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

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

November 27, 2022

Abstract

The Levenberg-Marquardt algorithm was developed in the early 1960's to solve nonlinear least squares problems. Least squares problems arise in the context of fitting a parameterized mathematical model to a set of data points by minimizing an objective expressed as the sum of the squares of the errors between the model function and a set of data points. If a model is linear in its parameters, the least squares objective is quadratic in the parameters. This objective may be minimized with respect to the parameters in one step via the solution to a linear matrix equation. If the fit function is not linear in its parameters, the least squares problem requires an iterative solution algorithm. Such algorithms reduce the sum of the squares of the errors between the model function and the data points through a sequence of well-chosen updates to values of the model parameters. The Levenberg-Marquardt algorithm combines two numerical minimization algorithms: 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 in the parameters, and finding the minimum of this 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 model function $\hat{y}(t;\boldsymbol{p})$ of an independent variable t and a vector of n parameters $\boldsymbol{p}$ 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 data y_i and the curve-fit function $\hat{y}(t;\boldsymbol{p})$.

$$\chi^2(\boldsymbol{p}) = \sum_{i=1}^{m} \left[\frac{y(t_i) - \hat{y}(t_i;\boldsymbol{p})}{\sigma_{y_i}} \right]^2 \tag{1}$$

$$= (\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{p}))^{\mathsf{T}} \boldsymbol{W} (\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{p})) \tag{2}$$

$$= \boldsymbol{y}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{y} - 2\boldsymbol{y}^{\mathsf{T}} \boldsymbol{W} \hat{\boldsymbol{y}} + \hat{\boldsymbol{y}}^{\mathsf{T}} \boldsymbol{W} \hat{\boldsymbol{y}} \tag{3}$$

where σ_{y_i} is the measurement error for datum $y(t_i)$. Typically the weighting matrix $\boldsymbol{W}$ is diagonal with $W_{ii} = 1/\sigma_{y_i}^2$. More formally, $\boldsymbol{W}$ can be set to the inverse of the measurement

error covariance matrix, in the unusual case that it is known. More generally, the weights W_{ii}, can be set to pursue other curve-fitting goals. This scalar-valued goodness-of-fit measure is called the *chi-squared error criterion* because the sum of squares of normally-distributed random variables is distributed as the chi-squared distribution.

If the function $\hat{y}(t;\boldsymbol{p})$ is nonlinear in the model parameters $\boldsymbol{p}$, 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 $\boldsymbol{h}$ to the parameters $\boldsymbol{p}$ 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 [6, 7]. For problems with thousands of parameters, gradient descent methods are sometimes the only viable choice.

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

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

where the $m \times n$ Jacobian matrix $[\partial\hat{\boldsymbol{y}}/\partial\boldsymbol{p}]$ represents the local sensitivity of the function $\hat{y}$ to variation in the parameters $\boldsymbol{p}$. For notational simplicity the variable $\boldsymbol{J}$ will be used for $[\partial\hat{\boldsymbol{y}}/\partial\boldsymbol{p}]$. Note that in models that are linear in the parameters, $\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{p}$, the Jacobian $[\partial\hat{\boldsymbol{y}}/\partial\boldsymbol{p}]$ is the matrix of model basis vectors $\boldsymbol{X}$. The parameter update $\boldsymbol{h}$ that moves the parameters in the direction of steepest descent is given by

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

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 [2]. For moderately-sized problems the Gauss-Newton method typically converges much faster than gradient-descent methods [8].

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

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

Substituting the approximation $\hat{\boldsymbol{y}}(\boldsymbol{p}+\boldsymbol{h}) \approx \hat{\boldsymbol{y}}(\boldsymbol{p}) + \boldsymbol{J}\boldsymbol{h}$ into equation (3) for $\chi^2(\boldsymbol{p}+\boldsymbol{h})$,

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

The first-order Taylor approximation (8) results in an approximation for χ^2 that is quadratic in the perturbation $\boldsymbol{h}$. The Hessian of the chi-squared fit criterion is approximately $\boldsymbol{J}^\mathsf{T}\boldsymbol{W}\boldsymbol{J}$.

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

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

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

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

Note that the right hand side vectors in normal equations for the gradient descent method (7) and the Gauss-Newton method (11) are identical.

4 The Levenberg-Marquardt Method

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

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

where small values of the *damping parameter* λ result in a Gauss-Newton update and large values of λ result in a gradient descent update. The damping parameter λ is initialized to be large so that first updates are small steps in the steepest-descent direction. If any iteration happens to result in a worse approximation ($\chi^2(\boldsymbol{p}+\boldsymbol{h}_{\mathsf{lm}}) > \chi^2(\boldsymbol{p})$), then λ is increased. Otherwise, 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 [6, 7, 8].

In Marquardt's update relationship [8], the damping parameter λ is scaled by the diagonal of the Hessian $\boldsymbol{J}^\mathsf{T}\boldsymbol{W}\boldsymbol{J}$ for each parameter.

