Curve Fitting Toolbox

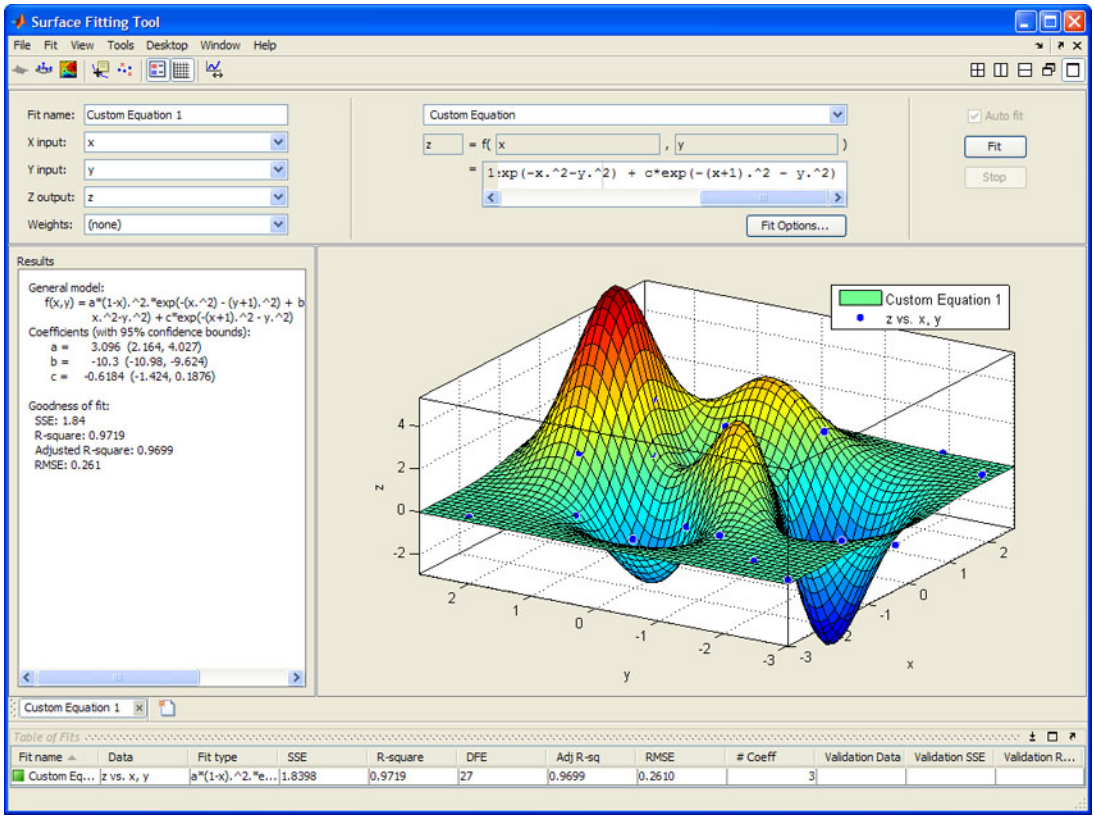
Fit curves and surfaces to data using regression, interpolation, and smoothing

Curve Fitting Toolbox™ provides an app and functions for fitting curves and surfaces to data. The toolbox lets you perform exploratory data analysis, preprocess and post-process data, compare candidate models, and remove outliers. You can conduct regression analysis using the library of linear and nonlinear models provided or specify your own custom equations. The library provides optimized solver parameters and starting conditions to improve the quality of your fits. The toolbox also supports nonparametric modeling techniques, such as splines, interpolation, and smoothing.

After creating a fit, you can apply a variety of post-processing methods for plotting, interpolation, and extrapolation; estimating confidence intervals; and calculating integrals and derivatives.

Key Features

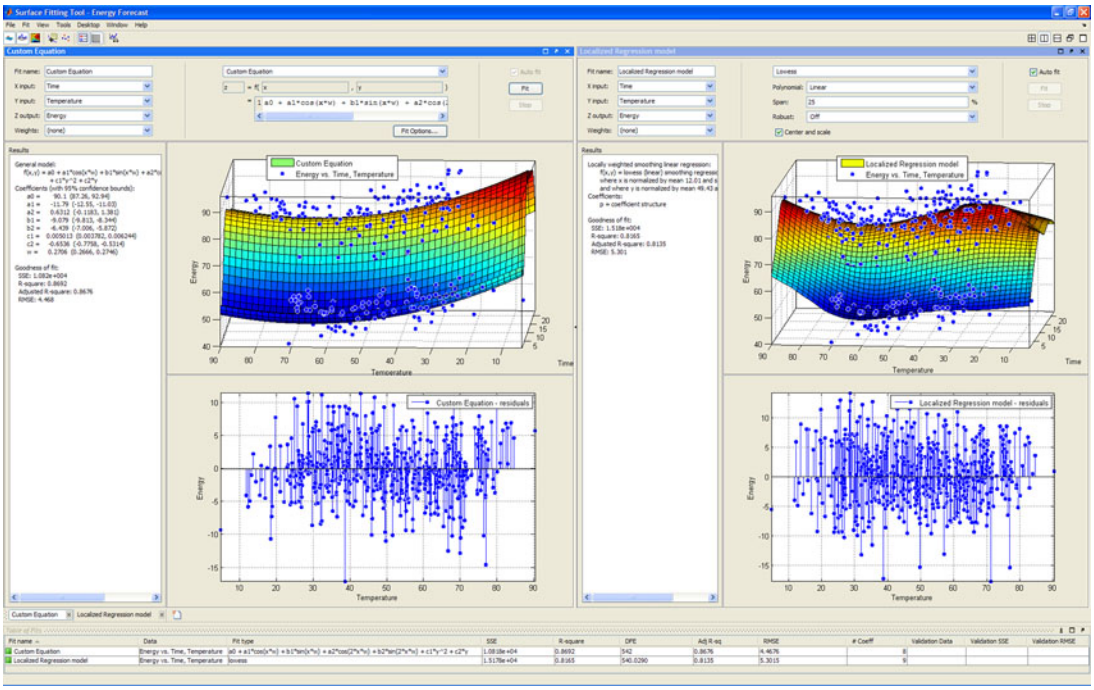
- Curve Fitting app for curve and surface fitting
- Linear and nonlinear regression with custom equations
- Library of regression models with optimized starting points and solver parameters
- Interpolation methods, including B-splines, thin plate splines, and tensor-product splines
- Smoothing techniques, including smoothing splines, localized regression, Savitzky-Golay filters, and moving averages
- Preprocessing routines, including outlier removal and sectioning, scaling, and weighting data
- Postprocessing routines, including interpolation, extrapolation, confidence intervals, integrals and derivatives



Surface generated using the Surface Fitting Tool. The tool supports a variety of fitting methods, including linear regression, nonlinear regression, interpolation, and smoothing.

Working with Curve Fitting Toolbox

Curve Fitting Toolbox provides the most widely used techniques for fitting curves and surfaces to data, including linear and nonlinear regression, splines and interpolation, and smoothing. The toolbox supports options for robust regression to fit data sets that contain outliers. All algorithms can be accessed through functions or the Curve Fitting app.

