Part II

Inverse Model

The MOBIIR scheme reconstructs the optical parameters given the measured and predicted detector readings on the tissue boundary. It utilizes a forward model for light propagation in tissue and an inverse model for recovering the optical parameters. In Part I we discussed the forward model. The fluence was calculated based on the ERT for a given distribution of optical parameters and source positions. We found good agreement between the model prediction and the experiment. What is still missing is the formulation of the inverse problem, which is presented in this second part of the thesis.

In the following chapters, we first define an objective function, which is a function of the optical parameters. Then, we employ gradient-based optimization techniques to find the minimum of the objective function. We compare the well-established nonlinear conjugate gradient technique used extensively in OT with **quasi-Newton** methods, which were implemented in this work.

Gradient-based optimization methods employ the derivative of the objective function with respect to the optical parameters. A major obstacle in using these techniques is the computationally efficient calculation of the gradient. In Chapter 6, we apply an **adjoint differentiation technique** to obtain the gradient in a computationally efficient way.

Finally, the MOBIIR scheme is utilized for imaging of finger joints as a means of monitoring the progress of **rheumatoid arthritis**, as discussed in Chapter 9. We perform a numerical study on a finger joint model to show the ability of differentiating between healthy and early rheumatoid conditions.

Chapter 5

Image Reconstruction as Nonlinear Optimization Problem

Using MOBIIR schemes, the inverse problem in OT can be formulated as a nonlinear optimization problem, where optimization techniques minimize an objective function. The objective function is a measure of difference between the predicted and experimental data. The predicted data are determined by the forward model. The experimental data are obtained by measuring the fluence on the tissue boundary. Once the minimum of the objective function is found, the cross-sectional distribution of the unknown optical parameters is displayed as an image. The quality of these reconstructed images depends on the accuracy of the predicted fluence of the forward model, as well as on the optimization scheme.

5.1 Objective Function

The disparity between the actual fixed measurements $\boldsymbol{m}$ and the current predictions $\boldsymbol{p}$ for D source-detector pairs is mapped onto a scalar $\tilde{\varphi}$ by the objective function $\tilde{\Phi}(\boldsymbol{p})$. One of the most commonly used objective functions is the *least-square error norm*:

$$\tilde{\Phi} : \mathbb{R}^{\mathrm{D}} \to \mathbb{R}$$
$$\boldsymbol{p} \mapsto \tilde{\varphi} = \tilde{\Phi}(\boldsymbol{p}) = \frac{1}{2} \sum_{\mathrm{d}=1}^{\mathrm{D}} \left(\frac{p_{\mathrm{d}} - m_{\mathrm{d}}}{\kappa_{\mathrm{d}}} \right)^2 . \tag{5.1}$$

The predictions $\boldsymbol{p}$ are evaluated for all D source-detector pairs, using the forward model F and given the N-dimensional vector $\boldsymbol{\mu}$ of optical parameters:

$$F : \mathbb{R}^{\mathrm{N}} \to \mathbb{R}^{\mathrm{D}}$$
$$\boldsymbol{\mu} \mapsto \boldsymbol{p}(\boldsymbol{\mu}), \tag{5.2}$$

where $\mathrm{N} = 2 \times \mathrm{I} \times \mathrm{J}$. I and J denote the number of grid points of a finite-difference grid along the x-axis and y-axis, respectively. The vector $\boldsymbol{\mu}$ contains the optical parameters, specifically the scattering (μ_s) and absorption (μ_a) coefficients. The parameter κ is used for normalizing the predictions and measurements. We usually set $\kappa_{\mathrm{d}} = m_{\mathrm{d}}$.

Using Definitions 5.1 and 5.2, we get the composite function $\Phi(\boldsymbol{\mu}) = \tilde{\Phi}(\boldsymbol{p}(\boldsymbol{\mu}))$. Clearly, the objective function Φ is nonlinear because the predictions depend nonlinearly on the optical parameters. In turn, the goal of a optimization technique is to determine a vector $\boldsymbol{\mu}$ that minimizes the objective function $\Phi(\boldsymbol{\mu})$. This vector will be a solution to the minimization problem and is displayed as a two-dimensional image.

