The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems

©Henri P. Gavin
Department of Civil and Environmental Engineering
Duke University

September 29, 2015

Abstract

The Levenberg-Marquardt method is a standard technique used to solve nonlinear least squares problems. Least squares problems arise when fitting a parameterized function to a set of measured data points by minimizing the sum of the squares of the errors between the data points and the function. Nonlinear least squares problems arise when the function is not linear in the parameters. Nonlinear least squares methods involve an iterative improvement to parameter values in order to reduce the sum of the squares of the errors between the function and the measured data points. The Levenberg-Marquardt curve-fitting method is actually a combination of two minimization methods: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the steepest-descent direction. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and finding the minimum of the quadratic. The Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. This document describes these methods and illustrates the use of software to solve nonlinear least squares curve-fitting problems.

1 Introduction

In fitting a function $\hat{y}(t;\mathbf{p})$ of an independent variable t and a vector of n parameters $\mathbf{p}$ to a set of m data points (t_i, y_i), it is customary and convenient to minimize the sum of the weighted squares of the errors (or weighted residuals) between the measured data $y(t_i)$ and the curve-fit function $\hat{y}(t_i;\mathbf{p})$. This scalar-valued goodness-of-fit measure is called the chi-squared error criterion.

$$\begin{aligned}
\chi^2(\mathbf{p}) &= \sum_{i=1}^{m} \left[\frac{y(t_i) - \hat{y}(t_i;\mathbf{p})}{w_i} \right]^2 && (1) \\
&= (\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}))^{\mathsf{T}} \mathbf{W} (\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p})) && (2) \\
&= \mathbf{y}^{\mathsf{T}} \mathbf{W} \mathbf{y} - 2\mathbf{y}^{\mathsf{T}} \mathbf{W} \hat{\mathbf{y}} + \hat{\mathbf{y}}^{\mathsf{T}} \mathbf{W} \hat{\mathbf{y}} && (3)
\end{aligned}$$

The value w_i is a measure of the error in measurement $y(t_i)$. The weighting matrix $\mathbf{W}$ is diagonal with $W_{ii} = 1/w_i^2$. If the function $\hat{y}$ is nonlinear in the model parameters $\mathbf{p}$, then the minimization of χ^2 with respect to the parameters must be carried out iteratively. The goal of each iteration is to find a perturbation $\mathbf{h}$ to the parameters $\mathbf{p}$ that reduces χ^2.

2 The Gradient Descent Method

The steepest descent method is a general minimization method which updates parameter values in the "downhill" direction: the direction opposite to the gradient of the objective function. The gradient descent method converges well for problems with simple objective functions [3, 4]. For problems with thousands of parameters, gradient descent methods are sometimes the only viable method.

The gradient of the chi-squared objective function with respect to the parameters is

$$\begin{aligned} \frac{\partial}{\partial \mathbf{p}} \chi^2 &= (\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}))^{\mathsf{T}} \mathbf{W} \frac{\partial}{\partial \mathbf{p}} (\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p})) & (4) \\ &= -(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}))^{\mathsf{T}} \mathbf{W} \left[\frac{\partial \hat{\mathbf{y}}(\mathbf{p})}{\partial \mathbf{p}} \right] & (5) \\ &= -(\mathbf{y} - \hat{\mathbf{y}})^{\mathsf{T}} \mathbf{W} \mathbf{J} & (6) \end{aligned}$$

where the $m \times n$ Jacobian matrix $[\partial \hat{\mathbf{y}} / \partial \mathbf{p}]$ represents the local sensitivity of the function $\hat{y}$ to variation in the parameters $\mathbf{p}$. For notational simplicity $\mathbf{J}$ will be used for $[\partial \hat{\mathbf{y}} / \partial \mathbf{p}]$. The parameter update $\mathbf{h}$ that moves the parameters in the direction of steepest descent is given by

$$\mathbf{h}_{\mathsf{gd}} = \alpha \mathbf{J}^{\mathsf{T}} \mathbf{W} (\mathbf{y} - \hat{\mathbf{y}}) \ , \qquad (7)$$

where the positive scalar α determines the length of the step in the steepest-descent direction.

3 The Gauss-Newton Method

The Gauss-Newton method is a method for minimizing a sum-of-squares objective function. It presumes that the objective function is approximately quadratic in the parameters near the optimal solution [1]. For moderately-sized problems the Gauss-Newton method typically converges much faster than gradient-descent methods [5].

The function evaluated with perturbed model parameters may be locally approximated through a first-order Taylor series expansion.

$$\hat{\mathbf{y}}(\mathbf{p} + \mathbf{h}) \approx \hat{\mathbf{y}}(\mathbf{p}) + \left[\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{p}} \right] \mathbf{h} = \hat{\mathbf{y}} + \mathbf{J}\mathbf{h} \ , \qquad (8)$$

Substituting the approximation for the perturbed function, $\hat{\mathbf{y}} + \mathbf{J}\mathbf{h}$, for $\hat{\mathbf{y}}$ in equation (3),

$$\chi^2(\mathbf{p} + \mathbf{h}) \approx \mathbf{y}^{\mathsf{T}} \mathbf{W} \mathbf{y} + \hat{\mathbf{y}}^{\mathsf{T}} \mathbf{W} \hat{\mathbf{y}} - 2 \mathbf{y}^{\mathsf{T}} \mathbf{W} \hat{\mathbf{y}} - 2 (\mathbf{y} - \hat{\mathbf{y}})^{\mathsf{T}} \mathbf{W} \mathbf{J} \mathbf{h} + \mathbf{h}^{\mathsf{T}} \mathbf{J}^{\mathsf{T}} \mathbf{W} \mathbf{J} \mathbf{h} \ . \qquad (9)$$

This shows that χ^2 is approximately quadratic in the perturbation $\mathbf{h}$, and that the Hessian of the chi-squared fit criterion is approximately $\mathbf{J}^{\mathsf{T}} \mathbf{W} \mathbf{J}$.