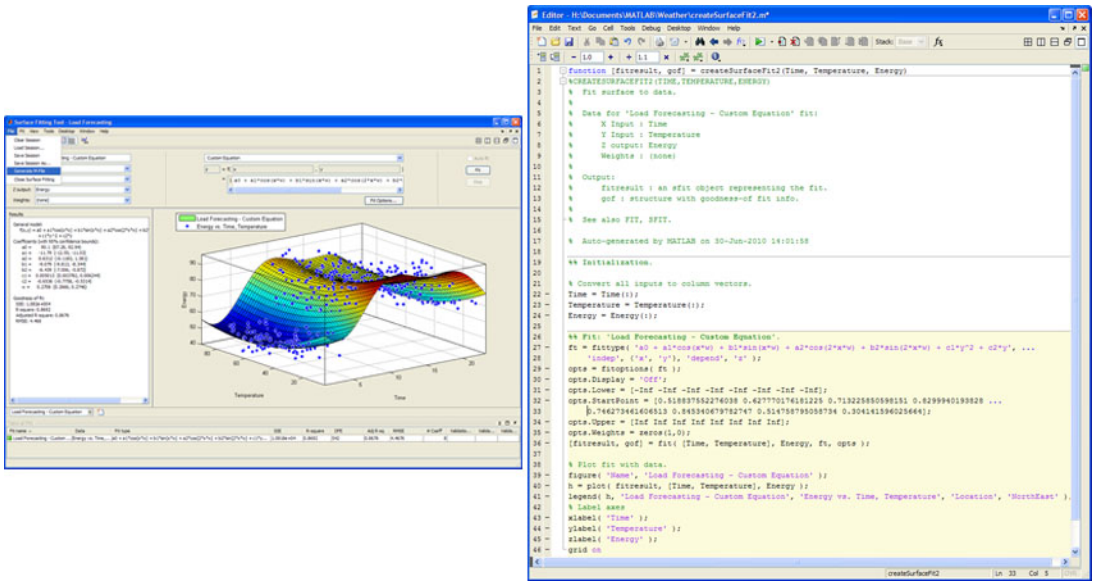


Fitting multiple candidate models to a single data series using the Curve Fitting app. You can compare the fitted surfaces visually or use goodness-of-fit metrics such as R^2, adjusted R^2, sum of the squared errors, and root mean squared error.

Fitting Data Interactively

The Curve Fitting app simplifies common tasks that include:

- Importing data from the MATLAB® workspace
- Visualizing your data to perform exploratory data analysis
- Generating fits using multiple fitting algorithms
- Evaluating the accuracy of your models
- Performing postprocessing analysis that includes interpolation and extrapolation, generating confidence intervals, and calculating integrals and derivatives
- Exporting fits to the MATLAB workspace for further analysis
- Automatically generating MATLAB code to capture work and automate tasks



MATLAB function generated with the Surface Fitting Tool.

Working at the Command Line

Working at the command line lets you develop custom functions for analysis and visualization. These functions enable you to:

- Duplicate your analysis with a new data set
- Replicate your analysis with multiple data sets (batch processing)
- Embed a fitting routine into MATLAB functions
- Extend the base capabilities of the toolbox

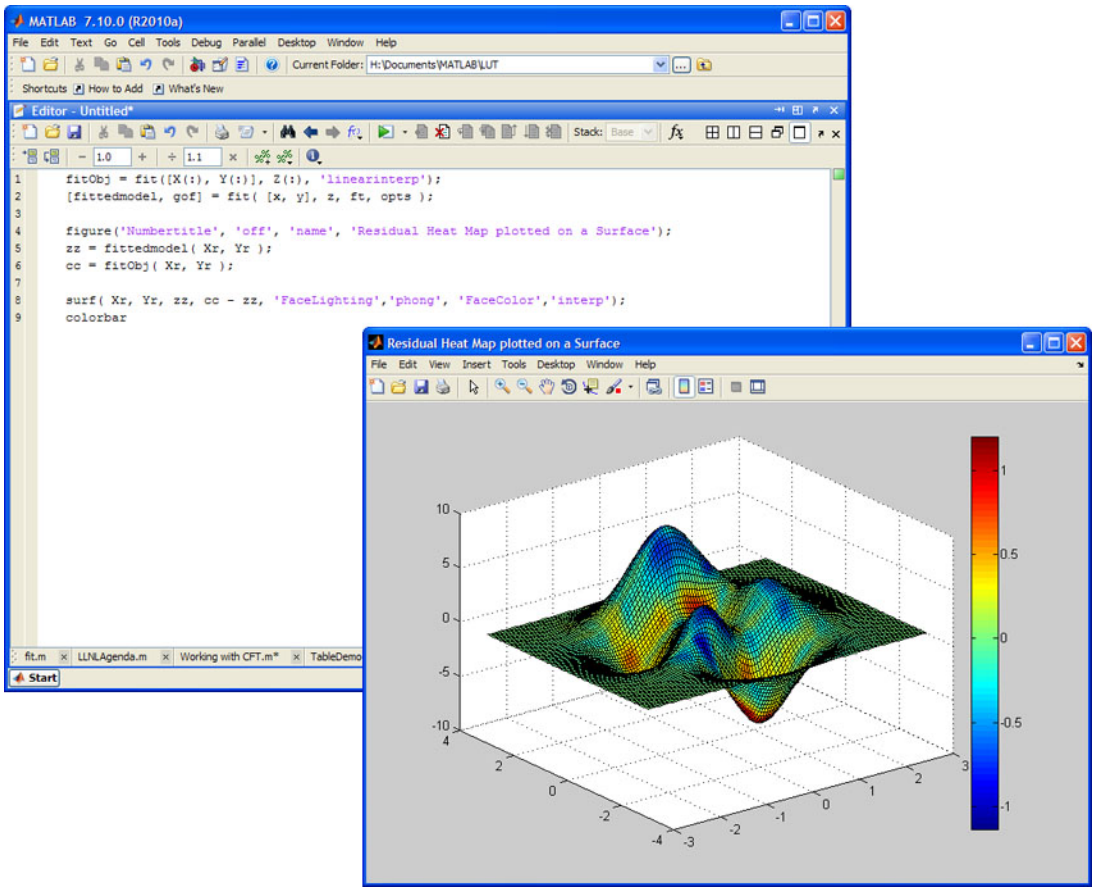
Curve Fitting Toolbox provides a simple intuitive syntax for command-line fitting, as in the following examples:

- Linear Regression: `fittedmodel = fit([X,Y], Z, 'poly11');`
- Nonlinear Regression: `fittedmodel = fit(X, Y, 'fourier2');`
- Interpolation: `fittedmodel = fit([Time,Temperature], Energy, 'cubicinterp');`
- Smoothing: `fittedmodel = fit([Time,Temperature], Energy, 'lowess', 'span', 0.12);`

The results of a fitting operation are stored in an object called "`fittedmodel`." Postprocessing analysis, such as plotting, evaluation, and calculating integrals and derivatives, can be performed by applying a method to this object, as in these examples:

- Plotting: `plot(fittedmodel)`
- Differentiation: `differentiate(fittedmodel, X, Y)`
- Evaluation: `fittedmodel(80, 40)`

Curve Fitting Toolbox lets you move interactive fitting to the command line. Using the app, you can automatically generate MATLAB code. You can also create fit objects with the app and export them to the MATLAB workspace for further analysis.



Extending the capabilities of the toolbox with a custom visualization. The color of the heat map corresponds to the deviation between the fitted surface and a reference model.

Regression

Curve Fitting Toolbox supports linear and nonlinear regression.