5.2 Optimization Methods

In general, optimization methods find a minimum of the defined objective function $\Phi(\boldsymbol{\mu})$. At the minimum, the predictions $\boldsymbol{p}(\boldsymbol{\mu})$ will most closely match the measurements $\boldsymbol{m}$, and $\boldsymbol{\mu}$ represents the distribution of the unknown optical parameters we set out to find.

Before we explain how to minimize the objective function, we give a brief overview of how a minimum is defined and what are the requirements that lead to a minimum.

A condition for a minimum $\boldsymbol{\mu}^*$ of the nonlinear and differentiable function $\Phi(\boldsymbol{\mu})$ is the *first-order necessary condition* $\nabla\Phi(\boldsymbol{\mu}^*) = 0$. In addition, a *second-order sufficient condition* is imposed by the second derivative $\nabla^2\Phi(\boldsymbol{\mu}^*)$ that guarantees that μ^* is a minimum. Specifically, if $\nabla^2\Phi(\boldsymbol{\mu}^*)$ is positive definite then $\boldsymbol{\mu}^*$ is a minimum of Φ. Therefore, by using the first-order necessary condition, the task of finding the minimum of $\Phi(\boldsymbol{\mu})$ is obtained by solving the equation

$$\nabla\Phi(\boldsymbol{\mu}) = 0. \tag{5.3}$$

In general, optimization methods begin with an initial guess $\boldsymbol{\mu}_0$ and generate a sequence $\{\boldsymbol{\mu}_\mathrm{k}\}$ of estimates until a solution $\boldsymbol{\mu}^*$ is reached. These iterative algorithms are classified by their robustness, efficiency and accuracy. One of the most commonly applied techniques are the gradient-based optimization methods.

Gradient-based optimization techniques have been proven to be computationally efficient for large-scale problems [Luenberger84] [Nash96] [Nocedal99] such as in OT, where between 10^3 and 10^5 unknown optical parameters are common. These optimization techniques use the gradient $\nabla\Phi(\boldsymbol{\mu})$ of the objective function for calculating a *search direction* $\boldsymbol{u}_\mathrm{k}(\nabla\Phi)$ and a *step length* α_k. An updating scheme determines a new estimate of the sequence $\{\boldsymbol{\mu}_k\}$:

$$\boldsymbol{\mu}_{\mathrm{k}+1} = \boldsymbol{\mu}_\mathrm{k} + \alpha_\mathrm{k}\boldsymbol{u}_\mathrm{k}. \tag{5.4}$$

Calculating the new estimate $\boldsymbol{\mu}_{\mathrm{k}+1}$ is broken up into two tasks:

- finding a proper step length α_k (see Subsection 5.2.1),
- calculating the search direction $\boldsymbol{u}_\mathrm{k}$ (see Subsections 5.2.2 and 5.2.4).

Therefore, gradient-based optimization algorithms share the following general form:

1. Start with an initial guess $\boldsymbol{\mu}_0$ of the solution $\boldsymbol{\mu}^*$
2. If the objective function Φ at $\boldsymbol{\mu}_\mathrm{k}$ is minimal, then stop.
3. Determine an improved estimate $\boldsymbol{\mu}_{\mathrm{k}+1} = \boldsymbol{\mu}_\mathrm{k} + \alpha_\mathrm{k}\boldsymbol{u}_\mathrm{k}$
4. go to 2.

Gradient techniques differ mainly in the way the step length α_k and the search direction $\boldsymbol{u}_\mathrm{k}$ are calculated. In OT the most common method to determine the search direction has been the nonlinear conjugate gradient (CG) technique [Arridge98] [Hielscher99]. After a brief introduction of the nonlinear CG method (see Subsection 5.2.2) we give an overview of the more complex quasi-Newton (QN) methods that were adapted as part of this thesis to provide better reconstruction results (see Subsection 5.2.4). All gradient-based optimization techniques make use of a line search for determining the step length α_k as discussed below.

5.2.1 Line Search

A line search is employed by the CG and QN methods to find a new update $\boldsymbol{\mu}_{\mathrm{k}+1} = \boldsymbol{\mu}_\mathrm{k} + \alpha_\mathrm{k}\boldsymbol{u}_\mathrm{k}$ along the search direction $\boldsymbol{u}_\mathrm{k}$. It chooses a sequence of step lengths α and accepts the one that fulfills certain conditions. A simple condition is that the line search provides a new value of the objective function with $\Phi(\boldsymbol{\mu}_\mathrm{k} + \alpha\boldsymbol{u}_\mathrm{k}) < \Phi(\boldsymbol{\mu}_\mathrm{k})$. However, this condition does not always lead to a sufficient reduction in Φ.