The parameter update $\mathbf{h}$ that minimizes χ^2 is found from $\partial \chi^2 / \partial \mathbf{h} = 0$.

$$\frac{\partial}{\partial \mathbf{h}} \chi^2(\mathbf{p} + \mathbf{h}) \approx -2 (\mathbf{y} - \hat{\mathbf{y}})^{\mathsf{T}} \mathbf{W} \mathbf{J} + 2 \mathbf{h}^{\mathsf{T}} \mathbf{J}^{\mathsf{T}} \mathbf{W} \mathbf{J} \ , \qquad (10)$$

update and the resulting normal equations for the Gauss-Newton update are

$$\left[\mathbf{J}^{\mathsf{T}} \mathbf{W} \mathbf{J} \right] \mathbf{h}_{\mathsf{gn}} = \mathbf{J}^{\mathsf{T}} \mathbf{W} (\mathbf{y} - \hat{\mathbf{y}}) \ . \qquad (11)$$

4 The Levenberg-Marquardt Method

The Levenberg-Marquardt algorithm adaptively varies the parameter updates between the gradient descent update and the Gauss-Newton update,

$$\left[\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J} + \lambda \mathbf{I}\right] \mathbf{h}_{\mathsf{lm}} = \mathbf{J}^{\mathsf{T}}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}) \ , \tag{12}$$

where small values of the algorithmic parameter λ result in a Gauss-Newton update and large values of λ result in a gradient descent update. The parameter λ is initialized to be large so that first updates are small steps in the steepest-descent direction. If an iteration happens to result in a worse approximation, λ is increased. As the solution improves, λ is decreased, the Levenberg-Marquardt method approaches the Gauss-Newton method, and the solution typically accelerates to the local minimum [3, 4, 5].

Marquardt's suggested update relationship [5]

$$\left[\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J} + \lambda \ \mathsf{diag}(\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J})\right] \mathbf{h}_{\mathsf{lm}} = \mathbf{J}^{\mathsf{T}}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}) \ . \tag{13}$$

makes the effect of particular values of λ less problem-specific, and is used in the Levenberg-Marquardt algorithm implemented in the MATLAB function `lm.m`

4.1 Numerical Implementation

Many variations of the Levenberg-Marquardt have been published in papers and in code. This document borrows from some of these, including the enhancement of a rank-1 Jacobian update. In iteration i, the step $\mathbf{h}$ is evaluated by comparing $\chi^2(\mathbf{p})$ to $\chi^2(\mathbf{p}+\mathbf{h})$. The step is accepted if the metric ρ_i [6] is greater than a user-specified threshold, $\epsilon_4 > 0$. This metric is a measure of the actual improvement in χ^2 as compared to the improvement of an LM update assuming the approximation (8) were exact.

$$\begin{aligned}
\rho_i(\mathbf{h}_{\mathsf{lm}}) &= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathsf{lm}})}{(\mathbf{y}-\hat{\mathbf{y}})^{\mathsf{T}}(\mathbf{y}-\hat{\mathbf{y}}) - (\mathbf{y}-\hat{\mathbf{y}}-\mathbf{J}\mathbf{h}_{\mathsf{lm}})^{\mathsf{T}}(\mathbf{y}-\hat{\mathbf{y}}-\mathbf{J}\mathbf{h}_{\mathsf{lm}})} && (14)\\
&= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathsf{lm}})}{\mathbf{h}_{\mathsf{lm}}^{\mathsf{T}}\left(\lambda_i \mathbf{h}_{\mathsf{lm}} + \mathbf{J}^{\mathsf{T}}\mathbf{W}\left(\mathbf{y}-\hat{\mathbf{y}}(\mathbf{p})\right)\right)} \quad \text{if using eq'n (12) for } \mathbf{h}_{\mathsf{lm}} && (15)\\
&= \frac{\chi^2(\mathbf{p}) - \chi^2(\mathbf{p}+\mathbf{h}_{\mathsf{lm}})}{\mathbf{h}_{\mathsf{lm}}^{\mathsf{T}}\left(\lambda_i \mathsf{diag}(\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J})\mathbf{h}_{\mathsf{lm}} + \mathbf{J}^{\mathsf{T}}\mathbf{W}\left(\mathbf{y}-\hat{\mathbf{y}}(\mathbf{p})\right)\right)} \quad \text{if using eq'n (13) for } \mathbf{h}_{\mathsf{lm}} && (16)
\end{aligned}$$

If in an iteration $\rho_i(\mathbf{h}) > \epsilon_4$ then $\mathbf{p}+\mathbf{h}$ is sufficiently better than $\mathbf{p}$, $\mathbf{p}$ is replaced by $\mathbf{p}+\mathbf{h}$, and λ is reduced by a factor. Otherwise λ is increased by a factor, and the algorithm proceeds to the next iteration.

4.1.1 *Initialization and update of the L-M parameter, λ, and the parameters $\mathbf{p}$*

In `lm.m` users may select one of three methods for inializing and updating λ and $\mathbf{p}$.

1. $\lambda_0 = \lambda_o$; λ_o is user-specified [5].
   use eq'n (13) for $\mathbf{h}_{\mathsf{lm}}$ and eq'n (16) for ρ
   if $\rho_i(\mathbf{h}) > \epsilon_4$: $\mathbf{p} \leftarrow \mathbf{p}+\mathbf{h}$; $\lambda_{i+1} = \max[\lambda_i/L_\downarrow, 10^{-7}]$;
   otherwise: $\lambda_{i+1} = \min\left[\lambda_i L_\uparrow, 10^7\right]$;

2. $\lambda_0 = \lambda_o \max \left[\mathsf{diag}[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J}] \right]$; λ_o is user-specified.
   use eq'n (12) for $\mathbf{h}_{\mathsf{lm}}$ and eq'n (15) for ρ
   $\alpha = \left(\left(\mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p})) \right)^\mathsf{T} \mathbf{h} \right) / \left(\left(\chi^2(\mathbf{p}+\mathbf{h}) - \chi^2(\mathbf{p}) \right)/2 + 2 \left(\mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p})) \right)^\mathsf{T} \mathbf{h} \right)$;
   if $\rho_i(\alpha\mathbf{h}) > \epsilon_4$: $\mathbf{p} \leftarrow \mathbf{p} + \alpha\mathbf{h}$; $\lambda_{i+1} = \max\left[\lambda_i/(1+\alpha), 10^{-7}\right]$;
   otherwise: $\lambda_{i+1} = \lambda_i + |\chi^2(\mathbf{p}+\alpha\mathbf{h}) - \chi^2(\mathbf{p})|/(2\alpha)$;

3. $\lambda_0 = \lambda_o \max \left[\mathsf{diag}[\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J}] \right]$; λ_o is user-specified [6].
   use eq'n (12) for $\mathbf{h}_{\mathsf{lm}}$ and eq'n (15) for ρ
   if $\rho_i(\mathbf{h}) > \epsilon_4$: $\mathbf{p} \leftarrow \mathbf{p} + \mathbf{h}$; $\lambda_{i+1} = \lambda_i \max\left[1/3, 1 - (2\rho_i - 1)^3\right]; \nu_i = 2$;
   otherwise: $\lambda_{i+1} = \lambda_i \nu_i; \quad \nu_{i+1} = 2\nu_i$;

For the examples in section 4.4, method 1 [5] with $L_\uparrow \approx 11$ and $L_\downarrow \approx 9$ has good convergence properties.