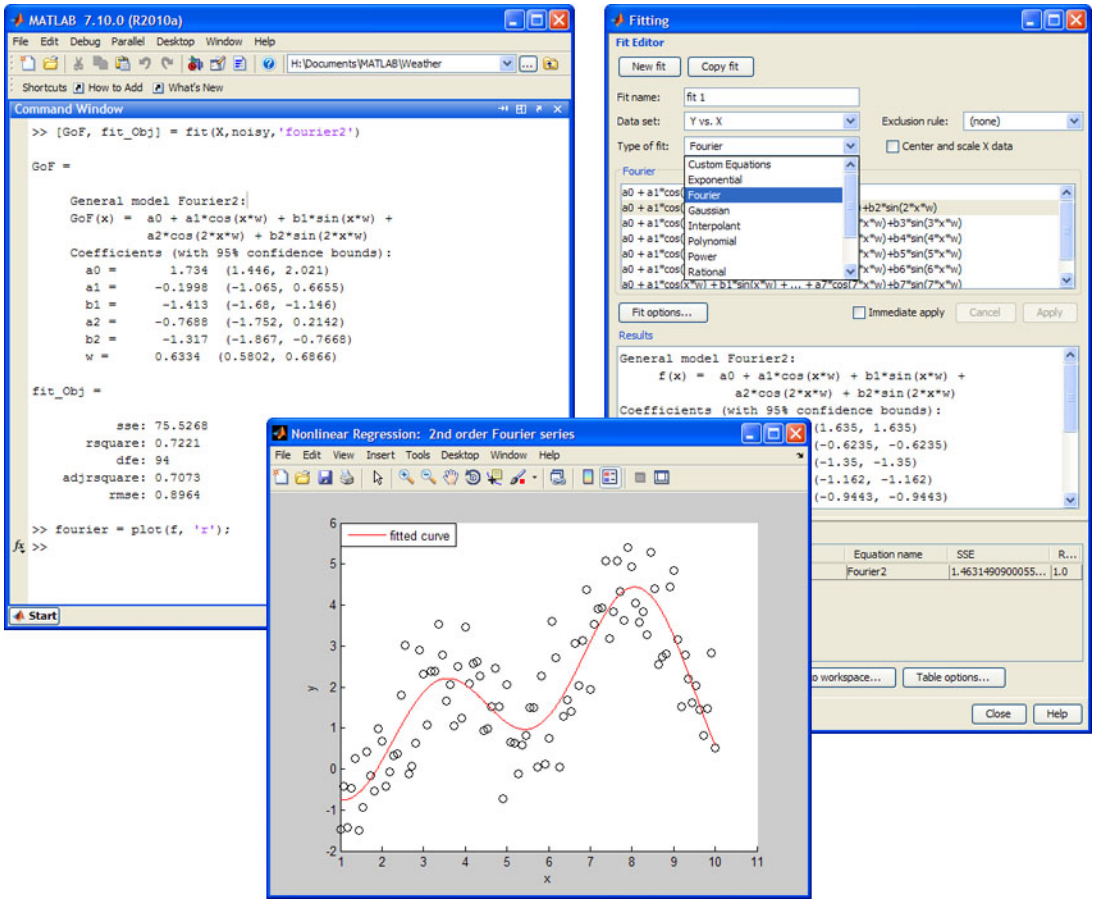
Linear Regression

The toolbox supports over 100 regression models, including:

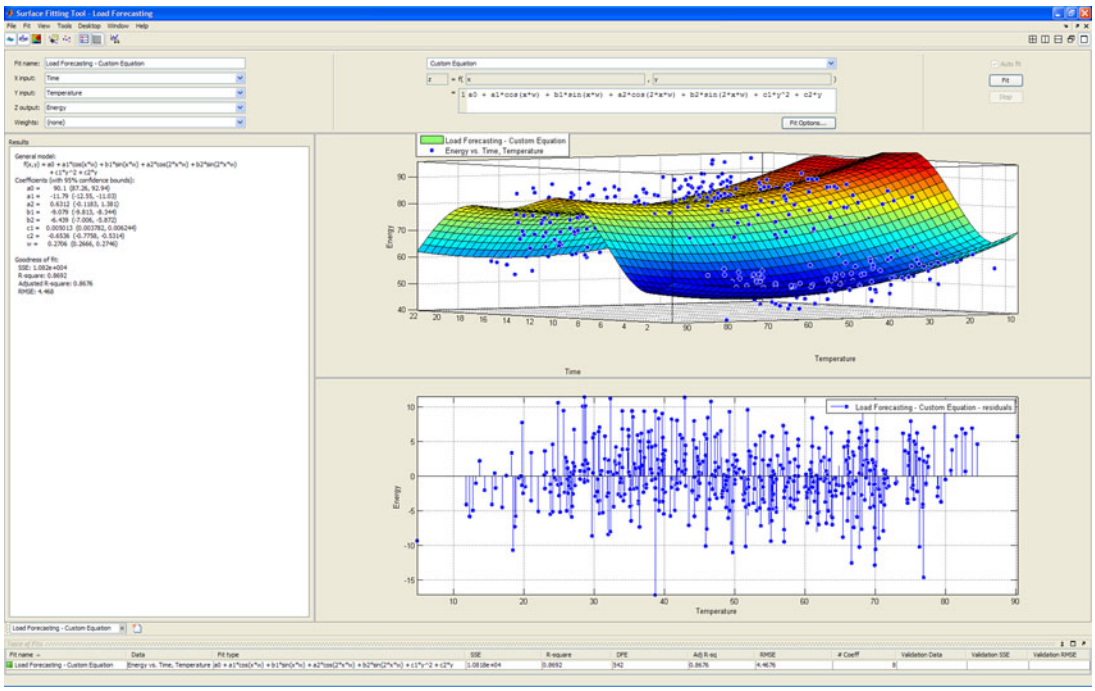
- Lines and planes
- High order polynomials (up to ninth degree for curves and fifth degree for surfaces)
- Fourier and power series
- Gaussians
- Weibull functions
- Exponentials
- Rational functions
- Sum of sines

All of these standard regression models include optimized solver parameters and starting conditions to improve fit quality. Alternatively, you can use the Custom Equation option to specify your own regression model.

In the Curve Fitting app you can generate fits based on complicated parametric models by using a drop-down menu. At the command line you can access the same models using intuitive names.



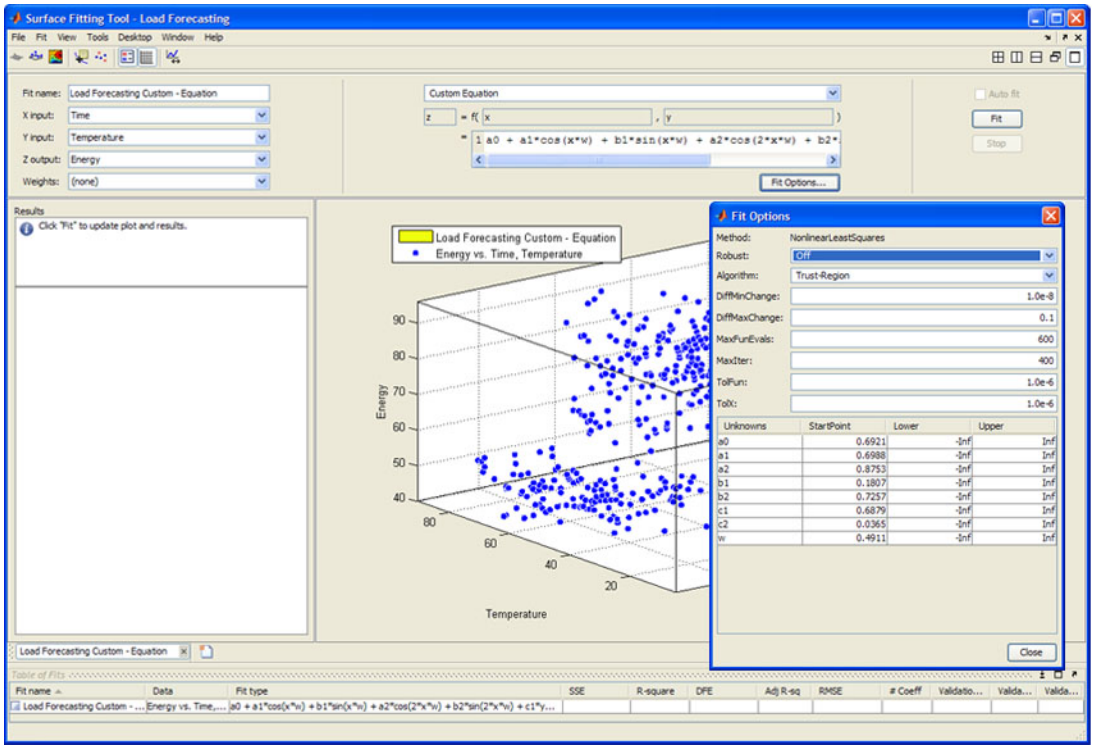
Nonlinear regression using a second-order Fourier series. You can pass the argument "fourier2" to the fit command (top, left) or select a second-order Fourier series in the Fit Editor (top, right).