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

4.1 Numerical Implementation

Many variations of the Levenberg-Marquardt have been published in papers and in code, e.g., [4, 6, 9, 10, 11]. This document borrows from some of these. In iteration i, the step $\boldsymbol{h}$ is evaluated by comparing $\chi^2(\boldsymbol{p})$ to $\chi^2(\boldsymbol{p}+\boldsymbol{h})$. The step is accepted if the metric ρ_i [9] 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.

$$\rho_i(\boldsymbol{h}_{\mathsf{lm}}) = \frac{\chi^2(\boldsymbol{p}) - \chi^2(\boldsymbol{p}+\boldsymbol{h}_{\mathsf{lm}})}{\mid (\boldsymbol{y}-\hat{\boldsymbol{y}})^{\mathsf{T}}\boldsymbol{W}(\boldsymbol{y}-\hat{\boldsymbol{y}}) - (\boldsymbol{y}-\hat{\boldsymbol{y}}-\boldsymbol{J}\boldsymbol{h}_{\mathsf{lm}})^{\mathsf{T}}\boldsymbol{W}(\boldsymbol{y}-\hat{\boldsymbol{y}}-\boldsymbol{J}\boldsymbol{h}_{\mathsf{lm}}) \mid} \tag{14}$$

$$= \frac{\chi^2(\boldsymbol{p}) - \chi^2(\boldsymbol{p}+\boldsymbol{h}_{\mathsf{lm}})}{\mid \boldsymbol{h}_{\mathsf{lm}}^{\mathsf{T}}\left(\lambda_i\boldsymbol{h}_{\mathsf{lm}} + \boldsymbol{J}^{\mathsf{T}}\boldsymbol{W}\left(\boldsymbol{y}-\hat{\boldsymbol{y}}(\boldsymbol{p})\right)\right) \mid} \quad \text{if using eq'n (12) for } \boldsymbol{h}_{\mathsf{lm}} \tag{15}$$

$$= \frac{\chi^2(\boldsymbol{p}) - \chi^2(\boldsymbol{p}+\boldsymbol{h}_{\mathsf{lm}})}{\mid \boldsymbol{h}_{\mathsf{lm}}^{\mathsf{T}}\left(\lambda_i\mathsf{diag}(\boldsymbol{J}^{\mathsf{T}}\boldsymbol{W}J)\boldsymbol{h}_{\mathsf{lm}} + \boldsymbol{J}^{\mathsf{T}}\boldsymbol{W}\left(\boldsymbol{y}-\hat{\boldsymbol{y}}(\boldsymbol{p})\right)\right) \mid} \quad \text{if using eq'n (13) for } \boldsymbol{h}_{\mathsf{lm}} \tag{16}$$

If in an iteration $\rho_i(\boldsymbol{h}) > \epsilon_4$ then $\boldsymbol{p}+\boldsymbol{h}$ is sufficiently better than $\boldsymbol{p}$, $\boldsymbol{p}$ is replaced by $\boldsymbol{p}+\boldsymbol{h}$, 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 initializing and updating λ and $\boldsymbol{p}$.

1. $\lambda_0 = \lambda_\mathsf{o}$; λ_o is user-specified [8].
   use eq'n (13) for $\boldsymbol{h}_{\mathsf{lm}}$ and eq'n (16) for ρ
   if $\rho_i(\boldsymbol{h}) > \epsilon_4$: $\boldsymbol{p} \leftarrow \boldsymbol{p}+\boldsymbol{h}$; $\lambda_{i+1} = \max[\lambda_i/L_\downarrow, 10^{-7}]$;
   otherwise: $\lambda_{i+1} = \min\left[\lambda_i L_\uparrow, 10^7\right]$;

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

3. $\lambda_0 = \lambda_\mathsf{o} \max\left[\mathsf{diag}[\boldsymbol{J}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{J}]\right]$; λ_o is user-specified [9].
   use eq'n (12) for $\boldsymbol{h}_{\mathsf{lm}}$ and eq'n (15) for ρ
   if $\rho_i(\boldsymbol{h}) > \epsilon_4$: $\boldsymbol{p} \leftarrow \boldsymbol{p}+\boldsymbol{h}$; $\lambda_{i+1} = \lambda_i \max\left[1/3, 1-(2\rho_i-1)^3\right]$; $\nu_i = 2$;
   otherwise: $\lambda_{i+1} = \lambda_i\nu_i$; $\quad \nu_{i+1} = 2\nu_i$;

For the examples in section 4.4, method 1 [8] with $L_\uparrow \approx 11$ and $L_\downarrow \approx 9$ exhibits good convergence properties.

