SOLVING NONLINEAR LEAST-SQUARES PROBLEMS WITH THE GAUSS-NEWTON AND LEVENBERG-MARQUARDT METHODS

ALFONSO CROEZE, LINDSEY PITTMAN, AND WINNIE REYNOLDS

Abstract. We will analyze two methods of optimizing least-squares problems; the Gauss-Newton Method and the Levenberg Marquardt Algorithm. In order to compare the two methods, we will give an explanation of each methods' steps, as well as show examples of two different function types. The advantages and disadvantages will then be explored for both methods.

1. Introduction

1.1. **Overview.** An optimization problem begins with a set of independent variables and often includes conditions or restrictions that define acceptable values of the variables. Such restrictions are called *restraints*. The essential component is the objective function, which depends in some way on the variables. The solution of an optimization problem is a set of allowed values of the variables for which the objective function assumes an optimal value. In mathematical terms, optimization usually involves maximizing or minimizing; for example, maximizing profit or minimizing cost. In a large number of practical problems, the objective function $f(x)$ is a sum of squares of nonlinear functions

$$f(x) = \frac{1}{2}\sum_{j=1}^{m}(r_j(x))^2 = \frac{1}{2}||r(x)||_2^2$$

that needs to be minimized. We consider the following problem

$$\min_x f(x) = \sum_{j=1}^{m}(r_j(x))^2.$$

This is a nonlinear least squares unconstrained minimization problem. It is called *least squares* because we are minimizing the sum of squares of these functions. Problems of this type occur when fitting model functions to data: if $\phi(x;t)$ represents the model function with t as an independent variable, then each $r_j(x) = \phi(x;t_j) - y_j$, where $d(t_j, y_j)$ is the given set of data points. Two common algorithms for solving such least-squares problems are the Gauss-Newton (GN) Method and the Levenberg-Marquardt Algorithm (LMA).

1.2. **Terminology.** The *gradient* ∇ of a multivariable function f is a vector consisting of the function's partial derivatives:

$$\nabla f(x_1, x_2) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right).$$

The *Hessian matrix* $H(f)$ of a function $f(x)$ is the square matrix of second-order partial derivatives of $r(x)$:

$$H(f(x_1, x_2)) = \begin{pmatrix} \dfrac{\partial f}{\partial x_1^2} & \dfrac{\partial f}{\partial x_1 \partial x_2} \\ \dfrac{\partial f}{\partial x_1 \partial x_2} & \dfrac{\partial f}{\partial x_2^2} \end{pmatrix}.$$

The *transpose* $A^\top$ of a matrix A is the matrix created by reflecting A over its main diagonal:

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^\top = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Lastly, a matrix A is *positive-definite* if, for all real non-zero vectors z, $z^\top Az > 0$. Equivalently, A is positive-definite if every upper-left submatrix of A, including A itself, has a positive determinant.

1.3. **Newton's Method.** Newton's method is an algorithm for locating roots that serves as the basis for the GN method. It is derived from the Taylor series expansion of a function $f(x)$ at a point $x = x_0 + \delta$:

$$f(x_0 + \delta) = f(x_0) + f'(x_0)\delta + \frac{1}{2} f''(x_0)\delta^2 + \ldots.$$

Newton's method uses the first-order approximation

$$f(x_0 + \delta) \approx f(x_0) + f'(x_0)\delta,$$

which is the equation of the tangent line to the curve at an initial guess x_0. The point where this tangent intersects the x-axis will be the next guess x_1 and is given by

$$x_1 = x_0 + \delta_0 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{1.1}$$

.

Thus, Newton's method is the iterative process

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Newton's method can be used to approximate the roots of a differentiable function provided that the initial guess is reasonably close to the true root and the function's derivative is not zero or very small in the neighborhood of the root. A slightly different version of Newton's method can be used to find the extreme points of a function rather than its roots:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

This formulation of Newton's method serves as the basis of the Gauss-Newton Method.

