MSI Lecture
5.5
Practical solution of the Nonlinear Least Squares Problem

Per Rutquist

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_y of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_\theta \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\begin{aligned}
\theta_{[k+1]} &= \arg\min_\theta \frac{1}{2}\Big\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 && (5.31)\\
&= \arg\min_\theta \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\Big\|_2^2 && (5.32)\\
&= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) && (5.33)\\
&= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) && (5.34)\\
&= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) && (5.35)
\end{aligned}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$\begin{aligned}
f(\theta) &= \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^N R_i(\theta)^2 && (5.36)\\
\nabla f(\theta) &= \sum_{i=1}^N \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) && (5.37)\\
\nabla^2 f(\theta) &= \underbrace{J(\theta)^\top J(\theta)}_{=:B_{GN}(\theta)} + \sum_{i=1}^N \nabla^2 R_i(\theta) R_i(\theta) && (5.38)
\end{aligned}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{GN}(\theta_{[k]})^{-1}\nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{GN}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{GN}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{GN}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_y). Following the linear least squares analysis, we could then directly use $B_{GN}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_y, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

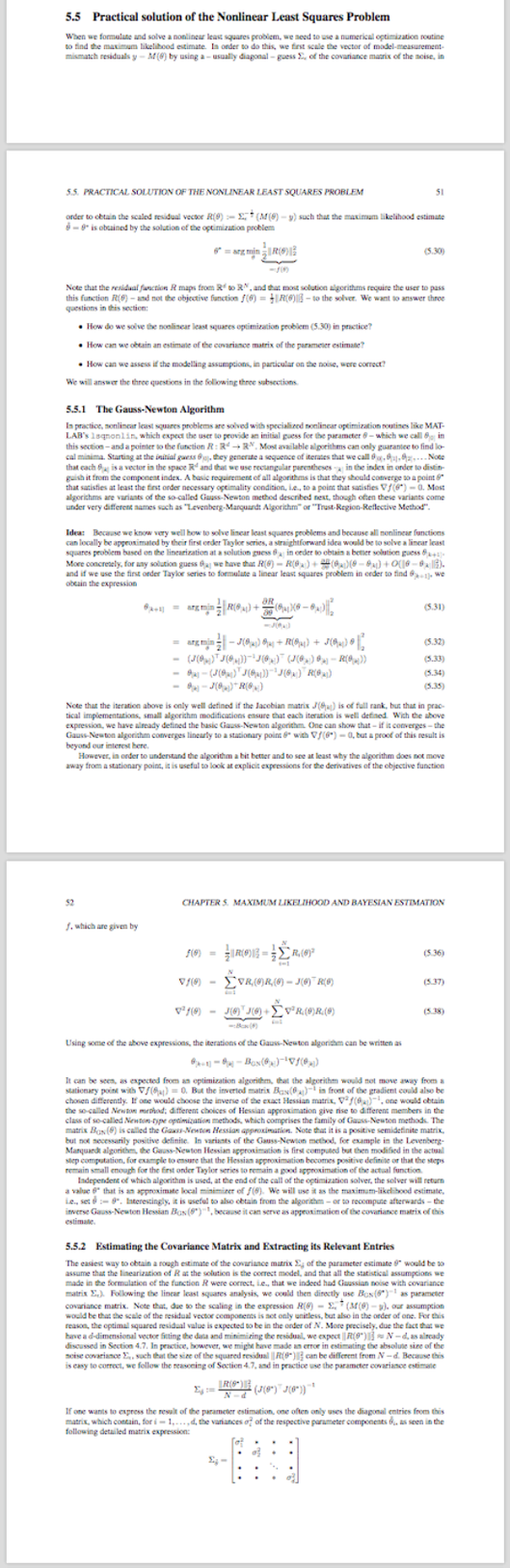
Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.



5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_* of the covariance matrix of the noise, in

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 51

order to obtain the scaled residual vector $R(\theta) := \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\Big\|\underbrace{R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]})}_{=:J(\theta_{[k]})}\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta \Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top \left(J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})\right) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

52 CHAPTER 5. MAXIMUM LIKELIHOOD AND BAYESIAN ESTIMATION

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_*). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_*, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 53

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

ML estimation

- For all possible parameter values:
  - Compute likelihood of the given observation(s)
- Pick the most likely!