4.1.2 *Computation and rank-1 update of the Jacobian,* $[\partial\mathbf{y}/\partial\mathbf{p}]$

In the first iteration, in every $2n$ iterations, and in iterations where $\chi^2(\mathbf{p}+\mathbf{h}) > \chi^2(\mathbf{p})$, the Jacobian ($\mathbf{J} \in \mathbb{R}^{m \times n}$) is numerically approximated using forward differences,

$$J_{ij} = \frac{\partial \hat{y}_i}{\partial p_j} = \frac{\hat{y}(t_i;\mathbf{p}+\delta\mathbf{p}_j) - \hat{y}(t_i;\mathbf{p})}{||\delta\mathbf{p}_j||} \ , \tag{17}$$

or central differences (default)

$$J_{ij} = \frac{\partial \hat{y}_i}{\partial p_j} = \frac{\hat{y}(t_i;\mathbf{p}+\delta\mathbf{p}_j) - \hat{y}(t_i;\mathbf{p}-\delta\mathbf{p}_j)}{2||\delta\mathbf{p}_j||} \ , \tag{18}$$

where the j-th element of $\delta\mathbf{p}_j$ is the only non-zero element and is set to $\Delta_j(1+|p_j|)$. In all other iterations, the Jacobian is updated using the Broyden rank-1 update formula,

$$\mathbf{J} = \mathbf{J} + \left((\hat{\mathbf{y}}(\mathbf{p}+\mathbf{h}) - \hat{\mathbf{y}}(\mathbf{p}) - \mathbf{J}\mathbf{h})\,\mathbf{h}^\mathsf{T} \right) / \left(\mathbf{h}^\mathsf{T}\mathbf{h} \right) \ . \tag{19}$$

For problems with many parameters, a finite differences Jacobian is computationally expensive. Convergence can be achieved with fewer function evaluations if the Jacobian is re-computed using finite differences only occaisonally. The rank-1 Jacobian update equation (19) requires no additional function evaluations.

4.1.3 *Convergence criteria*

Convergence is achieved when one of the following three criteria is satisfied,

- Convergence in the gradient, $\max \left| \mathbf{J}^\mathsf{T}\mathbf{W}(\mathbf{y} - \hat{\mathbf{y}}) \right| < \epsilon_1$;

- Convergence in paramters, $\max |h_i/p_i| < \epsilon_2$; or

- Convergence in χ^2, $\chi^2/(m-n+1) < \epsilon_3$.

Otherwise, iterations terminate when the iteration count exceeds a pre-specified limit.

4.2 Error Analysis

Once the optimal curve-fit parameters $\mathbf{p}_{\text{fit}}$ are determined, parameter statistics are computed for the converged solution using weight values, w_i^2, equal to the mean square measurement error, σ_y^2,

$$w_i^2 = \sigma_y^2 = \frac{1}{m-n+1}(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}_{\text{fit}}))^{\mathsf{T}}(\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}_{\text{fit}})) \quad \forall \; i \; . \tag{20}$$

The parameter covariance matrix is then computed from

$$\mathbf{V}_{\mathbf{p}} = [\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J}]^{-1} \; , \tag{21}$$

and the asymptotic standard parameter errors are given by

$$\sigma_{\mathbf{p}} = \sqrt{\mathsf{diag}([\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J}]^{-1})} \; , \tag{22}$$

The asymptotic standard parameter error is a measure of how unexplained variability in the data propagates to variability in the parameters, and is essentially an error measure for the parameters. The standard error of the fit is given by

$$\sigma_{\hat{\mathbf{y}}} = \sqrt{\mathsf{diag}(\mathbf{J}[\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J}]^{-1}\mathbf{J}^{\mathsf{T}})} \; . \tag{23}$$

The standard error of the fit indicates how variability in the parameters affects the variability in the curve-fit. The asymptotic standad prediction error reflects the standard error of the fit as well as the mean square measurement error.

$$\sigma_{\hat{\mathbf{y}}_{\mathbf{p}}} = \sqrt{\sigma_y^2 + \mathsf{diag}(\mathbf{J}[\mathbf{J}^{\mathsf{T}}\mathbf{W}\mathbf{J}]^{-1}\mathbf{J}^{\mathsf{T}})} \; . \tag{24}$$

4.3 Matlab code: lm.m

The MATLAB function `lm.m` implements the Levenberg-Marquardt method for curve-fitting problems. The code with examples are available here:

- http://www.duke.edu/~hpgavin/lm.m
- http://www.duke.edu/~hpgavin/lm_examp.m
- http://www.duke.edu/~hpgavin/lm_func.m
- http://www.duke.edu/~hpgavin/lm_plots.m

```
function [p,X2,sigma_p,sigma_y,corr_p,R_sq,cvg_hst] = lm(func,p,t,y_dat,weight,dp,p_min,p_max,c,opts)
% [p,X2,sigma_p,sigma_y,corr_p,R_sq,cvg_hst] = lm(func,p,t,y_dat,weight,dp,p_min,p_max,c,opts)
%
% Levenberg Marquardt curve-fitting: minimize sum of weighted squared residuals
%---------- INPUT VARIABLES -----------
% func   = function of n independent variables, 't', and m parameters, 'p',
%          returning the simulated model: y_hat = func(t,p,c)
% p      = n-vector of initial guess of parameter values
% t      = m-vectors or matrix of independent variables (used as arg to func)
% y_dat  = m-vectors or matrix of data to be fit by func(t,p)
% weight = weighting vector for least squares fit ( weight >= 0 ) ...
%          inverse of the standard measurement errors
%          Default:  sqrt(d.o.f. / ( y_dat' * y_dat ))
% dp     = fractional increment of 'p' for numerical derivatives
%          dp(j)>0 central differences calculated
%          dp(j)<0 one sided 'backwards' differences calculated
%          dp(j)=0 sets corresponding partials to zero; i.e. holds p(j) fixed
%          Default:  0.001;
% p_min  = n-vector of lower bounds for parameter values
% p_max  = n-vector of upper bounds for parameter values
% c      = an optional matrix of values passed to func(t,p,c)
% opts   = vector of algorithmic parameters
%             parameter    defaults    meaning
% opts(1)  =  prnt            3        >1 intermediate results; >2 plots
% opts(2)  =  MaxIter      10*Npar     maximum number of iterations
% opts(3)  =  epsilon_1       1e-3     convergence tolerance for gradient
% opts(4)  =  epsilon_2       1e-3     convergence tolerance for parameters
% opts(5)  =  epsilon_3       1e-3     convergence tolerance for Chi-square
% opts(6)  =  epsilon_4       1e-1     determines acceptance of a L-M step
% opts(7)  =  lambda_0        1e-2     initial value of L-M paramter
% opts(8)  =  lambda_UP_fac   11       factor for increasing lambda
% opts(9)  =  lambda_DN_fac    9       factor for decreasing lambda
% opts(10) =  Update_Type      1       1: Levenberg-Marquardt lambda update
%                                      2: Quadratic update
%                                      3: Nielsen's lambda update equations
%
%---------- OUTPUT VARIABLES -----------
% p       = least-squares optimal estimate of the parameter values
% X2      = Chi squared criteria
% sigma_p = asymptotic standard error of the parameters
% sigma_y = asymptotic standard error of the curve-fit
% corr_p  = correlation matrix of the parameters
% R_sq    = R-squared cofficient of multiple determination
% cvg_hst = convergence history
```

