[MATLAB] Piecewise Curve Fitting for Any Dataset

## Exist Curve Fitting Tools Limitation

There are many existing curve fitting tools in Excel/Python/MATLAB. However, none of them could support piecewise equations curve fitting. Plus, the mathematical models are limited. It would be infeasible for you to use a complexed custom mathematical model to fit your dataset.

Here is an example. The dataset is given below:

|  |  |
| --- | --- |
| Figure 1 Original dataset | Figure 2 Original dataset in Excel |

In here, we selected 3 piecewise equation and 2nd order polynomial equation to fit the data. However, you may use any number of piecewise equations and any equation combinations you like.

The goal is to find the optimal solution for all coefficients () and breakpoints () so that the model could best fit the raw data. To express this goal in mathematical way, we need to minimize the objective function:

Boundary conditions are:

* All coefficients boundary:
* Breakpoints boundary:

Because 3 piecewise equations must be continuous at their breakpoints, that implies the values are breakpoints must be equal, and gradients at breakpoints must be equal as well. Therefore, we may write the equality constraints as:

We could derive the gradient equation for the 3-piecewise equation as below:

There is no inequality constraint in this example.

This is a very typical system optimization or machine learning question.

## Solution 1: fmincon() for local optimization

MATLAB Optimization Toolbox provides a built-in function called fmincon() designed to find a local minima solution for system optimization problems. You may select many algorithms for fmincon(), such as

* 'interior-point' (default)
* 'trust-region-reflective'
* 'sqp'
* 'sqp-legacy'
* 'active-set'

However, a local minima usually is not the most optimal solution (called global minima). This is because of the gradient descent algorithm could stuck at the local minima and stop iteration.

Please read code from GitHub: find\_best\_fit\_3piecewise\_equ\_fmincon\_local.m

## Solution 2: fmincon() for global optimization

Based on solution 1, there are some improvements to making sure that you will find a global minima solution.

Please read code from GitHub: find\_best\_fit\_3piecewise\_equ\_fmincon\_global.m

The following 2 graphs show the curve fitting result.

* Black circles: raw data
* Red curve: 1st piecewise equation
* Green curve: 2nd piecewise equation
* Blue curve: 3rd piecewise equation
* Left graph: fmincon() to get local minima solution
* Right graph: fmincon() to get global minima solution

We could visually see the global minima has better curve fitting than local minima solution.

|  |  |
| --- | --- |
| Figure 3 fmincon() for local minima | Figure 4 fmincon() for global minima |

The numerical results are given below. Note that everything you run the code, the result will be slightly different. This is due to the nature behind machine learning and system optimization algorithm, which is the probability. However, the result should NOT vary too much.

As you can see, the objective function value at global minima is much less than the local minima. It approves that the curve fitting result is better.

|  |  |  |
| --- | --- | --- |
|  | **fmincon() for local minima** | **fmincon() for global minima** |
| **Final objective function value** | 0.001538723 | 0.000137076 |
| **Equation 1** | y1 =  -0.0000224932\*x^2 + 0.0031112500\*x + 4.0187622232 | y1 =  -0.0001649649\*x^2 + 0.0070842073\*x + 3.9975295101 |
| **Equation 2** | y2 =  0.0000025373\*x^2 + 0.0000485654\*x + 4.1124484546 | y2 =  -0.0000210524\*x^2 + 0.0027668325\*x + 4.0299098208 |
| **Equation 3** | y3 =  0.0000002261\*x^2 + 0.0005712989\*x + 4.0828909146 | y3 =  0.0000004615\*x^2 + 0.0004863585\*x + 4.0903423830 |
| **Breakpoint 1** | 61.17915627 | 15 |
| **Breakpoint 2** | 113.0883872 | 53 |

# How to Use the Code

In order to find the most suitable piecewise curve fitting model, you need to

1. Decide the number of piecewise equation
2. Select the suitable mathematical model
3. Select the lower and upper boundary conditions for all coefficients and breakpoints
4. Choose the suitable step size for breakpoints. The smaller step size, more computational time.
5. Run the code with different above combinations and compare final objective function result

To see the following 4 combinations and result table:

1. 2 piecewise equation with fmincon() local minima
2. 2 piecewise equation with fmincon() global minima
3. 3 piecewise equation with fmincon() local minima
4. 3 piecewise equation with fmincon() global minima

It’s obvious to see that option D has the minimum objective function, which means the curving fitting is most accurate. Meanwhile, there is no over-fitting issue by visual inspection.

|  |  |
| --- | --- |
| Figure 6 fmincon: 2 piecewise best fit equation | Figure 7 fmincon scan: 2 piecewise best fit equation |
| Figure 3 fmincon() for local minima | Figure 4 fmincon() for global minima |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **2 piecewise**  **fmincon() for local minima** | **2 piecewise**  **fmincon() for global minima** | **3 piecewise**  **fmincon() for local minima** | **3 piecewise**  **fmincon() for global minima** |
| **Final objective function value** | 0.000678959 | 0.000522159 | 0.001538723 | 0.000137076 |
| **Equation 1** | y1 = -0.0000624627\*x^2 + 0.0049620077\*x + 4.0058089647 | y1 =  -0.0000472164\*x^2 + 0.0043732929\*x + 4.0085296766 | y1 =  -0.0000224932\*x^2 + 0.0031112500\*x + 4.0187622232 | y1 =  -0.0001649649\*x^2 + 0.0070842073\*x + 3.9975295101 |
| **Equation 2** | y2 = -0.0000000150\*x^2 + 0.0006255810\*x + 4.0810903548 | y2 =  0.0000000510\*x^2 + 0.0005919077\*x + 4.0841573823 | y2 =  0.0000025373\*x^2 + 0.0000485654\*x + 4.1124484546 | y2 =  -0.0000210524\*x^2 + 0.0027668325\*x + 4.0299098208 |
| **Equation 3** | / | / | y3 =  0.0000002261\*x^2 + 0.0005712989\*x + 4.0828909146 | y3 =  0.0000004615\*x^2 + 0.0004863585\*x + 4.0903423830 |
| **Breakpoint 1** | 34.72042092 | 40 | 61.17915627 | 15 |
| **Breakpoint 2** | / | / | 113.0883872 | 53 |