UNIVERSITY OF CALIFORNIA
SANTA CRUZ

Estimation of Distributed Parameters by Multiresolution Optimization

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER AND INFORMATION SCIENCES

by

Koji Amakawa

December 1994

The dissertation of Koji Amakawa is approved:

Alex T. Pang

Suresh K. Lodha

Manfred K. Warmuth

Dean of Graduate Studies and Research

Contents

Abstract **ix**

Acknowledgments **xi**

1. Introduction **1**

1.1 Parameter Estimation by Numerical Optimization 2

1.2 Local Search Methods 4

1.3 Global Search Methods 4

1.4 Proposed Method: Multiresolution Local Search 6

1.5 Contributions of This Dissertation 9

2. Globalizing a Local Search by Multiresolution **10**

2.1 General Framework 10

2.1.1 Step Scheme 11

2.1.2 Weight Scheme 13

2.1.3 Estimating Transform Coefficients vs. Filtering the Gradient . 14

2.2 The Conjugate Gradient Method 17

2.2.1 When Parameters are Real 17

2.2.2 When Parameters are Complex 18

2.3 The Discrete Fourier Transform 22

2.3.1 For One-dimensional Field 22

2.3.2 For n-dimensional Field 29

2.3.3 Filtering the Gradient 30

2.4 The Haar Wavelet Transform 39

2.4.1 For One-dimensional Field 39

2.4.2 For n-dimensional Field 46

2.4.3 Filtering the Gradient 53

3. Calculation of the Gradient **63**

3.1 The Back-Propagation Algorithm 63

3.2 When the Parameter is Constant 67

3.3 For Converging Forward Calculation 69

3.4 The Gradient for Fourier Coefficients 70

3.5 The Gradient for Haar Wavelet Coefficients 72

4. Application: Electrical Impedance Tomography **74**

4.1 Model 74

4.2 Forward Solution 76

4.3 Obtaining the Gradient 80

4.4 Implementation 86

4.5 Experiments 87

4.5.1 Experimental Conditions 87

4.5.2 Conventional Single-resolution Method 91

4.5.3 Step Scheme with the Fourier Transform 95

4.5.4 Weight Scheme with the Fourier Transform 100

4.5.5 Step Scheme with the Haar Wavelet Transform 104

4.5.6 Weight Scheme with the Haar Wavelet Transform 108

4.5.7 Comparison among Best Results 112

5. Discussion **116**

5.1 Multiresolution vs. Single-resolution 116

5.2 Causes of Difficulties 117

5.3 Fourier Transform vs. Wavelet Transform 118

5.4 Conditions for Multiresolution Optimization to Work 120

6. Conclusions **122**

6.1 Future Work 123

References **124**

List of Figures

1.1 General estimation algorithm 3

2.1 Step scheme of multiresolution estimation 11

2.2 Weight scheme of multiresolution estimation 14

2.3 Step scheme using filtered gradient 16

2.4 Weight scheme using filtered gradient 16

2.5 Multiresolution estimation with Fourier transform 28

2.6 The Haar basis 39

2.7 The Haar wavelet 39

2.8 One-dimensional Haar wavelet transform 44

2.9 One-dimensional inverse Haar wavelet transform 44

2.10 Multiresolution estimation with Haar wavelet transform 45

2.11 n-dimensional Haar wavelet transform 48

2.12 n-dimensional inverse Haar wavelet transform 48

3.1 Layers of forward calculation 64

4.1 Node numbering 88

4.2 Current directions 88

4.3 Injection current patterns 88

4.4 Pattern A 90

4.5 Pattern B 90

4.6 History of single-resolution estimation 92

4.7 Single-resolution method (Pattern A) 93

4.8 Single-resolution method (Pattern B) 94

4.9 History of Fourier step scheme 96

4.10 Fourier step scheme (Pattern A, advance=0.1) 97

4.11 Fourier step scheme (Pattern B, advance=0.1) 98

4.12 History of Fourier weight scheme 101

4.13 Fourier weight scheme (Pattern A, weight=freq^{-2}) 102

4.14 Fourier weight scheme (Pattern B, weight=freq^{-2}) 103

4.15 History of Haar step scheme 105

4.16 Haar step scheme (Pattern A, advance=0.01) 106

4.17 Haar step scheme (Pattern B, advance=0.1) 107

4.18 History of Haar weight scheme 109

4.19 Haar weight scheme (Pattern A, weight=res^{-2}) 110

4.20 Haar weight scheme (Pattern B, weight=res^{-2}) 111

4.21 Comparison of history curves 113

4.22 Comparison for Pattern A 114

4.23 Comparison for Pattern B 115

5.1 Search paths 116

5.2 Wavelet transform vs. Fourier transform 118

5.3 Multiresolution steps and frequency 119

List of Tables

2.1 Fourier coefficients and frequency 25

Estimation of Distributed Parameters by Multiresolution Optimization

Koji Amakawa

ABSTRACT

This dissertation proposes, develops and evaluates multiresolution optimization methods for estimation of distributed parameters of mathematical models. The methods are based on the assumption that the distributed parameters are continuous almost everywhere in the defined field.

The main idea in employing multiresolution optimization is to give priority to large-scale characteristics of the parameter distribution over smaller-scale ones in the estimation process. This allows the overall structure of the distribution to be found more quickly, which results in rapid approach of the estimation to the true distribution. This in turn allows more reliable search for the details.

The multiresolution optimization method consists of a local search method and a multiresolution scheme that controls the resolution of the search. This dissertation employs the conjugate gradient method as a local search method, and the discrete Fourier transform and the Haar wavelet transform as a multiresolution scheme. Since the coefficients of these transforms are inherently sorted in frequency or scale, multiresolution estimation can be realized by estimating the transform coefficients with some controlled weights and inversely transforming the coefficients back to the parameter distribution. Two methods of controlling the resolution are devised. One is the "step scheme" in which the threshold frequency for a low-pass filter steps up every time the search converges. The other method is the "weight scheme" in which fixed weights are assigned so that the coefficients of low frequencies get larger weights than the coefficients of higher frequencies.

