Chapter 7

Nonlinear Least Squares Problems

7.1. Introduction

Given a smooth function $R : \mathbb{R}^n \to \mathbb{R}^m$ with component functions R_i, $i = 1, \ldots, m$, we consider the solution of the nonlinear least squares problem

$$\min \tfrac{1}{2}\|R(x)\|_2^2. \tag{7.1}$$

If we define the objective function

$$f(x) = \tfrac{1}{2}\|R(x)\|_2^2,$$

then gradient and the Hessian of f are given by

$$\nabla f(x) = R'(x)^T R(x) \tag{7.2}$$

and

$$\nabla^2 f(x) = R'(x)^T R'(x) + \sum_{i=1}^{m} \nabla^2 R_i(x)\, R_i(x), \tag{7.3}$$

respectively, where $R'(x)$ denotes the Jacobian of R and $\nabla^2 R_i(x)$ is the Hessian of the ith component function. Note that the first part of the Hessian, $R'(x)^T R'(x)$, uses derivative information already required for the computation of $\nabla f(x)$. In this chapter we study a variation of Newton's method, called the Gauss-Newton method, for the minimization of $f(x) = \frac{1}{2}\|R(x)\|_2^2$ in which the Hessian $\nabla^2 f(x)$ is replaced by $R'(x)^T R'(x)$. Before we discuss the Gauss-Newton method, we present a class of problems that leads to nonlinear least squares problems (7.1) in the next Section 7.2 and then we discuss the special case of linear least squares problems $R(x) = Ax + b$ in Section 7.3. The Gauss-Newton method is presented and analyzed in Section 7.4. The final Section 7.5 of this chapter treat a more complicated nonlinear least squares problem, parameter identification in ordinary differential equations.

7.2. Least Squares Curve Fitting

Suppose we are given m measurements

$$(t_i, b_i), \quad i = 1, \ldots, m,$$

and we want to fit a function

$$b(t) = \varphi(t; x_1, \ldots, x_n)$$

to the data points. The model–function φ depends on n parameters $x_1, \ldots, x_n$ which have to be determined from the measurements.

Typically the measurements b_i are not exact but are of the form $b_i = b_i^{ex} + \delta b_i$, where b_i^{ex} is the exact data and δb_i represents the measurement error. Moreover, the selection of the model–function $\varphi(t; x_1, \ldots, x_n)$ is often based on simplifying assumptions and $\varphi(t; x_1, \ldots, x_n)$ may not represent the correct relation between b and t. Therefore we try to find $x = (x_1, \ldots, x_n)$ such that the sum of squares $\sum_{i=1}^m r_i^2(x)$ of the residuals

$$r_i(x) = \varphi(t_i; x) - b_i, \quad i = 1, \ldots, m,$$

is minimized. If we define

$$R(x) = \begin{pmatrix} \varphi(t_1; x) - b_1 \\ \varphi(t_2; x) - b_2 \\ \vdots \\ \varphi(t_m; x) - b_m \end{pmatrix},$$

then

$$\sum_{i=1}^{m} r_i^2(x) = \|R(x)\|_2^2$$

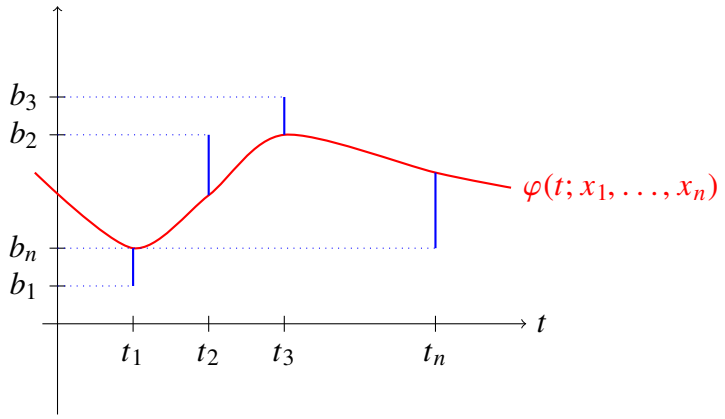


Figure 7.1: Least squares curve fitting. Given data (t_i, b_i), $i = 1, \ldots, m$, we want to find a function $\varphi(t; x_1, \ldots, x_n)$ parameterized by $x = (x_1, \ldots, x_n)^T$ such that the sum of squares of the residuals $\varphi(t_i; x) - b_i$, $i = 1, \ldots, m$, indicated by solid blue lines is minimal.

and we arrive at the minimization problem

$$\min \tfrac{1}{2}\|R(x)\|_2^2. \tag{7.4}$$

If the function R or, equivalently, φ depends linearly on x, then we call (7.4) a *linear least squares problem*. Otherwise, we call (7.4) a *nonlinear least squares problem*.

For linear least squares problems the model function φ is of the form

$$\varphi(t; x_1, \ldots, x_n) = x_1\varphi_1(t) + x_2\varphi_2(t) + \ldots + x_n\varphi_n(t)$$

with some functions $\varphi_1, \ldots, \varphi_n$. Notice that the function φ depends linearly on the parameters $x_1, \ldots, x_n$ but it may be a nonlinear function of t! If we introduce

$$A = \begin{pmatrix} \varphi_1(t_1) & \varphi_2(t_1) & \ldots & \varphi_n(t_1) \\ \varphi_1(t_2) & \varphi_2(t_2) & \ldots & \varphi_n(t_2) \\ \vdots & \vdots & & \vdots \\ \varphi_1(t_m) & \varphi_2(t_m) & \ldots & \varphi_n(t_m) \end{pmatrix} \in \mathbb{R}^{m\times n}, \tag{7.5}$$

and

$$b = (b_1, \ldots, b_m)^T \in \mathbb{R}^m, \tag{7.6}$$

then

$$R(x) = Ax - b$$

and, thus, a linear least squares problem can be written in the form

$$\min \tfrac{1}{2}\|Ax - b\|_2^2. \tag{7.7}$$

Example 7.2.1 ([Esp81, Sec. 6]) The temperature T dependence of the rate constant k for an elementary chemical reaction is almost always expressed by a relation of the form

$$k(T; C, U) = CT^n \exp\left(-\frac{U}{RT}\right), \tag{7.8}$$

where $R = 8.314$ [J/(kmol K)] is the general gas constant. Usually, n is assigned one of the values $0, 1/2$, or 1. Depending on the choice of n, the notation in (7.8) varies. For example, for $n = 0$ we obtain the *Arrhenius equation*, which is commonly written as

$$k(T; A, E) = A \exp\left(-\frac{E}{RT}\right). \tag{7.9}$$

In (7.9), A is called the pre-exponential factor and E is called the activation energy, and both have to estimated from experiments.

Consider the reaction

$$\mathrm{NO} + \mathrm{C1NO_2} \rightarrow \mathrm{NO_2} + \mathrm{C1NO}.$$

Measurements of the rate constant k (measured in $\mathrm{cm^3 mol^{-1} sec^{-1}}$) for various temperatures T (measured in K) are shown in the following table.

T	300	311	323	334	344
k	$0.79 * 10^7$	$1.25 * 10^7$	$1.64 * 10^7$	$2.56 * 10^7$	$3.4 * 10^7$

The coefficients C and U in (7.8) can be computed by solving the nonlinear least squares problem

$$\min_{C,U} \tfrac{1}{2} \sum_{i=1}^{5} \Big(k_i - k(T_i; C, U)\Big)^2. \tag{7.10}$$

Alternatively, we can divide both sides in (7.8) by T^n and take the logarithm. This gives

$$\ln\left(k(T; C, U)/T^n\right) = \ln(C) - \frac{U}{RT}.$$

The quantities $x_1 = \ln(C)$ and $x_2 = U$ can be estimated by solving the linear least squares problem

$$\min_{x} \tfrac{1}{2} \left\| \begin{pmatrix} \vdots & \vdots \\ 1 & -1/(RT_i) \\ \vdots & \vdots \end{pmatrix} x - \begin{pmatrix} \vdots \\ \ln(k_i/T_i^n) \\ \vdots \end{pmatrix} \right\|_2^2. \tag{7.11}$$

The problems (7.10) and (7.11) are related by are different, because in former we match $k(T_i; C, U)$ with k_i while in the latter we match $\ln\left(k(T_i; C, U)/T^n\right)$ with $\ln\left(k_i/T^n\right)$.

We first estimate C, U from the linear least squares problem (7.11) and then use these estimates as starting values in an optimization routine[1] to solve (7.10). This results are shown in the following table.

	Solution of (7.11)		
n	C	U	Res
0	$6.167e+11$	$2.808e+04$	$2.705e+12$
1/2	$2.087e+10$	$2.674e+04$	$2.673e+12$
1	$7.062e+08$	$2.541e+04$	$2.642e+12$

	Solution of (7.10)		
n	C	U	Res
0	$6.167e+11$	$2.807e+04$	$2.661e+12$
1/2	$2.087e+10$	$2.673e+04$	$2.632e+12$
1	$8.621e+08$	$2.595e+04$	$2.464e+12$

Here, Res $= \sum_{i=1}^{5}(k_i - k(T_i; C, U))^2$. ◇

So far we have considered scalar measurements $(t_i, b_i) \in \mathbb{R} \times \mathbb{R}$, $i = 1, \ldots m$. Everything that was said before can be extended to the case $t_i \in \mathbb{R}^k$, $b_i \in \mathbb{R}^\ell$, $i = 1, \ldots m$.

We will return to the solution of linear and nonlinear least squares problems. For some applications and statistical aspects of linear and nonlinear least squares problems we refer to the book [BW88].

7.3. Linear Least Squares Problems

An important unconstrained quadratic programming problem is the linear least squares problem

$$\min \tfrac{1}{2}\|Ax - b\|_2^2. \tag{7.12}$$

Since

$$\tfrac{1}{2}\|Ax - b\|_2^2 = \tfrac{1}{2}(Ax - b)^T(Ax - b) = \tfrac{1}{2}x^T A^T Ax - x^T A^T b + \tfrac{1}{2}b^T b$$

(7.12) is an instance of (4.9) with $H = A^T A$ and $c = -A^T b$. Hence we can apply Theorem 4.3.6. A vector x_* solves (7.12) if and only if x_* solves

$$A^T Ax = A^T b. \tag{7.13}$$

The equations (7.13) are called the *normal equations*. Using the singular value decomposition of A one can easily show that

$$\mathcal{R}(A^T A) = \mathcal{R}(A^T), \quad \mathcal{N}(A^T A) = \mathcal{N}(A).$$

Since $A^T b \in \mathcal{R}(A^T) = \mathcal{R}(A^T A)$, the normal equations are solvable. We obtain the following result.

[1] We have used the Matlab function `ls`.

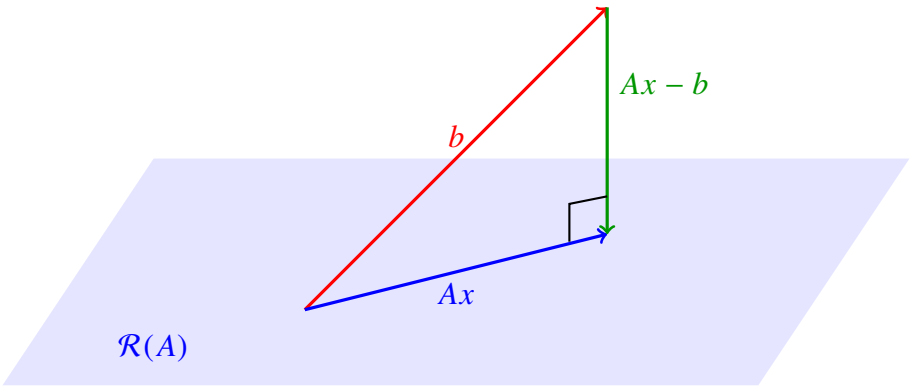


Figure 7.2: The Linear Least Squares Problem

Theorem 7.3.1 *A vector x_* solves (7.12) if and only if x_* solves the* normal equation *(7.13).*

The normal equation has at least one solution x_*. The set of solutions of (7.13) is given by

$$\mathcal{S}_b = x_* + \mathcal{N}(A),$$

where x_* denotes a particular solution of (7.13) and $\mathcal{N}(A)$ denotes the null space of A.

If $\mathcal{N}(A) \neq \{0\}$, the set of solutions of (7.12) forms a manifold in $\mathbb{R}^n$. In this case the *minimum norm solution* $x_\dagger$ is of interest. It is the solution of (7.12) with smallest norm. Mathematically, the minimum norm solution $x_\dagger$ is the solution of

$$\min_{x \in \mathcal{S}_b} \|x\|_2.$$

It can be shown that the minimum norm solution $x_\dagger$ is the solution of the least squares problem which is perpendicular to the null-space of A. See also Figure 7.3 and (7.18) below.

If $A \in \mathbb{R}^{m\times n}$ has rank n, which implies $m \geq n$, then $A^T A$ is invertible and the solution of the least squares problem (7.12) (or equivalently the normal equation (7.13)) is unique and given by

$$x = (A^T A)^{-1} A^T b. \tag{7.14}$$

If $A \in \mathbb{R}^{m\times n}$ has rank m, which implies $m \leq n$, then $A^T A$ is not invertible and the least squares problem has infinitly many solutions. The matrix AA^T is invertible and it is easy to verify that $x = A^T(A\,A^T)^{-1}b$ is a solution of the least squares problem (7.12) (or equivalently the normal equation (7.13)). In fact, since $A^T(A\,A^T)^{-1}b \in \mathcal{R}(A^T) = \mathcal{R}(A^T A) = \mathcal{N}(A^T A)^\perp = \mathcal{N}(A)^\perp$,

$$x_\dagger = A^T(A\,A^T)^{-1}b \tag{7.15}$$

is the minimum norm solution of (7.12).

7.3.1. Solution of Linear Least Squares Problems Using the SVD

The singular value decomposition (SVD) of A is a powerful tool in the analysis and solution of (7.12). Given $A \in \mathbb{R}^{m\times n}$ there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ and a 'diagonal' matrix $\Sigma \in \mathbb{R}^{m\times n}$ with diagonal entries

$$\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_{\min\{m,n\}} = 0$$

such that

$$A = U\Sigma V^T. \tag{7.16}$$

The decomposition (7.16) of A is called the singular value decomposition of A. The scalars σ_i, $i = 1, \ldots, \min\{m, n\}$ are called the singular values of A.

Using the orthogonality of U and V we find that

$$\begin{aligned} \|Ax - b\|_2^2 &= \|U^T(AV^Tx - b)\|_2^2 \\ &= \|\Sigma V^T x - U^T b\|_2^2 \\ &= \sum_{i=1}^{r}(\sigma_i z_i - u_i^T b)^2 + \sum_{i=r+1}^{m}(u_i^T b)^2, \end{aligned}$$

where we have set $z = V^T x$. Thus,

$$\min_x \|Ax - b\|_2^2 = \min_{z=V^T x} \sum_{i=1}^{r}(\sigma_i z_i - u_i^T b)^2 + \sum_{i=r+1}^{m}(u_i^T b)^2.$$

The solutions are given by

$$\begin{aligned} z_i &= \frac{u_i^T b}{\sigma_i}, \qquad i = 1, \ldots, r, \\ z_i &= \text{arbitrary}, \quad i = r+1, \ldots, n. \end{aligned}$$

Moreover,

$$AVz = U\Sigma z = \sum_{i=1}^{r} z_i u_i = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} u_i \tag{7.17}$$

and

$$\min \tfrac{1}{2}\|Ax - b\|_2^2 = \tfrac{1}{2}\sum_{i=r+1}^{m}(u_i^T b)^2.$$

Since V is orthogonal, we find that

$$\|x\|_2 = \|VV^T x\|_2 = \|V^T x\|_2 = \|z\|_2.$$

Hence, the minimum norm solution of the linear least squares problem is given by

$$x_\dagger = V z_\dagger,$$

where $z_\dagger \in \mathbb{R}^n$ is the vector with entries

$$\begin{aligned} z_i^\dagger &= \frac{u_i^T b}{\sigma_i}, \quad i = 1, \ldots, r, \\ z_i^\dagger &= 0, \qquad i = r+1, \ldots, n, \end{aligned}$$

i.e.

$$x_\dagger = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i. \tag{7.18}$$

Since $\{v_1, \ldots, v_r\}$ is an orthonormal basis for $\mathcal{N}(A)^\perp$, we see that $x_\dagger \perp \mathcal{N}(A)$. Moreover, since $\{u_1, \ldots, u_r\}$ is an orthonormal basis for $\mathcal{R}(A)$, the projection $P_{\mathcal{R}(A)} b$ of b onto $\mathcal{R}(A)$ is given by

$$P_{\mathcal{R}(A)} b = \sum_{i=r+1}^{m} (u_i^T b)\, u_i$$

we see that $Ax_* = P_{\mathcal{R}(A)} b$ for all solutions x_* of (7.12) (see (7.17)). The structure of the solution of the linear least squares problem is sketched in Figure 7.3.