A sufficient decrease in the objective function Φ is given by the inequality

$$\Phi(\boldsymbol{\mu}_\mathrm{k} + \alpha\boldsymbol{u}_\mathrm{k}) \leq \Phi(\boldsymbol{\mu}_\mathrm{k}) + c_1\alpha\nabla\Phi(\boldsymbol{\mu}_\mathrm{k})^T\boldsymbol{u}_\mathrm{k} \tag{5.5}$$

for some constant $c_1 \in (0,1)$, which is also called the *Armijo condition* or *sufficient decrease condition*. Yet, this condition alone does not ensure that the algorithm progresses reasonably. Therefore, a second condition is employed, called the *curvature condition*, which requires α to satisfy

$$\nabla\Phi(\boldsymbol{\mu}_\mathrm{k} + \alpha\boldsymbol{u}_\mathrm{k})^T\boldsymbol{u}_\mathrm{k} \geq c_2\nabla\Phi(\boldsymbol{\mu}_\mathrm{k})^T\boldsymbol{u}_\mathrm{k} \tag{5.6}$$

for some constant $c_2 \in (c_1, 1)$. Both conditions, Equations 5.5 and 5.6, are known as the *Wolfe conditions*. Typical values of c_2 are 0.9 when the search direction is chosen by a QN method, and 0.1 when $\boldsymbol{u}_\mathrm{k}$ is chosen by the nonlinear CG technique.

Furthermore, line searches are distinguished between exact and inexact searches depending on the method employed for calculating the search direction $\boldsymbol{u}_\mathrm{k}$. An exact line search performs an one-dimensional *line-minimization* $\Phi(\alpha) = \Phi(\boldsymbol{\mu}_\mathrm{k} + \alpha\boldsymbol{u}_\mathrm{k})$ along the direction $\boldsymbol{u}_\mathrm{k}$, finding the appropriate step length α. This method is mostly used by the CG method. Here, the line search is accomplished in two steps. The first step is called *bracketing* [Press92]. It brackets the minimum by finding three points $\boldsymbol{\mu}_a < \boldsymbol{\mu}_b < \boldsymbol{\mu}_c$ along the search direction $\boldsymbol{u}_\mathrm{k}$ such that $\Phi(\boldsymbol{\mu}_a) > \Phi(\boldsymbol{\mu}_b)$ and $\Phi(\boldsymbol{\mu}_c) > \Phi(\boldsymbol{\mu}_b)$. Thus, the minimum is somewhere within the interval $[\boldsymbol{\mu}_a, \boldsymbol{\mu}_c]$. This bracketing can be accomplished by an efficient algorithm such as the *golden section rule* [Press92]. The second step locates the minimum itself, in the interval $[\boldsymbol{\mu}_a, \boldsymbol{\mu}_c]$. In this work we typically perform a parabolic interpolation, which entails fitting a quadratic polynomial to the objective function at the three points $\boldsymbol{\mu}_a$, $\boldsymbol{\mu}_b$, and $\boldsymbol{\mu}_c$. The minimum of this parabola is found. This process is repeated by fitting a new parabola at the new point and the previous two points with the smallest objective function. A commonly used method for performing this task is *Brent's* algorithm [Press92]. At the minimum we obtain the step length α_k.

An inexact line search along $\boldsymbol{u}_\mathrm{k}$ does not require that $\Phi(\alpha) = \Phi(\boldsymbol{\mu}_\mathrm{k} + \alpha\boldsymbol{u}_\mathrm{k})$ is

minimal. In this case the second condition, Equation 5.6, is not applied and we employ a *backtracking* approach [Press92]. This technique is used by the QN methods, which do not require exact line searches. The backtracking algorithm chooses its candidate step length α_k only until the sufficient decrease condition, Equation 5.5, is satisfied. We find the best results for a $c_1 = 10^{-8}$.