Surface generated using the Custom Equation option of the Surface Fitting Tool. You can specify a custom equation or input a MATLAB function.

The regression analysis options in Curve Fitting Toolbox enable you to:

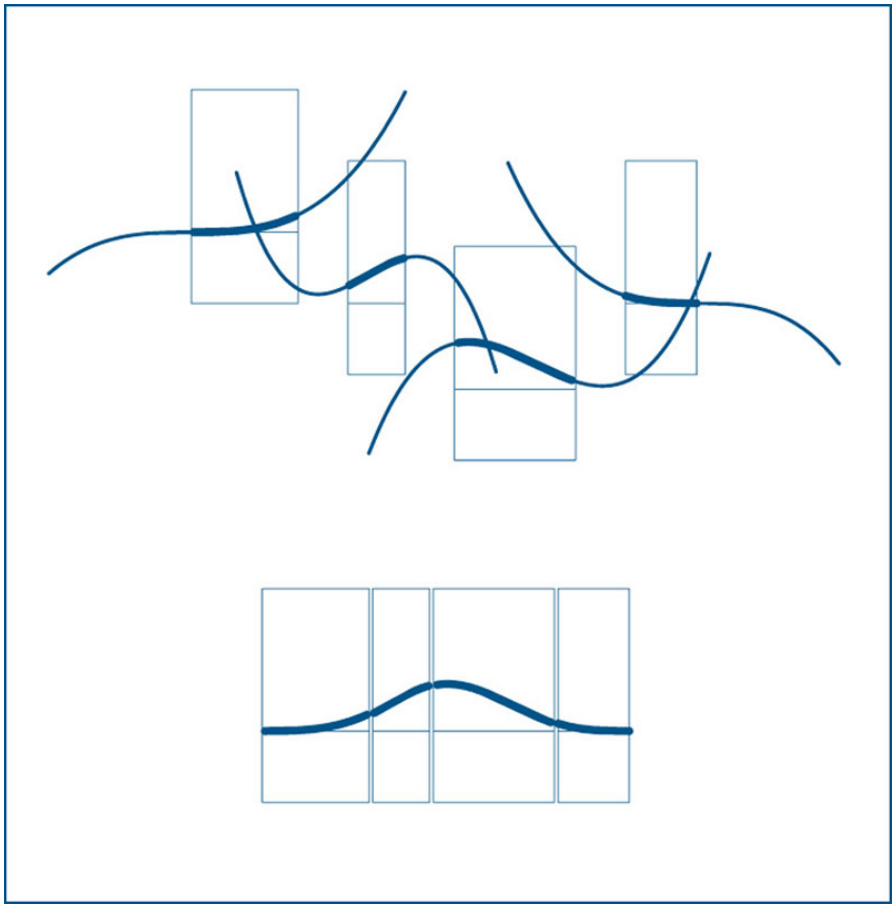
- Choose between two types of robust regression: bisquare or least absolute residual
- Specify starting conditions for the solvers
- Constrain regression coefficients
- Choose Trust-Region or Levenberg-Marquardt algorithms



Fit Options GUI in the Surface Fitting Tool. You can control the type of robust regression, the choice of optimization solver, and the behavior of the optimization solver with respect to starting conditions and constraints.

Splines and Interpolation

Curve Fitting Toolbox supports a variety of interpolation methods, including B-splines, thin plate splines, and tensor product splines. Curve Fitting Toolbox provides functions for advanced spline operations, including break/knot manipulation, optimal knot placement, and data-point weighting.



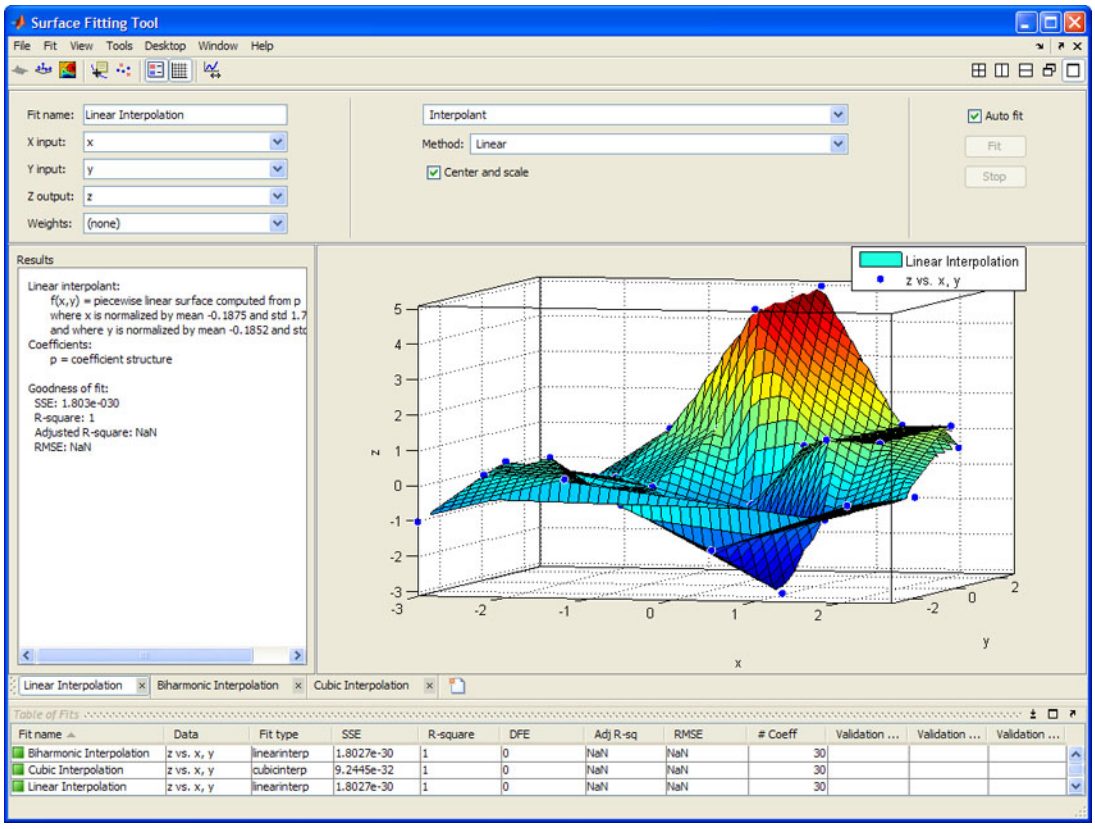
A cubic B-spline and the four polynomials from which it is made. Splines are smooth piecewise polynomials used to represent functions over large intervals.