This dissertation gives a proof that estimating the transform coefficients in either scheme is equivalent to directly estimating the desired parameter distribution by using the gradient that is filtered by the transform. In other words, we can realize a multiresolution optimization by simply filtering the gradient in a multiresolution manner to control *the resolution of the search direction.*

The developed methods are evaluated in simulation of the so-called electrical impedance tomography, which is one of many potential applications. It is shown that the multiresolution optimization yields better estimates more rapidly than the conventional single-resolution method.

Keywords: multiresolution optimization, parameter estimation, distributed parameter, Fourier transform, Haar wavelet transform, filtering gradient, electrical impedance tomography

Acknowledgments

I am very glad to be able to successfully finish my student life with this dissertation. Looking back, I feel it has been a long and winding road, and the very fact that I have walked such a road gives me confidence in myself along with the degree as a researcher's license. In the following, I would like to thank people who helped me accomplish this goal.

My advisor Prof. Alex Pang has been patiently watching me proceed in this research. The idea of multiresolution search originated in his suggestions. I am deeply grateful for his advice and patience.

I encountered wavelets through Prof. Suresh Lodha's powerful lectures. His speedy talk was a great stimulus that helped me decide to explore the wavelets. I would like to thank him for the intellectual stimulation.

A large part of this dissertation is related to my interest in machine learning. My thanks are due to Prof. Manfred Warmuth and Prof. David Haussler for their direct and indirect help to my understanding of the field.

I would like to thank Kimmen Sjölander for helping me improve my English writing.

I am also grateful to the staff of the Computer and Information Sciences board for their everyday support of the student life and the computer systems.

Finally, I would like to thank my wife Minako and our parents Tetsuya & Sachiko Amakawa and Ken & Haruko Takimoto. Minako has been enjoying, or rather enduring, the life as a wife of a poor graduate student for all these years. I am really grateful for her having walked all the way with me. Our parents helped and supported my student life, both mentally and financially, for which I am deeply indebted.

1. Introduction

Numerical simulation by the computer has become a powerful tool for investigating various phenomena that can be described by mathematical models. In the ordinary simulation, the user first sets the values of the model parameters and then calculates the corresponding results on a mathematical model. The direction of the information flow is from *cause* to *effect* in this case, because the parameter values are the cause of the simulated phenomenon which is the effect. The reverse process that seeks for a probable cause of a given effect on a given mathematical model is the parameter estimation problem in which the unknown model parameter values are to be estimated from a given phenomenon.

Such estimation problems, especially for distributed parameters, are found in various areas. For instance, one may want to get information about the internal mechanism of a physical system by measuring its externally observable states because it is difficult or impossible to directly measure the interior. In this case, the distributed parameter that represents the interior of the system has to be estimated from certain quantities measured on the exterior. Some examples of this type are image reconstruction problems in ultrasound, light scattering, impedance, diffuse tomography and biomagnetism [3, 12, 24, 35]. In another case, one may want to determine the aspects of a system, which are in themselves difficult to measure directly, by inferring these aspects from other more accessible quantities which may actually be in the same physical position. An example of this type is estimation of the diffusion coefficient distribution of a heart muscle tissue from its electrical activities [1].

This dissertation deals with such estimation problems for parameters distributed over a multi-dimensional field. Assumed as a basic property of the problems is that the parameter is continuous almost everywhere in the field. In other words, a parameter element is expected to have a value close to its neighbors' in most of the regions in the field.

1.1 Parameter Estimation by Numerical Optimization

Numerical optimization techniques are often employed to solve parameter estimation problems, especially when the mathematical model of the phenomenon is nonlinear [34, 37]. The common framework of these techniques is to optimize the unknown parameter distribution so that the simulated results of the phenomenon approach the given data.

Calculating the resulting state of the model variables based on given parameter values is called the *direct problem* or the *forward problem.* The direct problem can be expressed as:

$$\boldsymbol{V} := G(\boldsymbol{r}, \boldsymbol{V}_{\text{o}}) \tag{1.1}$$

$G()$	:	function based on the mathematical model
$\boldsymbol{r}$	:	model parameter(s)
$\boldsymbol{V}$	:	final state of variables
$\boldsymbol{V}_{\text{o}}$	:	initial state of variables

If the final state of variables does not depend on their initial state, we can write:

$$\boldsymbol{V} := G(\boldsymbol{r}) \tag{1.2}$$

The parameter $\boldsymbol{r}$ of the problems studied in this dissertation is distributed over a multi-dimensional field, continuous almost everywhere, and may be constant or time-varying. For simplicity, we refer to $\boldsymbol{r}$ as if a single distributed parameter is being estimated; in fact, more than one is possible.

Estimating the distributed parameter $\boldsymbol{r}$ from the given final state $\boldsymbol{V}$ of the variables, and possibly the given initial state $\boldsymbol{V}_\circ$, is an *inverse problem.* When the mathematical model is nonlinear, solving (1.1) with respect to $\boldsymbol{r}$ is often impossible. With a numerical optimization technique, however, we can estimate the parameter $\boldsymbol{r}$ by optimizing it so that the variable $\boldsymbol{V}$ calculated by (1.1) becomes close to a given $\boldsymbol{V}$. Let $\boldsymbol{V}$ denote the calculated, and $\widehat{\boldsymbol{V}}$ denote the given target. Then, we estimate $\boldsymbol{r}$ by minimizing the *cost* $H(\boldsymbol{V}, \widehat{\boldsymbol{V}})$ that represents the error between $\boldsymbol{V}$ and $\widehat{\boldsymbol{V}}$.

```
1  Input target state $\widehat{\boldsymbol{V}}$ (and possibly initial state $\boldsymbol{V}_\circ$)
2  Set initial guess of $\boldsymbol{r}$
3  Repeat until cost is minimized:
   3.1  $\boldsymbol{V} := G(\boldsymbol{r}, \boldsymbol{V}_\circ)$
   3.2  $cost := H(\boldsymbol{V}, \widehat{\boldsymbol{V}})$
   3.3  Adjust $\boldsymbol{r}$
4  Output $\boldsymbol{r}$
```

Figure 1.1: General estimation algorithm

Figure 1.1 shows the general estimation algorithm that uses optimization. In the repeat loop **3**, line **3.1** solves the direct problem based on the present $\boldsymbol{r}$, line **3.2** calculates the cost, and line **3.3** adjusts $\boldsymbol{r}$ so that eventually, after some number of iterations, a (local) minimum of the cost is reached. At this point, estimation of the distributed parameter $\boldsymbol{r}$ terminates.