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

For problems with many parameters, a finite differences Jacobian is computationally expensive. If the Jacobian is re-computed using finite differences only occasionally, convergence can be achieved with fewer function evaluations. In the first iteration, in every $2n$

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

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

or central differences (default)

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

where the j-th element of $\delta\boldsymbol{p}_j$ 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,

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

This 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: $\max \left| \boldsymbol{J}^\mathsf{T}\boldsymbol{W}(\boldsymbol{y}-\hat{\boldsymbol{y}}) \right| < \epsilon_1$;
- Convergence in parameters: $\max |h_i/p_i| < \epsilon_2$; or
- Convergence in χ^2: uses the value of the *reduced* χ^2, $\chi^2_\nu = \chi^2/(m-n+1) < \epsilon_3$.

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

4.2 Error Analysis

Once the optimal curve-fit parameters $\boldsymbol{p}_{\text{fit}}$ are determined, parameter statistics are computed for the converged solution. If the measurement error covariance matrix $\boldsymbol{V}_y$, or its diagonal, $\sigma^2_{y_i}$, is known a priori, (prior to the curve-fit), the weighting matrix should be set to the inverse of the measurement error covariance, $\boldsymbol{W} = \boldsymbol{V}_y^{-1}$, in estimating the model parameters and in the following error analysis. Note that if the actual measurement errors vary significantly across the measurement points (i.e., $\max_i(\sigma_{y_i})/\min_i(\sigma_{y_i}) > 10$), any error analysis that presumes equal measurement errors will be incorrect.

The reduced χ^2 error criterion,

$$\chi^2_\nu = \frac{\chi^2}{m-n+1} = \frac{1}{m-n+1}(\boldsymbol{y}-\hat{\boldsymbol{y}}(\boldsymbol{p}_{\text{fit}}))^\mathsf{T}\boldsymbol{W}(\boldsymbol{y}-\hat{\boldsymbol{y}}(\boldsymbol{p}_{\text{fit}})) \tag{20}$$

is a measure of the quality of the fit. Large values, $\chi^2_\nu \gg 1$, indicate a poor fit, $\chi^2_\nu \approx 1$ indicates that the fit error is of the same order as the measurement error (as desired), and

$\chi^2_\nu < 1$ indicates that the model is over-fitting the data; that is, the model is fitting the measurement noise.

The parameter covariance matrix is computed from

$$\boldsymbol{V}_p = \left[\boldsymbol{J}^\mathsf{T}\boldsymbol{W}\boldsymbol{J}\right]^{-1} \quad . \tag{21}$$

The asymptotic standard parameter errors,

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

give 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 asymptotic standard error of the fit,

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

indicates how variability in the parameters affects the variability in the curve-fit.

The asymptotic standard prediction error,

$$\sigma_{\hat{y}_p} = \sqrt{\mathsf{diag}\left(\boldsymbol{V}_y + \boldsymbol{J}\left[\boldsymbol{J}^\mathsf{T}\boldsymbol{W}\boldsymbol{J}\right]^{-1}\boldsymbol{J}^\mathsf{T}\right)} \, . \tag{24}$$

reflects the standard error of the fit as well as the measurement error.

If the measurement error covariance, or individual measurement errors are *not known* in advance of the analysis, the error analysis can be carried out assuming the same measurement error for every measurement point, as estimated from the fit,

$$\hat{\sigma}_y^2 = \frac{1}{m-n+1}(\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{p}_{\mathsf{fit}}))^\mathsf{T}(\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{p}_{\mathsf{fit}})) \, . \tag{25}$$

In this case $\boldsymbol{V}_y$ is set to $\hat{\sigma}_y^2\boldsymbol{I}$ and $\boldsymbol{W}$ is set to $\boldsymbol{I}/\hat{\sigma}_y^2$ in equations (21) to (24). Note also, that if $\boldsymbol{W} = \boldsymbol{I}/\hat{\sigma}_y^2$, then $\chi^2_\nu = 1$, by definition.

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

```
function [p,redX2,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,redX2,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      = initial guess of parameter values                                  (n x 1)
% t      = independent variables (used as arg to func)                        (m x 1)
% y_dat  = data to be fit by func(t,p)                                        (m x 1)
% weight = weights or a scalar weight value ( weight >= 0 ) ...               (m x 1)
%          inverse of the standard measurement errors
%          Default:  ( 1 / ( 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  = lower bounds for parameter values                                  (n x 1)
% p_max  = upper bounds for parameter values                                  (n x 1)
% 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-1      convergence tolerance for red. 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
% redX2   = reduced Chi squared error criteria - should be close to 1
% 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 ... see lm_plots.m
```

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.

Tips for successful use of lm.m:

- The data vector `t` *should* be a *column* vector, or columns of `t` *must* correspond to columns of `y_dat`.
- The data vector `y_dat` *should* be a *column* vector.
- Your .m-function `lm_func.m` *must* return the vector `y_hat` as a *column* vector.
- The vectors `p_init`, `p_min`, and `p_max` *must* be *column* vectors.
- Parameter values should be scaled to values in a compact range, for example, such that absolute parameters values are between 1 and 100.