The m-file to solve a least-squares curve-fit problem with `lm.m` can be as simple as:

```
my_data = load('my_data_file');   % load the data
t     = my_data(:,1);             % if the independent variable is in column 1
y_dat = my_data(:,2);             % if the   dependent variable is in column 2

p_min  = [ -10  0.1  5 0.1 ];     % minimum expected parameter values
p_max  = [  10  5.0 15 0.5 ];     % maximum expected parameter values
p_init = [   3  2.0 10 0.2 ];     % initial guess for parameter values

[ p_fit, X2, sigma_p, sigma_y, corr, R_sq, cvg_hst ] =  ...
                      lm ( 'lm_func', p_init, t, y_dat, 1, -0.01, p_min, p_max )
```

where the user-supplied function `lm_func.m` could be, for example,

```
function y_hat = lm_func(t,p,c)
   y_hat = p(1) * t .* exp(-t/p(2)) .* cos(2*pi*( p(3)*t - p(4) ));
```

It is common and desirable to repeat the same experiment two or more times and to estimate a single set of curve-fit parameters from all the experiments. In such cases the data file may arranged as follows:

```
  %   t-variable   y (1st experiment)   y (2nd experiemnt)  y (3rd experiemnt)
      0.50000      3.5986               3.60192             3.58293
      0.80000      8.1233               8.01231             8.16234
      0.90000     12.2342              12.29523            12.01823
       :            :                    :                   :
      etc.         etc.                 etc.                etc.
```

If your data is arranged as above you may prepare the data for `lm.m` using the following lines.

```
my_data = load('my_data_file');        % load the data
t_column = 1;                          % column  of the independent variable
y_columns = [ 2 3 4 ];                 % columns of the measured dependent variables

y_dat = my_data(:,y_columns);          % the measured data
y_dat = y_dat(:);                      % a single column vector

t     = my_data(:,t_column);           % the independent variable
t     = t*ones(1,length(y_columns));   % a column of t for each column of y
t     = t(:);                          % a single column vector
```

Note that the arguments `t` and `y_dat` to `lm.m` may be matrices as long as the dimensions of `t` match the dimensions of `y_dat`. The columns of `t` need not be identical. Results may be plotted with lm_plots.m:

```
function lm_plots ( t, y_dat, y_fit, sigma_y, cvg_hst, filename )
% lm_plots ( t, y_dat, y_fit, sigma_y, cvg_hst, filename )
% Plot statistics of the results of a Levenberg-Marquardt least squares
% analysis with lm.m

y_dat = y_dat(:);
y_fit = y_fit(:);

[max_it,n] = size(cvg_hst); n = n-3;

figure(101); % ---------- plot convergence history of parameters, chi^2, lambda
 clf
 subplot(211)
  plot( cvg_hst(:,1), cvg_hst(:,2:n+1), '-o','linewidth',4);
  for i=1:n
   text(1.02*cvg_hst(max_it,1),cvg_hst(max_it,1+i), sprintf('%d',i) );
  end
   ylabel('parameter values')
 subplot(212)
  semilogy( cvg_hst(:,1) , [ cvg_hst(:,n+2) cvg_hst(:,n+3)], '-o','linewidth',4)
   text(0.8*cvg_hst(max_it,1),0.2*cvg_hst(max_it,n+2), '\chi^2/(m-n+1)}' );
   text(0.8*cvg_hst(max_it,1),0.5*cvg_hst(max_it,n+3), '\lambda' );
%  legend('\chi^2/(m-n+1)}', '\lambda', 3);
   ylabel('\chi^2/(m-n+1)} and \lambda')
   xlabel('function calls')


figure(102); % ------------ plot data, fit, and confidence interval of fit
 clf
   plot(t,y_dat,'og', t,y_fit,'-b',
        t,y_fit+1.96*sigma_y,'.k', t,y_fit-1.96*sigma_y,'.k');
    legend('y_{data}','y_{fit}','95% c.i.','',0);
    ylabel('y(t)')
    xlabel('t')
% subplot(212)
%  semilogy(t,sigma_y,'-r','linewidth',4);
%     ylabel('\sigma_y(t)')

figure(103); % ------------ plot histogram of residuals, are they Gaussean?
 clf
 hist(real(y_dat - y_fit))
  title('histogram of residuals')
  axis('tight')
  xlabel('y_{data} - y_{fit}')
  ylabel('count')
```

4.4 Numerical Examples

The robustness of `lm.m` is tested in three numerical examples by curve-fitting simulated experimental measurements. Noisy experimental measurements y are simulated by adding random measurement noise to the curve-fit function evaluated with a set of "true" parameter values $\hat{y}(t; \mathbf{p}_{\text{true}})$. The random measurement noise is normally distributed with a mean of zero and a standard deviation of 0.20.

$$y_i = \hat{y}(t_i; \mathbf{p}_{\text{true}}) + N(0, 0.20). \tag{25}$$

The convergence of the parameters from an erroneous initial guess $\mathbf{p}_{\text{initial}}$ to values closer to $\mathbf{p}_{\text{true}}$ is then examined.

Each numerical example below has four parameters ($n = 4$) and one-hundred measurements ($m = 100$). Each numerical example has a different curve-fit function $\hat{y}(t; \mathbf{p})$, a different "true" parameter vector $\mathbf{p}_{\text{true}}$, and a different vector of initial parameters $\mathbf{p}_{\text{initial}}$.