2. Least-Squares Problems

Least-Squares problems minimize the difference between a set of data and a model function that approximates this data. Given a set of data $d(t_j, y_j)$ and a model function $\phi(x; t_j)$, we obtain the difference of the functions with the equation $r_j(x) = \phi(x; t_j) - y_j$, where y_j is y component of the data point at t_j. The objective function of least-squares problems is therefore

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x). \tag{2.1}$$

By minimizing $f(x)$, we can find the parameters that most accurately match the model to the observed data. Each r_j is called a *residual* and is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}$. This equation includes the sum of all components r_j of the *residual vector* r of m components given by the vector

$$r(x) = (r_1(x), r_2(x), ..., r_m(x))^T$$

.
We can then rewrite (2.1) using the residual vector: $f(x) = \frac{1}{2}||r(x)||_2^2$. When calculating the gradient of $f(x)$, it is necessary to find the gradient of the residual vector. The *Jacobian* $J(x)$ is a matrix of all $\nabla r_j(x)$:

$$J(x) = \left[\frac{\partial r_j}{\partial x_i}\right]_{j=1,\ldots,m;i=1,\ldots,n} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix}$$

The gradient and Hessian of the objective can be expressed in terms of the Jacobian:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^T r(x) \tag{2.2}$$

$$\begin{aligned} \nabla^2 f(x) &= \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) \\ &= J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) \end{aligned} \tag{2.3}$$

In many cases the gradient of each residual r_j is relatively easy to calculate, making the Jacobian a convenient substitution. $J(x)$ and $J(x)^T$ also comprise the first term of the Hessian matrix as seen in (2.3). The Hessian matrix must be positive definite for all least-squares problems. In cases where the residual is extremely close to the solution, $\nabla^2 f(x)$ can be approximated by the first term, thus eliminating a rather lengthy calculation of $\nabla^2 r(x)$ in the second term. This approximation is used for both the Gauss-Newton and Levenberg-Marquardt methods.

3. The Gauss-Newton Method

The Gauss-Newton method is based on the basic equation from Newton's method (1.1), except that it uses a search direction vector p_k^{GN} and a step size α_k in the revised equation

$$x_{k+1} = x_k + \alpha_k p_k. \tag{3.1}$$

The values that are being altered in this case are the variables of the model function $\phi(x; t_j)$. Like Newton's method, GN is an iterative process, repeating equation (3.1) until the model function fits the data points satisfactorily. Using $\nabla f(x)$ as $f'(x)$ and the Hessian $\nabla^2 f(x)$ as $f''(x)$, this method generalizes Newton's method for multiple dimensions. It also uses a different approximation of $\nabla^2 f(x)$,

$$\nabla^2 f(x) \approx J(x)^T J(x),$$

which eliminates the second term of the general equation (2.3) involving the sum of the residual Hessians $\nabla^2 r_j(x)$, for $j = 1, 2, ..., m$. This often saves a considerable amount of computational time. The gradient equation remains the same as the general equation involving $J(x)$ in (2.2). These approximations can be used to find the search direction p_k^{GN} for each iteration with the equation

$$J_k^T J_k p_k^{GN} = -J_k^T r_k. \tag{3.2}$$

Once we find p_k^{GN} we must find the step length α_k for each iteration. We do this by minimizing $\phi(x_k + \alpha_k p_k^{GN})$; we minimize the model function $\phi(x)$ at the new point $x_k + \alpha_k p_k$ with respect to the variable α_k (where $\alpha_k > 0$). To find the optimal step length, we find α_k that satisfies the condition

$$\phi(x_k + \alpha_k p_k^{GN}) \leq \phi(x_k) + c_1 \alpha_k \nabla \phi(x_k)^T p_k^{GN}. \tag{3.3}$$

This method works for any function $\phi(x)$ that is bounded below, with any constant c_1 that satisfies the condition $0 < c_1 < 1$. This is an iterative process where we choose an initial guess for α_k, plug it into (3.3), and alter it until the inequality is satisfied. There are other methods with more constraints that find even more optimal values of α_k. These are explored in [4].