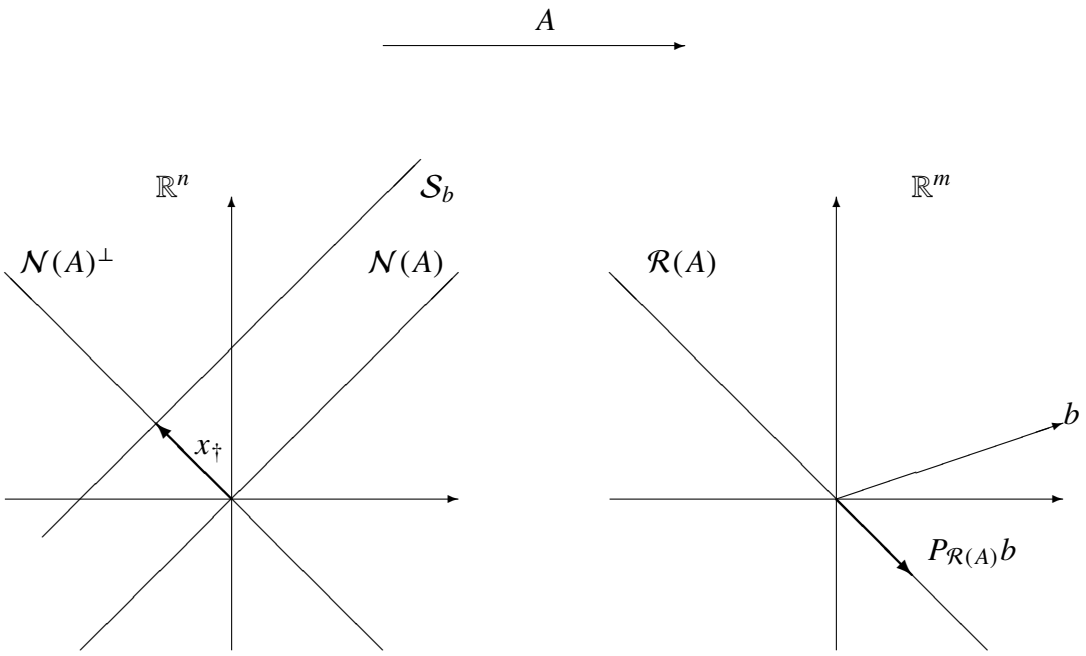
Given the SVD (7.16) of A, the matrix,

$$A^\dagger = V \Sigma^\dagger U^T, \tag{7.19}$$

where $\Sigma^\dagger \in \mathbb{R}^{n \times m}$ is the diagonal matrix with diagonal entries $1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0$, is called the *Moore–Penrose pseudo inverse*. The minimum norm solution (7.18) of the linear least squares problem is given by

$$x_\dagger = A^\dagger b. \tag{7.20}$$

Figure 7.3: Solution of the Linear Least Squares Problem



7.3.2. Solution of Linear Least Squares Problems Using the QR–Decomposition

Instead of using the SVD, we can also use the QR–decomposition of $A \in \mathbb{R}^{m\times n}$ to solve the linear least squares problem (7.12). Suppose that $A \in \mathbb{R}^{m\times n}$ has rank $r \leq \min\{m, n\}$. The QR–decomposition with column pivoting yields

$$AP = QR, \tag{7.21}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal, $P \in \mathbb{R}^{n\times n}$ is a permutation matrix, and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix of the form

$$R = \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix}, \tag{7.22}$$

with nonsingular upper triangular $R_1 \in \mathbb{R}^{r\times r}$, and $R_2 \in \mathbb{R}^{r\times(n-r)}$. See, e.g., [Bjö96, Sec. 1.3], [GL89, Sec. 5.4.], Since $Q^TQ = I$ and $P^TP = I$, we can write

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Q^T(APP^Tx - b)\|_2^2 \\ &= \left\| \begin{pmatrix} R_1R_2 \\ 0 \end{pmatrix} P^Tx - Q^Tb \right\|_2^2. \end{aligned}$$

Now we partition

$$Q^Tb = \begin{pmatrix} c_1 \\ c_2 \\ d \end{pmatrix} \begin{matrix} \}r \\ \}n-r \\ \}m-n \end{matrix} \tag{7.23}$$

and we set

$$y = P^Tx \quad \text{i.e.} \quad x = Py.$$

Let

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad y_1 \in \mathbb{R}^r, \quad y_2 \in \mathbb{R}^{n-r}. \tag{7.24}$$

This yields

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| \begin{pmatrix} R_1y_1 + R_2y_2 - c_1 \\ c_2 \\ d \end{pmatrix} \right\|_2^2 \\ &= \|R_1y_1 + R_2y_1 - c_1\|_2^2 + \|c_2\|_2^2 + \|d\|_2^2 . \end{aligned}$$

The least squares problem can be written as

$$\min_x \|Ax - b\|_2^2 = \min_y \|R_1y_1 + R_2y_2 - c_1\|_2^2 + \|c_2\|_2^2 + \|d\|_2^2.$$

If y solves the minimization problem on the right hand side, then $x = Py$ solves the minimization problem on the left hand side and vice versa. Since $R_1 \in \mathbb{R}^{r\times r}$ is nonsingular, we can compute

$$y_1 = -R_1^{-1}(R_2 y_2 - c_1)$$

for given $y_2 \in \mathbb{R}^{n-r}$. Thus, the set of solutions of

$$\min_y \|R_1 y_1 + R_2 y_2 - c_1\|_2^2 + \|c_2\|_2^2 + \|d\|_2^2$$

is given by

$$\left\{ \begin{pmatrix} -R_1^{-1}(R_2 y_2 - c_1) \\ y_2 \end{pmatrix} \mid y_2 \in \mathbb{R}^{n-r} \right\}$$

and we find that

$$\min_y \|R_1 y_1 + R_2 y_2 - c_1\|_2^2 + \|c_2\|_2^2 + \|d\|_2^2 = \|c_2\|_2^2 + \|d\|_2^2.$$

Consequently, the solutions to the linear least squares problem $\min \|Ax - b\|_2$ is given by

$$\mathcal{S}_b = \left\{ P \begin{pmatrix} -R_1^{-1}(R_2 y_2 - c_1) \\ y_2 \end{pmatrix} \mid y_2 \in \mathbb{R}^{n-r} \right\} \tag{7.25}$$

and

$$\min_x \|Ax - b\|_2^2 = \|c_2\|_2^2 + \|d\|_2^2.$$

A particular solution can be found by setting $y_2 = 0$ which yields

$$y = \begin{pmatrix} R_1^{-1} c_1 \\ 0 \end{pmatrix} \quad \text{and} \quad x = P \begin{pmatrix} R_1^{-1} c_1 \\ 0 \end{pmatrix}.$$

The minimum norm solution $x_\dagger$ is defined as the solution of

$$\min_{x\in\mathcal{S}_b} \|x\|_2.$$

Using (7.25) and the fact that $\|Py\|_2 = \|y\|_2$ for all $y \in \mathbb{R}^n$, we find that

$$\begin{aligned} \min_{x\in\mathcal{S}_b} \|x\|_2 &= \min_{y_2\in\mathbb{R}^{n-r}} \left\| P \begin{pmatrix} -R_1^{-1}(R_2 y_2 - c_1) \\ y_2 \end{pmatrix} \right\|_2 \\ &= \min_{y_2\in\mathbb{R}^{n-r}} \left\| \begin{pmatrix} -R_1^{-1} R_2 \\ I \end{pmatrix} y_2 + \begin{pmatrix} R_1^{-1} c_1 \\ 0 \end{pmatrix} \right\|_2. \end{aligned} \tag{7.26}$$

The right hand side in (7.26) is just another linear least squares problem in y_2. Its solution can be obtained by solving the normal equations

$$\left((R_1^{-1} R_2)^T R_1^{-1} R_2 + I \right) y_2 = (R_1^{-1} R_2)^T R_1^{-1} c_1 \tag{7.27}$$

corresponding to (7.26) or by computing the QR–decomposition of

$$\begin{pmatrix} -R_1^{-1}R_2 \\ I \end{pmatrix} \in \mathbb{R}^{n\times(n-r)} \tag{7.28}$$

and proceeding as above. This matrix in (7.28) has full rank $n - r$. Consequently, (7.27) or, equivalently, (7.26) has a unique solution y_2^*. The minimum norm solution of (7.12) is given by

$$x_\dagger = P \begin{pmatrix} -R_1^{-1}(R_2 y_2^* - c_1) \\ y_2^* \end{pmatrix},$$

where y_2^* is the solution of (7.26) or, equivalently, (7.27).

An important issue in computation of a solution to the rank deficient least squares problem is the determination of the rank of A. In practice, the computed singular values will never be exactly zero. In this case one has to decide which singular values are numerically zero. Similarly, in practice the upper triangular matrix R in the QR–decomposition

$$Q^T AP = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

will usually be of the form

$$R = \begin{pmatrix} R_1 & R_2 \\ 0 & \tilde{R}_3 \end{pmatrix}.$$

The lower right block $\tilde{R}_3$ will in general not be exactly equal to zero. One has to decide which submatrix is considered to be numerically zero. The numerical rank determination can be a difficult problem. We refer to the book [Bjö96, § 2.7] by Björck for a discussion of rank deficient problems.

7.4. The Gauss–Newton Method

7.4.1. Derivation of the Gauss–Newton Method

Let $R : \mathbb{R}^n \to \mathbb{R}^m$ be a smooth function with component functions R_i, $i = 1, \ldots, m$. The Jacobian of R is denoted by $R'(x) \in \mathbb{R}^{m\times n}$.

To minimize

$$f(x) = \tfrac{1}{2}\|R(x)\|_2^2$$

we compute the gradient

$$\nabla f(x) = R'(x)^T R(x) \tag{7.29}$$

and the Hessian

$$\nabla^2 f(x) = R'(x)^T R'(x) + \sum_{i=1}^{m} R_i(x)\nabla^2 R_i(x). \tag{7.30}$$

We note that once the Jacobian $R'(x)$ is computed, we can compute the gradient of f and we can compute the first term in the Hessian of f. If $R'(x)^T R'(x)$ is large compared to $\sum_{i=1}^m R_i(x)\nabla^2 R_i(x)$, then $\nabla^2 f(x) \approx R'(x)^T R'(x)$. This will be the case if, e.g., $R_i(x)$, $i = 1, \ldots, m$, is small, or if $\nabla^2 R_i(x)$, $i = 1, \ldots, m$ is small. The latter conditions means that R_i is almost linear.

If we omit the second order derivative information, then the approximate Newton system is of the form

$$R'(x_k)^T R'(x_k)s = -R'(x_k)^T R(x_k).$$

This system is always solvable and it has a unique solution if and only if the Jacobian $R'(x_k)$ has rank n. The previous system is the normal equation for the linear least squares problem

$$\min_{s\in\mathbb{R}^n} \tfrac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2.$$

This leads to the Gauss-Newton method.

Algorithm 7.4.1 (Gauss-Newton Method)

Input: Starting value $x_0 \in \mathbb{R}^n$, tolerance tol.

For $k = 0, \ldots$

Compute $R(x_k)$ and $R'(x_k)$.
Check truncation criteria.
Compute a solution s_k of $\min \tfrac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2$.
Set $x_{k+1} = x_k + s_k$.

End

7.4.2. Full Rank Problems

The Gauss–Newton method can be viewed as a Newton method with inexact Hessian information applied to the minimization of $f(x) = \frac{1}{2}\|R(x)\|_2^2$. The Hessian $\nabla^2 f(x_k) = R'(x_k)^T R'(x_k) + \sum_{i=1}^m R_i(x_k)\nabla^2 R_i(x_k)$ is approximated by $R'(x_k)^T R'(x_k)$. Our local convergence analysis for Newton's method with inexact derivative information, Theorem 5.2.7 can be applied to this setting. We have

$$\Delta(x_k) = -\sum_{i=1}^m R_i(x_k)\nabla^2 R_i(x_k), \quad \delta(x_k) = 0$$

Theorem 5.2.7 requires that

$$\nabla^2 f(x_k) + \Delta(x_k) = R'(x_k)^T R'(x_k)$$

are invertible,

$$\left\|\left(R'(x_k)^T R'(x_k)\right)^{-1}\right\|_2 \le M,$$

and that

$$\|(\nabla^2 f(x_k) + \Delta(x_k))^{-1}\Delta(x_k)\|_2 \le \alpha_k \le \alpha < 1,$$

i.e.,

$$\left\|\left(R'(x_k)^T R'(x_k)\right)^{-1} \sum_{i=1}^{m} R_i(x_k)\nabla^2 R_i(x_k)\right\|_2 \leq \alpha_k \leq \alpha < 1. \tag{7.31}$$

Under these assumptions the Gauss–Newton method is locally convergent and the iterates obey

$$\|x_{k+1} - x_*\|_2 \leq \alpha_k \, \|x_k - x_*\|_2 + \frac{ML}{2} \, \|x_k - x_*\|_2^2$$

for all k, where L is the Lipschitz constant of the Hessian of $\|R(x)\|_2^2$.

The matrices $R'(x_k)^T R'(x_k)$ are invertible if and only of $R'(x_k) \in \mathbb{R}^{m\times n}$ has rank n. This implies that $m \geq n$.

The convergence analysis for approximate Newton methods as sketched above gives a good description of the local convergence behavior of the Gauss–Newton method for problems in which the Jacobians $R'(x)$ have full column rank. We briefly call problems for which this is the case *full rank problems*. A more refined convergence result is given below. It avoids second derivatives altogether. However, the assumptions in the following theorem are closely related to the assumptions needed when using the convergence analysis for approximate Newton methods, as we will see later.