For several values of p_2 and p_4, the χ^2 error criterion is calculated and is plotted as a surface over the $p_2 - p_4$ plane. The "bowl-shaped" nature of the objective function is clearly evident in each example. The objective function may not appear quadratic in the parameters and the objective function may have multiple minima. The presence of measurement noise does not affect the smoothness of the objective function.

The gradient descent method endeavors to move parameter values in a down-hill direction to minimize $\chi^2(\mathbf{p})$. This often requires small step sizes but is required when the objective function is not quadratic. The Gauss-Newton method approximates the bowl shape as a quadratic and endeavors to move parameter values to the minimum in a small number of steps. This method works well when the parameters are close to their optimal values. The Levenberg-Marquardt method retains the best features of both the gradient-descent method and the Gauss-Newton method.

The evolution of the parameter values, the evolution of χ^2, and the evolution of λ from iteration to iteration is plotted for each example.

The simulated experimental data, the curve fit, and the 95-percent confidence interval of the fit are plotted, the standard error of the fit, and a histogram of the fit errors are also plotted.

The initial parameter values $\mathbf{p}_{\text{initial}}$, the true parameter values $\mathbf{p}_{\text{true}}$, the fit parameter values $\mathbf{p}_{\text{fit}}$, the standard error of the fit parameters $\sigma_{\mathbf{p}}$, and the correlation matrix of the fit parameters are tabulated. The true parameter values lie within the confidence interval $p_{\text{fit}} - 1.96\sigma_p < p_{\text{true}} < p_{\text{fit}} + 1.96\sigma_p$ with a confidence level of 95 percent.

4.4.1 *Example 1*

Consider fitting the following function to a set of measured data.

$$\hat{y}(t; \mathbf{p}) = p_1 \exp(-t/p_2) + p_3 t \exp(-t/p_4) \tag{26}$$

The m-function to be used with `lm.m` is simply:

```
function y_hat = lm_func(t,p,c)
   y_hat = p(1)*exp(-t/p(2)) + p(3)*t.*exp(-t/p(4));
```

The "true" parameter values $\mathbf{p}_{\text{true}}$, the initial parameter values $\mathbf{p}_{\text{initial}}$, resulting curve-fit parameter values $\mathbf{p}_{\text{fit}}$ and standard errors of the fit parameters $\sigma_{\mathbf{p}}$ are shown in Table 1. The R^2 fit criterion is 98 percent. The standard parameter errors are less than one percent of the parameter values except for the standard error for p_2, which is 1.5 percent of p_2. The parameter correlation matrix is given in Table 2. Parameters p_3 and p_4 are the most correlated at -96 percent. Parameters p_1 and p_4 are the least correlated at -35 percent.

The bowl-shaped nature of the χ^2 objective function is shown in Figure 1(a). This shape is nearly quadratic and has a single minimum.

The convergence of the parameters and the evolution of χ^2 and λ are shown in Figure 1(b).

The data points, the curve fit, and the curve fit confidence band are plotted in Figure 1(c). Note that the standard error of the fit is smaller near the center of the fit domain and is larger at the edges of the domain.

A histogram of the difference between the data values and the curve-fit is shown in Figure 1(d). Ideally these curve-fit errors should be normally distributed.

Table 1. Parameter values and standard errors.

$\mathbf{p}_{\text{initial}}$	$\mathbf{p}_{\text{true}}$	$\mathbf{p}_{\text{fit}}$	$\sigma_{\mathbf{p}}$	$\sigma_{\mathbf{p}}/\mathbf{p}_{\text{fit}}$ (%)
5.0	20.0	19.918	0.150	0.75
2.0	10.0	10.159	0.152	1.50
0.2	1.0	0.9958	0.005	0.57
10.0	50.0	50.136	0.209	0.41

Table 2. Parameter correlation matrix.

	p_1	p_2	p_3	p_4
p_1	1.00	-0.74	0.40	-0.35
p_2	-0.74	1.00	-0.77	0.71
p_3	0.40	-0.77	1.00	-0.96
p_4	-0.35	0.71	-0.96	1.00