You can represent a polynomial spline in ppform and B-form. The ppform describes the spline in terms of breakpoints and local polynomial coefficients, and is useful when the spline will be evaluated extensively. The B-form describes a spline as a linear combination of B-splines, specifically the knot sequence and B-spline coefficients.

Curve Fitting Toolbox also supports other types of interpolation, including:

- Linear interpolation
- Nearest neighbor interpolation
- Piecewise cubic interpolation
- Biharmonic surface interpolation
- Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)

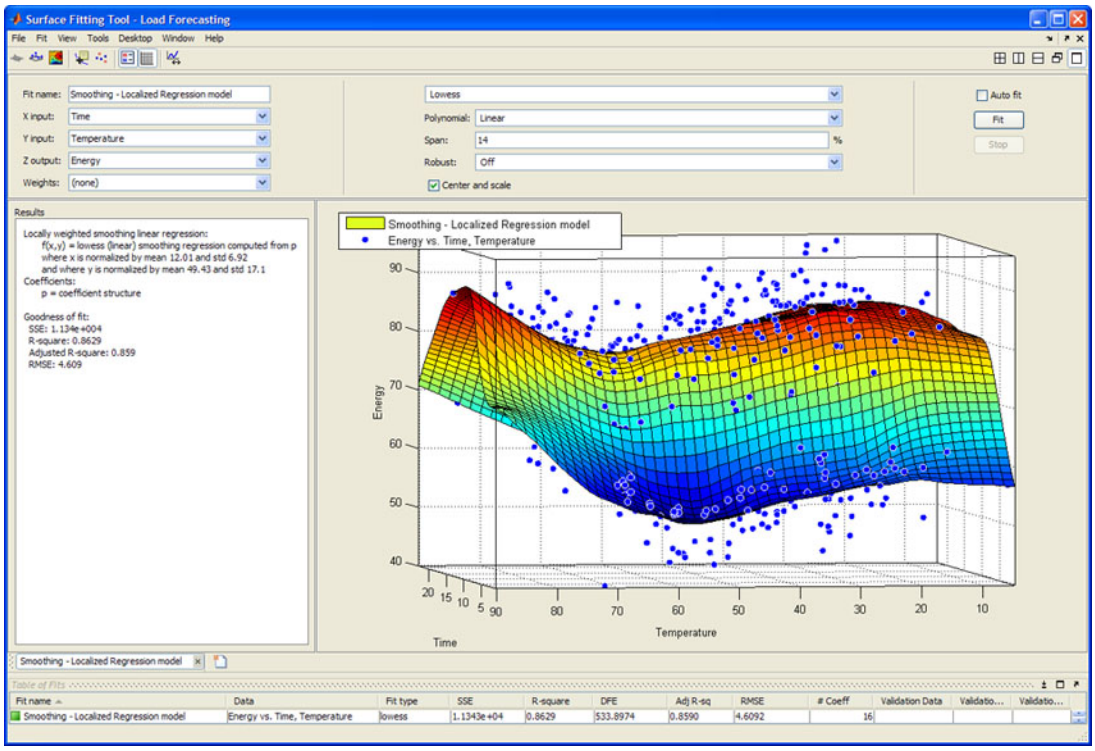
The Curve Fitting Toolbox commands for constructing spline approximations accommodate vector-valued gridded data, enabling you to create curve and surfaces in any number of dimensions.



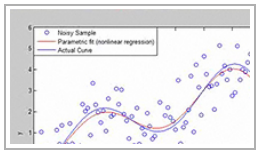
Linear interpolation using the Curve Fitting app.

Smoothing

Smoothing algorithms are widely used to remove noise from a data set while preserving important patterns. Curve Fitting Toolbox supports both smoothing splines and localized regression, which enable you to generate a predictive model without specifying a functional relationship between the variables.

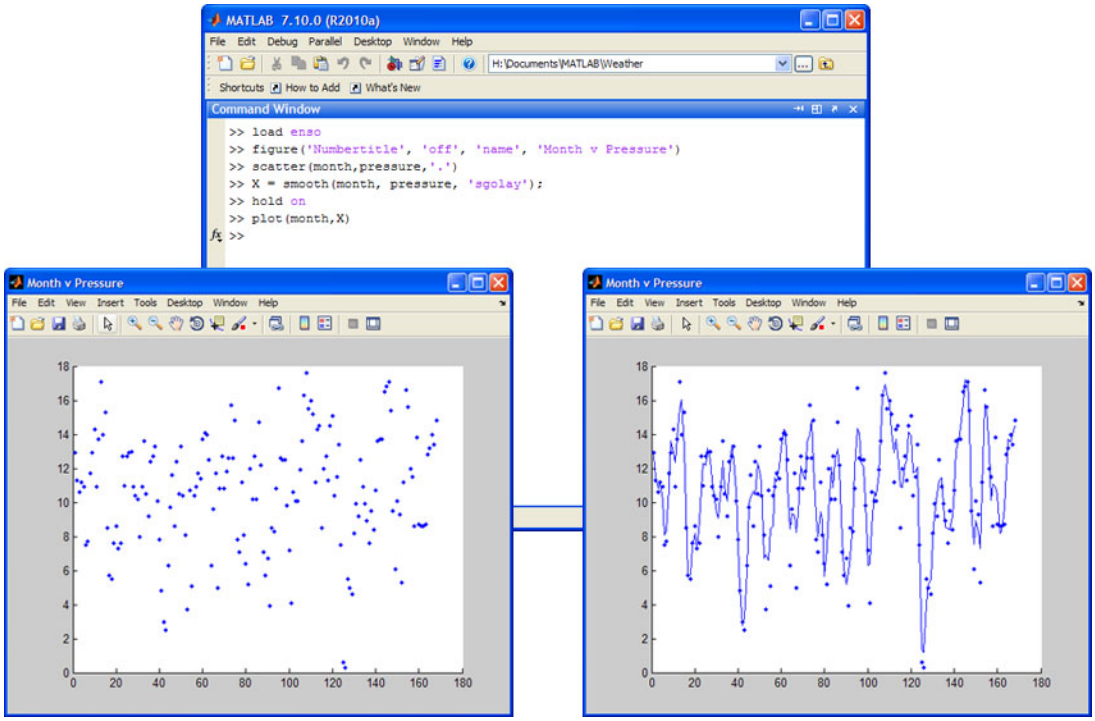


Localized regression model. Smoothing techniques can be used to generate predictive models without specifying a parametric relationship between the variables.



Nonparametric Fitting 4:07
Develop a predictive model without specifying a function that describes the relationship between variables.

Curve Fitting Toolbox supports localized regression using either a first-order polynomial (lowess) or a second-order polynomial (loess). The toolbox also provides options for robust localized regression to accommodate outliers in the data set. Curve Fitting Toolbox also supports moving average smoothers such as Savitzky-Golay filters.