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

%   Henri Gavin, Dept. Civil & Environ. Engineering, Duke Univ. 2 May 2016

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

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

figure(101);  % plot convergence history of parameters, reduced 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(cvg_hst(1,1),cvg_hst(1,n+2), '\chi^2_\nu','FontSize',16,'color','k');
   text(cvg_hst(1,1),cvg_hst(1,n+3), '\lambda', 'FontSize',16, 'color','k');
   text(cvg_hst(max_it,1),cvg_hst(max_it,n+2), '\chi^2_\nu','FontSize',16,'color','k');
   text(cvg_hst(max_it,1),cvg_hst(max_it,n+3), '\lambda', 'FontSize',16, 'color','k');
   ylabel('\chi^2_\nu and \lambda')
   xlabel('function calls')
   print(sprintf('%sA.pdf', filename),'-dpdfcrop');

figure(102); % ------------ plot data, fit, and confidence interval of fit

 patchColor95 = [ 0.95, 0.95, 0.1 ];
 patchColor99 = [ 0.2,  0.95, 0.2 ];
 tp  = [ t   ; t(end:-1:1)   ; t(1)   ];            % x coordinates for patch
 yps95 =  y_fit + 1.96*sigma_y;                      %  + 95 CI
 yms95 =  y_fit - 1.96*sigma_y;                      %  - 95 CI
 yps99 =  y_fit + 2.58*sigma_y;                      %  + 99 CI
 yms99 =  y_fit - 2.58*sigma_y;                      %  - 99 CI
 yp95  = [ yps95 ; yms95(end:-1:1) ; yps95(1) ];    % y coordinates for patch
 yp99  = [ yps99 ; yms99(end:-1:1) ; yps99(1) ];    % y coordinates for patch

 clf
 hold on
   hc99 = patch(tp, yp99, 'FaceColor', patchColor99, 'EdgeColor', patchColor99);
   hc95 = patch(tp, yp95, 'FaceColor', patchColor95, 'EdgeColor', patchColor95);
   hd   = plot(t,y_dat,'ob');
   hf   = plot(t,y_fit,'-k');
 hold off
  axis('tight')
  legend([hd,hf,hc95,hc99], 'y_{data}','y_{fit}','95% c.i.','99% c.i.');
  ylabel('y(t)')
  xlabel('t')
  print(sprintf('%sB.pdf', filename),'-dpdfcrop');