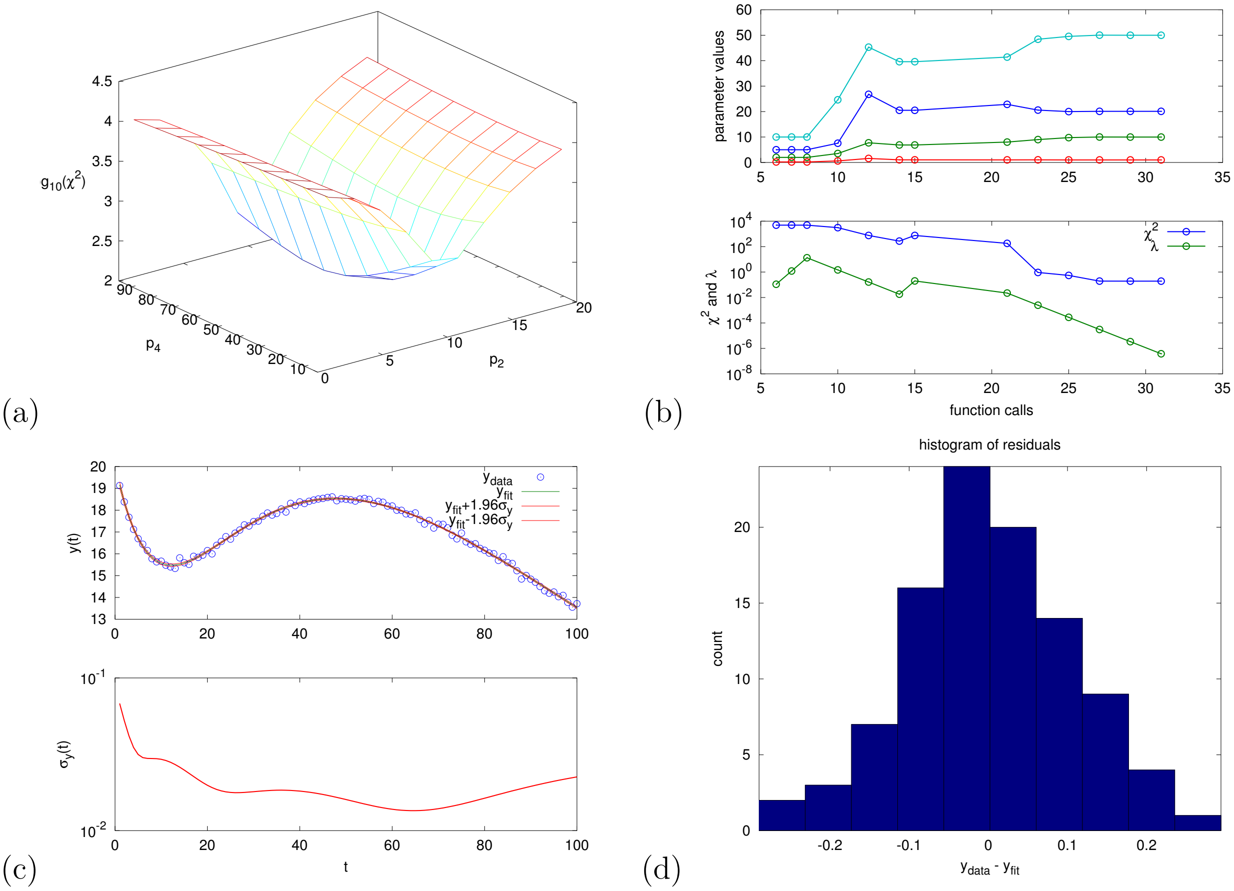

Figure 1. (a) The sum of the squared errors as a function of p_2 and p_4. (b) Top: the convergence of the parameters with each iteration, (b) Bottom: values of χ^2 and λ each iteration. (c) Top: data y, curve-fit $\hat{y}(t; \mathbf{p}_{\text{fit}})$, curve-fit+error, and curve-fit-error; (c) Bottom: standard error of the fit, $\sigma_{\hat{y}}(t)$. (d) Histogram of the errors between the data and the fit.

4.4.2 Example 2

Consider fitting the following function to a set of measured data.

$$\hat{y}(t;\mathbf{p}) = p_1(t/\max(t)) + p_2(t/\max(t))^2 + p_3(t/\max(t))^3 + p_4(t/\max(t))^4 \qquad (27)$$

This function is linear in the parameters and may be fit using methods of linear least squares. The m-function to be used with `lm.m` is simply:

```
function y_hat = lm_func(t,p,c)
   mt = max(t);
   y_hat = p(1)*(t/mt) + p(2)*(t/mt).^2 + p(3)*(t/mt).^3 + p(4)*(t/mt).^4;
```

The "true" parameter values $\mathbf{p}_{\text{true}}$, the initial parameter values $\mathbf{p}_{\text{initial}}$, resulting curve-fit parameter values $\mathbf{p}_{\text{fit}}$ and standard errors of the fit parameters $\sigma_{\mathbf{p}}$ are shown in Table 3. The R^2 fit criterion is 99.9 percent. In this example, the standard parameter errors are larger than in example 1. The standard error for p_2 is 12 percent and the standard error of p_3 is 17 percent. Note that a very high value of the R^2 coefficient of determination does not necessarily mean that parameter values have been found with great accuracy. The parameter correlation matrix is given in Table 4. These parameters are highly correlated with one another, meaning that a change in one parameter will almost certainly result in changes in the other parameters.

The bowl-shaped nature of the χ^2 objective function is shown in Figure 2(a). This shape is nearly quadratic and has a single minimum. The correlation of parameters p_2 and p_4, for example, is easily seen from this figure.

The convergence of the parameters and the evolution of χ^2 and λ are shown in Figure 2(b). The parameters converge monotonically to their final values.

The data points, the curve fit, and the curve fit confidence band are plotted in Figure 2(c). Note that the standard error of the fit approaches zero at $t = 0$ and is largest at $t = 100$. This is because $\hat{y}(0;\mathbf{p}) = 0$, regardless of the values in $\mathbf{p}$.

A histogram of the difference between the data values and the curve-fit is shown in Figure 1(d). Ideally these curve-fit errors should be normally distributed, and they appear to be so in this example.

Table 3. Parameter values and standard errors.

$\mathbf{p}_{\text{initial}}$	$\mathbf{p}_{\text{true}}$	$\mathbf{p}_{\text{fit}}$	$\sigma_{\mathbf{p}}$	$\sigma_{\mathbf{p}}/\mathbf{p}_{\text{fit}}$ (%)
4.0	20.0	19.934	0.506	2.54
-5.0	-24.0	-22.959	2.733	11.90
6.0	30.0	27.358	4.597	16.80
10.0	-40.0	-38.237	2.420	6.33

Table 4. Parameter correlation matrix.

	p_1	p_2	p_3	p_4
p_1	1.00	-0.97	0.92	-0.87
p_2	-0.97	1.00	-0.99	0.95
p_3	0.92	-0.99	1.00	-0.99
p_4	-0.87	0.95	-0.99	1.00