Gauss-Newton has the useful characteristic that when J_k has full rank and the gradient ∇f_k is non-zero, p_k^{GN} is a descent direction and thus a suitable direction for a line search. However, when $J_k^T p_k^{GN} = 0$, we know $J_k^T r_k = \nabla f_k = 0$, which indicates that x_k is a stationary point and no further iterations are necessary. A restriction on this method is the requirement that J_k must have full rank. GN is notable for its fast convergence close to the solution, but like Newton's method, its efficiency depends on having an accurate initial guess.

4. Gradient Descent

Gradient descent, also known as steepest descent, is an optimization method which involves taking steps proportional to the negative gradient of the function at the current point. Its iteration scheme is

$$x_{k+1} = x_k - \lambda_k \nabla f(x_k).$$

This method will not be discussed in detail here, but it is worth noting that it performs in a manner opposite to that of Gauss-Newton: gradient descent will quickly approach the solution from a distance, but its convergence will become very slow when it is close to the solution.

5. The Levenberg-Marquardt Method

Another method commonly used to minimize least-squares problems is the Levenberg-Marquardt method. This method uses the same approximation for the Hessian matrix as Gauss-Newton but implements a trust region strategy instead of a line search technique. At each iteration we must minimize p_k in the equation

$$\frac{1}{2}||J_k p_k + r_k||^2, \qquad \text{subject to } ||p_k|| \leq \Delta_k, \tag{5.1}$$

where $\Delta_k > 0$ is the trust region radius, making a spherical trust region. The value for Δ_k is chosen for each iteration depending on the error value of the corresponding p_k. Given a step p_k we then consider the ratio

$$\rho_k = \frac{d(x_k) - d(x_k + p_k)}{\phi_k(0) - \phi_k(p_k)}$$

which is a comparison of the *actual reduction* in the numerator and the *predicted reduction* in the denominator. The *actual reduction* gives the difference of values in the set of data $d(x)$ at points x_k and $x_k + p_k$, while the *predicted reduction* gives the difference of values in the model function ϕ_k at x_k (where the direction vector is 0) and at $x_k + p_k$ (after taking step p_k). If ρ_k is close to 1, the model is an accurate approximation of the data and it is safe to expand the trust-region radius Δ_k. If ρ_k is positive but significantly smaller than 1 (meaning both the model and the objective functions are decreasing, but the model function is decreasing faster than the objective function), we can keep our value of Δ_k. If ρ_k is close to zero or negative, we must reduce Δ_k at the next iteration. Once a good Δ_k is found we can solve for p_k.

When the search direction p_k^{GN} of the GN method lies inside this trust region (when $||p_k^{GN}|| \leq \Delta_k$), then p_k^{GN} can be used to minimize (5.1). If p_k^{GN} does not lie inside the trust region, we use the following equation to find a new search direction vector p_k^{LM} that lies inside the trust region ($||p_k^{LM}|| = \Delta_k$).

$$(J_k^T J_k + \lambda I)p_k^{LM} = -J_k^T r_k, \text{ where } \lambda > 0. \tag{5.2}$$

In order to find an appropriate λ for this equation, we reorder the variables to make it a function of λ:

$$p_k(\lambda) = -(J_k^T J_k + \lambda I)^{-1} J(x)^T r_k$$

Since we are trying to find a value of p_k that satisfies the equality $||p_k^{LM}|| = \Delta_k$, we want to find the norm of both sides of the equation and set it equal to our established value of Δ_k. This is a complicated process that involves finding a new equation of diagonal and orthogonal matrices that are equivalent to $\nabla^2 f(x)$ and $\nabla f(x)$ and whose norms can be easily evaluated. For more information of this method refer to [4]. For simplification, we will assume a value for λ and alter it based on error of previous iterations. Typically λ_1 is chosen to be small. If after the first iteration of (5.2) $||p_k|| > \Delta$, then λ_2 is chosen to be less than λ_1. Likewise, if $||p_k|| < \Delta$, then λ_2 is chosen to be greater than λ_1. After finding p_k^{LM} we will substitute it into (3.1) and update our values for the variables of the model function. We do not need to worry about the step size α_k, since $||p_k||$ is already established by the trust region.