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')
  print(sprintf('%sC.pdf', filename),'-dpdfcrop');
```

6 Examples

In this section, the use of `lm.m` is illustrated in three curve-fitting examples in which experimental measurements are numerically simulated. Noisy experimental measurements y are simulated by adding random measurement noise to the curve-fit function evaluated with a set of "true" parameter values $\hat{y}(t; \boldsymbol{p}_{\mathsf{true}})$. The random measurement noise is normally distributed with a mean of zero and a standard deviation of 0.50.

$$y_i = \hat{y}(t_i; \boldsymbol{p}_{\mathsf{true}}) + \mathcal{N}(0, 0.50). \tag{26}$$

The convergence of the parameters from an erroneous initial guess $\boldsymbol{p}_{\mathsf{init}}$ to values closer to $\boldsymbol{p}_{\mathsf{true}}$ 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 $\hat{y}(t; \boldsymbol{p})$, a different "true" parameter vector $\boldsymbol{p}_{\mathsf{true}}$, and a different vector of initial parameters $\boldsymbol{p}_{\mathsf{init}}$.

For several values of p_2 and p_4, the log of the reduced 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. In some cases, the objective function is not quadratic in the parameters or the objective function has 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 $\chi^2(\boldsymbol{p})$. 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_ν, and the evolution of λ from iteration to iteration is plotted for each example.

The simulated experimental data, the curve fit, and the 99-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 $\boldsymbol{p}_{\mathsf{init}}$, the true parameter values $\boldsymbol{p}_{\mathsf{true}}$, the fit parameter values $\boldsymbol{p}_{\mathsf{fit}}$, the standard error of the fit parameters $\sigma_{\boldsymbol{p}}$, and the correlation matrix of the fit parameters are tabulated. The true parameter values lie within the confidence interval $p_{\mathsf{fit}} - 2.58\sigma_p < p_{\mathsf{true}} < p_{\mathsf{fit}} + 2.58\sigma_p$ with a confidence level of 99 percent.

6.1 Example 1

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

$$\hat{y}(t;\boldsymbol{p}) = p_1 \exp(-t/p_2) + p_3 t \exp(-t/p_4) \tag{27}$$

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 $\boldsymbol{p}_{\mathsf{true}}$, the initial parameter values $\boldsymbol{p}_{\mathsf{init}}$, resulting curve-fit parameter values $\boldsymbol{p}_{\mathsf{fit}}$ and standard errors of the fit parameters $\sigma_{\boldsymbol{p}}$ are shown in Table 1. The R^2 fit criterion is 89 percent and the reduced $\chi^2_\nu = 1.004$. The standard parameter errors are one to five percent of the parameter values. 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_3 are the least correlated at +27 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_ν 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.