The performance of the algorithm in terms of computation time and estimation accuracy is dependent mostly on line **3.3**. There are various methods to "adjust" $\boldsymbol{r}$. These can be classified into two primary categories: *local search methods* that look for a better $\boldsymbol{r}$ in the vicinity of the present $\boldsymbol{r}$, and *global search methods* that explore much wider regions than the vicinity of the present $\boldsymbol{r}$.

1.2 Local Search Methods

Among various local optimization techniques, efficient and therefore often used are the *conjugate gradient* method and the *quasi-Newton* family [2, 25, 30, 37]. The conjugate gradient method may be more suitable if the distributed parameter to be estimated has a large number of elements, because it requires less memory than the quasi-Newton family and yet its convergence speed is comparable as follows. With N unknown elements of the parameter, the conjugate gradient method needs memory of size $O(N)$, while the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, which is a typical quasi-Newton method, requires $O(N^2)$. This difference becomes quite serious as N gets larger. However, both methods reach a minimum in $O(N)$ time when the cost is quadratic with respect to the unknowns. In many cases, the cost function is more complicated, but the quadratic approximation which is the basis of these methods is still useful because a smooth function becomes almost quadratic near a minimum.

The limitation of these methods is that they can perform only a local search, not a global one. Since they determine the next point of the search to lower the cost at every iteration, the search path is always downhill on the cost surface. Therefore, if there exist multiple valleys of the cost, these methods can find only a local minimum (the bottom of the current valley), not the global minimum (the lowest point of all the valleys), unless the user is lucky enough to happen to set the initial guess in the right valley.

1.3 Global Search Methods

The limitation of local search methods comes from the fact that the search is always downhill on the cost surface, which makes it impossible to investigate other valleys that might exist. Therefore, one way of realizing a global search is to allow somewhat random moves, including uphill (cost-increasing) ones, in a certain controlled manner to visit various valleys. The following two methods, which are the only general global optimization methods to date other than a brute force search, take this approach.

Simulated annealing [8, 20, 39], which was invented from analogy to physical annealing in which a material is heated to a high temperature and slowly cooled down until it rests in the most stable phase, allows cost-increasing moves with a probability controlled by the so-called *temperature*. The temperature is initially set high and gradually decreased based on a certain *annealing schedule*. Accordingly, the probability of accepting uphill moves is initially high and gradually decreased.

Genetic algorithms [7, 8, 10, 11, 43] also perform global minimization by somewhat random moves, but in a totally different way. The search starts with a population of random *chromosomes* (candidates of the best estimate of the parameters) and improves the population by creating new offspring by *genetic operations*. Typical genetic operations are *crossover* that exchanges parts of two chromosomes, and *mutation* that randomly modifies parts of a chromosome. The offspring created by such operations can be better than their parents, either in the same cost valley or in a different one. Even if some are worse than their parents, they might be on the way to better (deeper) valleys, although the worst chromosomes in the population are discarded. Genetic algorithms were invented from analogy to evolution in nature where random mutations in genes over many generations produce optimizations in the species.

These two global optimization methods are applied to many kinds of problems, often successfully. There is, however, a major difficulty in using them. Their optimization performance depends on settings of the control parameters that control the search behavior itself. The user often has to optimize the control parameters of the algorithm before successfully getting a good estimate of the desired model parameters.

In the case of simulated annealing, decreasing the temperature too fast may cause inaccurate parameter estimation, whereas decreasing it too slowly results in waste of computation. Similarly, in the case of a genetic algorithm, the population size, i.e. the number of chromosomes, and other control parameters determine the accuracy and efficiency of the estimation [11]. Another problem with genetic algorithms is that a vast amount of memory may be needed because many estimates (chromosomes) have to be stored.

1.4 Proposed Method: Multiresolution Local Search

This dissertation proposes, develops and tests *multiresolution local search* methods. The motivation is to develop a method that can perform a global, or nearly global, search in a deterministic manner as opposed to the stochastic approaches found in conventional global search methods. Using a deterministic process eliminates the need for control parameters for stochastic behavior used in conventional global search methods, and their attendant problems. As a result, it is hoped that the search behavior will be more predictable and the method will be easier to use.

Conducting a local search in a multiresolution manner may be a good answer, provided that the parameter to be estimated is continuous almost everywhere in its defined field. The basic idea is to first estimate the large-scale characteristics of the parameter distribution and gradually move on to the smaller-scale ones. Such a method can be realized by combining a conventional local search method with a multiresolution scheme. Intuitive reasons behind this approach are the following:

1. An estimate of a parameter distribution can be considered good if its large-scale characteristics agree with those of the true distribution but its small-scale ones (details) do not. On the other hand, an estimate with wrong large-scale

characteristics and some correct small-scale ones cannot be a good one. In this sense, large-scale characteristics should be given priority in the estimation.

2. In most estimation problems of distributed parameters, parameter values in any region of the defined field depend on, and are affected by, those in the other region. Consequently, for the estimate of any region to become good, the estimate of the other region has to be good, too. In other words, attempts to estimate the details may be useless or even harmful unless the larger-scale characteristics are already close to those of the true distribution.
3. If the distributed parameter is continuous almost everywhere in its defined field, its values at neighboring points may be close to one another. Then, we can restrict the degrees of freedom among neighboring parameter elements without losing much information of the parameter distribution. Restricting the degrees of freedom in this case is equivalent to suppressing the small-scale characteristics. It also leads to a smaller dimension of the search space.
4. In a local search method like the conjugate gradient method, the number of iterations needed to reach an optimum is roughly proportional to the dimension of the search space. Therefore, a search can be faster with a smaller dimension of the search space.