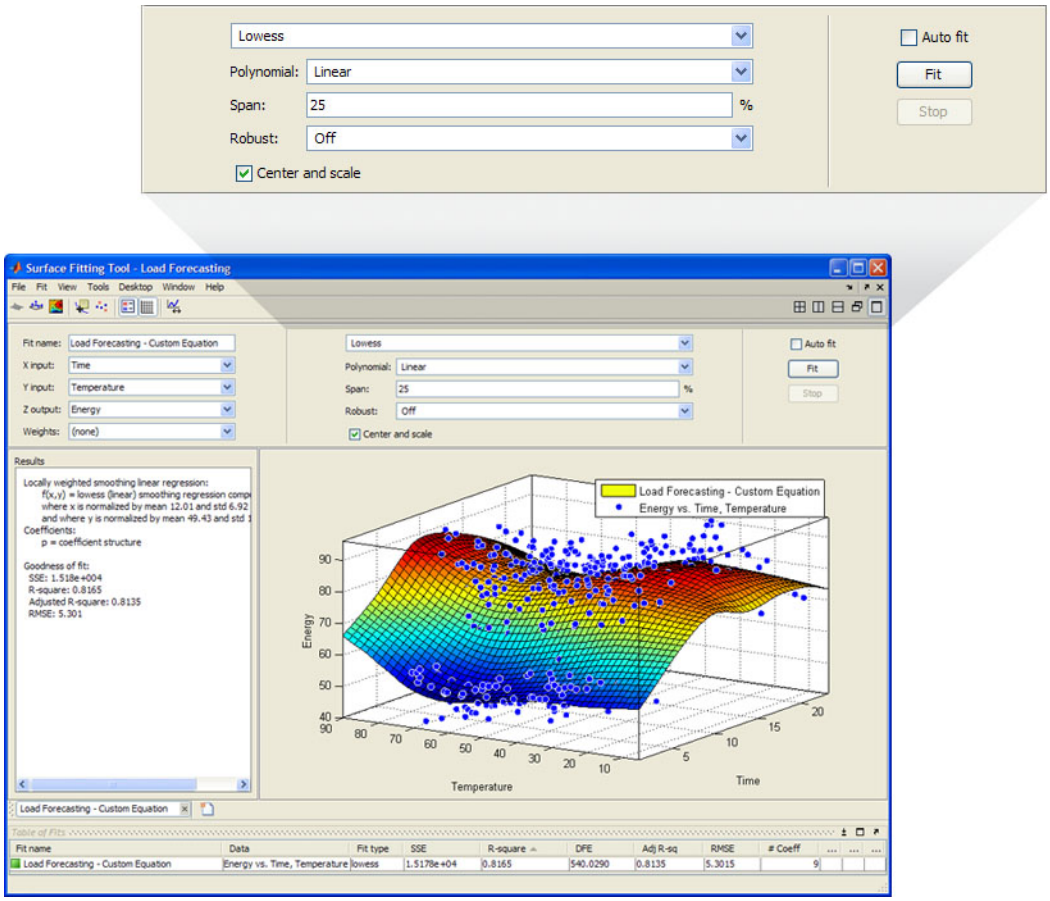
Exploratory data analysis using a Savitzky-Golay filter. Smoothing data enables you to identify periodic components.

Previewing and Preprocessing Data

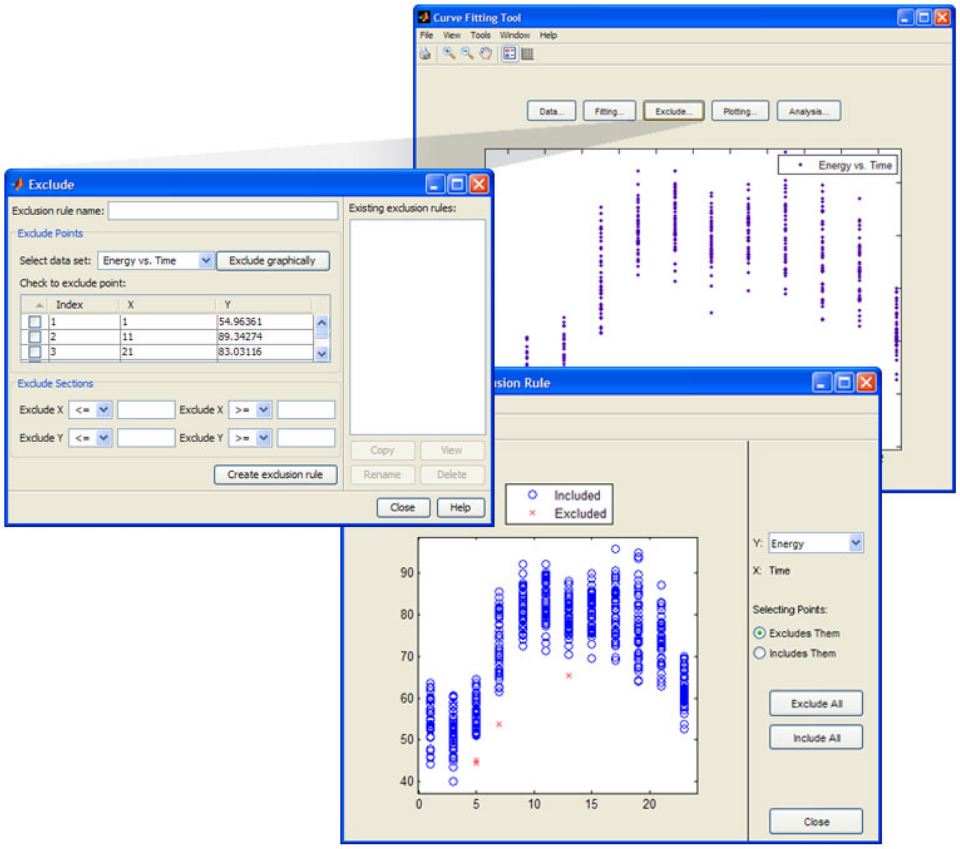
Curve Fitting Toolbox supports a comprehensive workflow that progresses from exploratory data analysis (EDA) through model development and comparison to postprocessing analysis.

You can plot a data set in two or three dimensions. The toolbox provides options to remove outliers, section data series, and weight or exclude data points.

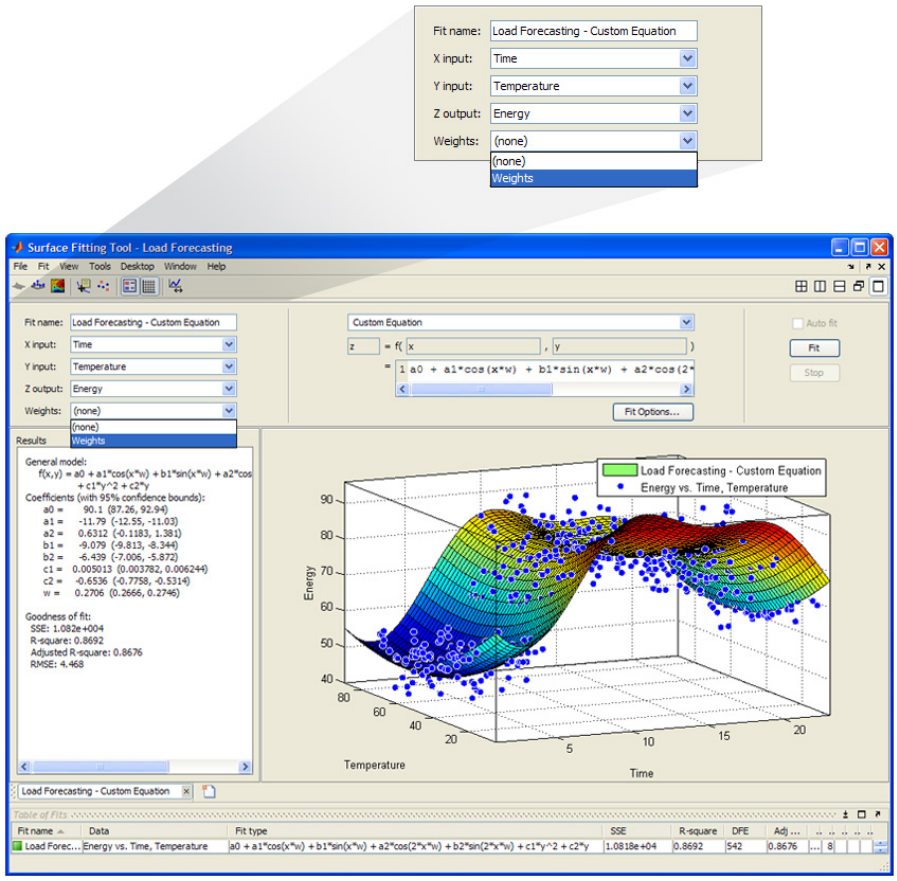
Curve Fitting Toolbox lets you automatically center and scale a data set to normalize the data and improve fit quality. The Center and scale option can be used when there are dramatic differences in variable scales or the distance between data points varies across dimensions.