Theorem 7.4.2 (Local Convergence of the GN Method for Full Rank Problems) *Let $D \subset \mathbb{R}^n$ be an open set and let $x_* \in D$ be a (local) solution of the nonlinear least squares problem. Suppose that $R : D \to \mathbb{R}^m$ is continuously differentiable with $R' \in Lip_L(D)$ and suppose that for all $x \in D$ the Jacobian $R'(x)$ has rank n. If there exist $\omega > 0$ and $\kappa \in (0, 1)$ such that for all $x \in D$ and all $t \in [0, 1]$ the following conditions hold*

$$\begin{aligned}
&\left\|\left(R'(x)^T R'(x)\right)^{-1}(R'(x)^T - R'(x_*)^T)R(x_*)\right\|_2 &\leq \kappa\|x - x_*\|_2, &\quad (7.32)\\
&\left\|\left(R'(x)^T R'(x)\right)^{-1} R'(x)^T \right.\\
&\qquad \left.\times(R'(x_* + t(x - x_*)) - R'(x))(x - x_*)\right\|_2 &\leq \omega t\|x - x_*\|_2^2, &\quad (7.33)
\end{aligned}$$

then there exists $\epsilon > 0$ such that if $x_0 \in B_\epsilon(x_*)$, then the iterates $\{x_k\}$ generated by the Gauss–Newton method convergence towards x_* and obey the estimate

$$\|x_{k+1} - x_*\|_2 \leq \frac{\omega}{2}\|x_k - x_*\|_2^2 + \kappa\|x_k - x_*\|_2. \tag{7.34}$$

Proof: i. First we show that for $x_k \in D$, the estimate (7.34) is valid.

To prove (7.34) recall that in the full rank case, the new Gauss–Newton iterate is given by $x_{k+1} = x_k - (R'(x_k)^T R'(x_k))^{-1} R'(x_k)^T R(x_k)$. The definition of the Gauss–Newton iterate and

$R'(x_*)^T R(x_*) = 0$ yields

$$\begin{aligned}
&x_{k+1} - x_* \\
&\quad = \; x_k - x_* - (R'(x_k)^T R'(x_k))^{-1} R'(x_k)^T R(x_k) \\
&\quad = \; (R'(x_k)^T R'(x_k))^{-1} \Big[R'(x_k)^T \{ R'(x_k)(x_k - x_*) - R(x_k) + R(x_*) \} \\
&\qquad + (R'(x_*) - R'(x_k))^T R(x_*) \Big] \\
&\quad = \; (R'(x_k)^T R'(x_k))^{-1} \Big[R'(x_k)^T \int_0^1 (R'(x_k) - R'(x_* + t(x_k - x_*))(x_k - x_*) dt \\
&\qquad + (R'(x_*) - R'(x_k))^T R(x_*) \Big].
\end{aligned}$$

Taking norms and applying (7.32) and (7.33) gives

$$\begin{aligned}
&\|x_{k+1} - x_*\|_2 \\
&\quad \leq \; \| \int_0^1 (R'(x_k)^T R'(x_k))^{-1} R'(x_k)^T \{ R'(x_k) - R'(x_* + t(x_k - x_*)) \} (x_k - x_*) dt \|_2 \\
&\qquad + \|(R'(x_k)^T R'(x_k))^{-1} (R'(x_*) - R'(x_k))^T R(x_*)\|_2 \\
&\quad \leq \; \frac{\omega}{2} \|x_k - x_*\|_2^2 + \kappa \|x_k - x_*\|_2.
\end{aligned}$$

ii. Let $\epsilon_1 > 0$ be such that $B_{\epsilon_1}(x_*) \subset D$ and let $\sigma \in (\kappa, 1)$ be arbitrary. If

$$\epsilon \leq \min\{\epsilon_1, \frac{2(\sigma - \kappa)}{\omega}\}$$

and $x_0 \in B_\epsilon(x_*)$, then

$$\|x_* - x_1\|_2 \leq \frac{\omega}{2} \|x_0 - x_*\|_2^2 + \kappa \|x_0 - x_*\|_2 < \left(\frac{\omega}{2} \epsilon + \kappa \right) \|x_0 - x_*\|_2 \leq \sigma \|x_0 - x_*\|_2$$

A simple induction argument shows that

$$\|x_{k+1} - x_*\|_2 < \sigma \|x_k - x_*\|_2 < \sigma^{k+1} \|x_0 - x_*\|_2.$$

This proves $\lim_{k \to \infty} x_k = x_*$. □

The condition (7.33) is implied by the Lipschitz continuity of R' and by the uniform boundedness of $\| \left(R'(x)^T R'(x) \right)^{-1} \|_2$. In fact if $R' \in \mathrm{Lip}_L(D)$ and

$$a = \sup_{x \in D} \| \left(R'(x)^T R'(x) \right)^{-1} R'(x)^T \|_2 < \infty,$$

then

$$\left\| \left(R'(x)^T R'(x) \right)^{-1} R'(x)^T (R'(x_* + t(x - x_*)) - R'(x))(x - x_*) \right\|_2$$
$$\leq \| \left(R'(x)^T R'(x) \right)^{-1} R'(x)^T \|_2 Lt \, \|x - x_*\|_2^2 \leq aLt \, \|x - x_*\|_2^2.$$

Thus (7.33) holds with $\omega \leq aL$. The condition (7.32) is more interesting. Clearly, if $R(x_*) = 0$ (zero residual problem) or if $R(x)$ is affine linear, then (7.32) is satisfied with $\kappa = 0$ and the Gauss–Newton method converges locally q–quadratic. We will show in Lemma 7.4.3 below that (7.32) is essentially equivalent to the condition (7.31) with $\alpha = \kappa$. Lemma 7.4.4 below relates (7.32) (via the results in Lemma 7.4.3) to the second order sufficient optimality condition. The analysis follows [Boc88, Sec. 3] and [Hei93].

Lemma 7.4.3 *Let $D \subset \mathbb{R}^n$ be an open set and let $x_* \in D$ be a (local) solution of the nonlinear least squares problem. Suppose that $R : D \to \mathbb{R}^m$ is continuously differentiable. Moreover, assume that R_i, $i = 1, \ldots, m$, is twice differentiable at x_* and that $R'(x_*)^T R'(x_*)$ is invertible.*

i. *If*

$$\|(R'(x_*)^T R'(x_*))^{-1}(R'(x)^T - R'(x_*)^T)R(x_*)\| \quad \leq \quad \kappa \|x - x_*\| \quad \forall x \in D,$$

then

$$\|(R'(x_*)^T R'(x_*))^{-1} \sum_{i=1}^{m} R_i(x_*) \nabla^2 R_i(x_*)\|_2 \leq \kappa.$$

ii. *If*

$$\|(R'(x_*)^T R'(x_*))^{-1} \sum_{i=1}^{m} R_i(x_*) \nabla^2 R_i(x_*)\|_2 \leq \hat{\kappa},$$

then for any $\kappa > \hat{\kappa}$ there exists $\epsilon > 0$ such that for all $x \in B_\epsilon(x_*)$

$$\|(R'(x_*)^T R'(x_*))^{-1}(R'(x)^T - R'(x_*)^T)R(x_*)\| \leq \kappa \|x - x_*\|.$$

Proof: i. Assume that

$$\|(R'(x_*)^T R'(x_*))^{-1}(R'(x)^T - R'(x_*)^T)R(x_*)\| \leq \kappa \|x - x_*\|$$

for all $x \in D$. In particular,

$$\|(R'(x_*)^T R'(x_*))^{-1}(R'(x_* + \frac{\delta}{\|h\|_2} h)^T - R'(x_*)^T)R(x_*)\| \leq \kappa\delta$$

for all $h \in \mathbb{R}^n$, $h \neq 0$ and all sufficiently small δ. Since R_i is twice differentiable at x_*, there exists $\phi : \mathbb{R} \to \mathbb{R}$ with $\lim_{t \to 0} \phi(t) = 0$ such that

$$\|(R'(x_*)^T R'(x_*))^{-1} \sum_{i=1}^{m} R_i(x_*) \nabla^2 R_i(x_*) \delta h\|_2 \; \leq \; (\kappa + \phi(\delta))\delta$$

for all $h \in \mathbb{R}^n$ with $\|h\|_2 = 1$. Since we can cancel δ on the left and on the right hand side of the previous inequality, this yields

$$\|(R'(x_*)^T R'(x_*))^{-1} \sum_{i=1}^{m} R_i(x_*)\nabla^2 R_i(x_*)\|_2 \quad \leq \quad (\kappa + \phi(\delta)).$$

If we take the limit $\delta \to 0$, we obtain the assertion.

ii. The second assertion can be proven in a similar way. □

If the assumptions of Lemma 7.4.3 are satisfied and if R_i, $i = 1, \ldots, m$, are twice differentiable and the Hessians $\nabla^2 R_i$, $i = 1, \ldots, m$, are Lipschitz continuous, then Lemma 7.4.3 ii. show that (7.32) implies (7.31) with $\alpha \in (\kappa, 1)$ for all x_k sufficiently close to x_*.

The next result relates (7.32) (via the results in Lemma 7.4.3) to the second order sufficient optimality condition.

Lemma 7.4.4 *Let $D \subset \mathbb{R}^n$ be an open set. Suppose that $R_i : D \to \mathbb{R}$, $i = 1, \ldots, m$, are twice continuously differentiable. If $R'(x_*)^T R'(x_*)$ is invertible, then the following statements are equivalent:*

i. There exists $\lambda > 0$ with

$$h^T R'(x_*)^T R'(x_*)h - |h^T \sum_{i=1}^{m} R_i(x_*)\nabla^2 R_i(x_*)h| \geq \lambda \|h\|_2^2 \quad \forall h \in \mathbb{R}^n. \tag{7.35}$$

ii. There exists $\kappa < 1$ with

$$\|(R'(x_*)^T R'(x_*))^{-1} \sum_{i=1}^{m} R_i(x_*)\nabla^2 R_i(x_*)\|_2 \leq \kappa. \tag{7.36}$$

Proof: First we prove that i. implies ii. Since $R'(x_*)^T R'(x_*)$ is invertible, it is symmetric positive definite. Hence, we can form $[R'(x_*)^T R'(x_*)]^{-1/2}$. With the variable transformation $h \to [R'(x_*)^T R'(x_*)]^{-1/2}h$, (7.35) implies that

$$\begin{aligned} & |h^T [R'(x_*)^T R'(x_*)]^{-1/2} \sum_{i=1}^{m} R_i(x_*)\nabla^2 R_i(x_*)[R'(x_*)^T R'(x_*)]^{-1/2}h| \\ \leq \quad & \left(1 - \frac{\lambda}{\|R'(x_*)^T R'(x_*)\|_2}\right) \|h\|_2^2 \end{aligned}$$

for all $h \in \mathbb{R}^n$. The latter inequality implies

$$\begin{aligned} & \|[R'(x_*)^T R'(x_*)]^{-1/2} \sum_{i=1}^{m} R_i(x_*)\nabla^2 R_i(x_*)[R'(x_*)^T R'(x_*)]^{-1/2}\|_2 \\ \leq \quad & 1 - \frac{\lambda}{\|R'(x_*)^T R'(x_*)\|_2}. \end{aligned}$$

Since

$$\begin{aligned} & \|[R'(x_*)^T R'(x_*)]^{-1/2} \sum_{i=1}^m R_i(x_*)\nabla^2 R_i(x_*)[R'(x_*)^T R'(x_*)]^{-1/2}\|_2 \\ = \; & \|R'(x_*)^T R'(x_*)^{-1} \sum_{i=1}^m R_i(x_*)\nabla^2 R_i(x_*)\|_2, \end{aligned}$$

(7.36) holds with $\kappa = 1 - \lambda/\|R'(x_*)^T R'(x_*)\|$.

Similar arguments can be used to show that ii. implies i. with $\lambda = (1-\kappa)/\|R'(x_*)^T R'(x_*)^{-1}\|_2$. □

7.4.3. Line Search Globalization

We have seen in Section 6.2.1 that if s_k satisfies

$$\tfrac{1}{2}\|R'(x_k)s_k + R(x_k)\|_2^2 < \tfrac{1}{2}\|R(x)\|_2^2, \tag{7.37}$$

then s_k is a descent direction. Hence we can use a line-search. The new iterate is

$$x_{k+1} = x_k + \alpha_k s_k,$$

where the step size $\alpha_k > 1$ is chosen according to the conditions in Section 6.2.2 applied to $f(x) = \frac{1}{2}\|R(x)\|_2^2$.

Often the special structure of the function can be used to find equivalent but more convenient representations of the line search conditions. For example, the sufficient decrease condition (6.5) for $f(x) = \frac{1}{2}\|R(x)\|_2^2$ is given by

$$\tfrac{1}{2}\|R(x_k + \alpha_k s_k)\|_2^2 \le \tfrac{1}{2}\|R(x_k)\|_2^2 + c_1\alpha_k R(x_k)^T R'(x_k)s_k. \tag{7.38}$$

If s_k is the exact solution of the linear least squares problem $\min_s \frac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2$, then the sufficient decrease condition (7.38) is equivalent to

$$\tfrac{1}{2}\|R(x_k + \alpha_k s_k)\|_2^2 \le \tfrac{1}{2}\|R(x_k)\|_2^2 + c_1\alpha_k \left(\|R'(x_k)s_k + R(x_k)\|_2^2 - \|R(x_k)\|_2^2\right). \tag{7.39}$$

See Problem 6.1. The representation (7.39) of the sufficient decrease condition only requires the quantities $\|R(x_k)\|_2$ and $\|R'(x_k)s_k + R(x_k)\|_2^2$ that have to be computed anyway during the Gauss-Newton algorithm.

7.4.4. Rank Deficient Problems

If $R'(x_k) \in \mathbb{R}^{m\times n}$ has rank $r < n$, then

$$\min_{s\in\mathbb{R}^n} \tfrac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2 \tag{7.40}$$

has infinitely many solutions. How do we choose the step s_k from the set of least squares solutions? It seems unreasonable to take arbitrarily large steps. We will take the minimum norm solution

$$s_k = -R'(x_k)^\dagger R(x_k) \tag{7.41}$$

as our step. This step can be computed with the methods described in Sections 7.3.1 or 7.3.2. Note that if $R'(x_k)^T R(x_k) = 0$, then $s_k = 0$ is the minimum norm solution of (7.40). Thus, if the first order necessary optimality conditions for $\frac{1}{2}\|R(x)\|_2^2$ are satisfied at x_k, in particular, if x_k is a local minimizer, then the Gauss-Newton method with the choice (7.41) will not move away from such a point.

Our convergence analysis of the Gauss-Newton method for the rank deficient case follows [Boc88, DH79]. See also [DH95] and [Deu04, Ch. 4]. If $R'(x) \in \mathbb{R}^{m\times n}$ has rank n, then $R'(x_k)^\dagger = (R'(x_k)^T R'(x_k))^{-1} R'(x_k)^T$.

We note that

$$R'(x_*)^T R(x_*) = 0 \quad \Longleftrightarrow R'(x_*)^\dagger R(x_*) = 0.$$

Clearly, if $R'(x_*)^T R(x_*) = 0$, the minimum norm solution of $\min \frac{1}{2}\|R'(x_*)s + R(x_*)\|_2^2$ is $R'(x_*)^\dagger R(x_*) = 0$. On the other hand, if the minimum norm solution $R'(x_*)^\dagger R(x_*)$ of $\min \frac{1}{2}\|R'(x_*)s + R(x_*)\|_2^2$ is zero, then $R'(x_*)^T R(x_*) = 0$.

