

# **Sigmoidal Curve Fitting with NonlinLeastSquares**

**IMSL Technical Report P10406**

**A White Paper by Visual Numerics, Inc.**  
March 2007

Visual Numerics<sup>®</sup>

Visual Numerics, Inc.  
2500 Wilcrest Dr., Suite 200  
Houston, TX 77042  
USA  
[www.vni.com](http://www.vni.com)

# Sigmoidal Curve Fitting with NonlinLeastSquares

IMSL Technical Report P10406

**by Visual Numerics, Inc.**

Copyright 2007 by Visual Numerics, Inc. All Rights Reserved  
Printed in the United States of America

**Publishing History:**

March 2007

---

## Trademark Information

Visual Numerics, IMSL and PV-WAVE are registered trademarks. JMSL TS-WAVE, and JWAVE are trademarks of Visual Numerics, Inc., in the U.S. and other countries. All other product and company names are trademarks or registered trademarks of their respective owners.

The information contained in this document is subject to change without notice. Visual Numerics, Inc. makes no warranty of any kind with regard to this material, included, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Visual Numerics, Inc. shall not be liable for errors contained herein or for incidental, consequential, or other indirect damages in connection with the furnishing, performance, or use of this material.

## Introduction

This report describes how to use the `NonlinLeastSquares` class in the JMSL™ Numerical Library or the IMSL® C# Numerical Library to do curve fitting. The applied example is a nonlinear function called a sigmoidal curve, but can be applied to any arbitrary user-defined function. The code examples are in Java and call the JMSL Library, but can be applied by .NET programmers using the C#.NET version of the IMSL Library.

## Overview

The goal is to fit a model of the form

$$y = A_b + \frac{A_t - A_b}{1 + e^{-\left(\frac{x-x_0}{w}\right)}}$$

to some experimental data.

The solution method is to define a `NonlinLeastSquares.Function` that accepts an array of parameters to be fit (of length four for this example) and an array of returned values. For this example, this user-supplied function in Java is:

```
NonlinLeastSquares.Function zsf = new NonlinLeastSquares.Function() {
    public void f(double theta[], double f[]) {
        // theta[0] = at
        // theta[1] = ab
        // theta[2] = x0
        // theta[3] = w
        for (int i=0; i<ox.length; i++) {
            f[i] = (theta[1] + (theta[0] - theta[1])/
                (1.0 + Math.exp(- (ox[i] - theta[2])/theta[3]))) - oy[i];
        }
    }
};
```

where `ox[]` and `oy[]` are arrays containing the observed data.

Next, one creates a `NonlinLeastSquares` object, passing in the number of observed data points and the number of parameters to solve for. An initial guess is set using the `setGuess()` method, and then the `solve()` method is called using the `NonlinLeastSquares.Function` defined previously. The `solve` method returns an

array of the parameters for the supplied function that best fit the data, evaluated using a modified Levenberg-Marquardt method. The Java code for this section is below:

```
params = new double[n];
double xguess[] = {0, 300, 50.0, 10.0};

NonlinLeastSquares zs = new NonlinLeastSquares(m,n);
zs.setGuess(xguess);
try {
    params = zs.solve(zsf);
} catch (NonlinLeastSquares.TooManyIterationsException tmie) {}
```

Note that the `solve()` method may throw an exception that one would want to deal with appropriately, instead of ignoring it as is done in this example.

## An Applied Example

The test data is historical sunflower growth data from Reed, H. S. and Holland, R. H. (1919), Growth of sunflower seeds, *Proceedings of the National Academy of Sciences*, 5, p. 140. The data contains weekly observations of the height of a sunflower and are shown below in Table 1.

**Table 1. Sunflower growth observations.**

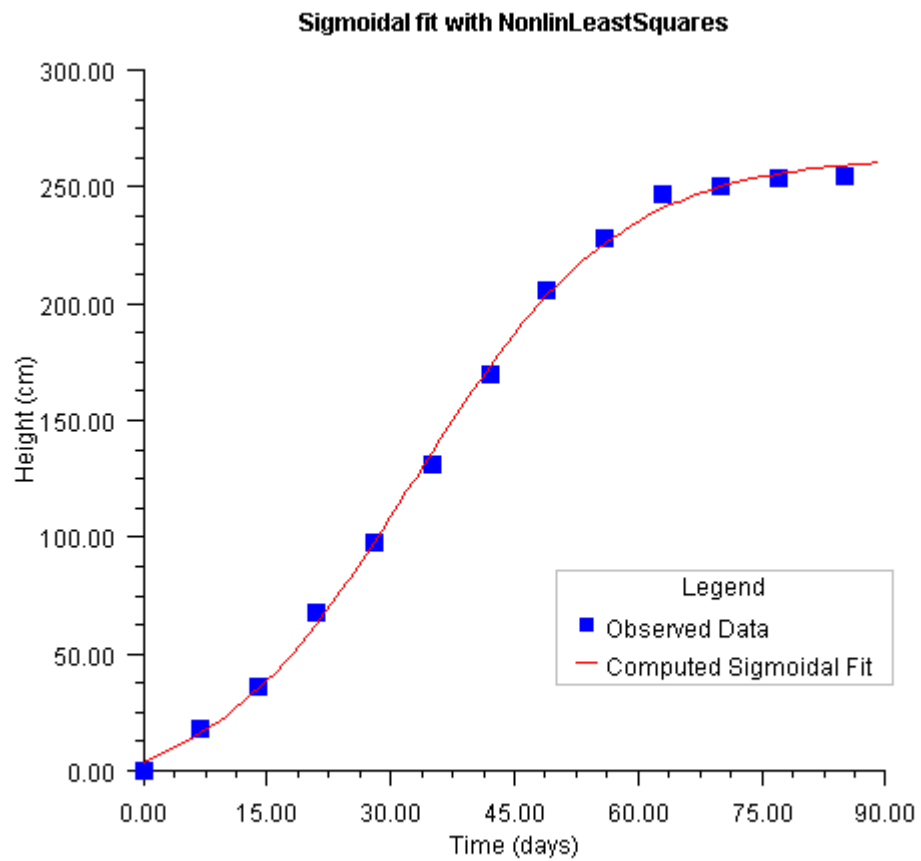
day	height, cm
0	0.00
7	17.93
14	36.36
21	67.76
28	98.1
35	131
42	169.5
49	205.5
56	228.3
63	247.1
70	250.5
77	253.8
84	254.5