Normalizing data with the Center and scale option to improve fit quality.



Outlier exclusion using the Curve Fitting app (top). You can create exclusion rules (middle) to remove all data points that match some specific criteria, and use the graphical exclusion option (bottom) to click on a data point and remove it from the sample.



Weighting data points using the Surface Fitting Tool.

Developing, Comparing, and Managing Models

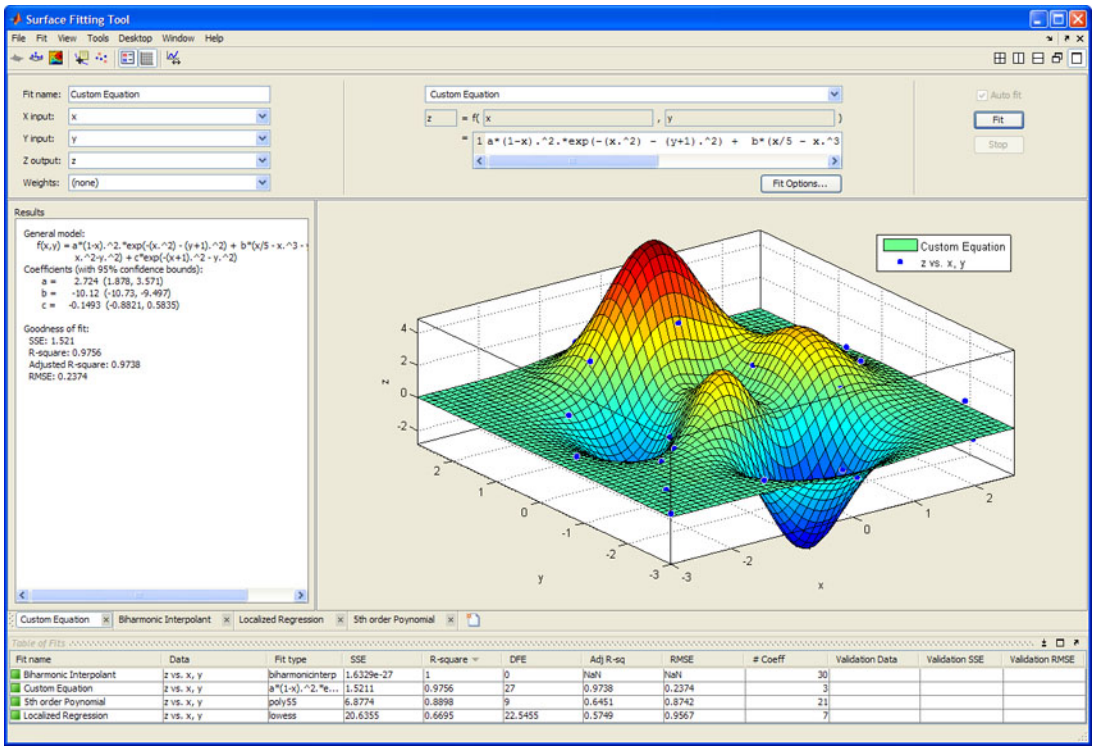
Curve Fitting Toolbox lets you fit multiple candidate models to a data set. You can then evaluate goodness of fit using a combination of descriptive statistics, visual inspection, and validation.

Descriptive Statistics

Curve Fitting Toolbox provides a wide range of descriptive statistics, including:

- R-square and adjusted R-square
- Sum of squares due to errors and root mean squared error
- Degrees of freedom

The Table of Fits lists all of the candidate models in a sortable table, enabling you to quickly compare and contrast multiple models.

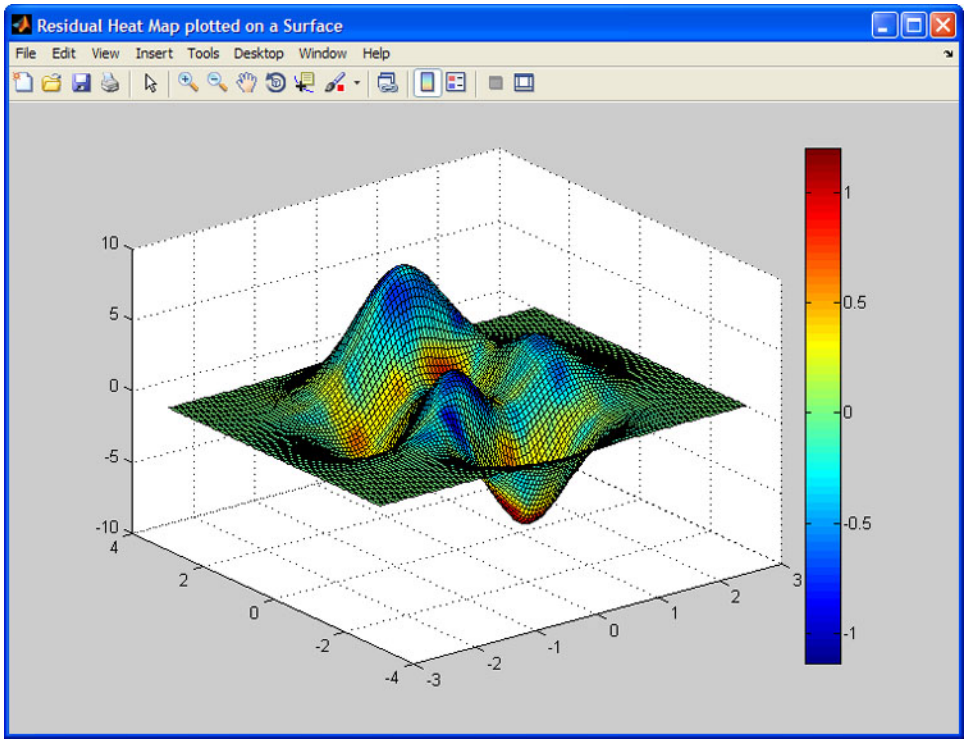


The Curve Fitting app, which provides a sortable table of candidate models.

Visual Inspection of Data

The toolbox enables you to visually inspect candidate models to reveal problems with fit that are not apparent in summary statistics. For example, you can:

- Generate side-by-side surface and residual plots to search for patterns in the residuals
- Simultaneously plot multiple models to compare how well they fit the data in critical regions
- Plot the differences between two models as a new surface



Surface generated with the Surface Fitting Tool. The color of the heat map corresponds to the deviation between the fitted surface and a reference model.

