit test to determine the significance of the computed reference variance σ_0^2
- disable covariance matrix scaling
- add an estimate of the beta coefficients as observation equations

Example Usage (*exampleLsr.m*)

Fit Unweighted 2D Data with Different Models

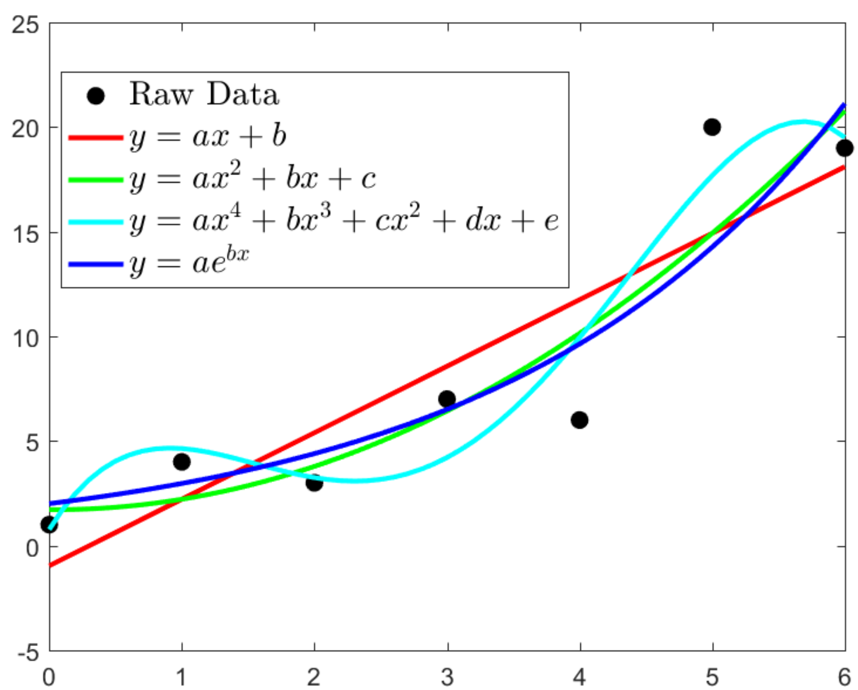
```
%% Fit Different Models to a set of unweighted data
% raw data
x = [0 1 2 3 4 5 6]';
y = [1 4 3 7 6 20 19]';

% Linear Trend  y = mx+b
modelfunLinear = @(b,x) b(1)*x + b(2);
betacoefLinear = lsr(x,y,modelfunLinear);

% 2nd Order Polynomial   y = ax^2+bx+c
modelfunPoly2 = @(b,x) b(1)*x.^2 + b(2)*x +b(3);
betacoefPoly2 = lsr(x,y,modelfunPoly2);

% 4th Order Polynomial  y = ax^4+bx^3+cx^2+dx+e
modelfunPoly4 = @(b,x) b(1)*x.^4 + b(2)*x.^3 + b(3)*x.^2 + b(4)*x.^1 + b(5);
betacoefPoly4 = lsr(x,y,modelfunPoly4);

% Exponential (NonLinear) y = ae^(-bx)
modelfunExp = @(b,x) b(1)*exp(b(2)*x);
betacoef0 = [3 .5]';
betacoefExponential = lsr(x,y,modelfunExp,betacoef0);
```



Sin Wave with known period

$$y = a \sin(\frac{2\pi}{T} x + \phi)$$

2D Conformal Coordinate Transformation

The 2D Conformal equations are:

$$
\begin{aligned}
X &= (S\cos(\theta))x - (S\sin(\theta))y + T_x \\
Y &= (S\sin(\theta))x + (S\cos(\theta))y + T_y
\end{aligned}
$$

By substituting:

$$
\begin{aligned}
a &= S\cos(\theta) \\
b &= S\sin(\theta)
\end{aligned}
$$

Where:

$$
\begin{aligned}
\theta &= \tan^{-1}(\frac{b}{a}) \\
S &= \frac{a}{\cos(\theta)}
\end{aligned}
$$

The observation equations become:

$$
\begin{aligned}
F: &\qquad X = ax - by + T_x \\
G: &\qquad Y = bx + ay + T_y
\end{aligned}
$$

Note that every $(X, Y) \rightarrow (x, y)$ correspondence produces 2 observation equations.

```
    'interpreter','latex','fontsize',14,'location','best')
saveas(f,'basicUsage.png');
%% Use Total Least Squares


%% Sin Wave With Known Period (Nonlinear Observation Equations)
```

```
modelfunPoly2 = @(b,x) b(1)*x.^2 + b(2)*x +b(3);
betacoefPoly2 = lsr(x,y,modelfunPoly2);

% 4th Order Polynomial  y = ax^4+bx^3+cx^2+dx+e
modelfunPoly4 = @(b,x) b(1)*x.^4 + b(2)*x.^3 + b(3)*x.^2 + b(4)*x.^1 + b(5);
betacoefPoly4 = lsr(x,y,modelfunPoly4);

% Exponential (NonLinear) y = ae^(-bx)
modelfunExp = @(b,x) b(1)*exp(b(2)*x);
betacoef0 = [3 .5]';
betacoefExponential = lsr(x,y,modelfunExp,betacoef0);

%plot
xi = 0:0.1:6;
f = figure(1);clf
plot(x,y,'k.','markersize',25);
hold on
plot(xi,modelfunLinear(betacoefLinear,xi),'r','linewidth',2);
plot(xi,modelfunPoly2(betacoefPoly2,xi),'g','linewidth',2);
```