Theorem 7.4.5 (Local Convergence of the GN Method) *Let $D \subset \mathbb{R}^n$ be an open set and let $R : D \to \mathbb{R}^m$ be continuously differentiable in D. If there exist $\omega > 0$ and $\kappa \in (0, 1)$ such that for all $x \in D$ and all $t \in [0, 1]$ the following conditions hold*

$$\left\|\left(R'(y)^\dagger - R'(x)^\dagger\right)\left(R(x) - R'(x)R'(x)^\dagger R(x)\right)\right\|_2 \leq \kappa\|y - x\|_2, \tag{7.42a}$$

$$\left\|R'(y)^\dagger(R'(x + t(y - x)) - R'(x))(y - x)\right\|_2 \leq \omega t\|y - x\|_2^2, \tag{7.42b}$$

then for all $x_0 \in D$ with

$$\delta_0 \stackrel{\text{def}}{=} \frac{\alpha_0\omega}{2} + \kappa < 1,$$

where $\alpha_0 \stackrel{\text{def}}{=} \|R'(x_0)^\dagger R(x_0)\|_2$, and

$$\overline{B_{\frac{\alpha_0}{1-\delta_0}}(x_0)} \subset D$$

the following statement are valid:

i. *The Gauss-Newton iteration $x_{k+1} = x_k - R'(x_k)^\dagger R(x_k)$ is well defined and $\{x_k\} \subset B_{\frac{\alpha_0}{1-\delta_0}}(x_0)$,*

ii. $\lim_{k\to\infty} x_k = x_*$ *and* x_* *satisfies* $R'(x_*)^\dagger R(x_*) = 0$,

iii. $\|x_k - x_*\|_2 \le \delta_0^k \frac{\alpha_0}{1-\delta_0}$,

iv. $\|x_{k+1} - x_k\|_2 \le \left(\frac{\alpha_k \omega}{2} + \kappa\right) \|x_k - x_{k-1}\|_2$, *where* $\alpha_k \stackrel{\text{def}}{=} \|R'(x_k)^\dagger R(x_k)\|_2 \le \alpha_0$.

Proof: By definition of α_0 we have

$$x_0, x_1 = x_0 - R'(x_0)^\dagger R(x_0) \in B_{\frac{\alpha_0}{1-\delta_0}}(x_0).$$

and $\|s_0\|_2 = \|R'(x_0)^\dagger R(x_0)\|_2 \le \alpha_0$. Suppose that $\|s_k\|_2 \le \alpha_0$ and $x_{k+1} \in B_{\frac{\alpha_0}{1-\delta_0}}(x_0)$. We will show that $\|s_{k+1}\|_2 \le \alpha_0$ and $x_{k+2} \in B_{\frac{\alpha_0}{1-\delta_0}}(x_0)$.

First we note that the identity $R'(x)^\dagger R'(x)\, R'(x)^\dagger = R'(x)^\dagger$ implies

$$\Big(R'(y)^\dagger - R'(x)^\dagger\Big)\Big(R(x) - R'(x)R'(x)^\dagger R(x)\Big) = R'(y)^\dagger\Big(R(x) - R'(x)R'(x)^\dagger R(x)\Big).$$

From the identities $s_{k+1} = -R'(x_{k+1})^\dagger R(x_{k+1})$ and $s_k = -R'(x_k)^\dagger R(x_k)$ we find that

$$\begin{aligned}
s_{k+1} =& - R'(x_{k+1})^\dagger R(x_{k+1}) \\
=& - R'(x_{k+1})^\dagger\Big(R(x_{k+1}) - R(x_k) - R'(x_k)s_k\Big) - R'(x_{k+1})^\dagger\Big(R(x_k) + R'(x_k)s_k\Big) \\
=& - \int_0^1 R'(x_{k+1})^\dagger (R'(x_k + ts_k) - R'(x_k))s_k\, dt \\
&+ R'(x_{k+1})^\dagger\Big(R(x_k) - R'(x_k)R'(x_k)^\dagger R(x_k)\Big).
\end{aligned}$$

Hence, using (7.42) we find

$$\begin{aligned}
\|s_{k+1}\|_2 \le& \int_0^1 \|R'(x_{k+1})^\dagger (R'(x_k + ts_k) - R'(x_k))s_k\|_2 dt \\
&+ \|R'(x_{k+1})^\dagger\Big(R(x_k) - R'(x_k)R'(x_k)^\dagger R(x_k)\Big)\|_2 \\
\le& \frac{\omega}{2}\|s_k\|_2^2 + \kappa\|s_k\|_2 \\
\le& \left(\frac{\omega}{2}\|R'(x_k)^\dagger R(x_k)\|_2 + \kappa\right)\|s_k\|_2 \\
\le& \left(\frac{\omega\alpha_k}{2} + \kappa\right)\|s_k\|_2 \le \left(\frac{\omega\alpha_0}{2} + \kappa\right)\|s_k\|_2 = \delta_0\|s_k\|_2 \le \delta_0\alpha_0 \le \alpha_0.
\end{aligned}$$

Moreover,

$$\|x_{k+2} - x_0\|_2 = \left\|\sum_{j=0}^{k+1} s_j\right\|_2 \le \sum_{j=0}^{k+1}\|s_j\|_2 \le \sum_{j=0}^{k+1}\delta_0^j\|s_0\|_2 \le \frac{\alpha_0}{1-\delta_0},$$

which means that $x_{k+2} \in B_{\frac{\alpha_0}{1-\delta_0}}(x_0)$.

The sequence $\{x_k\} \subset B_{\frac{\alpha_0}{1-\delta_0}}(x_0)$ is a Cauchy sequence since

$$\|x_{k+p} - x_k\|_2 = \left\| \sum_{j=0}^{p-1} s_{k+j} \right\|_2 \leq \sum_{j=0}^{p-1} \|s_{k+j}\|_2 \leq \sum_{j=0}^{p-1} \delta_0^{k+j} \|s_0\|_2 \leq \delta^k \frac{\alpha_0}{1-\delta_0},$$

and therefore has a limit $x_* \in \overline{B_{\frac{\alpha_0}{1-\delta_0}}(x_0)}$. The limit point x_* satisfies

$$\begin{aligned}
&\|R'(x_*)^\dagger \Big(R(x_k) - R'(x_k)R'(x_k)^\dagger R(x_k)\Big)\|_2 \\
&= \|\Big(R'(x_*)^\dagger - R'(x_k)^\dagger\Big)\Big(R(x_k) - R'(x_k)R'(x_k)^\dagger R(x_k)\Big)\|_2 \\
&\leq \kappa \|x_* - x_k\|_2 \to 0
\end{aligned}$$

and

$$R(x_k) - R'(x_k)R'(x_k)^\dagger R(x_k) \to R(x_*).$$

Hence $R'(x_*)^\dagger R(x_*) = 0$. Finally,

$$\|x_* - x_k\|_2 = \lim_{p\to\infty} \|x_{k+p} - x_k\|_2 \leq \delta^k \frac{\alpha_0}{1-\delta_0}.$$

□

Remark 7.4.6 *i. Note that the proof Theorem 7.4.5 ony required the property $R'(x_k)^\dagger R'(x_k)\, R'(x_k)^\dagger = R'(x_k)^\dagger$, not all of the four properties of the Moore-Penrose pseudo inverse (see Problem 7.1).*

ii. Theorem 7.4.5 not only establishes the convergence, more precise, the r–linear convergence of the Gauss-Newton iteration to a stationary point x_* of the nonlinear least squares problem, but it even establishes the existence of such a point.

7.4.5. Nearly Rank Deficient Problems

In many problems the numerical determination of the rank of $R'(x_k)$ is difficult and in other problems the matrix $R'(x_k)^T R'(x_k)$ might be nearly singular for some iterates. In such cases, the computation of the Gauss–Newton step as the solution of

$$\min \tfrac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2 \tag{7.43}$$

is problematic. Instead the problem (7.43) should be modified. To see how, we note that the solution of (7.43) is the solution of the normal equation

$$R'(x_k)^T R'(x_k)s = -R'(x_k)^T R(x_k)$$

and vice versa. If we add a multiple $\mu_k > 0$ of the identity to $R'(x_k)^T R'(x_k)$, then the resulting matrix $R'(x_k)^T R'(x_k) + \mu_k I$ is positive definite if $\mu_k > 0$ and $\|(R'(x_k)^T R'(x_k) + \mu_k I)^{-1}\|_2 \leq \mu_k^{-1}$. Furthermore, using the SVD $R'(x_k) = U\Sigma V^T$ we can show that the unique solution s_k of

$$\left(R'(x_k)^T R'(x_k) + \mu_k I\right) s = -R'(x_k)^T R(x_k) \tag{7.44}$$

is given by

$$s_k = -\sum_{i=1}^{\min\{m,n\}} \frac{\sigma_i}{\sigma_i^2 + \mu_k} \left(u_i^T R(x_k)\right) v_i. \tag{7.45}$$

It can be shown that $\mu \to \|s_k(\mu)\|_2^2$, where

$$s_k(\mu) = -\sum_{i=1}^{\min\{m,n\}} \frac{\sigma_i}{\sigma_i^2 + \mu} \left(u_i^T R(x_k)\right) v_i.$$

is monotonically decreasing (see Problem 7.3). Furthermore,

$$\lim_{\mu_k \to 0} -\sum_{i=1}^{\min\{m,n\}} \frac{\sigma_i}{\sigma_i^2 + \mu_k} \left(u_i^T R(x_k)\right) v_i = -R'(x_k)^\dagger R'(x_k)$$

(see (7.18)). Thus the parameter $\mu_k > 0$ can be used to control the size of the step s_k. In the nearly rank deficient case we use the step (7.45). However, in an implementation of this variation of the Gauss–Newton method we do not set up and solve (7.44). Instead we note that (7.44) are the necessary and sufficient optimality conditions for the linear least squares problem

$$\min \tfrac{1}{2} \left\| \begin{pmatrix} R'(x_k) \\ \sqrt{\mu_k} I \end{pmatrix} s + \begin{pmatrix} R(x_k) \\ 0 \end{pmatrix} \right\|_2^2. \tag{7.46}$$

Note also that

$$\tfrac{1}{2} \left\| \begin{pmatrix} R'(x_k) \\ \sqrt{\mu_k} I \end{pmatrix} s + \begin{pmatrix} R(x_k) \\ 0 \end{pmatrix} \right\|_2^2 = \frac{1}{2} \|R'(x_k)s + R(x_k)\|_2^2 + \frac{\mu_k}{2} \|s\|_2^2.$$

There exist methods for the solution of linear least squares problems of the type (7.46) that utilize the special structure of this problem.

We still have to discuss the choice of μ_k. Clearly (7.45) is a perturbation of the Gauss-Newton step. We do not want to add an unneccessarily large $\mu_k > 0$. On the other hand, we want to pick $\mu_k > 0$ to ensure that the size of the step s_k (or alternatively the size of $(R'(x_k)^T R'(x_k) + \mu_k I)^{-1}$) becomes artificially large because the rank of $R'(x_k)$ is difficult to determine. A method that chooses μ_k adaptively is the Levenberg–Marquardt method [Lev44, Mar63]. This method is closely related to trust–region methods which were discussed in Section 6.3. In fact, if s_k solves (7.44) then it solves

$$\begin{array}{ll} \min & \frac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2 \\ \text{s.t.} & \|s\|_2 \leq \Delta_k. \end{array}$$

where $\Delta_k = \|s_k\|_2$. See Lemma 6.3.5.

Instead of using $\sqrt{\mu_k} I$ in (7.46) it is better to use $\sqrt{\mu_k} D_k$ with a properly chosen scaling matrix D_k In this case we have

$$\tfrac{1}{2} \left\| \begin{pmatrix} R'(x_k) \\ \sqrt{\mu_k} D_k \end{pmatrix} s + \begin{pmatrix} R(x_k) \\ 0 \end{pmatrix} \right\|_2^2 = \frac{1}{2} \|R'(x_k)s + R(x_k)\|_2^2 + \frac{\mu_k}{2} \|D_k s\|_2^2.$$

and

$$\min \tfrac{1}{2} \left\| \begin{pmatrix} R'(x_k) \\ \sqrt{\mu_k} D_k \end{pmatrix} s + \begin{pmatrix} R(x_k) \\ 0 \end{pmatrix} \right\|_2^2$$

is equivalent to

$$\begin{array}{ll} \min & \frac{1}{2} \|R'(x_k)s + R(x_k)\|_2^2 \\ \text{s.t.} & \|D_k s\|_2 \le \Delta_k. \end{array}$$

where $\Delta_k = \|D_k s_k\|_2$. For the choice of scaling see [Mar63] and [Mor78].

A trust-region view of the Levenberg–Marquardt method is described in [Mor78]. NL2SOL is an older, but still popular code for the solution of nonlinear least squares problems [DGW81, DGE81, Gay83]. For example, it is part of R is a language and environment for statistical computing and graphics. See `http://www.r-project.org`

7.5. Parameter Identification in Ordinary Differential Equations

The evaluation of the residual function R in the nonlinear least squares problems of Section 7.2 involved the evaluation of elementary functions, such as $+, *, \exp, \sin$. The residual function R in the nonlinear least squares problem of this section involves the solution of an ordinary differential equation (ODE). Typically, the solution of the ODE is not known explicitly but only an approximation of the solution can be computed by a numerical algorithm.

7.5.1. Least Squares Formulation of the Parameter Identification Problem

As a motivating example we will study the identification of reaction rates in a chemical reaction. First, we describe the system of ordinary differential equations that model the chemical reactions. Then we formulate the nonlinear least squares problem and discuss the computation of the derivative of the residual function R.

A reaction involving the compounds with molecular formula A, B, C, D is written as

$$\sigma_A A + \sigma_B B \to \sigma_C C + \sigma_D D. \tag{7.47}$$

Here $\sigma_A, \sigma_B, \sigma_C, \sigma_D$ are the stoichiometric coefficients. The compounds A, B are the reactants, C, D are the products. The $\to$ indicates that the reaction is irreversible. For a reversible reaction

we use $\rightleftharpoons$. For example, the reversible reaction of carbon dioxide and hydrogen to form methane plus water is

$$CO_2 + 4H_2 \rightleftharpoons CH_4 + 2H_2O. \tag{7.48}$$

Notice that the number of atoms on the left and on the right hand side balance. For example, there is a single C atom and there are two O atoms. However, the appearance of two reactants and two products is accidental. The stoichiometric coefficients in (7.48) are $\sigma_{CO_2} = 1, \sigma_{H_2} = 4, \sigma_{CH_4} = 1, \sigma_{H_2O} = 2$.

For each reaction we have a rate r of the reaction that together with the stoichiometric coefficients determine the change in concentrations resulting from the reaction. Concentrations are typically measured in [mol/L]. The reaction rate is defined as the number of reactive events per second per unit volume and measured in [mol $\sec^{-1}$ L^{-1}]. For example, if the rate of the reaction (7.47) is r and if denote the concentration of compound $A, \ldots$ by $C_A, \ldots$, we have the following changes in concentrations:

$$\frac{d}{dt}C_A(t) = \ldots - \sigma_A r \ldots, \quad \frac{d}{dt}C_B(t) = \ldots - \sigma_B r \ldots,$$
$$\frac{d}{dt}C_C(t) = \ldots \sigma_C r \ldots, \quad \frac{d}{dt}C_D(t) = \ldots \sigma_D r \ldots.$$

The dots indicate that other reactions or inflows and outflows will in general also enter the change in concentration. For a reaction of the form (7.47) the rate of the reaction r is of the form

$$r = kC_A^\alpha C_B^\beta,$$

where k is the reaction rate constant and α, β are nonnegative parameters. The sum $\alpha + \beta$ is called the order of the reaction. The reaction rate constant depends on the temperature and is often given by the Arrhenius equation (7.9).

As a particular example we consider an autocatalytic reaction. This example is taken from [Ram97, S. 4.2]. 'Autocatalysis is a term commonly used to describe the experimentally observable phenomenon of a homogeneous chemical reaction which shows a marked increase in rate in time, reaches its peak at about 50 percent conversion, and the drops off. The temperature has to remain constant and all ingredients must be mixed at the start for proper observation.' We consider the catalytic thermal decomposition of a single compound A into two products B and C, of which B is the autocatalytic agent. A can decompose via two routes, a slow uncatalyzed one (r_1) and another catalyzed by B (r_3) The three essential kinetic steps are

$$\begin{aligned} A &\to B + C && \text{Start or background reaction,} \\ A + B &\to AB && \text{Complex formation,} \\ AB &\to 2B + C && \text{Autocatalytic step.} \end{aligned}$$

The autocatalytic agent B forms a complex AB (second reaction). Next, the complex AB decomposes, thereby releasing B in addition to forming B and C (third reaction). The last two reactions form the path by which most of A decomposes. The first reaction is the starter, but continues concurrently with the last two as long as there is any A.