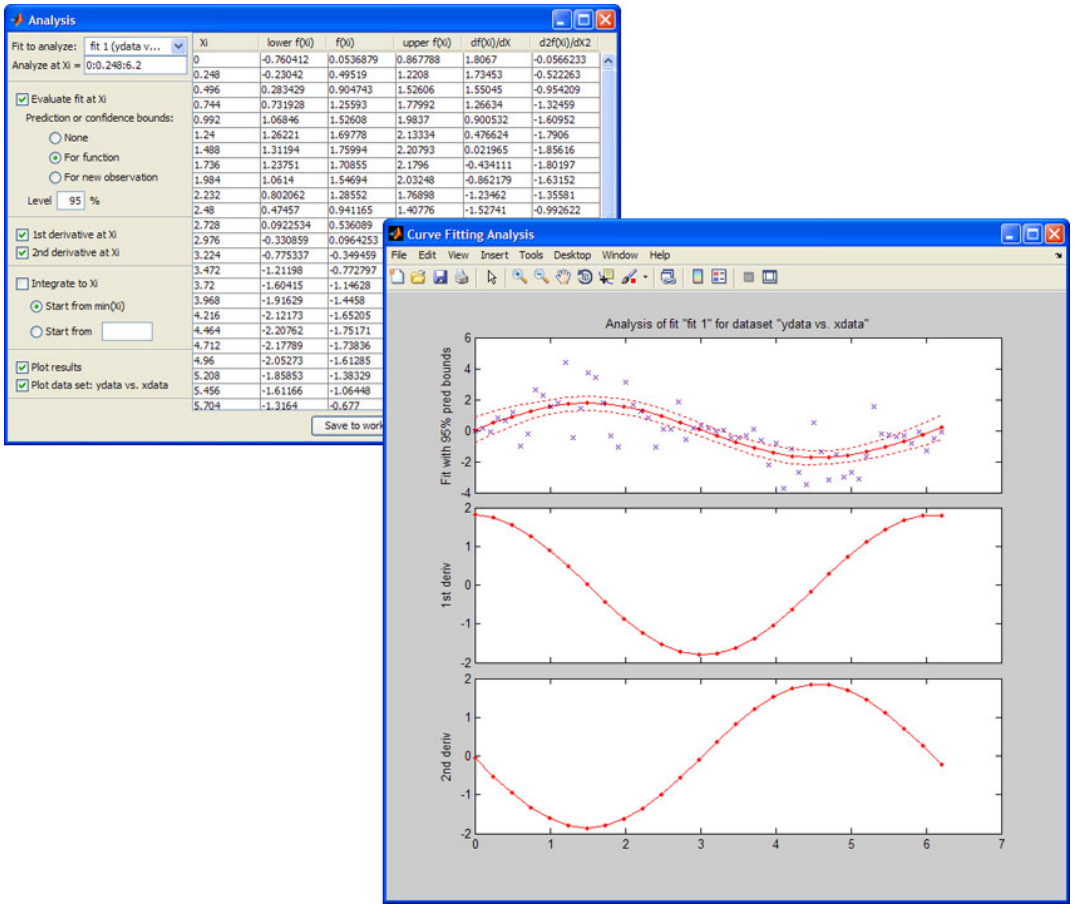
Validation Techniques

Curve Fitting Toolbox supports validation techniques that help protect against overfitting. You can generate a predictive model using a training data set, apply your model to a validation data set, and then evaluate goodness of fit.

Postprocessing Analysis

Once you have selected the curve or surface that best describes your data series you can perform postprocessing analysis. Curve Fitting Toolbox enables you to:

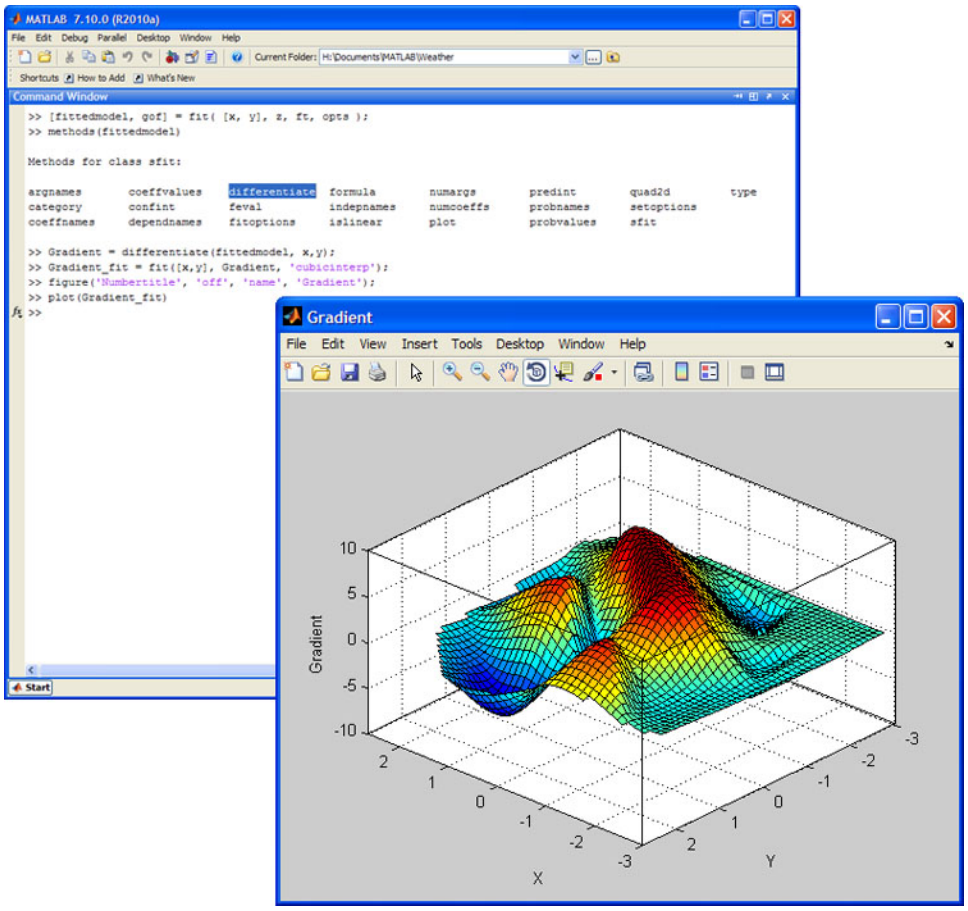
- Create plots
- Use your model to estimate values (evaluation)
- Calculate confidence intervals
- Create prediction bounds
- Determine the area under your curve (integration)
- Calculate derivatives



Postprocessing analysis with the Curve Fitting app, which automatically creates a scatter plot of the raw data along with the fitted curve. The first and second derivatives of the fitted curve are also displayed.

The following examples show how postprocessing at the command line applies intuitive commands to the objects created from a fitting operation:

- Evaluation: `EnergyConsumption = fittedmodel(X, Y)`
- Plotting: `EnergySurface = plot(fittedmodel)`
- Integration: `Volume_Under_Surface = quad2d(fittedmodel, Min_X, Max_X, Min_Y, Max_Y)`
- Differentiation: `Gradient = differentiate(fittedmodel, X,Y)`
- Computing `confidence intervals: Confidence_Intervals = confint(fittedmodel)`



Using command-line postprocessing to calculate and plot a gradient.

Resources

Product Details, Examples, and System Requirements
www.mathworks.com/products/curvefitting

Trial Software
www.mathworks.com/trialrequest

Sales
www.mathworks.com/contactsales

Technical Support
www.mathworks.com/support

Online User Community
www.mathworks.com/matlabcentral

Training Services
www.mathworks.com/training

Third-Party Products and Services
www.mathworks.com/connections

Worldwide Contacts
www.mathworks.com/contact