Equations Used

Linear Least Squares

Nonlinear Least Squares

Total Least Squares

Robust Least Squares (Iterative Re-weighted)

```
function [betacoef,R,J,CovB,MSE,ErrorModelInfo] = lsr(x,y,modelfun,varargin)
% LSR Least Squares Regression For Linear, Nonlinear, Robust, and Total LSR
%   LSR(X,Y,MODELFUN,VARARGIN) performs a least squares regression to
%   estimate the beta coefficients.  The Jacobian and partial derivatives
%   are computed using finite differencing.  This function mimics the
%   performance on NLINFIT without the reliance on the Statistics and
%   Machine Learning Toolbox.  This function also adds functionality to:
%    o perform linear least squares
%    o detect linearity from modelfun using finite differencing Hessian
%    o perform total least squares
%    o input analytical jacobians
%    o automatically perform chi2 goodness of fit test
%    o disable automatic covariance scaling
%    o add an estimate of the beta coefficients as observation equations
%
%   * See 'LSR.pdf' for more a more detailed description%
%   * See 'exampleLsr.m' for more example uses
%
% Inputs:
%   - x              : Predictor variables
%   - y              : Response values
%   - modelfun       : Model function handle @modelfun(betacoef,x)
%   - betacoef0      : Initial coefficient values
%
% Optional Parameters:
%   - 'betacoef0'            : Initial coefficient values
%   - 'type'                 : Type of Regression
%   - 'weights'              : Vector (weights) or Covariance matrix
%   - 'AnalyticalJacobian'   : Function @(b,x) for Jacobian wrt betacoef
%   - 'AnalyticalBfunction'  : Function @(b,x) for Jacobian wrt x
%   - 'noscale'              : true(false)/false to scale covariance matrix
%   - 'betaCoef0Cov'         : covariance of beta0 coefficient values
%   - 'chi2alpha'            : alpha values for confidence
%   - 'RobustWgtFun'         : Robust Weight Function
%   - 'Tune'                 : Robust Wgt Tuning Function
%   - 'RobustThresh'         : Threshold for Robust Iterations
%   - 'RobustMaxIter'        : Maximum iterations in Robust Least Squares
%   - 'maxiter'              : Maximum iterations for Nonlinear
%   - 'verbose'              : true/false print verbose output to screen
%   - 'DeriveStep'           : Difference for numerical jacobian
%
% Outputs:
%   - betacoef       : Estimated regression coefficients
%   - R              : Residuals
%   - J              : Jacobian
%   - CovB           : Estimated Variance Covariance Matrix
%   - MSE            : Mean Squared Error (Computed Reference Variance)
%   - ErrorModelInfo : Information about error model
```

Variable Definitions

$$
\begin{aligned}
m &= \text{Number of Observation Equations} \\
n &= \text{Number of Unknowns} \\
p &= \text{Number of Observation Equations per Observation} \\
q &= \text{Number of Predictor Variables per Observation} \\
\beta &= \text{Estimated Regression Coefficients} \\
x &= \text{Predictor Variables} \\
y &= \text{Response Variables} \\
V &= \text{Residuals} \\
V_{eq} &= \text{Equivalent Residuals (TLS)} \\
F(\beta, x) &= \text{Observation Equation} \\
w &= \text{Vector of Weights for each observation} \\
W &= \text{Weight Matrix} \\
W_{eq} &= \text{Equivalent Weight Matrix (TLS)} \\
\Sigma_{xx} &= \text{Covariance Matrix of Predictor Variables} \\
\Sigma_{yy} &= \text{Covariance Matrix of Response Variables} \\
\Sigma_{\beta\beta} &= \text{Covariance Matrix of Estimated Regression Coefficients} \\
\sigma_0^2 &= \text{Computed Reference Variance} \\
Q &= \text{Cofactor Matrix} \\
\sigma_\beta &= \text{Standard Deviation of Estimated Regression Coefficients} \\
\hat{y} &= \text{Predicted Response Variables} \\
R^2 &= \text{model skill (Does not apply to nonlinear equations)} \\
RMSE &= \text{Root Mean Square Error}
\end{aligned}
$$

Equations:

$$y = F(\beta, x)$$

Linear Least Squares Equations:

$$
\begin{aligned}
WA\beta &= y \\
\hat{\beta} &= (A^T WA)^{-1} WA^T y + WV
\end{aligned}
$$

NonLinear Least Squares Equations:

$$
\begin{aligned}
WJ_{y\beta}\Delta\beta &= K = y - F(x, \beta) + WV \\
\hat{\Delta\beta} &= (J_{y\beta}^T W J_{y\beta})^{-1} W J_{y\beta}^T K
\end{aligned}
$$

Total Least Squares Equations:

$$
\begin{aligned}
J_{yx}V + J_{y\beta}\Delta\beta &= K = y - F(\beta, x) + V_{eq} \\
W_{eq} &= (J_{yx}\Sigma_{xx}J_{yx}^T)^{-1} \\
\hat{\Delta\beta} &= (J_{y\beta}^T W_{eq} J_{y\beta})^{-1} W_{eq} J_{y\beta}^T K
\end{aligned}
$$

Robust Least Squares Equations:

$$\text{Hat Matrix: } H = J_{y\beta}(J_{y\beta}^T J_{y\beta})^{-1} J_{y\beta}$$
$$\text{Leverages: } h = diag(H)$$
$$\text{Adjusted Residuals: } R_{adj} = R * sqrt(h)$$
$$\text{Ensure minimum residuals: } R_{adj} = max(1e - 6, Radj)$$
$$\text{Estimated std (Median Absolute Residuals): } \sigma_{\hat{R_{adj}}} = MAD(R_{adj})/0.6745$$
$$\text{Normalized Residuals: } R_{norm} = R_{adj}/(RobustTune * \sigma_{R_{adj}})$$
$$\text{Weight Vector: } W = RobustWgtFun(R_{norm})$$