LMA is a superior method to GN when $f(x)$ is large, because we account for the ommitted value of the Hessian with λ. In effect, LMA uses a gradient-descent method when $f(x)$ is large, and switches to an altered GN method implementing a trust-search region when $f(x)$ is small. This trust-search region technique is more accurate, but often takes more iterations when it is far away from the minimum. LMA is also especially useful when J_k is rank deficient, in which case GN would not be effective.

6. Example: Exponential Data

Here is a set of data for the United States population (in millions) and the corresponding year [5]:

Year	Population
1815	8.3
1825	11.0
1835	14.7
1845	19.7
1855	26.7
1865	35.2
1875	44.4
1885	55.9

This is the set of data points $d(t_j, y_j)$ where t_j is the year and y_j is the population. For the sake of convenience, year 1815 will be labelled as 1, 1825 as 2, etc. The model function is $\phi(x;t) = x_1 e^{x_2 t}$. Here is a graph of the data points and the model with $x_1 = 6$ and $x_2 = .3$ as our initial guesses:

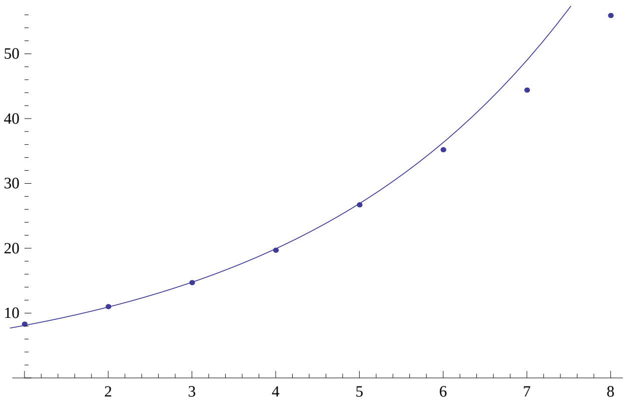

FIGURE 1. The exponential data and first approximation of the model function.

Our residual vector is a vector of the components $r_j = 6e^{.3t} - y_j$.

$$r(x) = \begin{pmatrix} 6e^{.3(1)} - 8.3 \\ 6e^{.3(2)} - 11 \\ 6e^{.3(3)} - 14.7 \\ 6e^{.3(4)} - 19.7 \\ 6e^{.3(5)} - 26.7 \\ 6e^{.3(6)} - 35.2 \\ 6e^{.3(7)} - 44.4 \\ 6e^{.3(8)} - 55.9 \end{pmatrix} = \begin{pmatrix} -0.200847 \\ -0.0672872 \\ 0.0576187 \\ 0.220702 \\ 0.190134 \\ 1.09788 \\ 4.59702 \\ 10.2391 \end{pmatrix}$$

The goal is to minimize the least-squares problem, $f(x) = \frac{1}{2}||r(x)||_2^2$. We find $||r(x)||^2$ by adding the squares of all the entries in $r(x)$. Our result is $||r(x)||^2 = 127.309$. Therefore $f(x) = 63.6545$.
The Jacobian is necessary for both the Gauss-Newton and Levenberg-Marquardt methods. Recall that the Jacobian is a matrix of all the components of $\nabla r(x)$. We compute each $\nabla r_j(x)$ by finding the gradient with respect to x_1 and x_2 and then substituting the original values for x_1 and x_2.

$$J(x) = \begin{pmatrix} e^{x_2} & e^{x_2}x_1 \\ e^{2x_2} & 2e^{2x_2}x_1 \\ e^{3x_2} & 3e^{3x_2}x_1 \\ e^{4x_2} & 4e^{4x_2}x_1 \\ e^{5x_2} & 5e^{5x_2}x_1 \\ e^{6x_2} & 6e^{6x_2}x_1 \\ e^{7x_2} & 7e^{7x_2}x_1 \\ e^{8x_2} & 8e^{8x_2}x_1 \end{pmatrix} = \begin{pmatrix} 1.34986 & 8.09915 \\ 1.82212 & 21.8654 \\ 2.4596 & 44.2729 \\ 3.32012 & 79.6828 \\ 4.48169 & 6010.1 \\ 6.04965 & 217.787 \\ 8.16617 & 342.979 \\ 11.0232 & 529.112 \end{pmatrix}$$