5.2.2 Nonlinear Conjugate Gradient Method

The nonlinear CG method generates a sequence $\{\boldsymbol{\mu}_k\}$ converging to the solution $\boldsymbol{\mu}^*$ of Equation 5.3. This method is derived from the linear CG method for minimizing convex quadratic functions $h(\boldsymbol{x})$ with a symmetric and positive definite matrix $\boldsymbol{A}$:

$$h(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}. \tag{5.7}$$

Using the first-order necessary condition ($\nabla h = 0$, see Equation 5.3), the linear CG method can also be considered as a technique for solving a linear system of equations:

$$\nabla h(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = 0. \tag{5.8}$$

The two problems, minimizing a convex quadratic function and solving a linear system of equations, are equivalent.

The linear CG method solves Equation 5.8 by using a set of search directions $\{\boldsymbol{a}_k\}$ and the one-dimensional minimizer α_k of $h(\boldsymbol{x})$ along $\boldsymbol{a}_k$ for determining $\boldsymbol{x}_{k+1}$ from the previous $\boldsymbol{x}_k$. The updating scheme (see Equation 5.4) becomes:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{a}_k. \tag{5.9}$$

The directions $\boldsymbol{a}_k$ are conjugate with respect to the matrix $\boldsymbol{A}$ by using the definition

$$\boldsymbol{a}_k^T\boldsymbol{A}\boldsymbol{a}_{k'} = 0 \text{ for } k \neq k' \tag{5.10}$$

for conjugate directions [Nash96] [Nocedal99]. Consequently, we find a new conjugate direction

$$\boldsymbol{a}_{\mathrm{k}} = -\boldsymbol{r}_{\mathrm{k}} + \beta_{\mathrm{k}} \boldsymbol{a}_{\mathrm{k}-1} \tag{5.11}$$

that uses the residual $\boldsymbol{r}_{\mathrm{k}}$

$$\boldsymbol{r}_{\mathrm{k}} = \boldsymbol{A}\boldsymbol{x}_{\mathrm{k}} - \boldsymbol{b} \tag{5.12}$$

and the scalar β_{k}

$$\beta_{\mathrm{k}} = \frac{\boldsymbol{r}_{\mathrm{k}}^T \boldsymbol{r}_{\mathrm{k}}}{\boldsymbol{r}_{\mathrm{k}-1}^T \boldsymbol{r}_{\mathrm{k}-1}}. \tag{5.13}$$

The one-dimensional minimizer α_{k} of the quadratic function $h(\boldsymbol{x})$ along $\boldsymbol{a}_{\mathrm{k}}$ is determined by

$$\alpha_{\mathrm{k}} = \frac{\boldsymbol{r}_{\mathrm{k}}^T \boldsymbol{r}_{\mathrm{k}}}{\boldsymbol{a}_{\mathrm{k}}^T \boldsymbol{A} \boldsymbol{a}_{\mathrm{k}}}. \tag{5.14}$$

However, we cannot use the linear CG method in OT, because this method only minimizes quadratic functions $h(\boldsymbol{x})$ well. For general nonlinear functions, such as the objective function, we have to use a nonlinear CG method. The nonlinear CG method differs in the way how the step length α_{k} and the residual $\boldsymbol{r}_{\mathrm{k}}$ are determined.

Instead of calculating α_{k} explicitly by using Equation 5.14, we replace its computation by an exact line search that minimizes $\Phi(\boldsymbol{\mu}_{\mathrm{k}} + \alpha \boldsymbol{u}_{\mathrm{k}})$. An appropriate line search, such as line-minimization, was explained in Subsection 5.2.1. Furthermore, the residual $\boldsymbol{r}_{\mathrm{k}}$ for the determination of the scalar β in Equation 5.13 is replaced by the gradient $\nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}})$ of the objective function:

$$\beta_{\mathrm{k}} = \frac{\nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}})^T \nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}})}{\nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}-1})^T \nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}-1})}. \tag{5.15}$$

Finally, the new conjugate direction is computed by modifying Equation 5.11

$$\boldsymbol{u}_{\mathrm{k}} = -\nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}}) + \beta_{\mathrm{k}} \boldsymbol{u}_{\mathrm{k}-1}. \tag{5.16}$$