Again, we denote the concentration of compound $A, \ldots$ by $C_A, \ldots$. The reaction rates for the three reactions are

$$r_1 = k_1 C_A, \quad r_2 = k_2 C_A C_B, \quad r_3 = k_3 C_{AB}.$$

This leads to a system of ODEs

$$\frac{dC_A}{dt} = -k_1 C_A - k_2 C_A C_B, \tag{7.49a}$$
$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_A C_B + 2k_3 C_{AB}, \tag{7.49b}$$
$$\frac{dC_{AB}}{dt} = k_2 C_A C_B - k_3 C_{AB}, \tag{7.49c}$$
$$\frac{dC_C}{dt} = k_1 C_A + k_3 C_{AB} \tag{7.49d}$$

with given initial values

$$C_A(0) = C_{A_0}, \quad C_B(0) = C_{B_0}, \quad C_{AB}(0) = C_{AB_0}, \quad C_C(0) = C_{C_0}. \tag{7.49e}$$

Here time is measured in [sec] and concentrations are measured in [kmol/L].

To put the previous system of ODEs into a more general mathematical framework we define

$$\begin{aligned} p &= (k_1, k_2, k_3)^T, \\ y(t) &= (C_A(t), C_B(t), C_{AB}(t), C_C(t))^T, \\ y_0 &= (C_{A_0}, C_{B_0}, C_{AB_0}, C_{C_0})^T. \end{aligned}$$

We see that the initial value problem (7.49a)–(7.49e) is a particular instance of the initial value problem

$$\begin{aligned} y'(t) &= F(t, y(t), p), \quad t \in [t_0, t_f] \\ y(t_0) &= y_0(p), \end{aligned} \tag{7.50}$$

where $F : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$, $y_0 : \mathbb{R}^l \to \mathbb{R}^n$.

We first review a result on the existence and uniqueness of the solution of the initial value problem (7.50).

Theorem 7.5.1 *Let $G \subset \mathbb{R} \times \mathbb{R}^n$ be an open connected set, let $P \subset \mathbb{R}^l$, and for each $p \in P$ let $F(\cdot, \cdot, p) : G \to \mathbb{R}^n$ be continuous and bounded by M. If*

$$J = \{(t, y) \in \mathbb{R} \times \mathbb{R}^n \ : \ |t - t_0| \le \delta, \ \|y - y_0(p)\|_2 \le \delta M\} \subset G$$

and F is Lipschitz continuous with respect to y on J, i.e., there exists $L > 0$ such that

$$\|F(t, y, p) - F(t, z, p)\|_2 \le L\|y - z\|_2 \text{ for all } (t, y), (t, z) \in R,$$

then there exists a unique solution $y(\cdot; p)$ of the initial value problem (7.50) on $I = [t_0 - \delta, t_0 + \delta]$.

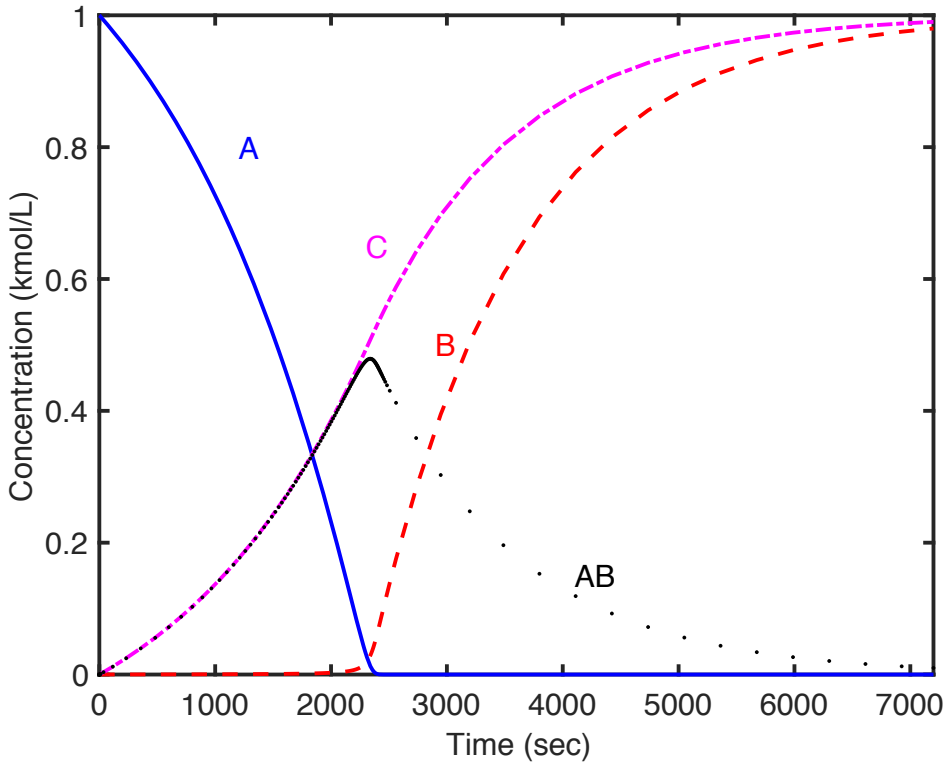


Figure 7.4: Solution of autocatalytic reaction (7.49) with initial values $C_A(0) = 1, C_B(0) = 0, C_{AB}(0) = 0, C_C(0) = 0$ and parameters $k_1 = 10^{-4}$, $k_2 = 1$, $k_3 = 8 * 10^{-4}$

Figure 7.4 shows the solution of (7.49a)–(7.49e) with initial values $C_A(0) = 1, C_B(0) = 0, C_{AB}(0) = 0, C_C(0) = 0$ and parameters $k_1 = 10^{-4}$, $k_2 = 1$, $k_3 = 8 * 10^{-4}$ on the time interval $t_0 = 0$ to $t_f = 7200$ secs. (2 hrs). The computations were done using the MATLAB ODE solver `ode23s` with the default options.

Now, suppose the reaction rates are not known, but have to be determined through an experiment. Given initial concentrations we will run the reaction and measure the concentrations $\hat{y}_i \in \mathbb{R}^4$ at times t_i, $i = 1, \ldots, m$. We try to fit the function $y(t; p)$ to the measurements, where $y(\cdot; p)$ is the solution of the initial value problem (7.50). This leads to a nonlinear least squares problem

$$\min_{p \in \mathbb{R}^l} \tfrac{1}{2} \|R(p)\|_2^2, \tag{7.51}$$

where

$$R(p) = \begin{pmatrix} y(t_1; p) - \hat{y}_1 \\ y(t_2; p) - \hat{y}_2 \\ \vdots \\ y(t_m; p) - \hat{y}_m \end{pmatrix} \in \mathbb{R}^{mn} \tag{7.52}$$

and $y(\cdot;p)$ is the solution of the initial value problem (7.50). For the evaluation of $R(p)$ at a given p we have to solve the ODE (7.50), evaluate the solution $y(\cdot;p)$ of this ODE at the points t_1, $i = 1,\dots,m$, and then assemble the vector $R(p)$ in (7.52). Thus, $R : \mathbb{R}^l \to \mathbb{R}^{mn}$ is a composition of functions

$$\begin{array}{ccccccc} p & \mapsto & y(\cdot;p) & \mapsto & y(t_i;p) & \mapsto & R(p), \\ \mathbb{R}^l & \mapsto & C(I,\mathbb{R}^n) & \mapsto & \mathbb{R}^n & \mapsto & \mathbb{R}^{mn}. \end{array}$$

By $C^\ell(S,\mathbb{R}^n)$ we denote the set of all functions $g : S \subset \mathbb{R}^j \to \mathbb{R}^n$ which are ℓ times continuously differentiable of S. If the ODE (7.50) is solved numerically, then we do not obtain the exact solution $y(\cdot;p)$, but only an approximation $y_h(\cdot;p)$. Consequently, we are only able to compute

$$p \mapsto y_h(\cdot;p) \mapsto y_h(t_i;p) \mapsto R_h(p).$$

The error between $R(p)$ and $R_h(p)$ depends on the accuracy of the ODE solver. Thus, while we want to minimize $\frac{1}{2}\|R(p)\|_2^2$, we do not have access to this function, but only to $\frac{1}{2}\|R_h(p)\|_2^2$.

7.5.2. Derivative Computation

Derivative Computation using Sensitivity Equations

Numerical algorithms for the solution of (7.51) require the Jacobian $R'(p) \in \mathbb{R}^{mn\times l}$ of R. For this purpose, we need to compute the derivative of

$$\begin{array}{ccc} p & \mapsto & y(\cdot;p), \\ \mathbb{R}^l & \mapsto & C(I,\mathbb{R}^n). \end{array}$$

The derivative $W(\cdot;p) = \frac{d}{dp}y(\cdot;p)$ is a matrix valued function

$$W(\cdot;p) : \mathbb{R} \to \mathbb{R}^{n\times l}$$

such that

$$\lim_{\|\delta p\|_2\to 0} \frac{1}{\|\delta p\|_2} \max_{t\in I} \|y(t;p+\delta p) - y(t;p) - W(t;p)\delta p\|_2 = 0. \tag{7.53}$$

The existence of the derivative can be established with the aid of the implicit function theorem, which also tells us how the derivative can be computed. For more details we refer to [Ama90], [HNW93, Sec. I.14], or [Wal98]. The following result is taken from [WA86, Thm. 3.2.16].

Theorem 7.5.2 *Let $G \subset \mathbb{R}\times\mathbb{R}^n$ be an open connected set, and $\bar p \in \mathbb{R}^l$. Further, let $\delta, \delta_1 > 0$ and define $I = [t_0-\delta, t_0+\delta]$ and $P = \left\{p \in \mathbb{R}^l \;:\; \|p-\bar p\|_2 < \delta_1\right\}$. Let $F \in C^\ell(G\times P,\mathbb{R}^n)$ and let $y \in C^\ell(P,\mathbb{R}^n)$ be given so that $(x_0, y_0(p)) \in G$ for all $p \in P$. Suppose that F is bounded on $G\times P$ by M and suppose that*

$$\|y_0(p) - y_0(\bar p)\|_2 \le \bar\epsilon \quad \text{for all } p \in P.$$

If

$$\{(t,y) \in \mathbb{R} \times \mathbb{R}^n \ : \ t \in I, \ \|y - y_0(p)\|_2 \le \bar{\epsilon} + M\|t - t_0\|_2\} \subset G,$$

then for each $p \in P$ there exists a unique solution $y(\cdot;p) \in C^\ell(I,\mathbb{R}^n)$ of the initial value problem (7.50). Moreover, the solution is ℓ times continuously differentiable with respect to p and the first derivative $w(\cdot;p) = \frac{d}{dp}y(\cdot;p)$ is the solution of

$$\begin{aligned} W'(t) &= \tfrac{\partial}{\partial y}F(t,y(t;p),p)W(t) + \tfrac{\partial}{\partial p}F(t,y(t;p),p) \quad t \in I \\ W(t_0) &= \tfrac{d}{dp}y_0(p). \end{aligned} \tag{7.54}$$

The linear differential equation (7.54) is sometimes also called the sensitivity equation. The function $W_{ij}(t)$ is the sensitivity of the ith component of the solution with respect to the parameter p_j. From (7.53) we see that

$$y_i(t;p + (\delta p)_j e_j) = y_i(t;p) + W_{ij}(t;p)(\delta p)_j + o((\delta p)_j) \qquad \text{for all } t \in I.$$

Here e_j denotes the jth unit vector and $h \in \mathbb{R}$.

For the ODE (7.49a)–(7.49e) written in the more abstract notation

$$\begin{pmatrix} \frac{d}{dt}y_1 \\ \frac{d}{dt}y_2 \\ \frac{d}{dt}y_3 \\ \frac{d}{dt}y_4 \end{pmatrix} = \begin{pmatrix} -k_1y_1 - k_2y_1y_2, \\ k_1y_1 - k_2y_1y_2 + 2k_3y_3, \\ k_2y_1y_2 - k_3y_3, \\ k_1y_1 + k_3y_3 \end{pmatrix}, \tag{7.55}$$

$$y_1(0) = y_{1,0}, \quad y_2(0) = y_{2,0}, \quad y_3(0) = y_{3,0}, \quad y_4(0) = y_{4,0}. \tag{7.56}$$

the sensitivity equations are given by

$$\begin{aligned} &\begin{pmatrix} \frac{d}{dt}W_{11} & \frac{d}{dt}W_{12} & \frac{d}{dt}W_{13} \\ \frac{d}{dt}W_{21} & \frac{d}{dt}W_{22} & \frac{d}{dt}W_{23} \\ \frac{d}{dt}W_{31} & \frac{d}{dt}W_{32} & \frac{d}{dt}W_{33} \\ \frac{d}{dt}W_{41} & \frac{d}{dt}W_{42} & \frac{d}{dt}W_{43} \end{pmatrix} \\ &= \begin{pmatrix} -k_1 - k_2y_2 & -k_2y_1 & 0 & 0 \\ k_1 - k_2y_2 & -k_2y_1 & 2k_3 & 0 \\ k_2y_2 & k_2y_1 & -k_3 & 0 \\ k_1 & 0 & k_3 & 0 \end{pmatrix} \begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \\ W_{41} & W_{42} & W_{43} \end{pmatrix} \\ &\quad + \begin{pmatrix} -y_1 & -y_1y_2 & 0 \\ y_1 & -y_1y_2 & 2y_3 \\ 0 & y_1y_2 & -y_3 \\ y_1 & 0 & y_3 \end{pmatrix}. \end{aligned} \tag{7.57}$$

Since the initial values (7.56) do not depend on the parameter p, the sensitivity W obeys the initial conditions

$$W_{ij}(0) = 0 \quad \text{for } i = 1, \ldots, 4, \ j = 1, \ldots, 3. \tag{7.58}$$

With the solution $W(\cdot; p)$ of the sensitivity equation (7.54) the Jacobian of R defined in (7.52) is given by

$$R'(p) = \begin{pmatrix} W(t_1; p) \\ W(t_2; p) \\ \vdots \\ W(t_m; p) \end{pmatrix} \in \mathbb{R}^{mn \times l}. \tag{7.59}$$

In practice the ODE (7.50) and the sensitivity equations have to be solved numerically. Since most ODE solvers are adaptive, one has to solve the original ODE (7.50) and the sensitivity equation (7.54) simultaneously for y and W. Instead of the exact solutions $y(\cdot; p)$ and $W(\cdot; p)$ of (7.50) and (7.54) one obtains approximations $y_h(\cdot; p)$ and $W_h(\cdot; p)$ thereof. Thus in practice one has only $R_h(p)$ and

$$(R'(p))_h = \begin{pmatrix} W_h(t_1; p) \\ W_h(t_2; p) \\ \vdots \\ W_h(t_m; p) \end{pmatrix} \in \mathbb{R}^{mn \times l} \tag{7.60}$$

available. It holds that $R_h(p) \approx R(p)$ and $(R'(p))_h \approx R'(p)$ and estimates for the errors $\|R_h(p) \approx R(p)\|_2$ and $\|(R'(p))_h \approx R'(p)\|_2$ are typically available. Usually, the approximation $(R'(p))_h$ of $R'(p)$ is not the derivative of $R_h(p)$, the approximation of $R(p)$. Therefore we have chosen the notation $(R'(p))_h$ over $(R_h(p))'$. [2] In fact, often we do not even know whether $R_h(p)$ is differentiable. Codes for the numerical solution of ODEs often chose the time steps adaptively. Rules for this adaptation involve min, max, $|\cdot|$. This might lead to the nondifferentiability of $R_h(p)$.

