Curve Fitting Software

Malavika Srinivasan

September 30, 2018

1 Revision History

Date	Version	Notes
Sep 28, 2018	1.0	First draft by Malavika
Date 2	1.1	Notes

2 Reference Material

This section records information for ease of reference. The information includes the units, symbols and abbreviations used in this document.

2.1 Table of Units

This library is designed to work for any set of data, irrespective of their units. So, table of units is not applicable.

2.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The symbols are listed in alphabetical order. [I am just noting down symbols, they are not yet in alphabetical order —Malavika]

symbol	unit	description
A_{in}	m^2	
Φ		
$\sum_{min}^{max}$		

[ISSUE 1: For libraries, we have only mathematical symbols which do not have units. Should we remove units or just say not applicable. —Malavika] [Use your problems actual symbols. The si package is a good idea to use for units. —SS]

2.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
CA	Commonality Analysis
CFS	Curve Fitting Software
T	Theoretical Model

[Add any other abbreviations or acronyms that you add —SS]

Contents

3 Introduction

Scientific Computation (SC) is the collection of tools, techniques, and theories that are required to solve problems in the field of science and engineering using computer-based mathematical models. The source data for scientific computation problems are often large sets of data from experiments conducted in a laboratory setup. This large set of data (such as time and temperature) is usually complex to analyze and require segmenting and curve-fitting.

Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points possibly subject to constraints Wiki.

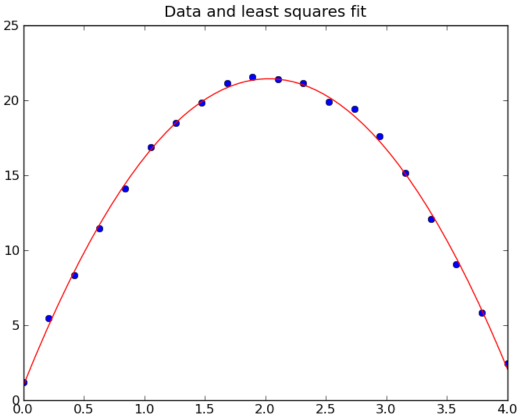

Figure 1: Typical example of a curve fitting process

A fit for the data can be obtained by different methods like interpolation, data smoothing and regression.

3.1 Interpolation

Interpolation is a process of fitting a function to given data so that the function has some values as the given data.

3.2 Smoothing

Smoothing is a process of creating an approximating function that attempts to capture important patterns in the data, while leaving out noise or other rapid phenomena.

3.3 Regression

Regression analysis is the process of finding the best fit parameters for a regression model for a given set of data points and thus obtain a curve through a set of data points.

3.4 Purpose of Document

3.5 Scope of the Family

3.6 Characteristics of Intended Reader

3.7 Organization of Document

4 General System Description

This section identifies the interfaces between the system and its environment, describes the potential user characteristics and lists the potential system constraints.

4.1 Potential System Contexts

[Your system context will likely include an explicit list of user and system responsibilities —SS]

- User Responsibilities:
  -

- CFS Responsibilities:
  - Detect data type mismatch, such as a string of characters instead of a floating point number
  -

4.2 Potential User Characteristics

The end user of CFS should have an understanding of undergraduate Level 1 Calculus and Physics.

4.3 Potential System Constraints

[You may not have any system constraints —SS]

5 Commonalities

5.1 Background Overview

5.2 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

-

5.3 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given. [Modify the examples below for your problem, and add additional definitions as appropriate. —SS]

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil (°C). T_W is the temperature of the water (°C). The heat flux out of the coil, q_C ($\mathrm{W\,m^{-2}}$), is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD**??**) is used on the surface of the coil, which has area A_C ($\mathrm{m^2}$) and heat transfer coefficient h_C ($\mathrm{W\,m^{-2}\,°C^{-1}}$). This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM**??**

5.4 Goal Statements

Given the [inputs —SS], the goal statements are:

GS1: [One sentence description of the goal. There may be more than one. Each Goal should have a meaningful label. —SS]

5.5 Theoretical Models

This section focuses on the general equations and laws that CFS is based on. [Modify the examples below for your problem, and add additional models as appropriate. —SS]

