Lecture 22 - Interpolation and Curve Fitting

- Prayer/Spiritual Thought
- Announcements

Outline

1. Interpolation
2. Linear Regression
3. Nonlinear Regression

1. Interpolation

A. Explanation

- Interpolation is the process of using (x,y) data at discrete points to infer the values in between those discrete points.
- We talked about two different types of interpolation:

    1. Linear interpolation -- lines connect each data point
    2. Cubic Spline -- cubic polynomials smoothly connect each data point

- Interpolation methods are valuable when the noise in the data is "small." How small is small enough is a judgement call.

B. Recipes

- Read data from a file using: READPRN("<filename>")
- Linear interpolation:

    1. Read/Define your x, y data points
    2. Define the x point(s) you want to interpolate at
    3. Find the y points from: y := linterp(<xdata>, <ydata>, <x interpolation points>)

- Cubic Spline:

    1. Read/Define your x, y data points
    2. Define the x point(s) you want to interpolate at
    3. Find the coefficients for the cubic spline by calling: coeffs := cspline(<xdata>, <ydata>)
    4. Find the y points from: y := interp(<coeffs>, <xdata>, <ydata>, <x interp points>)

C. Examples

(i) Linear interpolation,
read data from file

$$xy_{data} := \mathrm{READPRN}(\text{“Lec\_22–Ex\_A.dat”})$$

$$x_{data} := xy_{data}^{\langle 0 \rangle} \qquad y_{data} := xy_{data}^{\langle 1 \rangle}$$

$$x_{data} = \begin{bmatrix} 0 \\ 0.714 \\ 1.429 \\ 2.143 \\ 2.857 \\ 3.571 \\ 4.286 \\ 5 \\ 5.714 \\ 6.429 \\ 7.143 \\ 7.857 \\ \vdots \end{bmatrix} \qquad y_{data} = \begin{bmatrix} 2.404 \cdot 10^{-4} \\ 0.002 \\ 0.014 \\ 0.066 \\ 0.216 \\ 0.506 \\ 0.844 \\ 1 \\ 0.844 \\ 0.506 \\ 0.216 \\ 0.066 \\ \vdots \end{bmatrix}$$

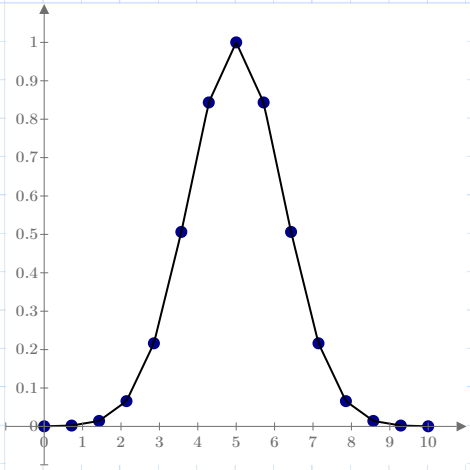
Tips:

- The header before the data (text before the columns of numbers) can't have a number in it. READPRN starts reading when it finds a number, and having numbers in the header will mess it up.
- Get the columns of the xy_data matrix using the column operator (Ribbon -> Vector/ Matrix Operators -> Matrix Column or the shortcut: <ctrl><shift>C).

(get the x points for the interpolation)

$N := 100 \quad i := 0..N \quad beg := 0 \quad end := 10 \quad x_{interp_i} := beg + (end - beg)\frac{i}{N}$

$y_{interp} := \text{linterp}(x_{data}, y_{data}, x_{interp})$



y_{data}

y_{interp}

x_{data}

x_{interp}

(ii) Cubic spline, data in vector

$i := 0 .. 10$

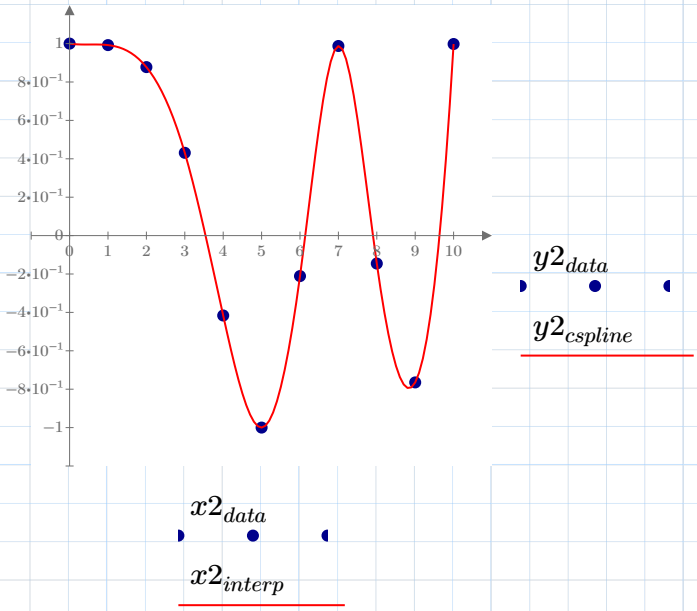
$x2_{data_i} := \frac{10 \cdot i}{10}$ $\quad y2_{data} := \cos\left(-\frac{x2_{data}^{\,2}}{8}\right)$

$j := 0 .. 100$

$x2_{interp_j} := \frac{10 \cdot j}{100}$

$f_{cspline} := \text{cspline}\left(x2_{data}, y2_{data}\right)$

$y2_{cspline} := \text{interp}\left(f_{cspline}, x2_{data}, y2_{data}, x2_{interp}\right)$



2. Polynomial Regression

A. Explanation

- Regression or curve fitting is a little bit different than the interpolations above. For a regression, we assume the data has some non-trivial noise.
- We did polynomial regression in Excel using "Linest" and in Python using "polyfit."