It was found that there were other researchers who conceived the same idea of using multiresolution optimization for parameter estimation [4, 23]. They used the *Haar wavelet* [5, 6, 31] as a multiresolution scheme combined with a quasi-Newton method (the BFGS algorithm) to estimate a distributed parameter of elliptic and parabolic models. It was shown that the multiresolution algorithm performed better than an ordinary single-resolution method. Their results were quite promising, although the models tested were only one-dimensional and the number of elements of the distributed parameter was limited to 32.

This dissertation develops methods that use the *discrete Fourier transform* [21, 30, 40] as well as the Haar wavelet transform for a multi-dimensional field. The major differences between the two transforms are: (1) the Haar wavelet transform has compact support, i.e., the effect of each coefficient of the transform is limited to a certain region of the field, whereas each coefficient of the Fourier transform has influence on the whole field, (2) the Fourier transform is smooth because it is based on sinusoidal waves, whereas the Haar wavelet transform is discontinuous, (3) the Fourier transform is more flexible in terms of the positions of the highs and the lows of a distribution at low resolution than the wavelet transform, because in the wavelet transform each coefficient is assigned to a fixed region, and (4) the number of different resolution levels that can be taken is much greater with the Fourier transform than with the wavelet transform, because the resolution goes up linearly in the Fourier transform and exponentially in the wavelet transform.

There are various ways of controlling resolution. In this dissertation, two methods are tested. One method uses a *step scheme* where the threshold frequency of a low-pass filter that cuts off the higher-frequency coefficients is moved one step higher each time the search converges. The other method uses a *weight scheme* that assigns larger weights to the coefficients of lower frequencies so that the search is guided more by the low-frequency information than by the high-frequency information.

The original idea of multiresolution optimization is to estimate the coefficients of the transform of the desired distributed parameter rather than the parameter itself. The reason is that the coefficients of the transform are inherently sorted by resolution and hence easy to manipulate in a multiresolution manner. However, this dissertation shows that the same results can be obtained by directly estimating the parameter using the gradient that is *filtered* by the transform. In other words, it is shown that a multiresolution search can be performed by simply filtering the gradient for a local search.

Performance of the developed methods is evaluated in the so-called electrical impedance tomography [16, 17, 22, 24, 26, 28, 29, 38, 41, 42, 44, 45], which is one of many application areas. The parameter to be estimated in the test problem of this dissertation is distributed over a large number (4096) of discretized nodes in a two-dimensional field. Since the number of unknowns is the dimension of the search space, estimation is expected to be difficult enough with this many unknown elements.

1.5 Contributions of This Dissertation

I would like to claim that the following are the contributions of this dissertation to advancement of the research area:

1. Developed the methods of multiresolution optimization that are based on the discrete Fourier transform for a multi-dimensional field.
2. Extended the multiresolution optimization method that is based on the Haar wavelet transform to be used for a multi-dimensional field.
3. Devised two schemes of controlling resolution, the "step scheme" and the "weight scheme", and investigated their performance.
4. Proved that multiresolution optimization can be simplified to filtering the gradient via the transform in a multiresolution manner.
5. Showed a generalized view of utilizing the so-called *back-propagation* algorithm for calculating the gradient.
6. Devised a practical method for calculating the gradient with a small amount of memory for a convergence-type forward solution scheme.
7. Examined fundamental differences between the Fourier transform and the wavelet transforms from the parameter estimation point of view.

2. Globalizing a Local Search by Multiresolution

2.1 General Framework

Multiresolution estimation of a distributed parameter can be realized by using a transform between the raw parameter field and the corresponding field in a frequency or scale domain. Instead of estimating the parameter field, we estimate the coefficients of such a transform. Then, since each coefficient of the transform corresponds to a specific frequency or scale, it is easy to control the resolution in estimating the parameter distribution.

Let $\mathcal{T}$ be a transform from the parameter field into the coefficient field in a frequency or scale domain, and $\mathcal{T}^{-1}$ be its inverse transform. That is:

$$\begin{aligned} \boldsymbol{R} &= \mathcal{T}(\boldsymbol{r}) \\ \boldsymbol{r} &= \mathcal{T}^{-1}(\boldsymbol{R}) \end{aligned}$$

$\boldsymbol{r}$: distributed parameter

$\boldsymbol{R}$: coefficients of the transform

Since the transform $\mathcal{T}$ performs a one-to-one mapping between the coefficients $\boldsymbol{R}$ and the parameter $\boldsymbol{r}$, we can *indirectly* estimate $\boldsymbol{r}$ by estimating $\boldsymbol{R}$. And in estimating $\boldsymbol{R}$, we can manipulate the resolution by utilizing the frequency or scale information inherently associated with $\boldsymbol{R}$.

In this dissertation, the discrete Fourier transform and the Haar wavelet transform are used as $\mathcal{T}$, and the conjugate gradient method is used for a local search. In the Fourier transform, it is "frequency" that is associated with each coefficient, and in the wavelet transform, it is "scale". The resolution goes up as the frequency increases or the scale decreases. In this sense, we consider "low frequency" and "large scale", or “high frequency” and “small scale”, interchangeable to simplify explanations that apply to both the Fourier transform and the wavelet transform.

As a method of manipulating the resolution, two schemes are devised as follows.