There are several versions of the nonlinear CG method that differ in the way one can calculate β_k (see Equation 5.15). In addition to the *Fletcher Reeves* formula as presented in Equation 5.15, the best known formulas with $\boldsymbol{y}_{k-1} = \nabla\Phi(\boldsymbol{\mu}_k) - \nabla\Phi(\boldsymbol{\mu}_{k-1})$ are the *Polak-Ribiere* formula

$$\beta_k = \frac{\boldsymbol{y}_{k-1}^T \nabla\Phi(\boldsymbol{\mu}_k)}{\nabla\Phi(\boldsymbol{\mu}_{k-1})^T \nabla\Phi(\boldsymbol{\mu}_{k-1})}, \tag{5.17}$$

and the *Hestenes-Stiefel* formula

$$\beta_k = \frac{\boldsymbol{y}_{k-1}^T \nabla\Phi(\boldsymbol{\mu}_k)}{\boldsymbol{y}_{k-1}^T \boldsymbol{u}_{k-1}}. \tag{5.18}$$

When applied to a mere quadratic function these versions are equivalent, but they behave differently on general nonlinear functions, such as the objective function in OT. We use the Polak-Ribiere formula for our work.

The algorithm for the nonlinear CG scheme can be formulated in general terms as follows:

ALGORITHM

`set convergence tolerance` $\epsilon > 0$

`initialize starting point` $\boldsymbol{\mu}_0$

`evaluate` $\Phi(\boldsymbol{\mu}_0)$ `and` $\nabla\Phi(\boldsymbol{\mu}_0)$

`set` $\boldsymbol{u}_0 = -\nabla\Phi(\boldsymbol{\mu}_0)$

$k = 0$

while: $\|\nabla\Phi(\boldsymbol{\mu}_k)\| > \epsilon$

`determine` α_k `performing a line-minimization of` $\Phi(\boldsymbol{\mu}_k + \alpha\boldsymbol{u}_k)$

`calculate a new estimate` $\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \alpha_k\boldsymbol{u}_k$

`calculate a new gradient` $\nabla\Phi(\boldsymbol{\mu}_{k+1})$

`determine` β_{k+1} `using Equation 5.17`

`calculate new search direction` $\boldsymbol{u}_{k+1}$ `using Equation 5.16`

`set` $k = k + 1$

end(while)

5.2.3 Newton's Method for Nonlinear Equations

To date mostly nonlinear CG methods have been applied to the image reconstruction problem in OT. However, nonlinear CG methods often suffer from slow convergence rates and require a relatively large number of function evaluations during the exact line search. These problems are compounded when the ERT is used as a forward model, because solving this equation is already a computationally intensive task.

Looking to overcome these problems, we focused on Newton-type optimization methods. These methods solve a sequence of linear equations, instead of solving a nonlinear equation (Equation 5.3) directly. Hence, the objective function $\Phi(\boldsymbol{\mu})$ is replaced by a quadratic model $Q(\boldsymbol{u}_k)$ at $\boldsymbol{\mu}_k$:

$$\Phi(\boldsymbol{\mu}_k + \boldsymbol{u}_k) \sim Q(\boldsymbol{u}_k) = \Phi(\boldsymbol{\mu}_k) + \boldsymbol{u}_k^T \nabla\Phi(\boldsymbol{\mu}_k) + \frac{1}{2}\boldsymbol{u}_k^T \nabla^2\Phi(\boldsymbol{\mu}_k)\boldsymbol{u}_k, \tag{5.19}$$

where $\nabla^2\Phi$ is called the *Hessian* matrix. By simply setting the derivative of $Q(\boldsymbol{u}_k)$ with respect to $\boldsymbol{u}_k$ to zero, we get a linear system of equations, called *Newton equations*, for the unknown variable $\boldsymbol{u}_k$:

$$\nabla^2\Phi(\boldsymbol{\mu}_k) \cdot \boldsymbol{u}_k = -\nabla\Phi(\boldsymbol{\mu}_k). \tag{5.20}$$

After computing $\boldsymbol{u}_k$, a new estimate of the optical parameters,

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \boldsymbol{u}_k, \tag{5.21}$$

is found.