Example problem

- Ferris wheel with (noisy) altimeters

Ferris Wheel

$$\sin(a) \approx -0.1$$

$$\sin(a + 30^\circ) \approx 0.6$$

$$\sin(a + 60^\circ) \approx 0.9$$

- Find the angle *a !*

Ferris Wheel

$$\sin(a) + \epsilon_1 = -0.1$$

$$\sin(a + 30^\circ) + \epsilon_2 = 0.6$$

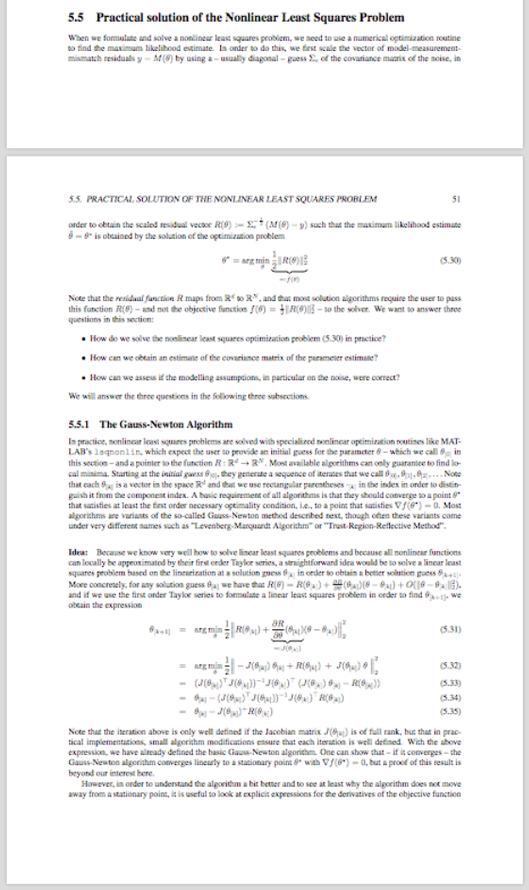
$$\sin(a + 60^\circ) + \epsilon_3 = 0.9$$

- Assume that all *ϵ are independent, and drawn from a normal distribution with zero mean and standard deviation 0.5*

- Find the most likely *a !*

Matlab demo

```
% Defining a nonlinear model M
M = @(a) sin(a + pi/6 * [0 1 2]')

% Our noise covariance matrix
C = diag(0.5^2 * [1 1 1])

% Square root of the covariance
S = sqrtm(C)

% Measurements
y = [-0.1, 0.6, 0.9]'

% The optimally weighted residual function
R = @(a) inv(S) * (M(a) - y)

% Likelihood
% (During the lecure, I forgot the "0.5" below. Sorry about that! /Per)
p = @(a) exp(-0.5*sum(R(a).^2))

% A vector of values to try for the angle a
as = -pi:0.01:pi;

% Evaluate p for each element of the vector
% Note: Matlab automatically "broadcasts" opearations like exp, * and + across dimensions
ps = p(as);

% Alternatively, we could have written:
% ps = arrayfun(p, as);

% Another way would be:
% ps = zeros(size(as));
% for i=1:length(ps)
%    ps(i) = p(as(i));
% end

% Plot p as funcition of a
plot(as, ps)

% Find the maximum and its location:
[pmax, ix] = max(ps)
a = as(ix)
```



5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_* of the covariance matrix of the noise, in

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 51

order to obtain the scaled residual vector $R(\theta) := \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$ in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\left\| R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]}) \right\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\left\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta \right\|_2^2 \tag{5.32}$$

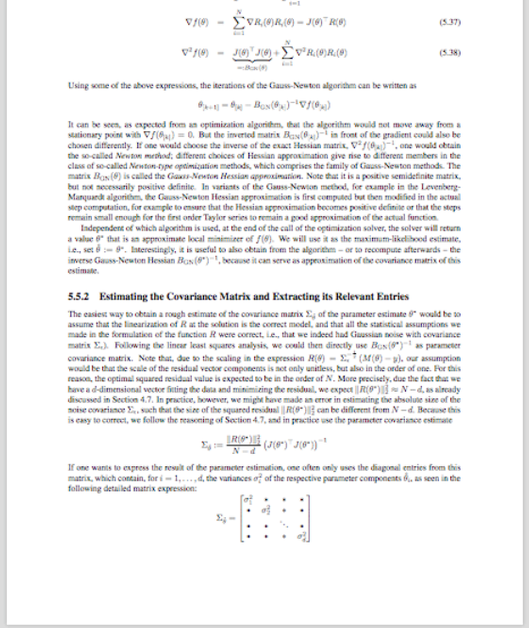
$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top \left(J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})\right) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function