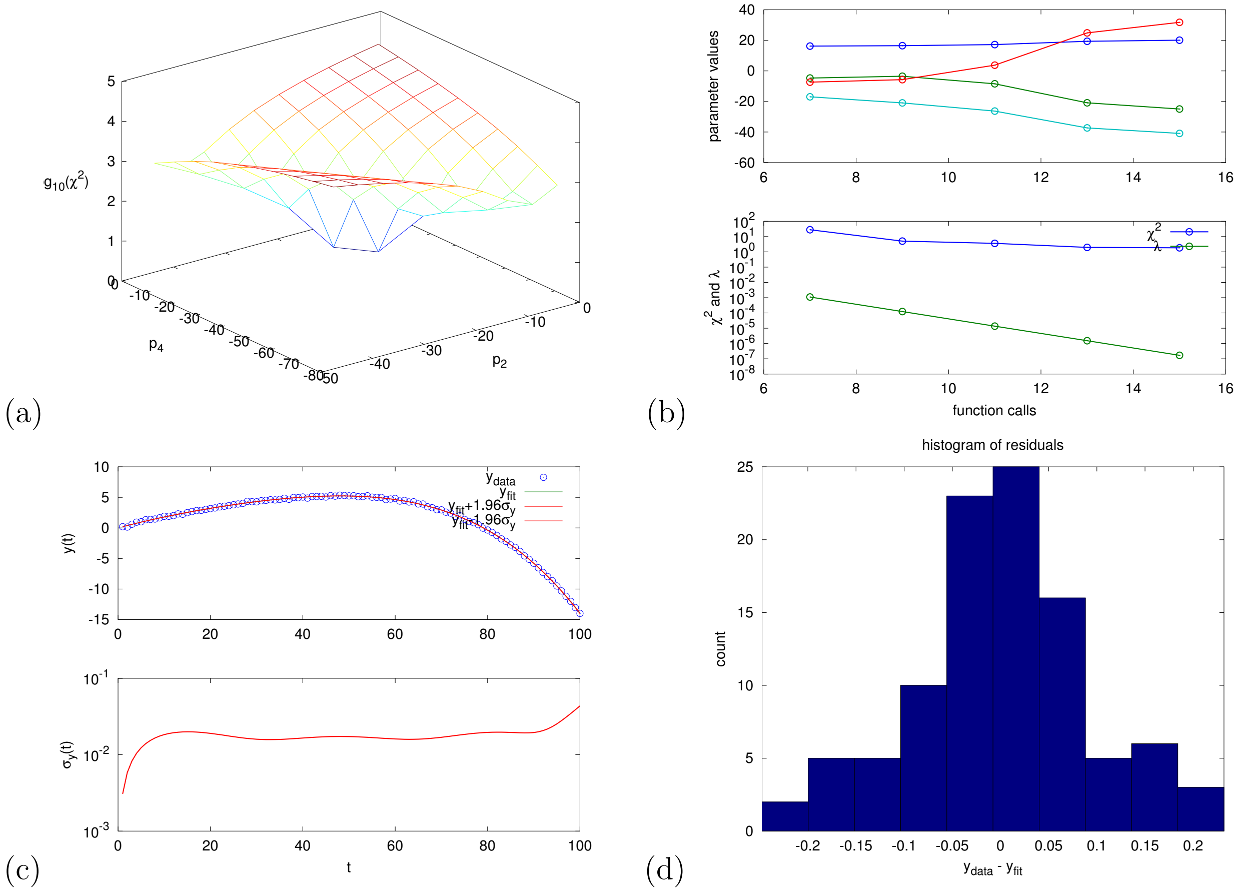

Figure 2. (a) The sum of the squared errors as a function of p_2 and p_4. (b) Top: the convergence of the parameters with each iteration, (b) Bottom: values of χ^2 and λ each iteration. (c) Top: data y, curve-fit $\hat{y}(t; \mathbf{p}_{\text{fit}})$, curve-fit+error, and curve-fit-error; (c) Bottom: standard error of the fit, $\sigma_{\hat{y}}(t)$. (d) Histogram of the errors between the data and the fit.

4.4.3 Example 3

Consider fitting the following function to a set of measured data.

$$\hat{y}(t;\mathbf{p}) = p_1 \exp(-t/p_2) + p_3 \sin(t/p_4) \tag{28}$$

This function is linear in the parameters and may be fit using methods of linear least squares. The m-function to be used with `lm.m` is simply:

```
function y_hat = lm_func(t,p,c)
   y_hat = p(1)*exp(-t/p(2)) + p(3)*sin(t/p(4));
```

The "true" parameter values $\mathbf{p}_{\text{true}}$, the initial parameter values $\mathbf{p}_{\text{initial}}$, resulting curve-fit parameter values $\mathbf{p}_{\text{fit}}$ and standard errors of the fit parameters $\sigma_{\mathbf{p}}$ are shown in Table 5. The R^2 fit criterion is 99.8 percent. In this example, the standard parameter errors are all less than one percent. The parameter correlation matrix is given in Table 6. Parameters p_4 is not correlated with the other parameters. Parameters p_1 and p_2 are most correlated at 73 percent.

The bowl-shaped nature of the χ^2 objective function is shown in Figure 3(a). This shape is clearly not quadratic and has multiple minima. In this example, the initial guess for parameter p_4, the period of the oscillatory component, has to be within ten percent of the true value, otherwise the algorithm in `lm.m` will converge to a very small value of the amplitude of oscillation p_3 and an erroneous value for p_4. When such an occurrence arises, the standard errors $\sigma_{\mathbf{p}}$ of the fit parameters p_3 and p_4 are quite large and the histogram of curve-fit errors (Figure 3(d)) is not normally distributed.

The convergence of the parameters and the evolution of χ^2 and λ are shown in Figure 3(b). The parameters converge monotonically to their final values.

The data points, the curve fit, and the curve fit confidence band are plotted in Figure 3(c).

A histogram of the difference between the data values and the curve-fit is shown in Figure 1(d). Ideally these curve-fit errors should be normally distributed, and they appear to be so in this example.

Table 5. Parameter values and standard errors.

$\mathbf{p}_{\text{initial}}$	$\mathbf{p}_{\text{true}}$	$\mathbf{p}_{\text{fit}}$	$\sigma_{\mathbf{p}}$	$\sigma_{\mathbf{p}}/\mathbf{p}_{\text{fit}}$ (%)
10.0	6.0	5.987	0.032	0.53
50.0	20.0	20.100	0.144	0.72
6.0	1.0	0.978	0.010	0.98
5.6	5.0	4.999	0.004	0.08

Table 6. Parameter correlation matrix.

	p_1	p_2	p_3	p_4
p_1	1.00	-0.74	-0.28	-0.02
p_2	-0.74	1.00	0.18	0.02
p_3	-0.28	0.18	1.00	-0.02
p_4	-0.02	0.02	-0.02	1.00