Number	T1
Label	**Conservation of thermal energy**
Equation	$-\nabla \cdot \mathbf{q} + g = \rho C \frac{\partial T}{\partial t}$
Description	The above equation gives the conservation of energy for transient heat transfer in a material of specific heat capacity C ($\mathrm{J\,kg^{-1}\,°C^{-1}}$) and density ρ ($\mathrm{kg\,m^{-3}}$), where $\mathbf{q}$ is the thermal flux vector ($\mathrm{W\,m^{-2}}$), g is the volumetric heat generation ($\mathrm{W\,m^{-3}}$), T is the temperature (°C), t is time (s), and ∇ is the gradient operator. For this equation to apply, other forms of energy, such as mechanical energy, are assumed to be negligible in the system (A**??**). In general, the material properties (ρ and C) depend on temperature.
Source	`http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm`
Ref. By	GD**??**

6 Variabilities

6.1 Assumptions

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to T, GD, DD etc., using commands like dref, ddref etc. —SS]

6.2 Calculation

6.3 Output

7 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

7.1 Functional Requirements

R1: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —SS]

R2: [It isn't always required, but often echoing the inputs as part of the output is a good idea. —SS]

R3: [Calculation related requirements. —SS]

R4: [Verification related requirements. —SS]

R5: [Output related requirements. —SS]

7.2 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —SS]

8 Likely Changes

LC1: [If there is a ranking of variabilities, or combinations of variabilities, that are more likely, this information can be included here. —SS]

9 Traceability Matrices and Graphs

[You will have to add tables. —SS]

References

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL http://www.ifi.unizh.ch/req/events/RE06/.

Wiki. Curve fitting. URL https://en.wikipedia.org/wiki/Curve_fitting.

10 Appendix

[Your report may require an appendix. For instance, this is a good point to show the values of the symbolic parameters introduced in the report. —SS]

10.1 Symbolic Parameters

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —SS]

[Points to be noted —Malavika]
[** —Malavika]
[Interpolation means fitting some function to given data so that the function has some values as the given data. —Malavika]
The Simplest 1d interpolation is given by:
for a given data (t_i, y_i), where $i = 1, 2, 3...m$,
f is an interpolating function such that,

$$f(t_i) = y_i \text{where i} = 1, 2...m$$

Some of the variabilities includes:

- What form should the function have?
- How should the function behave between data points?
- Should the function inherit properties of the data such as monotonicity, convexity or periodicity?
- Are we interested in the values of the parameters that define the interpolating function, or simply in evaluating the function at various points for plotting or other purposes?
- If the function and data are plotted, should the results be visually pleasing?

11 Selection of function

The selection of function for interpolation depends on the following factors:

- How easy is the interpolant or the function is to work with. Working means determining the parameters of the interpolant from the data, evaluating the interpolant at a given point, differentiating or integrating the interpolant.
- How well the properties of the interpolant match the properties of the data to be fit(smoothness, monotonicity, convexity, periodicity etc)

Some of the familiar functions commonly used for interpolation are:

- Polynomials
- Piecewise Polynomials
- Trignometric functions
- Exponential functions
- Rational functions

[Questions: Ask Dr.Smith if we need to have all these five variabilities for fitting data or its enough if we have piecewise and polynomial —Malavika]

To find an interpolating function for a set of data points, it is important to make sure that the interpolant exists. This comes down to the discussion of matching the number of parameters in the interpolant to number of data points to be fit.If there are too few parameters, then the interpolant does not exist. If there are too many points, then the interpolant is not unique.

For a gien set of data points (t_i, y_i), where i $= 1, 23...m$, an interpolant is chosen from a suitable set of basis functions $\Phi(t), \Phi_1(t), ...\Phi_n(t)$. Therefore, the interpolating function f can be expressed as a linear combination of these basis functions.

$$f(t) = \sum_{j=1}^{n} x_j \Phi_j(t)$$

where x_j are the parameters to be found. But inorder for f to be interpolating the data points (t_i, y_i),

$$f(t_i) = \sum_{j=1}^{n} x_j \Phi_j(t) = y_i$$

where i $= 1, 2, 3...m$ This can be expressed as a liner system of equations

$$Ax = y$$

where A is the basis matrix whose m X n entries are given by,

$$a_{ij} = \Phi_j(t_i)$$

where a_{ij} is the value of j^{th} basis function at i^{th} data point.