52 CHAPTER 5. MAXIMUM LIKELIHOOD AND BAYESIAN ESTIMATION

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_*). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_*, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 53

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\hat{\theta}_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

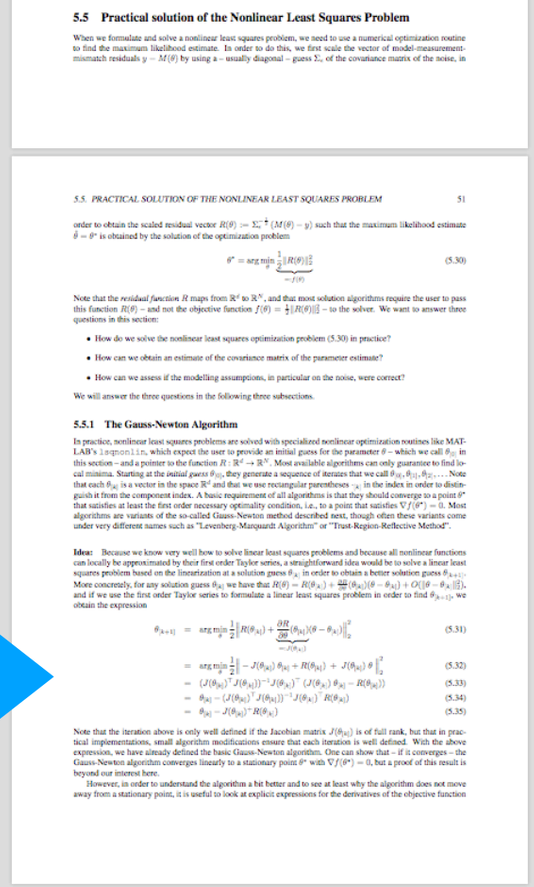
Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

- We cannot try all a.
- There are infinitely many.
- Let's solve a least squares problem!

- The Gauss-Newton method:
  - Linearise the residual function at a point (your best guess)
  - Solve a **linear** least squares problem (Find a better guess)
  - Repeat!

$$\theta_{[k+1]} = \theta_{[k]} - J(\theta_{[k]})^{\dagger} R(\theta_{[k]})$$



5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_y of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_\theta \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$ Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $\cdot_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_\theta \frac{1}{2}\Big\|\underbrace{R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]})}_{=:J(\theta_{[k]})}\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_\theta \frac{1}{2}\Big\| - J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\,\Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^N R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^N \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^N \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1}\nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_y). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_y, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

$$\theta_{[k+1]} = \arg\min_\theta \frac{1}{2}\Big\|\underbrace{R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]})}_{=:J(\theta_{[k]})}\Big\|_2^2$$

$$= \arg\min_\theta \frac{1}{2}\Big\| - J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\,\Big\|_2^2$$

… cf 4.2 …

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]}))$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]})$$

Note: We do not use the Hessian of R

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_y of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\Big\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\theta := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix Σ_θ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_y). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_y, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_\theta := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_\theta = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta)$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta)$$

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

Matlab demo

$$\theta = [a]$$

```
% The Jacobian of the R function:
J = @(a) inv(S) * cos(a + pi/6 .* [0 1 2]')

% Iterations of the Gauss-Newton algoritm:
a = a - J(a) \ R(a)
a = a - J(a) \ R(a)
a = a - J(a) \ R(a)
a = a - J(a) \ R(a)
% ...after a few iterations we should have a good estimate of the
% argmin of norm(R(a))^2
```

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_* of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\Big\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top \left(J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})\right) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_*). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_*, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