- In Mathcad we will use: "polyfit" and "polyfitc"
- To find the actual polynomial, use **polyfit**:
  <best fit polynomial function> = polyfit(<x_data>, <y_data>, <order of polynomial)
- To find the coefficients in the fit, use **polyfitc:**
  <matrix of coefficients> = polyfit(<x_data>, <y_data>, <order of polynomial)

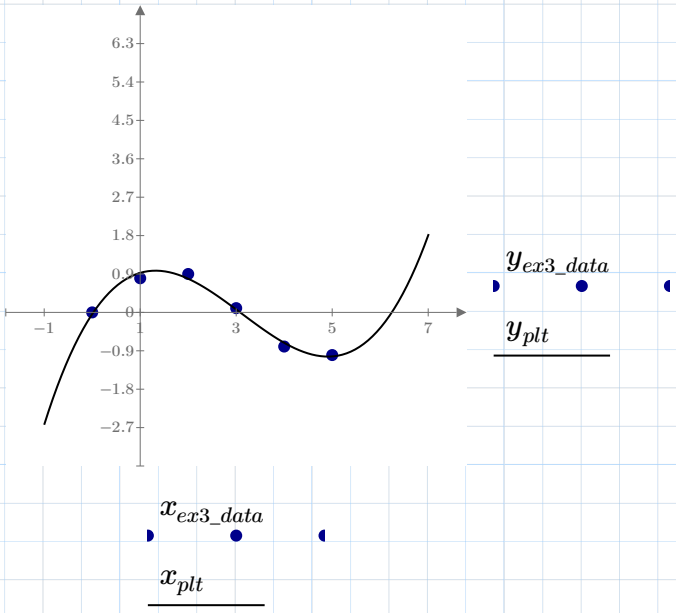
B. Example

$$x_{ex3\_data} := \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}^{T} \qquad y_{ex3\_data} := \begin{bmatrix} 0 & 0.8 & 0.9 & 0.1 & -0.8 & -1 \end{bmatrix}^{T}$$

$$f_{poly\_regress} := \text{polyfit}\left(x_{ex3\_data}, y_{ex3\_data}, 3\right)$$

$$\text{polyfitc}\left(x_{ex3\_data}, y_{ex3\_data}, 3\right) = \begin{bmatrix} \text{“Term”} & \text{“Coefficient”} & \text{“Std Error”} & \text{“95\% CI Low”} & \text{“95\% CI H} \\ \text{“Intercept”} & -0.04 & 0.138 & -0.634 & 0.554 \\ \text{“A”} & 1.693 & 0.268 & 0.539 & 2.847 \\ \text{“AA”} & -0.813 & 0.133 & -1.387 & -0.24 \\ \text{“AAA”} & 0.087 & 0.017 & 0.012 & 0.162 \end{bmatrix}$$

$$i := 0 \,..\, 100 \qquad x_{plt_i} := 9 \cdot \frac{i}{100} - 1 \qquad y_{plt_i} := f_{poly\_regress}\left(x_{plt_i}\right)$$



3. Nonlinear Regression

A. Explanation

- For a general nonlinear regression (i.e. not a polynomial) we must use a different fitting function.
- In Excel we minimized the sum of the squared error using solver, and in Python we used Scipy.optimize.curve_fit.
- In Mathcad we will use **genfit**.

B. Recipe

1. Define the function you are trying to fit with the indepent variable first, and a vector of coefficients to fit next:
   f(x, b) := fit_function(<x dep variable>, <b array of coefficents>)
2. Make an initial guess for the coefficients.
3. Call genfit:
   <fit coefficients> = genfit(<x data>, <y data>, <coefficients guess>, <fit function>)
4. (if needed) Generate the x_fit, y_fit pairs using the function defined in (1) and the coefficients obtained in (3).

C. Example - Fit the Antoine Equation to the given data.

$TP_{data} := \mathrm{READPRN}(\text{“Lec\_22–Ex\_C.dat”})$ (Read in the data)

$T_{data} := TP_{data}^{\langle 0 \rangle} \cdot \boldsymbol{K} \qquad P_{data} := TP_{data}^{\langle 1 \rangle} \cdot \boldsymbol{Pa} \qquad log\_P_{data} := \log\left(TP_{data}^{\langle 1 \rangle}\right)$

$log\_P_{sat}(T, b) := b_0 - \dfrac{b_1}{\dfrac{T}{\boldsymbol{K}} + b_2}$ (function to fit)

$b_{guess} := \begin{bmatrix} 10 \\ 1000 \\ 100 \end{bmatrix}$ (guess parameters)

$b_{fit} := \mathrm{genfit}\left(T_{data}, log\_P_{data}, b_{guess}, log\_P_{sat}\right) = \begin{bmatrix} 9.408 \\ 1.301 \cdot 10^3 \\ -33.723 \end{bmatrix}$ (get the fit)

$log\_P_{sat}(350\ \boldsymbol{K}, b_{fit}) = 5.293$ (Test it)

(Make a plot)

$i := 0 .. 500$

$T_{plot_i} := min\left(T_{data}\right) + \left(\max\left(T_{data}\right) - min\left(T_{data}\right)\right) \cdot \frac{i}{500}$

$P_{sat_i} := 10^{log\_P_{sat}\left(T_{plot_i}, b_{fit}\right)} \cdot Pa$