If (7.50) and (7.54) have to be solved simultaneously for y and W using a numerical ODE solver, the sensitivity equation (7.54) typically has to be written in vector form. For example the ODE resulting from (7.55) and (7.57) is given by

[2] Here $(R'(p))_h$ indicates the we differentiate first an then discretize the derivative, whereas $(R_h(p))'$ indicates that we discretize first and take the derivative of of the discretized $R_h(p)$.

$$\begin{pmatrix} \frac{d}{dt}y_1 \\ \frac{d}{dt}y_2 \\ \frac{d}{dt}y_3 \\ \frac{d}{dt}y_4 \\ \frac{d}{dt}W_{11} \\ \frac{d}{dt}W_{21} \\ \frac{d}{dt}W_{31} \\ \frac{d}{dt}W_{41} \\ \frac{d}{dt}W_{12} \\ \frac{d}{dt}W_{22} \\ \frac{d}{dt}W_{32} \\ \frac{d}{dt}W_{42} \\ \frac{d}{dt}W_{13} \\ \frac{d}{dt}W_{23} \\ \frac{d}{dt}W_{33} \\ \frac{d}{dt}W_{43} \end{pmatrix} = \begin{pmatrix} -k_1y_1 - k_2y_2y_3, \\ k_1y_1 - k_2y_1y_2 + 2k_3y_3, \\ k_2y_1y_2 - k_3y_3, \\ k_1y_1 + k_3y_3, \\ (-k_1 - k_2y_2)W_{11} - k_2y_1W_{21} - y_1, \\ (k_1 - k_2y_2)W_{11} - k_2y_1W_{21} + 2k_3W_{31} + y_1, \\ k_2y_2W_{11} + k_2y_1W_{21} - k_3W_{31}, \\ k_1W_{11} + k_3W_{31} + y_1, \\ (-k_1 - k_2y_2)W_{12} - k_2y_1W_{22} - y_1y_2, \\ (k_1 - k_2y_2)W_{12} - k_2y_1W_{22} + 2k_3W_{32} - y_1y_2, \\ k_2y_2W_{12} + k_2y_1W_{22} - k_3W_{32} + y_1y_2, \\ k_1W_{12} + k_3W_{32} + y_1, \\ (-k_1 - k_2y_2)W_{13} - k_2y_1W_{23}, \\ (k_1 - k_2y_2)W_{13} - k_2y_1W_{23} + 2k_3W_{33} + 2y_3, \\ k_2y_2W_{13} + k_2y_1W_{23} - k_3W_{33} - y_3, \\ k_1W_{13} + k_3W_{33} + y_3 \end{pmatrix}. \tag{7.61}$$

Figures 7.5 and 7.6 show the solution of (7.61). As before, the computations were done using the MATLAB ODE solver `ode23s` with the default options. The parameters were $k_1 = 10^{-4}$, $k_2 = 1$, $k_3 = 8 * 10^{-4}$ and $t_0 = 0$ to $t_f = 7200$ secs. (2 hrs). Note the different scales of the sensitivities with respect to k_1, k_3 and k_2. For example, since $k_1 = 10^{-4}$, the sensitivities dy_j/dk_1 can be about 10^4 times larger than y_j, $j = 1, \ldots, 4$. In this case scaling issues have to be dealt with when solving (7.61). When using a numerical solver such as the MATLAB ODE solver `ode23s` it might be necessary to choose different absolute tolerances `AbsTol` for each solution component in (7.61). In our computations we have used `AbsTol` $= 10^{-6}$ for all components (the default). A more sensible choice might be `AbsTol` $= 10^{-6}$ for components 1–4, `AbsTol` $= 10^{-6}/k_1$ for components 5–8, `AbsTol` $= 10^{-6}/k_2$ for components 9–12, and `AbsTol` $= 10^{-6}/k_3$ for components 13–16.

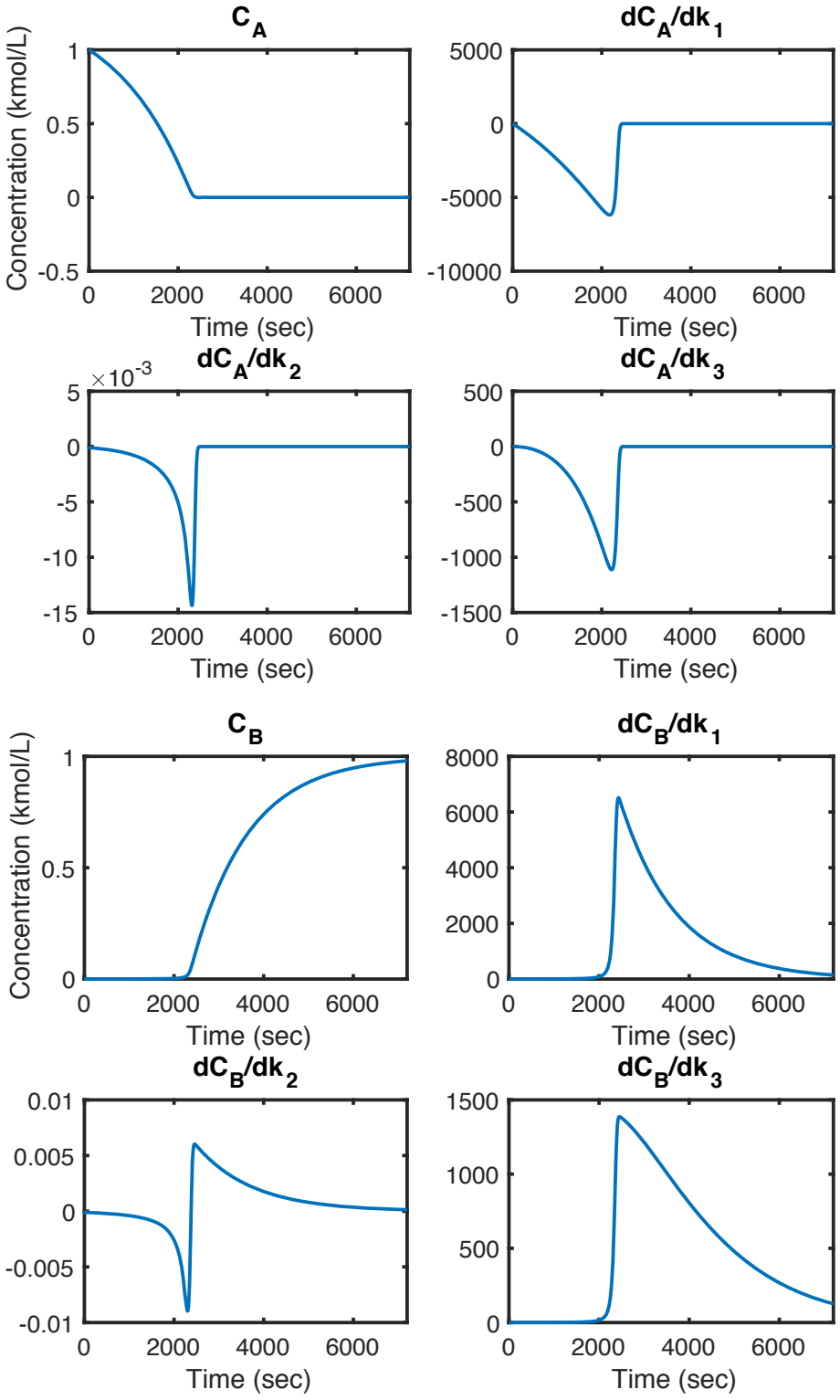


Figure 7.5: y_1, y_2 and corresponding sensitivities

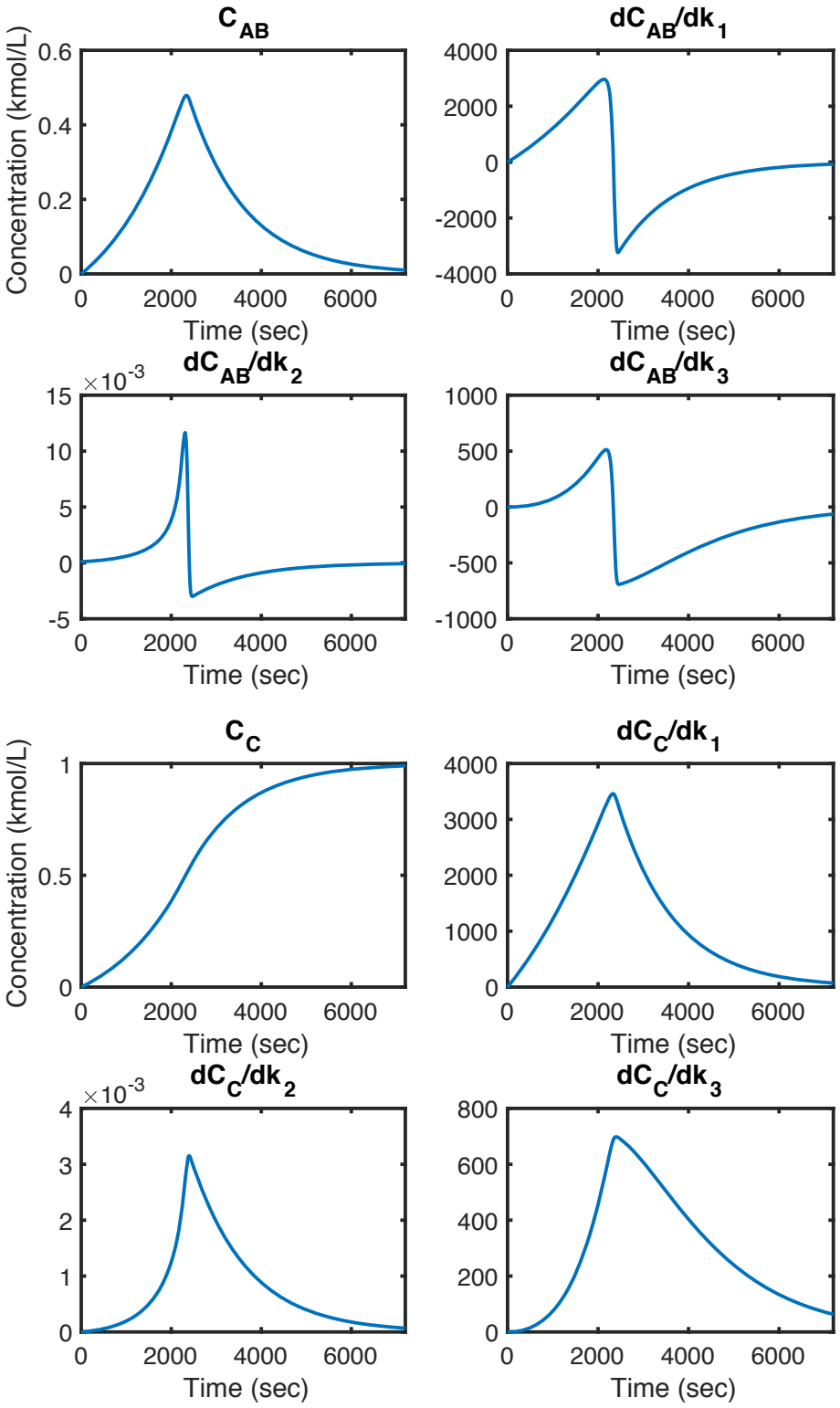


Figure 7.6: y_3, y_4 and corresponding sensitivities

DASSL and DASPK are two Fortran codes for the solution of ODEs [BCP95]. Actually, DASSL and DASPK solve differential–algebraic equations (DAEs), which are systems of ODEs coupled with nonlinear algebraic equations. Both codes have been augmented to solve the DAE and the corresponding sensitivity equations simultaneously. The original codes DASSL and DASPK and their augmentations DASSLSO and DASPKSO are available. For details we refer to the paper [MP96].

Derivative Computation using Finite Differences

Now we apply this to the approximation of the derivative $R'(p)$. This requires the approximation of the derivatives of

$$p \mapsto y(t_i; p).$$

The jth partial derivative can be approximated by

$$\frac{\partial}{\partial p_j} y(t_i; p) \approx \frac{1}{(\delta p)_j} (y(t_i; p + (\delta p)_j e_j) - y(t_i; p)),$$

where e_j is the jth unit vector and $(\delta p)_j \in \mathbb{R}$. The scalar $(\delta p)_j \in \mathbb{R}$ can and typically does vary with j. Thus, we can compute a finite difference approximation of $R'(p)$ as follows. For $j = 1, \ldots, k$ choose $(\delta p)_j \in \mathbb{R}$ sufficiently small and compute the solution $y(\cdot, p + (\delta p)_j e_j)$ of (7.50) with p replaced by $p + (\delta p)_j e_j$. The jth column $(R'(p))_j$ of $R'(p)$ is then approximated by

$$(R'(p))_j \approx \begin{pmatrix} (y(t_1; p + (\delta p)_j e_j) - y(t_1; p))/(\delta p)_j \\ (y(t_2; p + (\delta p)_j e_j) - y(t_2; p))/(\delta p)_j \\ \vdots \\ (y(t_m; p + (\delta p)_j e_j) - y(t_m; p))/(\delta p)_j \end{pmatrix} \in \mathbb{R}^{mn}. \tag{7.62}$$

As we have pointed out several times before, the exact solution of (7.50) can not be computed, but only an approximation thereof. Thus, instead of $y(t_i; p)$, $y(t_i; p + (\delta p)_j e_j)$, $j = 1, \ldots, k$, we only have $y_h(t_i; p)$, $y_h(t_i; p + (\delta p)_j e_j)$, $j = 1, \ldots, k$. In practice the jth column $(R'(p))_j$ of $R'(p)$ is approximated by

$$(R'(p))_j \approx \begin{pmatrix} (y_h(t_1; p + (\delta p)_j e_j) - y_h(t_1; p))/(\delta p)_j \\ (y_h(t_2; p + (\delta p)_j e_j) - y_h(t_2; p))/(\delta p)_j \\ \vdots \\ (y_h(t_m; p + (\delta p)_j e_j) - y_h(t_m; p))/(\delta p)_j \end{pmatrix} \in \mathbb{R}^{mn}. \tag{7.63}$$

Figures 7.7 to 7.9 below show the solution component y_4 of (7.55) and the sensitivities of this component with respect to the parameters $p_j = k_j$, $j = 1, 2, 3$. The solid plots are the sensitivities computed using the sensitivity equation method and the dashed plots are the finite difference approximations of the sensitivities. The approximate solutions of (7.55) were computed using the

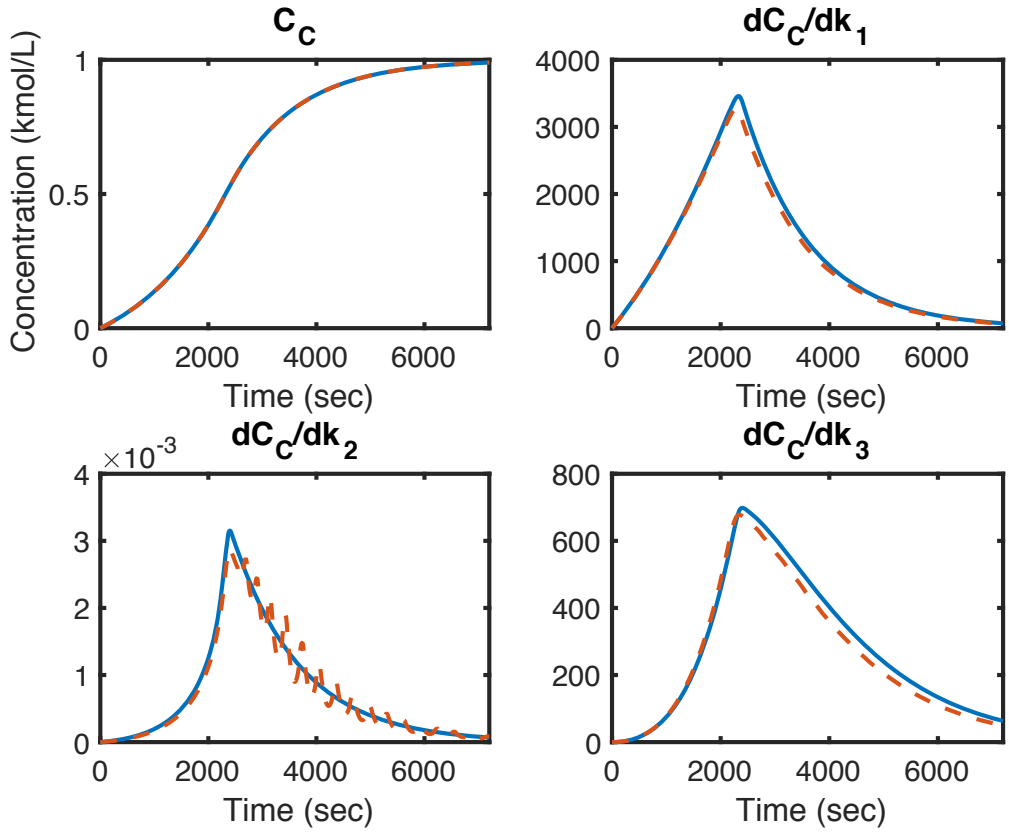


Figure 7.7: y_4 and corresponding sensitivities. The solid curves are the sensitivities computed using the sensitivity equation method (these are identical to the corresponding plots in Figure 7.6) and the dashed curves are the sensitivity approximations via finite differences with $(\delta p)_j = 10^{-1} p_j$, $j = 1, 2, 3$.