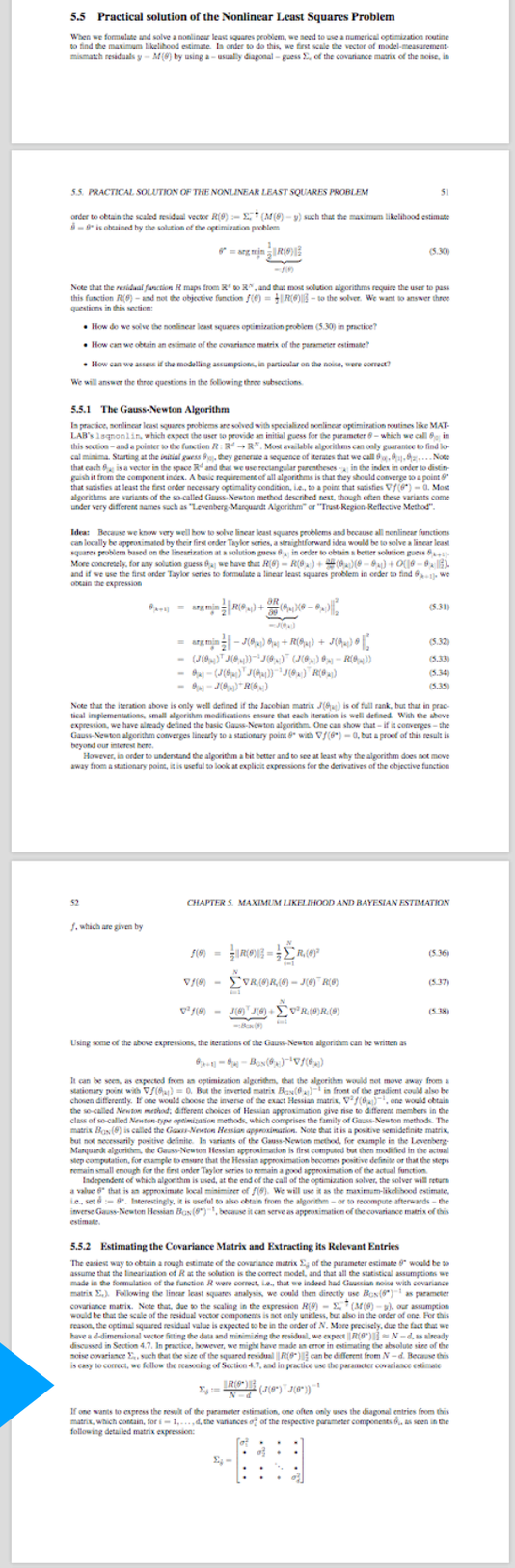
$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

- How certain is our estimate?



5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_y of the covariance matrix of the noise, in

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 51

order to obtain the scaled residual vector $R(\theta) := \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\left\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\right\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\left\|-J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\right\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

52 CHAPTER 5. MAXIMUM LIKELIHOOD AND BAYESIAN ESTIMATION

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_y). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_y, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 53

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

Estimate:

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

- Works for linear systems and Gaussian distributions
- May work for nonlinear systems
  - No guarantees!

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_* of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\Big\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{GN}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{GN}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{GN}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{GN}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{GN}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_*). Following the linear least squares analysis, we could then directly use $B_{GN}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_*, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}$$

In our case:

$$a = 0 \pm 0.07$$

Matlab demo

```
sigma_est = sqrt( (R(a)'*R(a))/(length(y)-1) * inv(J(a)'*J(a)) )

% We can generate a "measurement"
% by using M(0) as the "ground truth" and adding
% random noise to it.
% (Uncomment the line below to expermient with
% estimating a from a different measurement.)
% y = M(0) + S * randn(size(M(0)))

% … and re-run the simulation
```

MOAR PARAMETERS!

- Maybe there’s an offset in all the measurements?
- Let’s introduce another parameter!

New ML problem

$$\sin(a) + b + \epsilon_1 = -0.1$$

$$\sin(a + 30^\circ) + b + \epsilon_2 = 0.6$$

$$\sin(a + 60^\circ) + b + \epsilon_3 = 0.9$$

- All *ϵ are independent, and drawn from a normal distribution with zero mean and standard deviation 0.5*

- Find $\theta = [a, b]$ *!*

Matlab demo

```
M = @(a, b) sin(a + pi/6 * [0 1 2]') + b

% Note, if you want to see this script produce useful estimates,
% then the data would have to be drawn from a distribution with
% a lot smaller variance. (That is: C should be smaller.)
C = diag(0.5^2 * [1 1 1])

S = sqrtm(C)

y = [-0.1, 0.6, 0.9]'
%y = M(0, 0) + S * randn(size(M(0, 0)))

R = @(a, b) inv(S) * (M(a, b) - y)

p = @(a, b) exp(-0.5*sum(R(a, b).^2))

as = -pi:0.01:pi;
bs = -3:0.01:3;

ps = zeros(length(as), length(bs));
for i=1:length(as)
  for j=1:length(bs)
    ps(i,j) = p(as(i), bs(j));
  end
end

figure(1)
% Plot integral of p over all b values, as function of a
plot(as, sum(ps, 2))

figure(2)
% Plot integral of p over all a values, as function of b
plot(bs, sum(ps, 1))

figure(3)
% Plot p as function of a and b
contour(bs, as, ps)

[pmax, ix] = max(sum(ps, 2))
a = as(ix)

[pmax, ix] = max(sum(ps, 1))
b = bs(ix)

t = [a, b]'

J = @(t) inv(S) * [cos(a + pi/6 .* [0 1 2]'), [1 1 1]']
R2 = @(t) R(t(1), t(2))

t = t - J(t) \ R2(t)
t = t - J(t) \ R2(t)
t = t - J(t) \ R2(t)
t = t - J(t) \ R2(t)

C_est = (R2(t)'*R2(t))/(length(y)-2) * inv(J(t)'*J(t))

sigma_est = sqrt(diag(C_est))
```