In general, Newton's method is rarely used in its *classical* form for nonlinear problems. It can fail if the quadratic model $Q(\boldsymbol{u}_k)$ is not a good approximation to the nonlinear objective function $\Phi(\boldsymbol{\mu}_k + \boldsymbol{u}_k)$ at $\boldsymbol{\mu}_k$. Another problem we have to face is guaranteeing the positive-definiteness of the Hessian $\nabla^2\Phi$. When $\nabla^2\Phi$ is not positive definite, the Newton update $\boldsymbol{u}_k$ may not be defined, since the inverse $(\nabla^2\Phi)^{-1}$ may not exist. Furthermore, even when it is defined, it may not satisfy the descent property [Nocedal99]:

$$\boldsymbol{u}_k^T \cdot \nabla\Phi(\boldsymbol{\mu}_k) < 0. \tag{5.22}$$

Consequently, refinements to Newton's method have been proposed, which are called quasi-Newton (QN) methods. They differ in their computational speed and stability [Nocedal99]. Generally, all QN methods replace Equation 5.20 with various alternatives for calculating the search direction $\boldsymbol{u}_k$, which have the general form

$$\boldsymbol{\mathcal{B}}_k \cdot \boldsymbol{u}_k = -\nabla\Phi(\boldsymbol{\mu}_k), \tag{5.23}$$

where $\boldsymbol{\mathcal{B}}_k$ is a positive-definite matrix. Intuitively, $\boldsymbol{\mathcal{B}}_k$ should be some approximation to $\nabla\Phi^2(\boldsymbol{\mu}_k)$. The search direction $\boldsymbol{u}_k$ is required to be a descent direction (see Equation 5.22). Furthermore, Equation 5.21 is replaced by the updating formula given by Equation 5.4. In summary, QN methods require the first derivative $\nabla\Phi(\boldsymbol{\mu}_k)$, an approximation $\boldsymbol{\mathcal{B}}_k$ to the Hessian $\nabla^2\Phi(\boldsymbol{\mu}_k)$, the solution $\boldsymbol{u}_k$ of a linear system (see Equation 5.23), and the storage of the approximated Hessian $\boldsymbol{\mathcal{B}}_k$.

5.2.4 Quasi-Newton Method

In this subsection we derive two different QN methods from Newton's method for calculating the approximated Hessian $\boldsymbol{\mathcal{B}}_k$ and search direction $\boldsymbol{u}_k$. All QN methods employ generalizations of the *secant method* for computing the matrix $\boldsymbol{\mathcal{B}}_k$ [Nash96] with

$$\nabla^2\Phi(\boldsymbol{\mu}_{k+1}) \cdot (\boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k) \sim \nabla\Phi(\boldsymbol{\mu}_{k+1}) - \nabla\Phi(\boldsymbol{\mu}_k). \tag{5.24}$$

Replacing the Hessian with an approximated Hessian $\boldsymbol{\mathcal{B}}_{k+1}$ yields

$$\boldsymbol{\mathcal{B}}_{k+1} \cdot (\boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k) = \nabla\Phi(\boldsymbol{\mu}_{k+1}) - \nabla\Phi(\boldsymbol{\mu}_k). \tag{5.25}$$

Two additional vectors are defined that will be used repeatedly:

$$\begin{aligned} \boldsymbol{s}_k &= \boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k \\ \boldsymbol{y}_k &= \nabla\Phi(\boldsymbol{\mu}_{k+1}) - \nabla\Phi(\boldsymbol{\mu}_k). \end{aligned} \tag{5.26}$$

Substituting Equations 5.26 into Equation 5.25 gives the *secant condition* [Nocedal99] or *quasi-Newton condition* [Bishop97]

$$\boldsymbol{\mathcal{B}}_{k+1} \cdot \boldsymbol{s}_k = \boldsymbol{y}_k, \tag{5.27}$$

which has to be satisfied by the approximated Hessian $\boldsymbol{\mathcal{B}}_{k+1}$. Furthermore, the matrix $\boldsymbol{\mathcal{B}}_{k+1}$ is positive definite only if $\boldsymbol{s}_k$ and $\boldsymbol{y}_k$ satisfy the *curvature condition* [Nocedal99]

$$\boldsymbol{s}_k^T \cdot \boldsymbol{y}_k > 0. \tag{5.28}$$

A technique for calculating the matrix $\boldsymbol{\mathcal{B}}_{k+1}$ that satisfies the secant condition is given by the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) formula [Nash96]:

$$\boldsymbol{\mathcal{B}}_{k+1} = \boldsymbol{\mathcal{B}}_k - \frac{(\boldsymbol{\mathcal{B}}_k \boldsymbol{s}_k)(\boldsymbol{\mathcal{B}}_k \boldsymbol{s}_k)^T}{\boldsymbol{s}_k^T \boldsymbol{\mathcal{B}}_k \boldsymbol{s}_k} + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}. \tag{5.29}$$