2.1.1 Step Scheme

One way of giving priority to low-resolution components over high-resolution components is to first estimate only the low-frequency coefficients of the transform and gradually move on to the higher-frequency coefficients. It is realized by neglecting the coefficients whose frequencies are higher than a certain threshold and stepping up the threshold every time the estimation converges. We call this approach the “step scheme”.

```
1  Input target state $\widehat{\boldsymbol{V}}$ (and possibly initial state $\boldsymbol{V}_\circ$)
2  Set initial guess of parameter $\boldsymbol{r}$ and transform it by $\boldsymbol{R} := \mathcal{T}(\boldsymbol{r})$
3  Repeat from the lowest to the highest frequency threshold:
   3.1  Repeat until the advancing criterion is met:
        3.1.1  $\boldsymbol{r} := \mathcal{T}^{-1}(\boldsymbol{R})$
        3.1.2  $\boldsymbol{V} := G(\boldsymbol{r})$
        3.1.3  $cost := H(\boldsymbol{V}, \widehat{\boldsymbol{V}})$
        3.1.4  Adjust the elements of $\boldsymbol{R}$ below the threshold
4  Output $\boldsymbol{r}$
```

Figure 2.1: Step scheme of multiresolution estimation

The general framework of the step scheme is shown in Figure 2.1. Since we estimate the coefficients $\boldsymbol{R}$ instead of the distributed parameter $\boldsymbol{r}$, the framework of the estimation algorithm Figure 1.1 in page 3 is modified. Line **3.1.1** transforms the present estimate of the coefficients $\boldsymbol{R}$ into the distributed parameter $\boldsymbol{r}$, line **3.1.2** solves the direct problem, line **3.1.3** calculates the cost, and line **3.1.4** adjusts the elements of $\boldsymbol{R}$ whose frequencies are lower than the threshold.

We have to consider a criterion for advancing the frequency threshold in the step scheme. There are various ways for this, and the best way may be different for different estimation problems. Here, under an assumption, one criterion is presented.

The assumption is that the shape of the cost curve against the number of local search iterations is approximately similar in all the frequency steps, after being appropriately scaled and shifted. In other words, we assume that H_L, the cost with frequency limit L, can be roughly described by the following equation:

$$H_L(t_L) = a_L H_*(b_L t_L) + c_L \tag{2.1}$$

where t_L is the number of local search iterations with frequency limit L, H_* is the template curve assumed to be valid for all the frequency limits, and a_L, b_L and c_L are constants for L. Then, H'_L, the derivative of H_L with respect to t_L, is expressed by H'_*, the derivative of H_*, as follows:

$$H'_L(t_L) = a_L b_L H'_*(b_L t_L) \tag{2.2}$$

H'_L is always negative, since the cost H_L is always reduced in a local search.

We want to determine a frequency threshold advancing criterion that has the same strictness for all the frequency steps. This means that advancement should occur at the same point on the standardized curve H_*. Let such a point be $H_*(x^{\text{adv}})$. Then, from (2.1), we get $b_L t_L^{\text{adv}} = x^{\text{adv}}$ for frequency limit L, where t_L^{adv} is t_L at the advancing point. Using (2.2) for both $t_L = t_L^{\text{adv}}$ and $t_L = 0$, we get

$$\begin{aligned} H'_L(t_L^{\text{adv}}) &= a_L b_L H'_*(x^{\text{adv}}) \qquad (2.3)\\ H'_L(0) &= a_L b_L H'_*(0) \qquad (2.4) \end{aligned}$$

from which the following criterion is obtained:

$$\frac{H'_L(t_L^{\text{adv}})}{H'_L(0)} = \frac{H'_*(x^{\text{adv}})}{H'_*(0)} = C \tag{2.5}$$

where C is a constant for all L. Therefore, the search with frequency limit L should advance to the next frequency step (unless L is the highest frequency) when the following condition is satisfied:

$$\frac{H'_L(t_L)}{H'_L(0)} \leq C \tag{2.6}$$

There may be a problem with this criterion, however. That is, cost curves in reality are not perfectly similar and may sometimes come to the steepest point after several iterations. Then, using the actual steepest slope rather than the initial slope may make the criterion more consistent throughout all the frequency steps. Therefore, it may be often better to replace the criterion (2.6) with the following:

$$\frac{H'_L(t_L)}{\min(H'_L)} \leq C \tag{2.7}$$

For evaluation of each slope H'_L, it is better to use values of cost H_L that are two or more iterations apart rather than consecutive ones. The reason is that the descent of the cost fluctuates. Using cost values some iterations apart has an averaging effect and prevents premature advancement of frequency when the descent of the cost happens to be small at one iteration.

If the number of coefficients to be estimated is N_L with frequency limit L and the cost function is quadratic with respect to them, it takes $O(N_L)$ local search iterations to converge in the conjugate gradient method [25, 30]. Although the cost function in general is more complicated than quadratic, $O(N_L)$ might still be useful as a rough estimate of the number of local search iterations required in the step L.

2.1.2 Weight Scheme

Another way of giving priority to low-resolution components is to attenuate the adjustments of the transform coefficients according to their frequencies. This can be realized by assigning certain constant weights to the coefficients such that the weight decreases as the frequency goes up. We call this method the "weight scheme".

```
1   Input target state $\widehat{\boldsymbol{V}}$ (and possibly initial state $\boldsymbol{V}_\circ$)
2   Set initial guess of parameter $\boldsymbol{r}$ and transform it by $\boldsymbol{R} := \mathcal{T}(\boldsymbol{r})$
3   Assign weights to coefficients according to frequencies
4   Repeat until convergence:
    4.1   $\boldsymbol{r} := \mathcal{T}^{-1}(\boldsymbol{R})$
    4.2   $\boldsymbol{V} := G(\boldsymbol{r})$
    4.3   $cost := H(\boldsymbol{V}, \widehat{\boldsymbol{V}})$
    4.4   Adjust $\boldsymbol{R}$ using the weights
5   Output $\boldsymbol{r}$
```

Figure 2.2: Weight scheme of multiresolution estimation

Figure 2.2 shows the general framework of the weight scheme. It has no loop on frequency unlike the step scheme, since the weights are constant throughout the estimation. Line **3** assigns the weights to the coefficients of the transform. The loop in line **4** is an ordinary local search method for the coefficients $\boldsymbol{R}$ except that the adjustment of each coefficient is multiplied by its assigned weight in line **4.4**.

2.1.3 Estimating Transform Coefficients vs. Filtering the Gradient

So far, we have considered algorithms that *indirectly* estimate the desired parameter via estimating the transform coefficients. However, if the employed local search method determines search directions based on the gradient, we can devise another kind of multiresolution algorithm that *directly* estimates the distributed parameter.