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_y of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$ Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\Big\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta \Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^T J(\theta_{[k]}))^{-1} J(\theta_{[k]})^T (J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^T J(\theta_{[k]}))^{-1} J(\theta_{[k]})^T R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^T R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^T J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_y). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_y^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_y, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^T J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

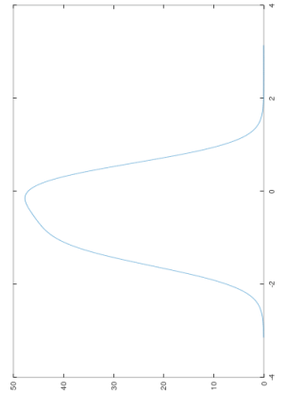
Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

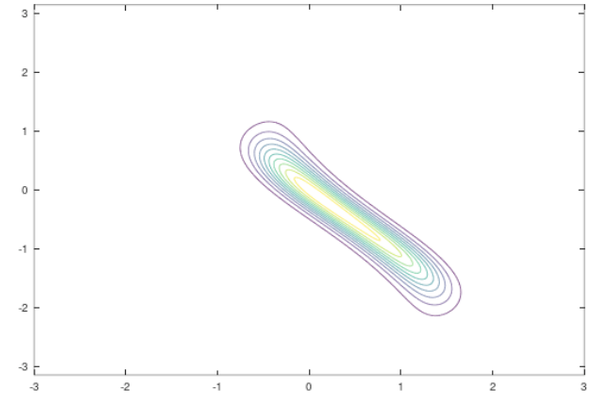
However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

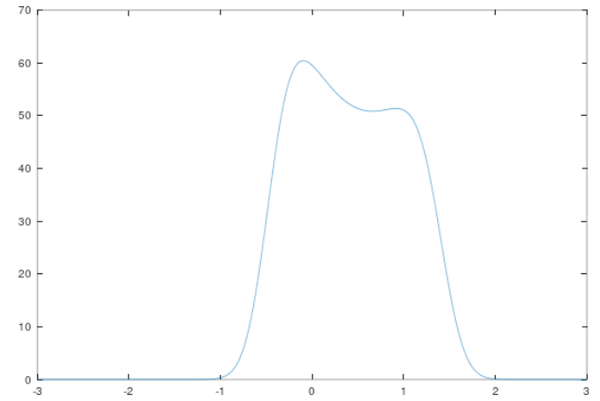
$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}$$

$$b = -0.13 \pm 0.41$$







$$a = -0.14 \pm 0.48$$

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_* of the covariance matrix of the noise, in

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 51

order to obtain the scaled residual vector $R(\theta) := \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[1]}, \theta_{[2]}, \ldots$. Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$, in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\theta_{[k+1]} = \arg\min_{\theta} \frac{1}{2}\Big\|R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 \tag{5.31}$$

$$= \arg\min_{\theta} \frac{1}{2}\Big\| - J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) + J(\theta_{[k]})\,\theta\Big\|_2^2 \tag{5.32}$$

$$= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top \left(J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})\right) \tag{5.33}$$

$$= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) \tag{5.34}$$

$$= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) \tag{5.35}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

52 CHAPTER 5. MAXIMUM LIKELIHOOD AND BAYESIAN ESTIMATION

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_*). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_*^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_*, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d}\left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

5.5. PRACTICAL SOLUTION OF THE NONLINEAR LEAST SQUARES PROBLEM 53

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because the whole analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

Looking at the residuals

```
>> R(theta(1), theta(2))
ans =
   0.46141
  -0.65579
   0.19438
```

- Should resemble a sample from a normal distribution with unity variance