11.1 For my understanding

[Important points to be noted here —Malavika]

- m data points (t_i, y_i) where $i = 1....m$
- n basis function $\Phi_1(t), \Phi_2(t), \Phi_3(t)...\Phi_n(t)$
- f is the interpolating function
- f is the linear combination of basis function as shown below
- $$f(t_i) = \sum_{j=1}^{n} x_j \Phi_j(t) = y_i$$

  Where i $= 1, 2..m$. Here they consider m because there are m data points
- x_j is the parameters of the fit to be found
- To find x_j, we use matrix solving method because the above form is a system of linear equations.

  $$Ax = y$$

  Where A is the basis matrix whose entries are given by a_{ij}
- a_{ij} is given by the following equation

  $$a_{ij} = \Phi_j(ti)$$

11.2 Polynomial interpolation

Polynomial function has different types of basis functions.

- Monomial
- Lagrange
- Newton
- Orthogonal
- Interpolating continuous functions

[I have worked out only monomial, lagrange and newton —Malavika] [Need to ask if Orthogonal and continous function can be left out —Malavika]

11.2.1 Monomial basis

To interpolate n data points (t_i, y_i), we choose $k = n - 1$ as the maximum degree of the polynomial. We define P_{n-1} as the set of polynomials of degree at most n-1 and is composed on first n monomials Φ_j. [Note: n data points, n basis function, max degree of polynomial = n-1 —Malavika]

$$\Phi_j(t) = t^{j-1}$$

where j $= 1, 2, ...n$
We can construct the polynomial from this which will have the form,

$$P_{n-1}(t) = x_1 + x_2 t + x_3 t^2 + x_4 t^3 + ...x_n t^{n-1}$$

The matrix form is given by, [To understand this, see in class notes. Example problem solved —Malavika]

$$Ax = \begin{bmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \dots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & t_n & t_n^2 & \dots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = y$$

[Note: A matrix whose columns are successive powers of some independent variable t is called Vandermonde matrix —Malavika] [Assumption: The vandermonde matrix should be non singular provided t_i are all distint, hence the interpolant exists. —Malavika] [Cost of computing interpolant is high —Malavika]

11.2.2 Lagrange Interpolation

For a given set of data points (t_i, y_i), $i = 1, 2, \dots$n, the Lagrange polynomial of degree n-1, P_{n-1} is given by,

$$P_{n-1}(t) = y_1 l_1(t) + y_2 l_2(t) + \dots y_n l_n(t).$$

where the basis functions(also called fundamental polynomials) for the l_j is given by,

$$l_j(t) = \frac{\Pi_{k=1,k\neq j}^{n}(t - t_k)}{\Pi_{k=1,k\neq j}^{n}(t_j - t_k)} = \frac{(t - t_1)}{(t_j - t_1)} \dots \frac{(t - t_{j-1})}{(t_j - t_{j-1})} \frac{(t - t_{j+1})}{(t_j - t_{j+1})} \dots \frac{(t - t_n)}{(t_j - t_n)}$$

where j $=1, 2, ...n$
The matrix form,

$$Ax = y$$

A is the Identity matrix I. [Expensive to evaluate and more difficult to differentiate and integrate —Malavika] [n data points, n basis function, n-1 is the maximum degree of polynomials. Refer to notes for example problem. —Malavika]

11.2.3 Newton Interpolation

For a given set of data points (t_i, y_i), $i = 1, 2, \ldots$n, the newton basis function for P_{n-1} is given by, [n data points, n basis points —Malavika]

$$\pi(t) = \Pi_{k=1}^{j-1}(t - t_k)$$

where j = 1,2,....n
When the limits make it vacuous, the product is taken to be 1
Also, $\pi_j(t) = 0$ for i¡j. The polynomial P_{n-1} is given by,

$$P_{n-1}(t) = x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2) + ...x_n(t - t_1)(t - t_2)...(t - t_{n-1})$$

The matrix form of basis in $Ax = y$ is lower triangular as $\pi_j(t) = 0$ for i¡j with its entries defined by $a_{ij} = \pi_j(t_i)$

11.3 Piecewise Polynomial Interpolation

Even though choosing an appropriate basis function and intetpolation points mitigate the difficulties associated with interpolation by a polynomial of higher degree, fitting a single polynomial to a large number of data points will still yield unsatisfactory oscillating behaviour in the interpolant. [I think this has reference to my thesis, because we chose Piecewise polynomials for data fitting. I need to add this to thesis as a place holder —Malavika] Piecewise polynomial provides an alternative to the difficulties associated with higher order polynomial interpolation. The main advantage is that a large number of data ppints can be fit with a low degree poynomials.