The key here is to *filter* the gradient so that the search direction is controlled in a multiresolution manner. In other words, it is possible to *guide* the search in a multiresolution manner by manipulating the *resolution of the gradient.*

Filtering the gradient $\nabla_{\boldsymbol{r}} H$ can be performed by using a transform $\mathcal{T}$ as follows:

$$\nabla_{\boldsymbol{r}}^{\text{flt}} H = \mathcal{T}^{-1}(\mathcal{W}(\mathcal{T}(\nabla_{\boldsymbol{r}} H))) \tag{2.8}$$

where $\mathcal{W}()$ represents multiplying the transform coefficients by certain weights and $\nabla_{\boldsymbol{r}}^{\text{flt}} H$ is the resulting filtered gradient. Namely, the original gradient is transformed into the frequency or scale domain, multiplied by certain weights based on the frequencies or scales, and then transformed back into the original domain to make the filtered gradient.

Figure 2.3 shows the step scheme that uses the filtered gradient. The loop in line **3** sets the frequency threshold the same way as in the indirect algorithm Figure 2.1, page 11. Line **3.1** sets the low-pass filter, which is the filter defined by (2.8) with weight $= 1$ for the coefficients below the threshold and weight $= 0$ for the others. Line **3.2.3** gets the filtered gradient using the filter, and line **3.2.4** adjusts the estimate using the filtered gradient.

Figure 2.4 is the weight scheme using the filtered gradient. It is the same as an ordinary single-resolution local search except that the gradient is filtered. Line **3** sets the filter defined by (2.8) with appropriate weights, line **4.3** gets the filtered gradient, and line **4.4** adjusts the estimate using the filtered gradient.

It will be shown later in Sections 2.3.3 and 2.4.3 that, as long as a local search is performed by the conjugate gradient method, the direct estimation of the parameter using the filtered gradient is equivalent to the indirect estimation via estimating the transform coefficients. Hence, the direct estimation with the filtered gradient is neither superior nor inferior to the indirect estimation in terms of the performance. However, the direct estimation is conceptually simpler and thus easier to implement.

```
1 Input target state $\widehat{\boldsymbol{V}}$ (and possibly initial state $\boldsymbol{V}_\circ$)
2 Set initial guess of parameter $\boldsymbol{r}$
3 Repeat from the lowest to the highest frequency threshold:
    3.1 Set the low-pass filter for the gradient using the threshold
    3.2 Repeat until the advancing criterion is met:
        3.2.1 $\boldsymbol{V} := G(\boldsymbol{r})$
        3.2.2 $cost := H(\boldsymbol{V}, \widehat{\boldsymbol{V}})$
        3.2.3 Get $\nabla^{\mathrm{flt}}_{\boldsymbol{r}} H$ by filtering the gradient $\nabla_{\boldsymbol{r}} H$
        3.2.4 Adjust $\boldsymbol{r}$ based on $\nabla^{\mathrm{flt}}_{\boldsymbol{r}} H$
4 Output $\boldsymbol{r}$
```

Figure 2.3: Step scheme using filtered gradient

```
1 Input target state $\widehat{\boldsymbol{V}}$ (and possibly initial state $\boldsymbol{V}_\circ$)
2 Set initial guess of parameter $\boldsymbol{r}$
3 Set the filter for the gradient using the frequency weights
4 Repeat until convergence:
    4.1 $\boldsymbol{V} := G(\boldsymbol{r})$
    4.2 $cost := H(\boldsymbol{V}, \widehat{\boldsymbol{V}})$
    4.3 Get $\nabla^{\mathrm{flt}}_{\boldsymbol{r}} H$ by filtering the gradient $\nabla_{\boldsymbol{r}} H$
    4.4 Adjust $\boldsymbol{r}$ based on $\nabla^{\mathrm{flt}}_{\boldsymbol{r}} H$
5 Output $\boldsymbol{r}$
```

Figure 2.4: Weight scheme using filtered gradient

Furthermore, it can possibly open up new ways of multiresolution optimization by introducing various types of filters for the gradient.

2.2 The Conjugate Gradient Method

This dissertation employs the conjugate gradient method [14, 25, 30, 37] for local search. The reason is that this method is suitable for estimating a large number of unknowns because its required storage is smaller than that of the quasi-Newton family, which is another powerful method, and yet its convergence speed is comparable (see Section 1.2). In the following, the method is explained for two cases. In the first ordinary case, the parameters to be estimated are real numbers. We extend the method to the second case, where the parameters are complex numbers.

2.2.1 When Parameters are Real

The conjugate gradient method is based on the assumption that the real function to be minimized, the cost H in our case, is approximately quadratic as follows:

$$H(\boldsymbol{x}) \approx \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + c \tag{2.9}$$

where $\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^T \in \mathcal{R}^N$ is a column vector of N real unknowns ($\mathcal{R}$: the set of real numbers), $\boldsymbol{A}$ is an $N \times N$ symmetric positive definite matrix, $\boldsymbol{b}$ is a column vector of N real constants, and c is a real scalar constant. This can be considered as an approximation of any smooth function, because the effects of the terms of degree 3 and higher become smaller as a minimum is approached.

Let us imagine the N-dimensional search space where each of the N coordinates represents one of the N elements of the unknown $\boldsymbol{x}$. That is, any one point in the search space determines the values of all the elements of $\boldsymbol{x}$, thus representing a specific $\boldsymbol{x}$.

Let $\boldsymbol{x}_j$ be the j-th point in the search space visited by the conjugate gradient method. The next point $\boldsymbol{x}_{j+1}$ is determined by searching for a minimum in the direction $\boldsymbol{d}_j$ from the present point $\boldsymbol{x}_j$. Namely, the relation between $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j+1}$ is:

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \sigma_j \boldsymbol{d}_j \tag{2.10}$$

such that a real scalar σ_j minimizes the function value $H(\boldsymbol{x}_{j+1})$. This is called a "line search" or a "line minimization".