$\boldsymbol{p}_{\text{init}}$	$\boldsymbol{p}_{\text{true}}$	$\boldsymbol{p}_{\text{fit}}$	$\sigma_{\boldsymbol{p}}$	$\sigma_{\boldsymbol{p}}/\boldsymbol{p}_{\text{fit}}$ (%)
5.0	20.0	19.375	0.389	2.01
2.0	10.0	10.367	0.377	3.64
0.2	1.0	0.991	0.015	1.56
10.0	50.0	50.472	0.540	1.06

Table 2. Parameter correlation matrix.

	p_1	p_2	p_3	p_4
p_1	1.00	-0.68	0.27	-0.34
p_2	-0.68	1.00	-0.78	0.80
p_3	0.27	-0.78	1.00	-0.96
p_4	-0.34	0.80	-0.96	1.00

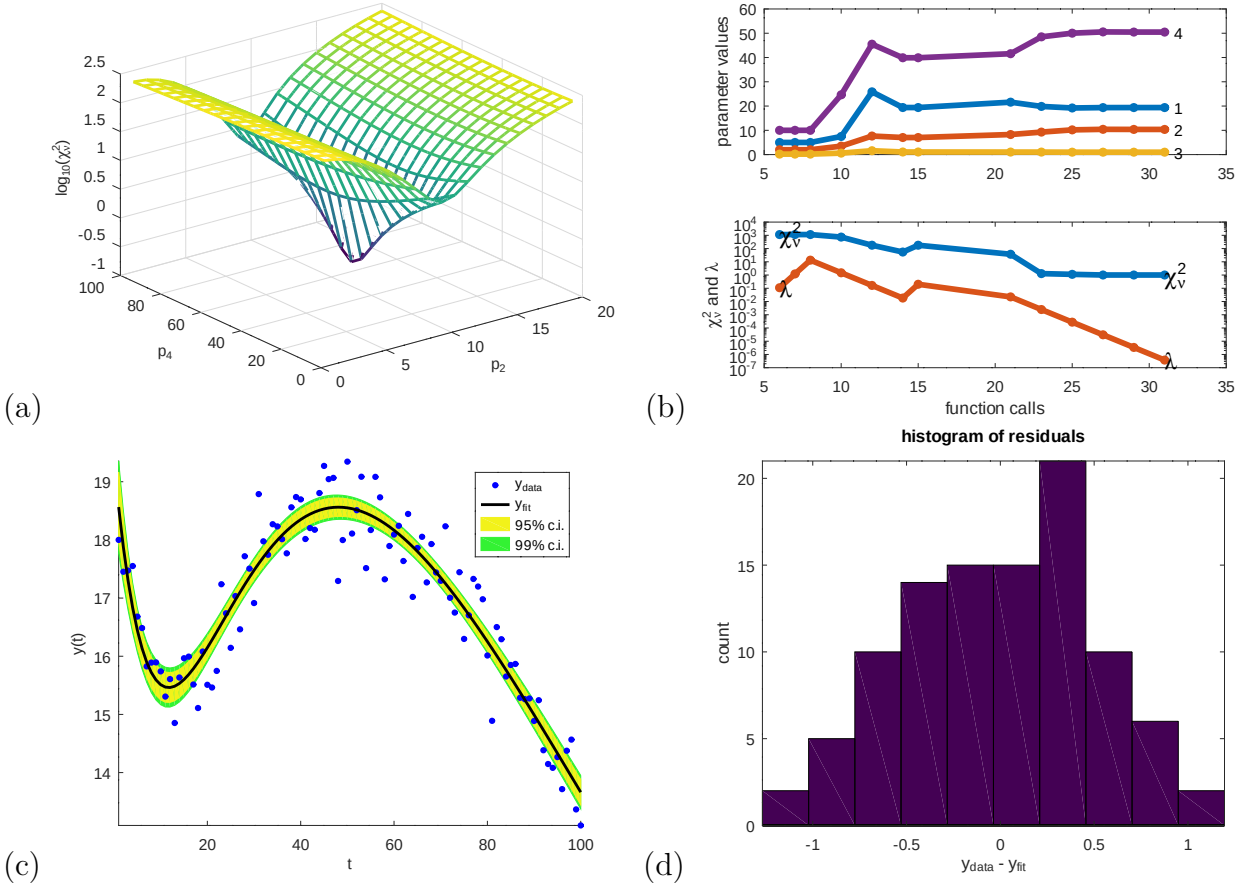

Figure 1. (a) The sum of the squared errors as a function of p_2 and p_4. (b) Top: the convergence of the parameters with each iteration, (b) Bottom: values of χ^2_ν and λ each iteration. (c) Top: data y, curve-fit $\hat{y}(t;\boldsymbol{p}_{\text{fit}})$, curve-fit confidence intervals; (d) Histogram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

6.2 Example 2

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

$$\hat{y}(t;\boldsymbol{p}) = p_1(t/\max(t)) + p_2(t/\max(t))^2 + p_3(t/\max(t))^3 + p_4(t/\max(t))^4 \qquad (28)$$

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)
   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 $\boldsymbol{p}_{\text{true}}$, the initial parameter values $\boldsymbol{p}_{\text{init}}$, resulting curve-fit parameter values $\boldsymbol{p}_{\text{fit}}$ and standard errors of the fit parameters $\sigma_{\boldsymbol{p}}$ are shown in Table 3. The R^2 fit criterion is 99.9 percent and $\chi^2_\nu = 1.025$. In this example, the standard parameter errors are larger than in example 1. The standard error for p_1 is 8 percent and the standard error of p_3 is 47 percent of the estimated value. 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 high value of R^2 merely indicates that the fit is highly correlated with the data. In this example the high R^2 value is a result of relatively low measurement noise (compared to the data values) and a fit that passes through the data points. 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 high values of parameter standard errors, parameter correlations, and χ^2_ν indicate that $\hat{y}(t;\boldsymbol{p})$ is over-parameterized.

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_ν 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 $\hat{y}(0;\boldsymbol{p}) = 0$, regardless of the values in $\boldsymbol{p}$.