MATLAB ODE solver `ode23s` with the default options. The parameters were $k_1 = 10^{-4}$, $k_2 = 1$, $k_3 = 8 * 10^{-4}$ and $t_0 = 0$ to $t_f = 7200$ secs. (2 hrs). If we want to extend the error analysis of finite difference approximations performed for the function g to $y(\cdot; p)$, then we will obtain a different constant L_j for each component p_j and $L_j \approx \max_t \| \frac{\partial^2}{\partial p_j^2} y(t; p) \|_2$. To compute the finite difference step size $(\delta p)_j$ we need an estimate for L_j and an estimate for the error level ϵ in the evaluation of $y(t; p)$, $t \in [0, 7200]$. Using our previous sensitivity computations we estimate that $\max_t \| \frac{\partial}{\partial p_j} y(t; p) \|_2 = O(1/p_j)$. (This seems to be reasonable for $j = 1, 3$, but it is too high for $j = 2$.) For the second partial derivatives we use the estimate $\max_t \| \frac{\partial^2}{\partial p_j^2} y(t; p) \|_2 = O(1/p_j^2)$. Thus, the optimal finite difference step size for the j parameter is $(\delta p_*)_j = (2/\sqrt{L_j})\sqrt{\epsilon} = O(p_j \sqrt{\epsilon})$. The default options in the MATLAB ODE solver `ode23s` attempt to compute an approximate solution that is within `AbsTol` $= 10^{-6}$ of the true solution. Thus, we estimate that $\epsilon \approx 10^{-6}$. This gives

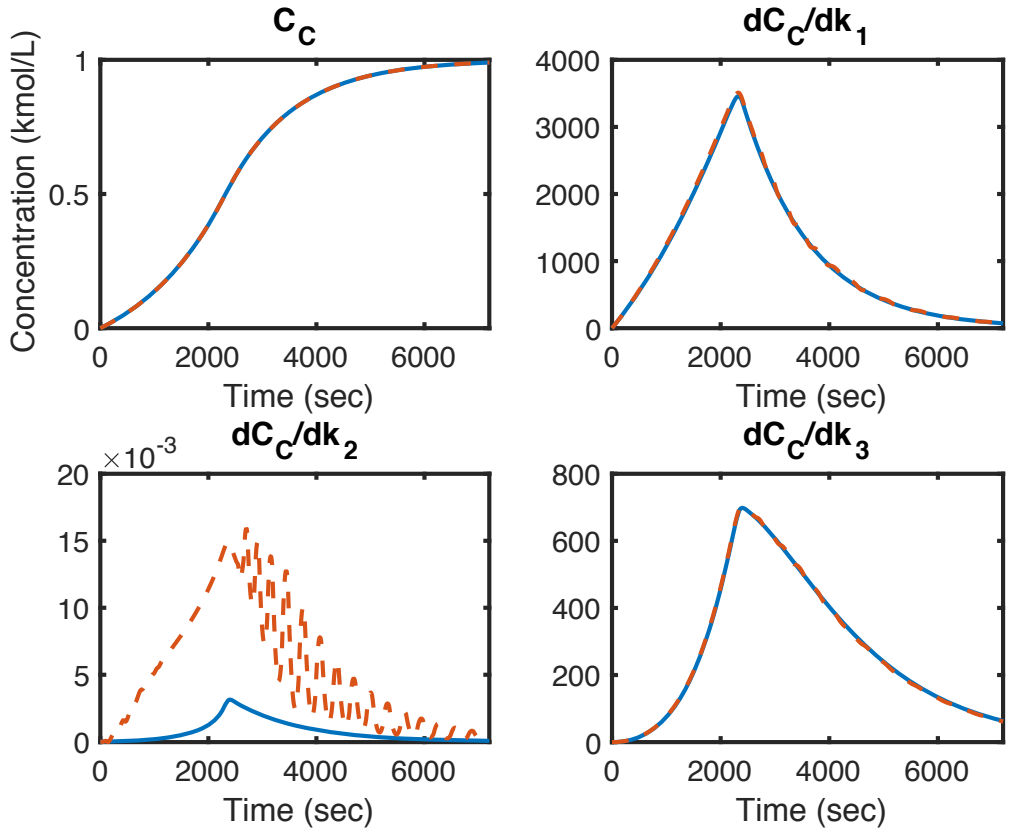


Figure 7.8: y_4 and corresponding sensitivities. The solid curves are the sensitivities computed using the sensitivity equation method (these are identical to the corresponding plots in Figure 7.6) and the dashed curves are the sensitivity approximations via finite differences with $(\delta p)_j = 10^{-2} p_j$, $j = 1, 2, 3$.

an estimate $(\delta p_*)_j = O(p_j 10^{-3})$ for the optimal step size. We see that for $j = 1, 3$ the best agreements between the the sensitivities computed using the sensitivity equation method and the finite difference approximations of the sensitivities are achieved for $(\delta p_*)_j = O(p_j 10^{-2})$, $j = 1, 3$. For $j = 2$, however, the best agreements are achieved for $(\delta p_*)_2 = O(p_2 10^{-1})$. The calculations for the solution components $y_1 - y_3$ gave similar results. This example indicates how difficult it is to approximate derivatives of vector valued functions using finite differences, especially if the variables and functions have different scales and the functions are not computed exactly.

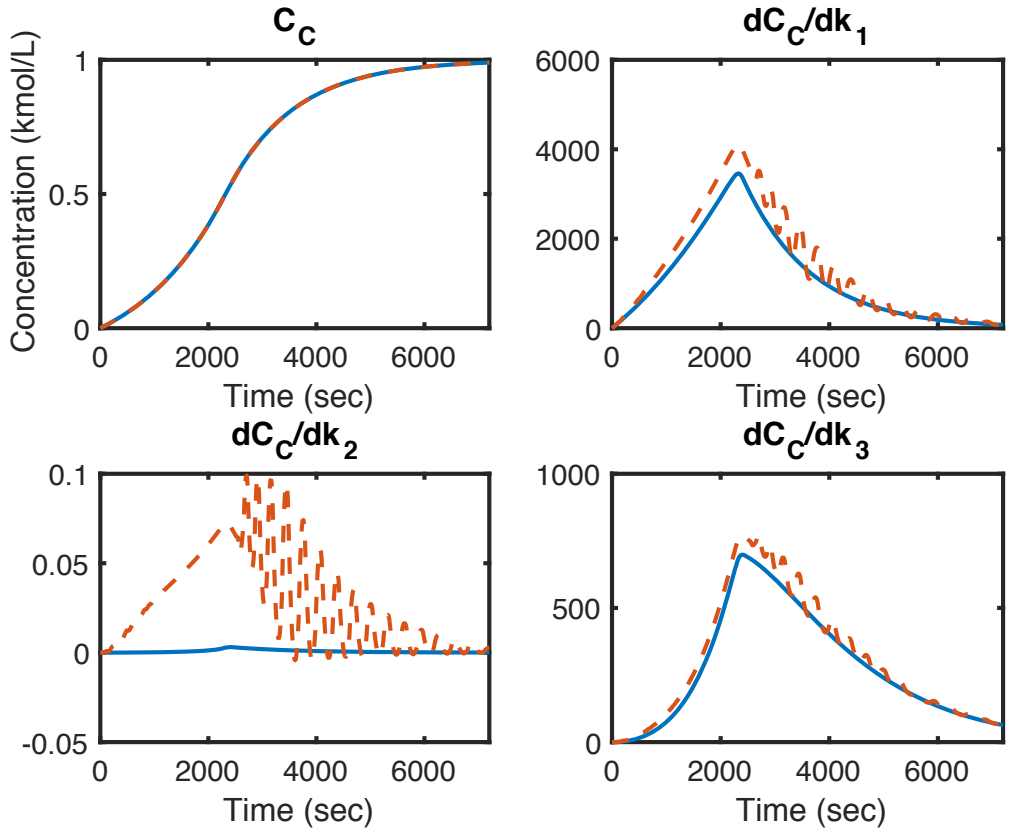


Figure 7.9: y_4 and corresponding sensitivities. The solid curves are the sensitivities computed using the sensitivity equation method (these are identical to the corresponding plots in Figure 7.6) and the dashed curves are the sensitivity approximations via finite differences with $(\delta p)_j = 10^{-3} p_j$, $j = 1, 2, 3$.

Derivative Computation using Automatic Differentiation

Another approach to the computation of derivatives of the solution of ODEs with respect to parameters is automatic differentiation (which is now also known as computational differentiation). Given the source code of a computer program (or a set of computer programs) for the solution of a differential equation, automatic differentiation tools take this program and generate source code for a new program that computes the solution of the ODE as well as the derivatives of the solution with respect to parameters. Actually, this technique is not limited to computer programs for the solution of ODEs. Automatic differentiation is based on the observation that inside computer programs only elementary functions such as $+, *, \sin, \log$ are executed. The derivatives of these elementary operations are known. A computer program can be viewed as a composition of such elementary functions. The derivative of a composition of functions is obtained by the chain rule. This is the basic mathematical observation underlying automatic differentiation. See the paper [Gri03]

by Griewank and the book [GW08] by Griewank and Walther. Of course computer programs also include operations that are not necessarily differentiable such as max, $|\cdot|$, and if-then-else statements. Automatic differentiation techniques will always generate an augmented program that also generates 'derivatives'. It is important that the user applies these tools intelligently.

ADIC (for the automatic differentiation of programs written in C/C++) and ADIFOR (for the automatic differentiation of programs written in Fortran 77) are available from https://wiki.mcs.anl.gov/autodiff[3].

[3]Accessed April 10, 2017

7.6. Problems

Problem 7.1 Let $A \in \mathbb{R}^{m\times n}$. Show that $A^\dagger$ defined in (7.19) satisfies

$$A\,A^\dagger A = A, \quad A^\dagger A\,A^\dagger = A^\dagger, \quad (AA^\dagger)^T = A\,A^\dagger, \quad (A^\dagger A)^T = A^\dagger A.$$

Note: It can actually be shown that there exists only one matrix $X \in \mathbb{R}^{n\times m}$ that satisfies

$$AXA = A, \quad XAX = X, \quad (AX)^T = AX, \quad (XA)^T = XA.$$

Hence these four identities are also used to define the Moore–Pensrose pseudo inverse. For more details see, e.g., [Bjö96, pp. 15-17] and [BIG74, CM79, Gro77, Nas76].

Problem 7.2 Let $A \in \mathbb{R}^{m\times n}$ have rank m ($m \le n$). Show that

$$A^\dagger = A^T(A\,A^T)^{-1}.$$

Problem 7.3 Consider the regularized linear least squares problem

$$\min \frac{1}{2}\|Ax - b\|_2^2 + \frac{\mu}{2}\|x\|_2^2, \tag{7.64}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $\mu \ge 0$.

i. Show that for each $\mu > 0$, (7.64) has a unique solution $x(\mu)$.

ii. Let $\mu_2 > \mu_1 > 0$ and let $x_1 = x(\mu_1)$, $x_2 = x(\mu_2)$ be the solutions of (7.64) with $\mu = \mu_1$ and $\mu = \mu_2$, respectively.

Show that

$$\begin{aligned}
\frac{1}{2}\|Ax_1 - b\|_2^2 + \frac{\mu_1}{2}\|x_1\|_2^2 &\le \frac{1}{2}\|Ax_2 - b\|_2^2 + \frac{\mu_2}{2}\|x_2\|_2^2,\\
\frac{1}{2}\|Ax_1 - b\|_2^2 &\le \frac{1}{2}\|Ax_2 - b\|_2^2,\\
\|x_2\|_2^2 &\le \|x_1\|_2^2.
\end{aligned}$$

Problem 7.4 In many applications the linear least squares problem

$$\min \tfrac{1}{2}\|R'(x_k)s + R(x_k)\|_2^2$$

has to be solved iteratively using, e.g., the conjugate gradient methods described in Sections 3.7.3 and 3.7.3. We assume that the computed step s_k satisfies

$$\|R'(x_k)^T(R'(x_k)s + R(x_k))\|_2 \leq \eta_k \|R'(x_k)^T R(x_k)\|_2.$$

Formulate and prove an extension of the local convergence Theorem 7.4.2 for this inexact Gauss-Newton method.

Hint: Revisit Theorem 5.3.1. One convergence result for inexact Gauss-Newton methods is presented in [Mar87].

Problem 7.5 (Taken from [KMN88, p. 380])

Type I supernovae have been found to follow a specific pattern of luminosity (brightness). Beginning a few days after the maximum luminosity this pattern may be described by

$$L(t) = C_1 \exp(-t/\alpha_1) + C_2 \exp(-t/\alpha_2),$$

where t is the time in days after the maximum luminosity and $L(t)$ is the luminosity relative to the maximum luminosity. The table in `lumi_data.m` gives the relative luminosity for the type I supernovae SN1939A measured in 1939. The peak luminosity occured at day 0.0, but all measurement before day 7.0 are omitted because the model above cannot account for the luminosity before and immediately after the maximum.

i. Plot the data. You should notice two distinct regions, and thus two exponentials are required to provide an adequate fit.

ii. Use the function `lsqcurvefit` from the MATLAB optimization toolbox to fit the data to the model above, i.e., to determine $C_1, C_2, \alpha_1, \alpha_2$. Plot the fit along with the data. Also plot the residuals. Do the residuals look random? Experience plays a role in choosing the starting values It is known that the time constants α_1 and α_2 are about 5.0 and 60.0, respectively. Try different values for C_1 and C_2. How sensitive is the resulting fit to these values?

Problem 7.6 The Gas–Oil–Cracking problem is a system of two ODEs given by

$$\begin{aligned} \begin{pmatrix} y_1'(t) \\ y_2'(t) \end{pmatrix} &= \begin{pmatrix} -(p_1 + p_3)y_1^2(t) \\ p_1 y_1^2(t) - p_2 y_2(t) \end{pmatrix} \quad \text{in } (0,4), \\ \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \end{aligned} \tag{7.65}$$

Formulate the system of ODEs for $W(\cdot;p) = \frac{d}{dp} y(\cdot;p)$.