After each iteration a new update of the matrix $\boldsymbol{\mathcal{B}}_{k+1}$, which is a function of $\boldsymbol{\mathcal{B}}_k$, can be computed by starting with $\boldsymbol{\mathcal{B}}_0 = \boldsymbol{I}$.

Instead of calculating $\boldsymbol{\mathcal{B}}_{k+1}$, and then evaluating its inverse, the QN method calculates directly the inverse $\boldsymbol{\mathcal{H}}_{k+1}$ of $\boldsymbol{\mathcal{B}}_{k+1}$. Given $\boldsymbol{\mathcal{H}}_{k+1}$, it will be even easier to compute the new search direction $\boldsymbol{u}_{k+1}$ using the Newton equation (5.23)

$$\boldsymbol{u}_{k+1} = -\boldsymbol{\mathcal{H}}_{k+1} \nabla\Phi(\boldsymbol{\mu}_{k+1}) \tag{5.30}$$

since one will not have to solve a system of linear equations $\boldsymbol{\mathcal{B}}_{k+1} \cdot \boldsymbol{u}_{k+1} = -\nabla\Phi(\boldsymbol{\mu}_{k+1})$. The inverse Hessian is derived in an iterative manner by starting with $\boldsymbol{\mathcal{H}}_0 = \boldsymbol{I}$[Nash96] and applying the following formula:

$$\boldsymbol{\mathcal{H}}_{k+1} = \boldsymbol{\mathcal{H}}_k + \frac{(\boldsymbol{y}_k - \boldsymbol{\mathcal{H}}_k \boldsymbol{s}_k)\boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{(\boldsymbol{y}_k - \boldsymbol{\mathcal{H}}_k \boldsymbol{s}_k)^T \boldsymbol{s}_k}{(\boldsymbol{y}_k^T \boldsymbol{s}_k)^2}(\boldsymbol{y}_k \boldsymbol{y}_k^T). \tag{5.31}$$

The BFGS method requires the storage of the inverse Hessian $\boldsymbol{\mathcal{H}}_k$ for use by the next iteration step. This storage space can be quite large and leads to a high computational burden [1]. Therefore, the *limited-memory BFGS* (lm-BFGS) method was devised that required no storage of the inverse Hessian $\boldsymbol{\mathcal{H}}_k$ [Nocedal80] [Liu89]. Consequently, the lm-BFGS method is very suitable for large-scale problems [Nash96]. It is based on the inverse formula of the BFGS method, the matrix $\boldsymbol{\mathcal{H}}_k$ is replaced by the identity matrix $\boldsymbol{I}$ at each iteration step. We make this substitution in Equation 5.31, and obtain the following expression for the search direction $\boldsymbol{u}_k$ by using Equation 5.30 and the vectors $\boldsymbol{s}_k$ and $\boldsymbol{y}_k$ [Bishop97]:

$$\boldsymbol{u}_k = -\nabla\Phi(\boldsymbol{\mu}_k) + \gamma \boldsymbol{s}_k + \lambda \boldsymbol{y}_k. \tag{5.32}$$

The scalars γ and λ are defined by

$$\gamma = -\left(1 + \frac{\boldsymbol{y}_k^T \boldsymbol{y}_k}{\boldsymbol{s}_k^T \boldsymbol{y}_k}\right) \frac{\boldsymbol{s}_k^T \nabla\Phi(\boldsymbol{\mu}_{k+1})}{\boldsymbol{s}_k^T \boldsymbol{y}_k} + \frac{\boldsymbol{y}_k^T \nabla\Phi(\boldsymbol{\mu}_{k+1})}{\boldsymbol{s}_k^T \boldsymbol{y}_k}, \tag{5.33}$$

$$\lambda = \frac{\boldsymbol{s}_k^T \nabla\Phi(\boldsymbol{\mu}_{k+1})}{\boldsymbol{s}_k^T \boldsymbol{y}_k}.$$

[1] An example with $N^2 = (2 \times I \times J)^2 = (2 \times 60 \times 60)^2 = 5.184 \cdot 10^7$ elements requires a total storage space of $4.05 \cdot 10^5$ KByte $\approx$ 396 MByte with 8 Byte each element.