Now find the search direction vector p_k using (3.2) for the GN method. For this first iteration $p_1 = (0.923529, -0.0368979)$. The approximations of x_1 and x_2 can now be updated:

$$x_{1_1} = 6 + 0.923529 = 6.92353$$
$$x_{2_1} = .3 - 0.0368979 = 0.263103$$

We repeat the GN method using these new approximations. At the third iteration,

$$x_{1_3} = 7.00009$$
$$x_{2_3} = 0.262078,$$

which yields the graph:

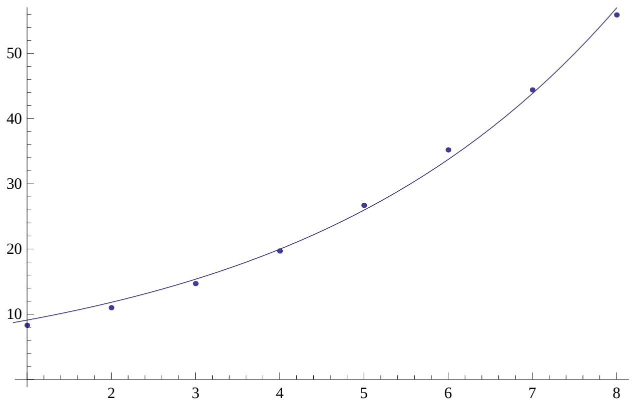

FIGURE 2. The exponential data and model function after the third iteration of the GN method.

At the third iteration we also have the residual vector

$$r = \begin{pmatrix} 0.797515 \\ 0.823383 \\ 0.665998 \\ 0.270077 \\ -0.746333 \\ -1.46989 \\ -0.563416 \\ 1.07124 \end{pmatrix}$$

so that $||r||^2 = 6.01308$ and $f(x) = 3.00654$, indicating that the model has become quite accurate.

The Levenberg-Marquardt method can also be used to minimize this least squares problem. First, we must establish a value for Δ. For simplicity, we will assume a small value for the trust-region radius, $\Delta = 0.05$. Since $||p_1^{GN}|| = 0.924266 > \Delta$ we need to use (5.1) to find an appropriate p_1^{LM}. Since the calculations for $||r||$ are the same as GN, we use (2.1) to calculate the objective function: $f(x) = 63.65305$. Because this number is relatively large, it is necessary to use equation (5.2). For the first iteration we substitute $\lambda = 1$ and use the same approximations for $J(x)$ and r to obtain the solution $p_1^{LM} = (0.851068, -0.0352124)$. The approximations of x_1 and x_2 can now be updated and used to find the new value of $f(x)$:

$$\begin{aligned} &x_{1_1} = 6 + 0.851068 = 6.85107 \\ &x_{2_1} = .3 - 0.0352124 = 0.264788 \\ &f(x) = 3.08479 \end{aligned}$$

Since $||p_1^{LM}|| = 0.851796 > \Delta$ and the error has decreased, we decrease the value for λ to 0.1 and repeat this method using these new approximations. At the second iteration we get $||p_2^{LM}|| = 0.150782 > \Delta$. At the third, $||p_3^{LM}|| = 0.000401997 < \Delta$ and

$$\begin{aligned} &x_{1_3} = 7.00005 \\ &x_{2_3} = 0.262079 \\ &f(x) = 3.00654 \end{aligned}$$

The graph of the model function and data points at this iteration is shown below.
If we were to continue this LMA process, we would no longer use (5.2) because our new p^{LM} is inside the trust region. Instead we would minimize each following p_k in (5.1).

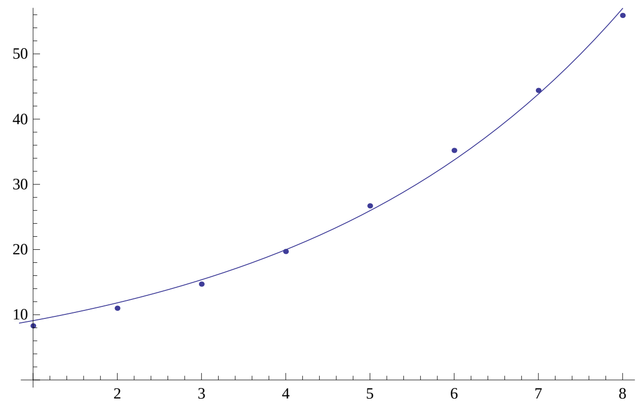

Figure 3. The exponential data and model function after the third iteration of the LMA.