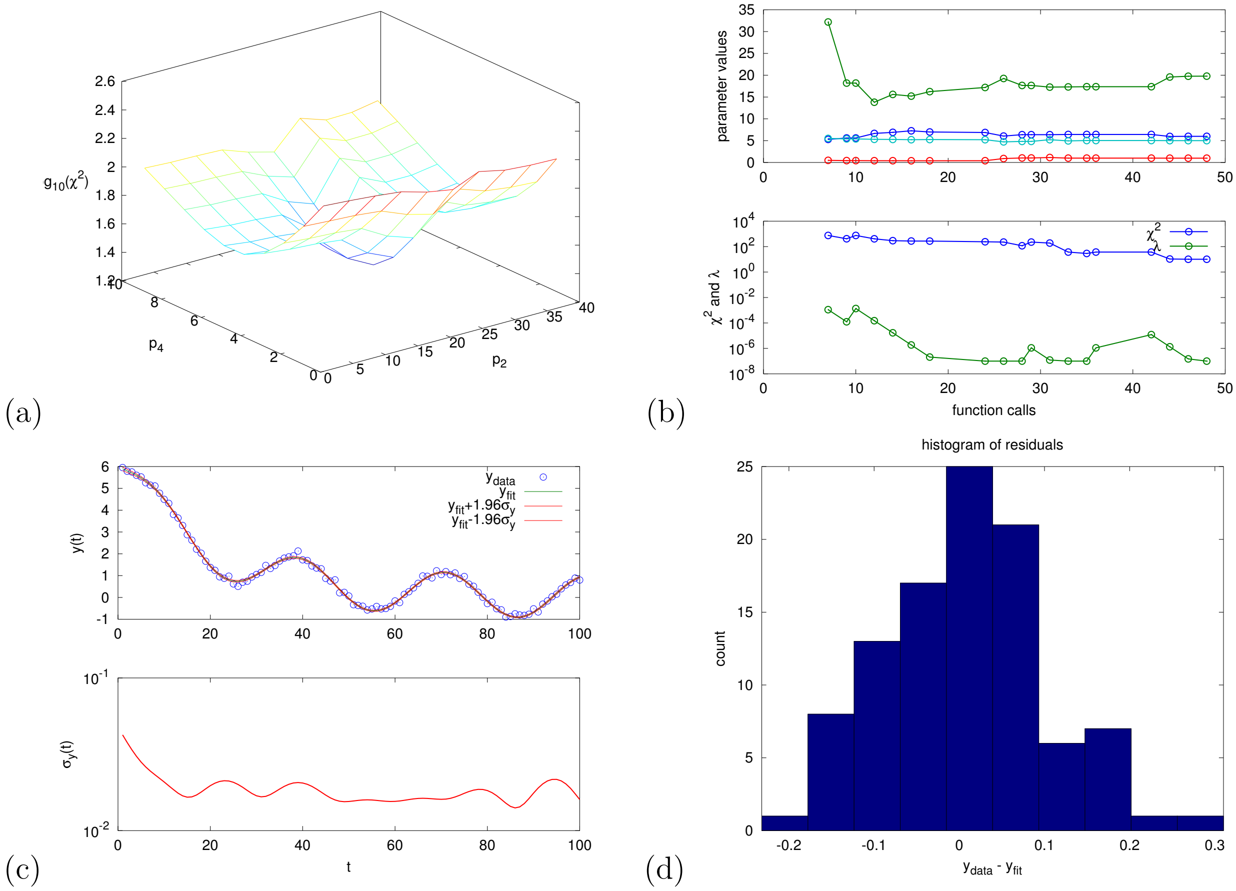

Figure 3. (a) The sum of the squared errors as a function of p_2 and p_4. (b) Top: the convergence of the parameters with each iteration, (b) Bottom: values of χ^2 and λ each iteration. (c) Top: data y, curve-fit $\hat{y}(t;\mathbf{p}_{\text{fit}})$, curve-fit+error, and curve-fit-error; (c) Bottom: standard error of the fit, $\sigma_{\hat{y}}(t)$. (d) Histogram of the errors between the data and the fit.

4.5 Fitting in Multiple Dimensions

The code `lm.m` can carry out fitting in multiple dimensions. For example, the function

$$\hat{z}(x,y) = (p_1 x^{p_2} + (1-p_1) y^{p_2})^{1/p_2}$$

may be fit to data points $z_i(x_i, y_i)$, $(i = 1, \cdots, m)$, using `lm.m` using an m-file such as

```
my_data = load('my_data_file');  % load the data
x_dat = my_data(:,1);            % if the independent variable x is in column 1
y_dat = my_data(:,2);            % if the independent variable y is in column 2
z_dat = my_data(:,3);            % if the   dependent variable z is in column 3

p_min  = [ 0.1  0.1  ];          % minimum expected parameter values
p_max  = [ 0.9  2.0  ];          % maximum expected parameter values
p_init = [ 0.5  1.0  ];          % initial guess for parameter values

t = [ x_dat y_dat ];     % x and y are column vectors of independent variables

[p_fit,Chi_sq,sigma_p,sigma_y,corr,R2,cvg_hst] = ...
                       lm('lm_func2d',p_init,t,z_dat,weight,0.01,p_min,p_max);
```

with the m-function `lm_func2d.m`

```
function z_hat = lm_func2d(t,p)
% example function used for nonlinear least squares curve-fitting
% to demonstrate the Levenberg-Marquardt function, lm.m,
% in two fitting dimensions

    x_dat = t(:,1);
    y_dat = t(:,2);
    z_hat = ( p(1)*x_dat.^p(2) + (1-p(1))*y_dat.^p(2) ).^(1/p(2));
```

5 Remarks

This text and lm.m were written in an attempt to understand and explain methods of nonlinear least squares for curve-fitting applications.

It is important to remember that the purpose of applying statistical methods for data analysis is primarily to estimate parameter statistics . . . not only the parameter values themselves. Reasonable parameter values can often be found using non-statistical minimization algorithms, such as random search methods, the Nelder-Mead simplex method, or simply griding the parameter space and finding the best combination of parameter values.

Nonlinear least squares problems can have objective functions with multiple local minima. Fitting algorithms will converge to different local minima depending upon values of the initial guess, the measurement noise, and algorithmic parameters. It is perfectly appropriate and good to use the best available estimate of the desired parameters as the initial guess. In the absence of physical insight into a curve-fitting problem, a reasonable initial guess may be found by coarsely griding the parameter space and finding the best combination of parameter values. There is no sense in forcing any statistical curve-fitting algorithm to work too hard by starting it with a poor initial guess. In most applications, parameters identified from neighboring initial guesses (±5%) should converge to similar parameter estimates (±0.1%). The fit statistics of these converged parameters should be the same, within two or three significant figures.

References

[1] A. Björck. *Numerical methods for least squares problems,* SIAM, Philadelphia, 1996.

[2] K. Levenberg. "A Method for the Solution of Certain Non-Linear Problems in Least Squares". *The Quarterly of Applied Mathematics,* 2: 164-168 (1944).

[3] M.I.A. Lourakis. *A brief description of the Levenberg-Marquardt algorithm implemented by* levmar*,* Technical Report, Institute of Computer Science, Foundation for Research and Technology - Hellas, 2005.

[4] K. Madsen, N.B. Nielsen, and O. Tingleff. *Methods for nonlinear least squares problems.* Technical Report. Informatics and Mathematical Modeling, Technical University of Denmark, 2004.

[5] D.W. Marquardt. "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics,* 11(2):431-441, 1963.

[6] H.B. Nielson, *Damping Parameter In Marquardt's Method,* Technical Report IMM-REP-1999-05, Dept. of Mathematical Modeling, Technical University Denmark.

[7] W.H. Press, S.A. Teukosky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C,* Cambridge University Press, second edition, 1992.

[8] Mark K.Transtrum, Benjamin B. Machta, and James P. Sethna, "Why are nonlinear fits to data so challenging?", *Phys. Rev. Lett.* 104, 060201 (2010),

[9] Mark K. Transtrum and James P. Sethna "Improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization," Preprint submitted to *Journal of Computational Physics,* January 30, 2012.