The direction $\boldsymbol{d}_j$ of the line minimization is initially (when $j = 0$) the negative gradient at the initial point $\boldsymbol{x}_0$. In the following cycles it is the present negative gradient deflected toward the previous search direction $\boldsymbol{d}_{j-1}$ as follows. Let $\boldsymbol{g}_j$ be the gradient at $\boldsymbol{x}_j$:

$$\boldsymbol{g}_j \equiv \nabla_{\boldsymbol{x}} H(\boldsymbol{x}_j) \tag{2.11}$$

Then, the search direction $\boldsymbol{d}_j$ for the j-th line minimization is determined by:

$$\boldsymbol{d}_j = \begin{cases} -\boldsymbol{g}_0 & \text{if } j = 0 \\ -\boldsymbol{g}_j + \tau_j \boldsymbol{d}_{j-1} & \text{if } j \geq 1 \end{cases} \tag{2.12}$$

The real coefficient τ_j that determines the amount of influence of the previous search direction on the present one is usually calculated by either of the two formulas below:

$$\tau_j = \begin{cases} \dfrac{\boldsymbol{g}_j \cdot \boldsymbol{g}_j}{\boldsymbol{g}_{j-1} \cdot \boldsymbol{g}_{j-1}} & \text{(Fletcher-Reeves)} \\ \dfrac{(\boldsymbol{g}_j - \boldsymbol{g}_{j-1}) \cdot \boldsymbol{g}_j}{\boldsymbol{g}_{j-1} \cdot \boldsymbol{g}_{j-1}} & \text{(Polak-Ribiere)} \end{cases} \tag{2.13}$$

where $\boldsymbol{x} \cdot \boldsymbol{y}$ denotes the inner product of the vectors. The Polak-Ribiere formula is said to be usually superior to the Fletcher-Reeves formula [30].

2.2.2 When Parameters are Complex

Let us consider the case where the cost H is a function of complex numbers. This consideration is necessary for estimating the Fourier coefficients, which are complex in general. We convert such a complex problem into a real problem by using the fact that a complex number consists of two real numbers, the real part and the imaginary part.

Let the cost H be a real-valued function of a distributed parameter $\boldsymbol{x}$ that has N unknown complex elements, namely $H : \mathcal{C}^N \rightarrow \mathcal{R}$ ($\mathcal{C}$: the set of complex numbers). The parameter vector $\boldsymbol{x}$ consists of the real part $\boldsymbol{x}_R$ and the imaginary part $\boldsymbol{x}_I$ such that

$$\boldsymbol{x} = \boldsymbol{x}_R + i\,\boldsymbol{x}_I \qquad (i = \sqrt{-1}) \tag{2.14}$$

where

$$\begin{aligned} \boldsymbol{x}_R &\equiv \mathrm{Re}(\boldsymbol{x}) \in \mathcal{R}^N \\ \boldsymbol{x}_I &\equiv \mathrm{Im}(\boldsymbol{x}) \in \mathcal{R}^N \end{aligned}$$

In this way, an N-component complex vector $\boldsymbol{x}$ is decomposed into two N-component real vectors $\boldsymbol{x}_R$ and $\boldsymbol{x}_I$. Let us create a $2N$-component real column vector from the two real column vectors $\boldsymbol{x}_R$ and $\boldsymbol{x}_I$ as follows:

$$\boldsymbol{x}_{RI} \triangleq \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_I \end{bmatrix} \tag{2.15}$$

That is, the upper half of $\boldsymbol{x}_{RI}$ is $\boldsymbol{x}_R$ and the lower half is $\boldsymbol{x}_I$, thus $\boldsymbol{x}_{RI} \in \mathcal{R}^{2N}$.

The cost function H can be considered to take either the original complex vector $\boldsymbol{x}$ or its converted real version $\boldsymbol{x}_{RI}$. The former is viewed as a function of N complex numbers and the latter as a function of $2N$ real numbers:

$$\begin{aligned} H(\boldsymbol{x}): &\qquad \mathcal{C}^N \rightarrow \mathcal{R} \\ H(\boldsymbol{x}_{RI}): &\qquad \mathcal{R}^{2N} \rightarrow \mathcal{R} \end{aligned}$$

From (2.15), the gradient of the real version $\nabla_{\boldsymbol{x}_{RI}} H$ consists of the gradient with respect to $\boldsymbol{x}_R$ and the gradient with respect to $\boldsymbol{x}_I$ as follows:

$$\nabla_{\boldsymbol{x}_{RI}} H \equiv \begin{bmatrix} \nabla_{\boldsymbol{x}_R} H \\ \nabla_{\boldsymbol{x}_I} H \end{bmatrix} \tag{2.16}$$

Since H, $\boldsymbol{x}_R$ and $\boldsymbol{x}_I$ are all real, both $\nabla_{\boldsymbol{x}_R} H$ and $\nabla_{\boldsymbol{x}_I} H$ must be real. Therefore, using the chain rule for partial differentiation, we get the following:

$$\begin{aligned}
\nabla_{\boldsymbol{x}_R} H \equiv \left[\frac{\partial H}{\partial \boldsymbol{x}_R}\right]^T &= \mathrm{Re}\left(\left[\frac{\partial H}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_R}\right]^T\right) \\
&= \mathrm{Re}\left(\left[\frac{\partial H}{\partial \boldsymbol{x}} \cdot 1\right]^T\right) \\
&= \mathrm{Re}\left(\left[\frac{\partial H}{\partial \boldsymbol{x}}\right]^T\right) \\
&\equiv \mathrm{Re}(\nabla_{\boldsymbol{x}} H) \qquad (2.17) \\
\nabla_{\boldsymbol{x}_I} H \equiv \left[\frac{\partial H}{\partial \boldsymbol{x}_I}\right]^T &= \mathrm{Re}\left(\left[\frac{\partial H}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_I}\right]^T\right) \\
&= \mathrm{Re}\left(\left[\frac{\partial H}{\partial \boldsymbol{x}} \cdot i\right]^T\right) \\
&= -\mathrm{Im}\left(\left[\frac{\partial H}{\partial \boldsymbol{x}}\right]^T\right) \\
&\equiv -\mathrm{Im}(\nabla_{\boldsymbol{x}} H) \qquad (2.18)
\end{aligned}$$