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.

$\boldsymbol{p}_{\text{init}}$	$\boldsymbol{p}_{\text{true}}$	$\boldsymbol{p}_{\text{fit}}$	$\sigma_{\boldsymbol{p}}$	$\sigma_{\boldsymbol{p}}/\boldsymbol{p}_{\text{fit}}$ (%)
4.0	20.0	19.769	1.741	8.80
-5.0	-24.0	-24.364	9.394	38.55
6.0	30.0	33.379	15.801	47.33
10.0	-40.0	-42.843	8.321	19.42

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

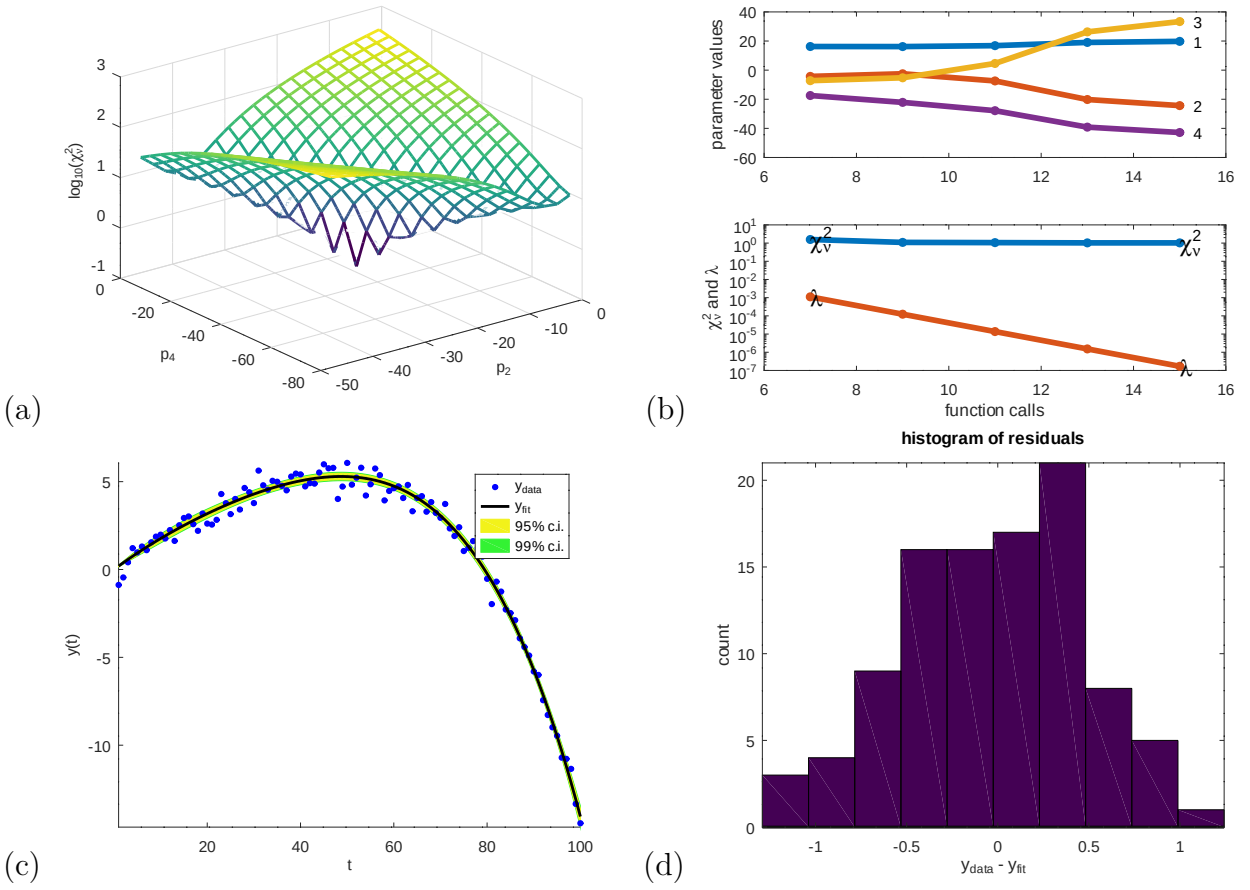

Figure 2. (a) The sum of the squared errors as a function of p_2 and p_4. (b) Top: the convergence of the parameters with each iteration, (b) Bottom: values of χ^2_ν and λ each iteration. (c) Top: data y, curve-fit $\hat{y}(t;\boldsymbol{p}_{\text{fit}})$, curve-fit confidence intervals; (d) Histogram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

6.3 Example 3

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

$$\hat{y}(t;\boldsymbol{p}) = p_1 \exp(-t/p_2) + p_3 \sin(t/p_4) \tag{29}$$

This function is linear p_1 and p_3 but not in p_2 and p_4. 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 $\boldsymbol{p}_{\text{true}}$, the initial parameter values $\boldsymbol{p}_{\text{init}}$, resulting curve-fit parameter values $\boldsymbol{p}_{\text{fit}}$ and standard errors of the fit parameters $\sigma_{\boldsymbol{p}}$ are shown in Table 5. The R^2 fit criterion is 93 percent and $\chi^2_\nu = 0.921$. In this example, the standard parameter errors are all less than ten 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 $\sigma_{\boldsymbol{p}}$ 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_ν 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.

$\boldsymbol{p}_{\text{init}}$	$\boldsymbol{p}_{\text{true}}$	$\boldsymbol{p}_{\text{fit}}$	$\sigma_{\boldsymbol{p}}$	$\sigma_{\boldsymbol{p}}/\boldsymbol{p}_{\text{fit}}$ (%)
10.0	6.0	5.486	0.230	4.20
50.0	20.0	21.995	1.286	5.85
6.0	1.0	1.120	0.074	6.57
5.7	5.0	5.016	0.027	0.53

Table 6. Parameter correlation matrix.

	p_1	p_2	p_3	p_4
p_1	1.00	-0.73	-0.28	-0.01
p_2	-0.73	1.00	0.20	0.01
p_3	-0.28	0.20	1.00	-0.03
p_4	-0.01	0.01	-0.03	1.00

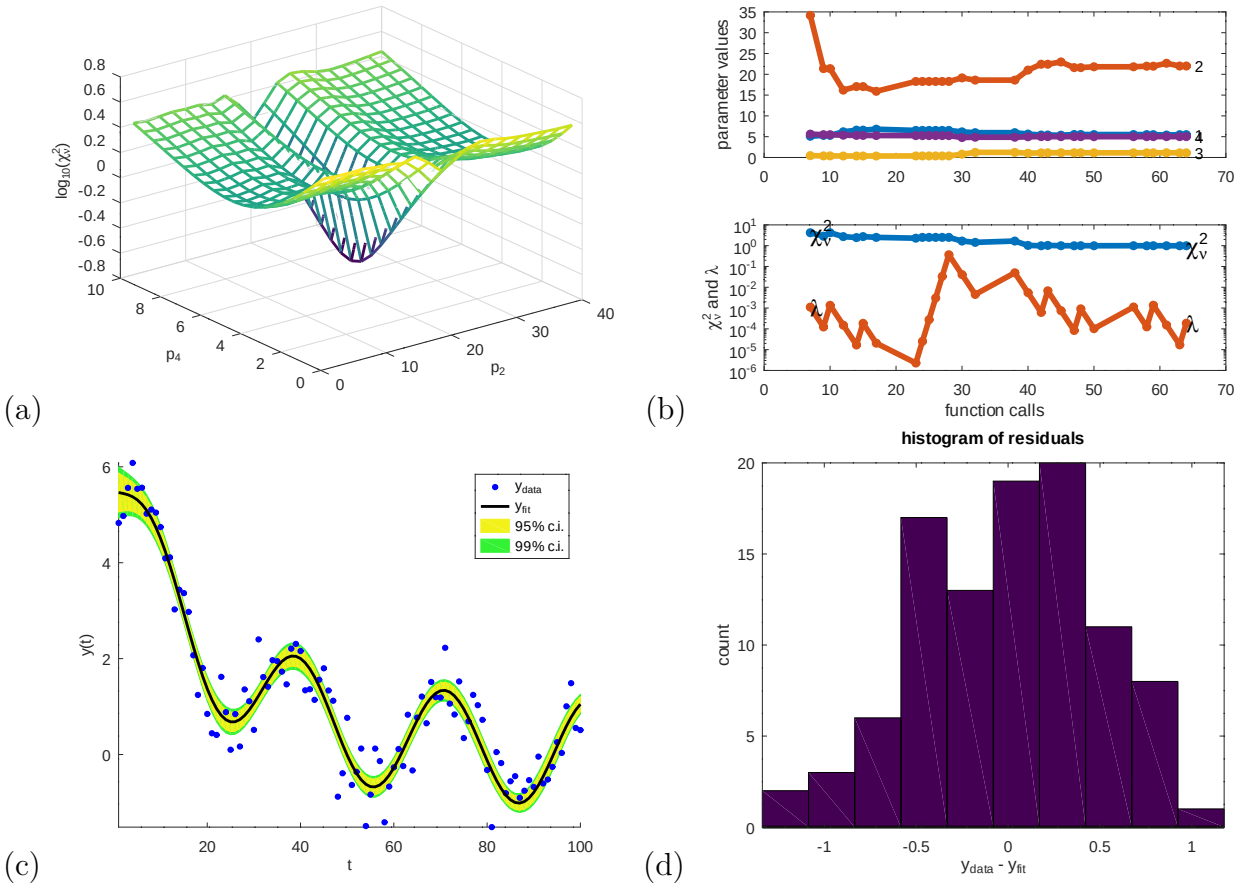

Figure 3. (a) The sum of the squared errors as a function of p_2 and p_4. (b) Top: the convergence of the parameters with each iteration, (b) Bottom: values of χ^2_ν and λ each iteration. (c) Top: data y, curve-fit $\hat{y}(t;\boldsymbol{p}_{\text{fit}})$, curve-fit confidence intervals; (d) Histogram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

6.4 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` with a `.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);
```