Now the results of these two methods can be compared to see which was more effective in this specific example. At the third iteration of each method $f(x) = 3.00824$ is the result for GN, and $f(x) = 3.00796$ is the result for LMA, demonstrating that LMA was more accurate in this example. The difference between these two methods is almost negligable in this case. Had the Jacobian of this function been rank-deficient, LMA would have been required.

7. Example: Sinusoidal Data

Here is a list of average monthly high temperatures for the city of Baton Rouge, LA[1]:

Jan	61	Jul	92
Feb	65	Aug	92
Mar	72	Sep	88
Apr	78	Oct	81
May	85	Nov	72
Jun	90	Dec	63

This is the set of data points $d(t_j, y_j)$ where t_j is the month and y_j is the temperature. January will be labelled as 1, February as 2, etc. The model function is $\phi(x;t) = x_1 sin(x_2 t + x_3) + x_4$. Here is a graph of the data points and the model with the initial guesses $(x_1, x_2, x_3, x_4) = (17, 0.5, 10.5, 77)$:

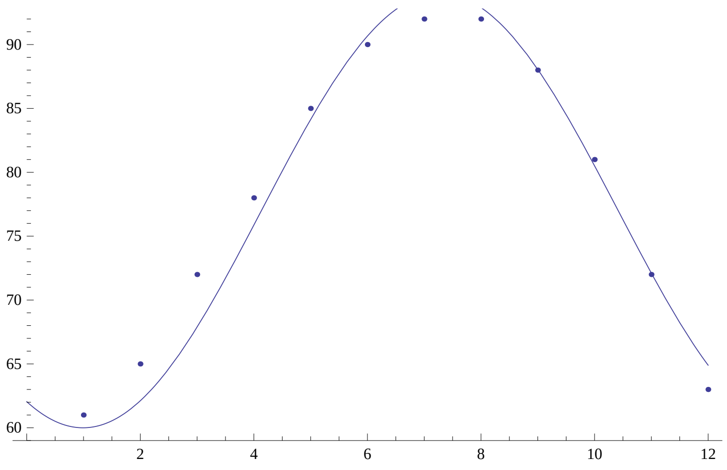

FIGURE 4. The sinusoidal data and first approximation of the model function.

The residual vector:

$$r(x) = \begin{pmatrix} 17\sin(.5(1)+10.5)+77-61 \\ 17\sin(.5(2)+10.5)+77-65 \\ 17\sin(.5(3)+10.5)+77-72 \\ 17\sin(.5(4)+10.5)+77-78 \\ 17\sin(.5(5)+10.5)+77-85 \\ 17\sin(.5(6)+10.5)+77-90 \\ 17\sin(.5(7)+10.5)+77-92 \\ 17\sin(.5(8)+10.5)+77-92 \\ 17\sin(.5(9)+10.5)+77-88 \\ 17\sin(.5(10)+10.5)+77-81 \\ 17\sin(.5(11)+10.5)+77-72 \\ 17\sin(.5(12)+10.5)+77-63 \end{pmatrix} = \begin{pmatrix} -0.999834 \\ -2.88269 \\ -4.12174 \\ -2.12747 \\ -0.85716 \\ 0.664335 \\ 1.84033 \\ 0.893216 \\ 0.0548933 \\ -0.490053 \\ 0.105644 \\ 1.89965 \end{pmatrix}$$