Write a program for the computation of approximate sensitivities $W_h^{\mathrm{S}}(\cdot;p)$ via the sensitivity equation method and for the computation of approximate sensitivities $W_h^{\mathrm{FD}}(\cdot;p)$ via finite

differences. Use the parameter values $p_1 = 0.9875, p_2 = 0.2566, p_3 = 0.3323$. Evaluate the sensitivities at $t_i = 0, 0, 1, 0.2, \ldots, 4$ (`tspan = [0:0.1:4]` if you use the Matlab ODE solvers). For the computation of the finite difference approximations use $(\delta p)_j = \delta p$, $j = 1, \ldots, 3$, with $\delta p = 10^{-3}, 10^{-6}, 10^{-9}$. For each δp plot the error $W_h^S(\cdot; p) - W_h^{FD}(\cdot; p)$ (in log-format).

Problem 7.7 Consider the continuously stirred tank reactor (CSTR) with heating jacket in

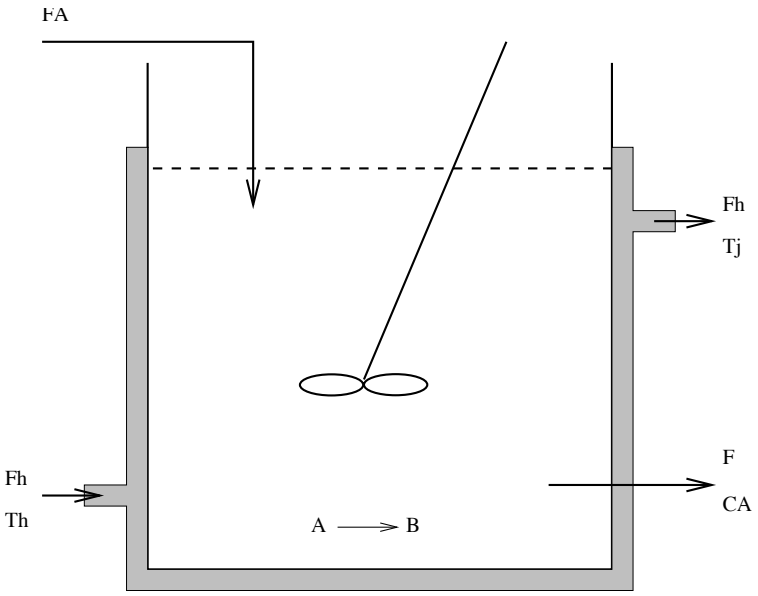


Figure 7.10: A continuously stirred tank reactor with heating jacket.

Figure 7.10. Reactant A is fed at a flow rate F, molar concentration C_{Ao}, and temperature T_A to the reactor, where the irreversible endothermic reaction $A \to B$ occurs. The rate of the reaction is given by the following relation

$$r = k_0 \exp\left(-\frac{E}{RT}\right) C_A,$$

where C_A is the molar concentration of A in the reactor holdup, T is the reactor temperature. The parameters k_0, E, and R are called the pre-exponential factor, the activation energy, and the gas constant respectively. In the CSTR, the product stream is withdrawn at flow rate F and heat is provided to the reactor through the heating jacket, where the heating fluid is fed at a flow rate F_h and temperature T_h. We assume that the flow rates F, and F_h are constant.

The system is described by the following set of differential equations

$$\frac{d}{dt}C_A(t) = \frac{F_A}{V}(C_{Ao} - C_A(t)) - k_0 \exp\left(-\frac{E_a}{RT(t)}\right) C_A(t), \tag{7.66a}$$

$$\frac{d}{dt}C_B(t) = -\frac{F_A}{V}C_B(t) + k_0 \exp\left(-\frac{E_a}{RT(t)}\right) C_A(t), \tag{7.66b}$$

$$\frac{d}{dt}T(t) = \frac{F_A}{V}(T_A - T(t)) - k_0 \exp\left(-\frac{E_a}{RT(t)}\right) C_A(t)\frac{\Delta H_r}{\rho c_p} + \frac{U\mathrm{A}(T_j(t) - T(t))}{\rho V c_p}, \tag{7.66c}$$

$$\frac{d}{dt}T_j(t) = \frac{F_h}{V_h}(T_h - T_j(t)) - \frac{U\mathrm{A}(T_j(t) - T(t))}{\rho_h V_h c_{ph}}. \tag{7.66d}$$

Here U is heat transfer rate and A denotes the heat transfer area. The other quantities and their values are described in Table 7.1.

Changes in the concentration C_A of chemical A are due to inflow and they are due to the reaction. In (7.66a) this is modeled by the terms $F_A(C_{Ao} - C_A(t))/V$ and

$$-k_0 \exp\left(-\frac{E_a}{RT(t)}\right) C_A(t),$$

respectively. Equation (7.66b) is interpreted analogously. The heat inside the reactor affected by the inflow, the reaction, and heat transfer between jacket and tank. In (7.66c) the term $F_A(T_A - T(t))/V$ models the change in heat due to inflow,

$$-k_0 \exp\left(-\frac{E_a}{RT(t)}\right) C_A(t)\frac{\Delta H_r}{\rho c_p}$$

models the heat generated by the reaction, and the term $U\mathrm{A}(T_j(t) - T(t))/(\rho V c_p)$ describes the change of heat due to differences between temperature in the reactor T and in the heating jacket T_j. The change in the jacket temperature T_j is modeled analogously, except that no reaction takes place in the jacket.

a. If we set

$$y(t) = (C_A(t), C_B(t), T(t), T_j(t))^T \tag{7.67}$$

and

$$p = (T_A, T_h, \rho, \rho_h)^T, \tag{7.68}$$

then (7.66a)–(7.66d) can be written as

$$\frac{d}{dt}y(t) = f(y(t), p), \quad y(0) = y_0. \tag{7.69}$$

Solve the system (7.66a)–(7.66d) on the time interval $[0, 60]$ (min) using the Matlab function `ode23s` and the initial conditions and problem parameters specified in Table 7.1.

Variable	Description	Value
C_{Ao}	feed reactant concentration [mol l^{-1}]	5.0
$C_A(0)$	reactant concentration in reactor at time $t = 0$ [mol l^{-1}]	1.596
$C_B(0)$	product concentration in reactor at time $t = 0$ [mol l^{-1}]	3.404
c_p	specific heat capacity [J g^{-1} K^{-1}]	6.0
c_{ph}	specific heat capacity of heating fluid [J g^{-1} K^{-1}]	6.0
E	activation energy [J mol^{-1}]	50000
R	gas constant [J mol^{-1} K^{-1}]	8.3144
k_0	pre-exponential factor in reaction rate [min^{-1}]	1.0×10^9
T_A	feed reactant temperature [K]	300
$T(0)$	reactor temperature at time $t = 0$ [K]	284.08
T_h	heating fluid temperature [K]	375
$T_j(0)$	jacket temperature at time $t = 0$ [K]	285.37
V	reactor holdup volume at time [l]	10.0
V_h	jacket volume [l]	1.0
F	inlet/outlet flow rate in/from reactor [l min^{-1}]	3.0
F_h	heating fluid flow rate [l min^{-1}]	0.1
ρ	liquid density [g l^{-1}]	600
ρ_h	liquid density of heating fluid [g l^{-1}]	600
ΔH_r	heat of reaction [J mol^{-1}]	20000
$\rho c_p/(U\mathrm{A})$	[min l^{-1}]	0.144

Table 7.1: Parameters for the CSTR with heating jacket

b. Formulate the system of ODEs for $W(\cdot;p) = \frac{d}{dp}y(\cdot;p)$.

Write a program for the computation of approximate sensitivities $W_h^S(\cdot;p)$ via the sensitivity equation method and for the computation of approximate sensitivities $W_h^{FD}(\cdot;p)$ via finite differences.

Use the parameter values specified in Table 7.1 for p in (7.68).

Problem 7.8 In this problem we will determine parameters in a simple model for an electrical furnace.

The mathematical model for the oven involves the following quantities: t: time (seconds), T: temperature inside the oven (^{0}C), C: 'heat capacity' of the oven including load (joule/^{0}C), Q: rate of loss of heat inside the oven to the environment (joule/sec), V: voltage of the source of electricity (volt), I: intensity of the electric current (amp), R: resistance of the heating of the oven plus regulation resistance (ohm) ($R = \infty$ corresponds to the 'open' (disconnected) circuit). Then, according to the laws of Physics:

- $I = V/R$,
- $VI = V^2/R$ = power = heat generated per second,
- $Q = kT$, where k is a constant of loss of heat per second and per degree of temperature difference between oven and the environment,
- $(V^2/R) - Q$ = heat gain of the oven per second,
- $[(V^2/R) - kT]/C$ = rate of increase of temperature of the oven (^{0} C per second).

The temperature of the oven will then evolve according to the differential equation

$$T'(t) = \frac{(V^2/R(t)) - kT(t)}{C}. \tag{7.70}$$

With $\alpha = k/C > 0$ and $\beta = V^2/C > 0$, this equation is of the form

$$T'(t) = -\alpha T(t) + \beta/R(t). \tag{7.71}$$

We set

$$u(t) = 1/R(t).$$

This gives the differential equation

$$T'(t) = -\alpha T(t) + \beta u(t). \tag{7.72}$$

The model (7.72) depends on the parameters α, β. These depend on the particular oven (geometry, material, ...). Our goal is to determine the parameters from measurements using the least squares formulation.

First, we note that the solution of (7.72) for *constant* u is given by

$$T(t) = T(t_0)e^{-\alpha(t-t_0)} + \frac{\beta u}{\alpha}(1 - e^{-\alpha(t-t_0)}). \tag{7.73}$$

For $u \equiv 1$, temperature measurements $\hat{T}_i$ at times t_i, $i = 0, \ldots, m$, are given in Table 7.2.

Table 7.2: Measurements for $u \equiv 1$.

i	t_i	$\hat{T}_i$	i	t_i	$\hat{T}_i$
0	0.0	1.0000	11	1.1	1.6672
1	0.1	1.0953	12	1.2	1.6988
2	0.2	1.1813	13	1.3	1.7275
3	0.3	1.2592	14	1.4	1.7534
4	0.4	1.3298	15	1.5	1.7770
5	0.5	1.3935	16	1.6	1.7982
6	0.6	1.4512	17	1.7	1.8174
7	0.7	1.5034	18	1.8	1.8348
8	0.8	1.5508	19	1.9	1.8505
9	0.9	1.5935	20	2.0	1.8647
10	1.0	1.6322			

i. Set up the least squares problem

$$\min_{\alpha,\beta,T_0 \in \mathbb{R}} \tfrac{1}{2}\|R(\alpha, \beta, T_0)\|_2^2, \tag{7.74}$$

i.e., determine R.

Note that since none of the temperature measurements above can assumed to be exact, we include $T_0 = T(t_0)$ as a variable into the least squares problem.

ii. Determine the Jacobian of R.

iii. Solve the least squares problem.

References

[Ama90] H. Amann. *Ordinary Differential Equations. An Introduction to Nonlinear Analysis*. De Gruyter Studies in Mathematics, Vol. 13. de Gruyter, Berlin, New York, 1990.

[BCP95] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics, Vol. 14. SIAM, Philadelphia, 1995.

[BIG74] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. John Wiley & Sons, New-York, Chicester, Brisbane, Toronto, 1974.

[Bjö96] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[Boc88] H. G. Bock. Randwertprobleme zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen. Preprint Nr. 442, Universität Heidelberg, Institut für Angewandte Mathematik, SFB 123, D–6900 Heidelberg, Germany, 1988.

[BW88] D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and its Applications*. John Wiley and Sons, Inc., Somerset, New Jersey, 1988.

[CM79] S. L. Campbell and C. D. Meyer. *Generalized Inverses of Linear Transformations*. Pitman, London, San Francisco, Melbourne, 1979.

[Deu04] P. Deuflhard. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*, volume 35 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2004.

[DGE81] J. E. Dennis, D. M. Gay, and R. E.Welsch. Algorithm 573 nl2sol – an adaptive nonlinear least-squares algorithm. *TOMS*, 7:369–383, 1981. Fortran code available from http://www.netlib.org/toms/573.

[DGW81] J. E. Dennis, D. M. Gay, and R. E. Welsch. An adaptive nonlinear least-squares algorithm. *TOMS*, 7:348–368, 1981.

[DH79] P. Deuflhard and G. Heindl. Affine invariant convergence theorems for Newton's method and extensions to related methods. *SIAM J. Numer. Anal.*, 16:1–10, 1979.

[DH95] P. Deuflhard and A. Hohmann. *Numerical Analysis. A First Course in Scientific Computation*. Walter De Gruyter, Berlin, New York, 1995.

[Esp81] J. H. Espenson. *Chemical Kinetics and Reaction Mechanisms*. Mc Graw Hill, New York, 1981.

[Gay83] D. M. Gay. Remark on "algorithm 573: NL2SOL—an adaptive nonlinear least-squares algorithm". *ACM Trans. Math. Softw.*, 9(1):139, 1983. doi:http://doi.acm.org/10.1145/356022.356031.

[GL89] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[Gri03] A. Griewank. A mathematical view of automatic differentiation. In A. Iserles, editor, *Acta Numerica 2003*, pages 321–398. Cambridge University Press, Cambridge, London, New York, 2003. URL: https://doi.org/10.1017/S0962492902000132, doi:10.1017/S0962492902000132.

[Gro77] C. W. Groetsch. *Generalized Inverses of Linear Operators*. Marcel Dekker, Inc., New York, Basel, 1977.

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2008. URL: https://doi.org/10.1137/1.9780898717761, doi:10.1137/1.9780898717761.

[Hei93] M. Heinkenschloss. Mesh independence for nonlinear least squares problems with norm constraints. *SIAM J. Optimization*, 3:81–117, 1993. URL: http://dx.doi.org/10.1137/0803005, doi:10.1137/0803005.

[HNW93] E. Hairer, S. O. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer Series in Computational Mathematics, Vol. 8. Springer Verlag, Berlin, Heidelberg, New York, second edition, 1993. URL: http://dx.doi.org/10.1007/978-3-540-78862-1, doi:10.1007/978-3-540-78862-1.

[KMN88] D. Kahaner, C.B. Moler, and S. Nash. *Numerical Methods and Software*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[Lev44] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quarterly Applied Mathematics*, 2:164–168, 1944.

[Mar63] D. W. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[Mar87] J. M. Martinez. An algorithm for solving sparse nonlinear least squares problems. *Computing*, 39:307–325, 1987. URL: https://doi.org/10.1007/BF02239974, doi:10.1007/BF02239974.

[Mor78] J. J. Moré. The Levenberg–Marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis, Proceedings, Biennial Conference, Dundee 1977*, pages 105–116, Berlin, Heidelberg, New-York, 1978. Springer Verlag.

[MP96] T. Maly and L. R. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics*, 20:57–79, 1996.

[Nas76] M. Z. Nashed. *Generalized Inverses and Applications*. Academic Press, Boston, San Diego, New York, London,, 1976.

[Ram97] W. F. Ramirez. *Computational Methods for Process Simulation*. Butterworth–Heinemann, Oxford, Boston, second edition, 1997.

[WA86] H. Werner and H. Arndt. *Gewöhnliche Differentialgleichungen. Eine Einführung in Theorie und Praxis*. Springer Verlag, Berlin, Heidelberg, New-York, 1986.

[Wal98] W. Walter. *Ordinary Differential Equations*. Graduate Texts in Mathematics. Springer Verlag, Berlin, Heidelberg, New York, 1998.