In piecewise polynomial interpolation of a given set of points (t_i, y_i), i $= 1, 2, ...n$, with $t_1 < t_2 < t_3...t_n$, a different polynomial is used in each subinterval $[t_i, t_{i+1}]$.The points at which the interpolant changes from one polynomial to another is called breakpoints or knots or control points.

11.3.1 Hermite Cubic interpolation

Hermite interpolation matches an unknown function both in observed value, and the observed value of its first m derivatives. This means that n(m + 1) values must be known, rather than just the first n values required for Newton interpolation.

Data to be known is given by,

(x_0, y_0),	(x_1, y_1),	(x_2, y_2),	$\ldots$,	x_{n-1}, y_{n-1}
$(x_0, y_0^{'})$,	$(x_1, y_1^{'})$,	$(x_2, y_2^{'})$,	$\ldots$,	$x_{n-1}, y_{n-1}^{'}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$(x_0, y_0^{'})$,	$(x_1, y_1^{'})$,	$(x_2, y_2^{'})$,	$\ldots$,	$x_{n-1}, y_{n-1}^{'}$,

The resulting polynomial may have degree at most $n(m + 1) - 1$.

In a simple case, using a divided difference to calculate hermite polynomial f, the first step is to copy each point m times.For a simple case $m = 1$ for all points, which means for the given n+1 data points, all the points have m derivatives (here m = 1).

This means data is of the form,

(x_0, y_0),	(x_1, y_1),	(x_2, y_2),	$\ldots$,	x_n, y_n
$(x_0, y_0^{'})$,	$(x_1, y_1^{'})$,	$(x_2, y_2^{'})$,	$\ldots$,	$x_n, y_n^{'}$

Note:
$y_0, y_1, y_2 \ldots y_n = f(x_0), f(x_1), f(x_2) \ldots f(x_n)$
$y_0^{'}, y_1^{'}, y_2^{'} \ldots y_n^{'} = f^{'}(x_0), f^{'}(x_1), f^{'}(x_2) \ldots f^{'}(x_n)$

To find:
f which is an interpolating polynomial

Steps to find:

- Create a new data set $z_0, z_1, z_2, z_3 \ldots z_{2n+1}$ such that

  $$z_{2i} = z_{2i+1} = x_i$$

  Which means at i = 0, $z_0 = z_1 = x_0$ Here we have only z_0, z_1 bacause we assumed m = 1, so each element xi repeats in z data set 2 times.
  For a general case, where we have k derivatives for each point, the data set z_0, z_1, z_2, z_N will contain k identical copies of x_i.

- Create a divided difference table for the data set $z_0, z_1, z_2, z_3 \ldots z_{2n+1}$. For the general case, $z_0, z_1, z_2, z_3 \ldots z_N$
  However, When creating the table, divided differences of $j = 2, 3, \ldots$ k identical values will be calculated as, $\frac{f^{(j)}(x_i)}{x!}$

- Then we generate the polynomial by taking the coefficients from the diagonal of the divided difference table, and multiplying the kth coefficient by $\Pi_{i=0}^{k-1}(x - z_i)$.

11.3.2 Cubic spline interpolation

[This notes is in bookmark named NOTES FOR CUBIC SPLINE —Malavika]
A spline is a piecewise polynomial of degree k that is continuously differentiable $k - 1$ times. For a cubic spline, $k = 3$ so it is continuously differentiable 2 times.

Let there be a set of points,

K $= x_0, x_1 ..., x_m$ where

$$a = x_0 < x_1 < x_2 < x_3 ... x_m = b$$

A function S $\in C^2[a, b]$[C2 denotes first and second derivative —Malavika] is called a cubic spline if S is a cubic polynomial S_i in each interval $[x_i, x_{i+1}]$. It is a cubic interpolating spline if $s(x_i) = y_i$ for given values of y_i.

The ansatz of m piecewise polynomials is given by:

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + di$$

For each s_i, we have 4 coefficients to be fixed a_i, b_i, c_i and d_i.
so for m polynomials $(0 to m - 1)$, we need to fix $4m$ coefficients to fix the spline.