$$||r||^2 = 40.0481$$

$$J(x) = \begin{pmatrix} \sin(x_2+x_3) & x_1\cos(x_2+x_3) & x_1 cos(x_2+x_3) & 1 \\ \sin(2x_2+x_3) & 2x_1\cos(2x_2+x_3) & x_1\cos(2x_2+x_3) & 1 \\ \sin(3x_2+x_3) & 3x_1\cos(3x_2+x_3) & x_1\cos(3x_2+x_3) & 1 \\ \sin(4x_2+x_3) & 4x_1\cos(4x_2+x_3) & x_1\cos(4x_2+x_3) & 1 \\ \sin(5x_2+x_3) & 5x_1\cos(5x_2+x_3) & x_1\cos(5x_2+x_3) & 1 \\ \sin(6x_2+x_3) & 6x_1\cos(6x_2+x_3) & x_1\cos(6x_2+x_3) & 1 \\ \sin(7x_2+x_3) & 7x_1\cos(7x_2+x_3) & x_1\cos(7x_2+x_3) & 1 \\ \sin(8x_2+x_3) & 8x_1\cos(8x_2+x_3) & x_1\cos(8x_2+x_3) & 1 \\ \sin(9x_2+x_3) & 9x_1\cos(9x_2+x_3) & x_1\cos(9x_2+x_3) & 1 \\ \sin(10x_2+x_3) & 10x_1\cos(10x_2+x_3) & x_1\cos(10x_2+x_3) & 1 \\ \sin(11x_2+x_3) & 11x_1\cos(11x_2+x_3) & x_1\cos(11x_2+x_3) & 1 \\ \sin(12x_2+x_3) & 12x_1\cos(12x_2+x_3) & x_1\cos(12x_2+x_3) & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -0.99999 & 0.0752369 & 0.0752369 & 1 \\ -0.875452 & 16.4324 & 8.21618 & 1 \\ -0.536573 & 43.0366 & 14.3455 & 1 \\ -0.0663219 & 67.8503 & 16.9626 & 1 \\ 0.420167 & 77.133 & 15.4266 & 1 \\ 0.803784 & 60.6819 & 10.1137 & 1 \\ 0.990607 & 16.2717 & 2.32453 & 1 \\ 0.934895 & -48.2697 & -6.03371 & 1 \\ 0.650288 & -116.232 & -12.9147 & 1 \\ 0.206467 & -166.337 & -16.6337 & 1 \\ -0.287903 & -179.082 & -16.2802 & 1 \\ -0.711785 & -143.289 & -11.9407 & 1 \end{pmatrix}$$

As before, we will use GN and find the direction vector p_k using (3.2). The first iteration yields $p_1 = (-0.904686, -0.021006, 0.230013, -0.17933)$, and so x_1 through x_4 can be updated:

$$\begin{aligned} x_{1_1} &= 17 - 0.904686 = 16.0953 \\ x_{2_1} &= .5 - 0.021006 = 0.478994 \\ x_{3_1} &= 10.5 + 0.230013 = 10.73 \\ x_{4_1} &= 77 - 0.17933 = 76.8207 \end{aligned}$$

At the second iteration,

$$\begin{aligned} x_{1_2} &= 17.5078 \\ x_{2_2} &= 0.463583 \\ x_{3_2} &= 10.8584 \\ x_{4_2} &= 76.1255 \end{aligned}$$

and $||r||^2 = 13.6556$.

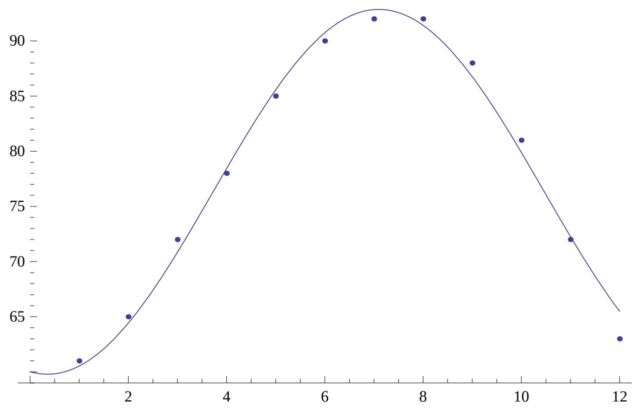

FIGURE 5. The sinusoidal data and model function after the second iteration of GN.