An algorithm based on the presented approach has the general form:

ALGORITHM

set convergence tolerance $\epsilon > 0$

initialize starting point $\boldsymbol{\mu}_0$

calculate gradient $\nabla\Phi(\boldsymbol{\mu}_0)$

initialize inverse approximated Hessian $\boldsymbol{\mathcal{H}}_0 = \boldsymbol{I}$ with identity matrix

while: $\|\nabla\Phi(\boldsymbol{\mu}_\mathrm{k})\| > \epsilon$

BFGS: calculate search direction $\boldsymbol{u}_\mathrm{k}$ by using Equation 5.30

lm-BFGS: calculate search direction $\boldsymbol{u}_\mathrm{k}$ by using Equation 5.32

determine α_k by performing a line search along $\boldsymbol{u}_\mathrm{k}$

calculate a new estimate $\boldsymbol{\mu}_{\mathrm{k}+1} = \boldsymbol{\mu}_\mathrm{k} + \alpha_\mathrm{k} \cdot \boldsymbol{u}_\mathrm{k}$

calculate a new gradient $\nabla\Phi(\boldsymbol{\mu}_{\mathrm{k}+1})$

define $\boldsymbol{s}_\mathrm{k}$ and $\boldsymbol{y}_\mathrm{k}$ by means of Equation 5.26

BFGS: calculate $\boldsymbol{\mathcal{H}}_{\mathrm{k}+1}$ by using Equation 5.31

lm-BFGS: calculate γ and λ by using Equation 5.33

$\mathrm{k} = \mathrm{k} + 1$

end(while)

5.2.5 Positive Definite Inverse Hessian

As already mentioned on page 78, the approximated Hessian $\boldsymbol{\mathcal{B}}_\mathrm{k}$ may not always be positive definite. In this case the quadratic model $Q(\boldsymbol{u})$ (see Equation 5.19) is a poor approximation to the objective function Φ, and the curvature condition (see Equation 5.28) is not satisfied. This can occur for example if the initial guess $\boldsymbol{\mu}_0$ is far from the solution $\boldsymbol{\mu}^*$. In the case of the BFGS method one commonly replaces the approximated inverse Hessian

$\mathcal{H}_k$ in Equation 5.30 by the identity matrix $\boldsymbol{I}$, which is positive definite. In the case of the lm-BFGS method, the scalars γ and λ are set to zero. After replacement, the search direction becomes $\boldsymbol{u}_k = -\nabla\Phi(\boldsymbol{\mu}_k)$, which is just the steepest descent direction.

5.3 Discussion

We have reviewed various gradient-based optimization methods for minimizing the objective function $\Phi(\boldsymbol{\mu})$. Typically, an initial guess $\boldsymbol{\mu}_0$ is iteratively updated along successive search directions $\boldsymbol{u}_k$. The nonlinear CG method, which utilizes only the first derivative $\nabla\Phi(\boldsymbol{\mu}_k)$ of the objective function for determining these search directions, often leads to a slow convergence towards the minimum. The BFGS method makes additional use of an approximation $\mathcal{H}_k$ to the inverse of the second derivative for computing the required search directions. The inverse Hessian is iteratively built up by using the first derivatives $\nabla\Phi(\boldsymbol{\mu}_k)$. Hence, it is expected that the BFGS method will find the minimum with less function evaluations than the CG method. On the other hand, a drawback of the BFGS method is the large memory requirement for the storage of $\mathcal{H}_k$. This obstacle can partly be overcome by employing the lm-BFGS method. The lm-BFGS method replaces the inverse Hessian by the identity matrix $\boldsymbol{I}$. It might, however, have a slightly slower convergence than the BFGS method. Overall we will see that the BFGS and lm-BFGS methods outperform the CG method despite their partly strong computational requirements. This will be subject of Chapter 7.