To fix these coefficients, we need 4m conditions [similar to how we need 2 equations to solve for 2 variables —Malavika]
The conditions are:

$$S_i(x_i) = Y_i....i = 0, 1, 2...m - 1 \tag{1}$$

$$S_{m-1} = Y_m \tag{2}$$

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}).....i = 0, 1, 2...m - 2 \tag{3}$$

$$S_i^{'}(x_{i+1}) = S_{i+1}^{'}(x_{i+1}).....i = 0, 1, 2...m - 2 \tag{4}$$

$$S_i^{''}(x_{i+1}) = S_{i+1}^{''}(x_{i+1}).....i = 0, 1, 2...m - 2 \tag{5}$$

These are $4m - 2$ conditions. We need 2 more for the spline to be unique which has several alternatives.

- Natural splines $S_0^{''}(x_0) = 0$ and $S_{m-1}^{''}(x_m) = 0$
- End slope spline $S_0^{'}(x_0) = Y_0^{'}$ and $S_{m-1}^{'}(x_m) = Y_m^{'}$
- Periodic spline $S_0^{'}(x_0) = S_{m-1}^{'}(x_m)$ and $S_0^{''}(x_0) = S_{m-1}^{''}(x_m)$.
- Not-a-Knot spline $S_0^{'''}(x_1) = S_1^{'''}(x_1)$ and $S_{m-2}^{'''}(x_{m-1}) = S_{m-1}^{'''}(x_{m-1})$

[If you want visually pleasing results, then choose Hermite cubic interpolation but if smoothness is important then cubic splines are better —Malavika]

Interpolating the given data and requiring continuity of first derivative imposes $3n - 4$ constraints on cubic spline.Requiring continuity at second derivative imposes n-2 additional constraints which leaves only 2 free parameters. The final two parameters can be fixed by floolowing ways:

-

11.3.3 B-splines

This allows the representation of arbitrary splines as a linear combination of basis functions. They can be defined in a number of ways like recursion, convolution and divided differences. For given set of knots or data points $..t_{-2} < t_{-1} < t_0 < t_1 < t_2 < ...$

Assumptions for notational convinience:

- We use recursion to define bsplines
- Assume infinite set of knots, even though in practice we have only finite set of knots
- Linear functions is assumed for notational convinience

$$v_i^k = \frac{t - t_i}{t_{i+k} - t_i}$$

To start the recursion we start with B-splines of degree 0 by:

$$B_i^0(t) = \begin{cases} 1, & \text{if } t_i \leq t < t_{i+1}. \\ 0, & \text{otherwise.} \end{cases}$$

For B-splines of degree k, $k > 0$,

$$B_i^k(t) = v_i^k(t)B_i^{k-1}(t) + (1 - v_{i+1}^k(t))B_{i+1}^{k-1}(t)$$

In general, B_i^k is a piecewise polynomial of degree = k

[** —Malavika]

[** —Malavika]

[** —Malavika]

[Only linear least squares, no non linear. This can go into assumptions. —Malavika]

12 Least Squares

12.1 Assumptions

- Only linear, no non linear.

The general form is given by,

$$Ax \approxeq b$$

where the approximation is understood to be in 2-norm or least squares sense.

12.2 Curve fitting

Given:
m data points, (t_i, y_i) where i $= 1, 2, 3...m$
To find:
Find n-vector x that it gives the best fit to th data by the model function f(t,x)
Best fit is defined by:

$$\min_x \sum_{i=1}^{m} (y_i - f(t_i, x))^2$$

Assuming linear data, which means f is linear in parameter x, hence f is a linear combination of Φ_j that depend only on t.

$$f(t, x) = x_1\Phi_1(t) + x_2\Phi_2(t) + ... + x_n\Phi_n(t)$$

Where Φ_j can be replaced by t^{j-1}, so the equation becomes,

$$f(t, x) = x1 + x2t + x_3t^2 + ... + x_nt^{n-1}$$

The matrix form is given by,

$$Ax \approxeq b$$

Where the entries of the matrix a_{ij} is defined by $a_{ij} = \Phi_j(t_i)$ and m vector b is given by components $b_i = y_i$.

Existence and uniqueness The solution to a mXn least squares problem $Ax \approxeq b$ is unique if and only if A has full column rank that is $rank(A) = n$. If $rank(A) < n$, then A is said to be rank deficient. Even though a solution must still exist, it is not unique.