Since $||r|| = 6.32836$ and $||p_1^{GN}|| = 0.95077$ initially, the LMA can also be used. Again starting with $\lambda = 1$, we find that $p_1 = (-0.7595, -0.0219004, 0.236647, -0.198876)$ and

$$\begin{aligned}
x_{1_1} &= 17 - 0.7595 = 16.2405\\
x_{2_1} &= .5 - 0.0219004 = 0.4781\\
x_{3_1} &= 10.5 + 0.236647, = 10.7366\\
x_{4_1} &= 77 - 0.198876 = 76.8011
\end{aligned}$$

Now $||r||^2 = 13.6458$ and $||p_1^{LM}|| = 0.820289$, so λ is decreased by a factor of 10 and the process continues. At the second iteration, $||p_2^{LM}|| = 0.566443$,

$$\begin{aligned}
x_{1_2} &= 16.5319\\
x_{2_2} &= 0.465955\\
x_{3_2} &= 10.8305\\
x_{4_2} &= 76.3247
\end{aligned}$$

and $||r||^2 = 13.0578$. Since $||r||^2 = 13.6556$ after two iterations of GN, LMA is slightly more efficient in this example. The process could be continued for $\Delta < ||p_2^{LM}|| = 0.566443$.

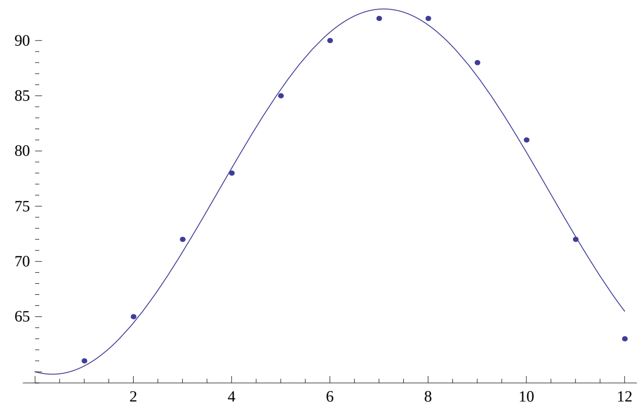

Figure 6. The sinusoidal data and model function after the second iteration of LMA.

8. Other Methods

In cases where the residual is large and the model does not fit the function well, methods other than Gauss-Newton and Levenberg-Marquardt must be used. This is because both GN and LMA approximate $\nabla^2 f(x)$ by eliminating the second term of (2.3) involving the calculation of $\nabla^2 r(x)$. Some of these methods are explored in [2].

Acknowledgements

We would like to thank our professor, Dr. Humberto Munoz, and our graduate student, Ladorian Latin, for helping guide us through this process. We would also like to thank Dr. Mark Davidson and the SMILE program for making this experience possible.

References

[1] "Average Weather for Baton Rouge, LA - Temperature and Precipitation." The Weather Channel. 1 July 2012 (http://www.weather.com/weather/wx climatology/monthly/graph/USLA0033)

[2] Gill, Philip E.; Murray, Walter. *Algorithms for the solution of the nonlinear least-squares problem.* SIAM Journal on Numerical Analysis 15 (5): 977-992. 1978.

[3] Griva, Igor; Nash, Stephen; Sofer Ariela. *Linear and Nonlinear Optimization.* 2nd ed. Society for Industrial Mathematics. 2008.

[4] Nocedal, Jorge; Wright, Steven J. *Numerical Optimization*, 2nd Edition. Springer, Berlin, 2006.

[5] "The Population of the United States." University of Illinois. 28 June 2012 (mste.illinois.edu/malcz/ExpFit/data.html).

[6] Ranganathan, Ananth. "The Levenberg-Marquardt Algorithm." Honda Research Institute, USA. 8 June 2004. 1 July 2012 (http://ananth.in/Notes _files/lmtut.pdf).

Louisiana State University, Baton Rouge, Louisiana
E-mail address: `acroez1@lsu.edu`

University of Mississippi, Oxford, Mississippi
E-mail address: `lbpittma@olemiss.edu`

Louisiana State University, Baton Rouge, Louisiana
E-mail address: `wreyno2@lsu.edu`