These data are hard coded into an example Java application that extends `com.imsl.chart.JFrameChart` to allow easy display of the results. The data are analyzed and the resulting parameter estimates are shown below in Table 2.

**Table 2. Computed parameters for the sigmoidal model.**

Parameter	Value
$A_t$	263.32981685253367
$A_b$	-14.865203610256076
$x_o$	32.82338522946577
$w$	12.463345262315638

Finally, Figure 1 shows the original data and the model fit graphically.



**Figure 1. Observed data and model fit.**

## Complete source code for SigmoidEx.java

```
import com.imsl.math.*;
import com.imsl.chart.*;

public class SigmoidEx extends JFrameChart {
    private final int m = 13;    // # data points
    private final int n = 4;    // # parameters
    private double[] params;    // holds the results of the n parameters
    private final double ox[] = {0,7,14,21,28,35,42,49,56,63,70,77,85};
    private final double oy[] = {0.00, 17.93, 36.36, 67.67, 98.10, 131.00,
        169.50, 205.50, 228.30, 247.10, 250.50, 253.80, 254.50};

    public SigmoidEx() {
        compute();
        draw();
    }

    public void compute() {

        NonlinLeastSquares.Function zsf = new NonlinLeastSquares.Function() {
            public void f(double theta[], double f[]) {
                // growth rate of a sunflower (Reed and Holland, 1919)
                // Reference: Reed, H. S. and Holland, R. H. (1919),
                // Growth of sunflower seeds; Proceedings of the National
                // Academy of Sciences, volume 5, p. 140.
                // theta[0] = at
                // theta[1] = ab
                // theta[2] = x0
                // theta[3] = w
                for (int i=0; i<ox.length; i++) {
                    f[i] = (theta[1] + (theta[0] - theta[1])/
                        (1.0 + Math.exp(- (ox[i] - theta[2])/theta[3]))) -
                        oy[i];
                }
            }
        };

        params = new double[n];
        double xguess[] = {0, 300, 50.0, 10.0};

        NonlinLeastSquares zs = new NonlinLeastSquares(m,n);
        zs.setGuess(xguess);
        try {
            params = zs.solve(zsf);
        } catch (NonlinLeastSquares.TooManyIterationsException tmie) {}
    }

    public void draw() {
        for (int i=0; i<n; i++)
            System.out.println("params[" + i + "] = " + params[i]);

        final int len = 90;
        double y[] = new double[len];
    }
}
```

```

double xx[] = new double[len];

for (int i=0; i<len; i++) {
    xx[i] = (double)i;
    y[i] = (params[1] + (params[0] - params[1])/
        (1.0 + Math.exp(- (xx[i] - params[2])/params[3])) ) ;
}
Chart chart = getChart();
chart.getChartTitle().setFontStyle(1);
chart.getChartTitle().setTitle("Sigmoidal fit with
    NonlinLeastSquares");
AxisXY axis = new AxisXY(chart);
axis.getAxisX().getAxisTitle().setTitle("Time (days)");
axis.getAxisY().getAxisTitle().setTitle("Height (cm)");

Legend legend = chart.getLegend();
legend.setPaint(true);
legend.setTitle("Legend");
legend.setViewport(0.55, 0.65, 0.6, 0.7);

Data obs = new Data(axis, ox, oy);
obs.setDataType(Data.DATA_TYPE_MARKER);
obs.setMarkerType(Data.MARKER_TYPE_FILLED_SQUARE);
obs.setMarkerColor(java.awt.Color.blue);
obs.setTitle("Observed Data");

Data fit = new Data(axis, xx, y);
fit.setDataType(Data.DATA_TYPE_LINE);
fit.setLineColor(java.awt.Color.red);
fit.setTitle("Computed Sigmoidal Fit");
}

public static void main(String argv[]) {
    new SigmoidEx().setVisible(true);
}
}

```