and 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));
```

7 Remarks

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

Least squares methods of curve-fitting are useful in computing the values *and* the standard errors of parameter estimates. Nonlinear least squares problems iterate from an initial parameter values, provided by a user based on their experience or possibly calculated from non-statistical minimization algorithms, such as random search methods, the Nelder-Mead simplex method, or coarsly griding the parameter space and finding the best combination of parameter values.

Least squares problems can be nonlinear and convex (exapmles 1), linear and convex (example 2) or non-convex (example 3) in which the χ^2 objective function has multiple local minima and in which fitting algorithms can converge to different local minima depending upon the initial parameter values, the measurement noise, and algorithmic parameters. It is always perfectly appropriate and good to set the initial guess to the best available parameter estimates. 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 iterative curve-fitting algorithm to work too hard by initializing it with a random or otherwise intentionally poor initial guess. In most applications of Levenberg Marquardt, parameters identified from neighboring initial guesses ($\pm 5\%$) should converge to similar parameter estimates ($\pm 0.1\%$). Further, the fit statistics of these converged parameters should be the same within two or three significant figures.

The implementation of Levemberg-Marquardt described here offers the user options to customize the method to their application. The starting value of the damping parameter, λ_0, can optionally be specified in `opts(7)` of `lm.m`. The updating method for λ and $\boldsymbol{h}$ can optionally be specified in `opts(8)`, `opts(9)` and `opts(10)`, as described in section 4.1.1 above. In the default update option (1), λ is scaled by the diagonal of the Hessian of χ^2, effectively providing a different value of λ for each parameter. This is helpful for problems involving broadly-ranging parameter values. In update options (2) and (3), λ and $\boldsymbol{h}$ are scaled by scalar maximum of the diagonal of the Hessian of χ^2. The three examples in this manuscript are run with the default case. In example 1 λ increases in the first two steps (Figure 1), but then decreases. In example 2, a *linear* least squares problem, λ decreases exponentially from its initial value of 0.01 (Figure 2). And in example 3, a non-convex problem, λ jumps around, and increases exponentially from 10^{-6} to 10^{-1} from function evaluation #22 to #28. The "best" initialization and updating method is problem-dependent. Only if convergence rates or converged results are somehow inadequate would it be worth trying different values of λ_0 or different updating methods.

References

[1] Y. Bard, *Nonlinear Parameter Estimation*, Academic Press, 1974.

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

[3] N.R. Draper and H. Smith, `Applied Regression Analysis`, John Wiley and Sons, 1981.

[4] Carsten Grammes. "fit.c" Gnuplot source code. http://www.gnuplot.info

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] Richard Shrager, Arthur Jutan, Ray Muzic, and Olaf Till, "leasqr.m" 1992-2016 *Octave-Forge, A collection of packages providing extra functionality for GNU Octave*

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

[13] 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.