Here, we treated $\nabla_{\boldsymbol{y}} H$ ($\boldsymbol{y} \equiv \boldsymbol{x}, \boldsymbol{x}_R$ or $\boldsymbol{x}_I$) as an N-component column vector, $\dfrac{\partial H}{\partial \boldsymbol{y}}$ as an N-component row vector, and $\dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}$ as an $N \times N$ matrix. Putting (2.17) and (2.18) into (2.16), we finally get the following relation between the gradient of the complex version $\nabla_{\boldsymbol{x}} H$ and the gradient of the real version $\nabla_{\boldsymbol{x}_{RI}} H$.

$$\nabla_{\boldsymbol{x}_{RI}} H = \begin{bmatrix} \mathrm{Re}(\nabla_{\boldsymbol{x}} H) \\ -\mathrm{Im}(\nabla_{\boldsymbol{x}} H) \end{bmatrix} \tag{2.19}$$

Note that the sign of the imaginary part of the complex gradient must be reversed.

The following is the conjugate gradient method for $\boldsymbol{x}_{RI}$, based on the method shown in Section 2.2.1. At each iteration $j \geq 0$, the next estimate $\boldsymbol{x}_{RI_{j+1}}$ is determined from the present estimate $\boldsymbol{x}_{RI_j}$ by line minimization in the direction $\boldsymbol{d}_{RI_j}$:

$$\boldsymbol{x}_{RI_{j+1}} = \boldsymbol{x}_{RI_j} + \sigma_j \boldsymbol{d}_{RI_j} \tag{2.20}$$

where σ_j is a real scalar that minimizes $H(\boldsymbol{x}_{RI_{j+1}})$. The line-search direction $\boldsymbol{d}_{RI_j}$ is determined by

$$\boldsymbol{d}_{RIj} = \begin{cases} -\boldsymbol{g}_{RI_0} & \text{if } j = 0 \\ -\boldsymbol{g}_{RI_j} + \tau_j \boldsymbol{d}_{RI_{j-1}} & \text{if } j \geq 1 \end{cases} \tag{2.21}$$

where $\boldsymbol{g}_{RI_j} \equiv \nabla_{\boldsymbol{x}_{RI}} H(\boldsymbol{x}_{RI_j})$. The real coefficient τ_j is calculated as follows:

$$\tau_j = \begin{cases} \dfrac{\boldsymbol{g}_{RI_j} \cdot \boldsymbol{g}_{RI_j}}{\boldsymbol{g}_{RI_{j-1}} \cdot \boldsymbol{g}_{RI_{j-1}}} & \text{(Fletcher-Reeves)} \\ \dfrac{(\boldsymbol{g}_{RI_j} - \boldsymbol{g}_{RI_{j-1}}) \cdot \boldsymbol{g}_{RI_j}}{\boldsymbol{g}_{RI_{j-1}} \cdot \boldsymbol{g}_{RI_{j-1}}} & \text{(Polak-Ribiere)} \end{cases} \tag{2.22}$$

From the above, we can obtain the equivalent expressions for the original complex parameter $\boldsymbol{x}$ as follows. At each iteration $j \geq 0$, the next estimate $\boldsymbol{x}_{j+1}$ is determined from the present estimate $\boldsymbol{x}_j$ by line minimization in the direction $\boldsymbol{d}_j$:

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \sigma_j \boldsymbol{d}_j \tag{2.23}$$

where σ_j is a real scalar that minimizes $H(\boldsymbol{x}_{j+1})$. The line-search direction $\boldsymbol{d}_j$ is determined as follows:

$$\boldsymbol{d}_j = \begin{cases} -\boldsymbol{g}_0 & \text{if } j = 0 \\ -\boldsymbol{g}_j + \tau_j \boldsymbol{d}_{j-1} & \text{if } j \geq 1 \end{cases} \tag{2.24}$$

Here, $\boldsymbol{g}_j$ is the complex conjugate of the gradient at $\boldsymbol{x}_j$:

$$\boldsymbol{g}_j \equiv \overline{\nabla_{\boldsymbol{x}} H(\boldsymbol{x}_j)} \tag{2.25}$$

because, as (2.19) shows, $\nabla_{\boldsymbol{x}_{RI}} H$ is equivalent to $\mathrm{Re}(\nabla_{\boldsymbol{x}} H) - i\,\mathrm{Im}(\nabla_{\boldsymbol{x}} H) \equiv \overline{\nabla_{\boldsymbol{x}} H}$. The real coefficient τ_j is calculated as follows:

$$\tau_j = \begin{cases} \dfrac{\boldsymbol{g}_j \cdot \boldsymbol{g}_j}{\boldsymbol{g}_{j-1} \cdot \boldsymbol{g}_{j-1}} & \text{(Fletcher-Reeves)} \\ \dfrac{\mathrm{Re}\Big((\boldsymbol{g}_j - \boldsymbol{g}_{j-1}) \cdot \boldsymbol{g}_j\Big)}{\boldsymbol{g}_{j-1} \cdot \boldsymbol{g}_{j-1}} & \text{(Polak-Ribiere)} \end{cases} \tag{2.26}$$

where the inner product of two complex column vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined by

$$\boldsymbol{x} \cdot \boldsymbol{y} \triangleq \overline{\boldsymbol{x}}^T \boldsymbol{y} \tag{2.27}$$

where $\overline{\boldsymbol{x}}$ is the complex conjugate of $\boldsymbol{x}$.

Note that (2.24), (2.25) and (2.26) can be used even if the parameter $\boldsymbol{x}$ is real, because in that case they become equivalent to (2.12), (2.11) and (2.13).

2.3 The Discrete Fourier Transform

This section introduces the *discrete Fourier transform* (DFT) as one of the transforms for multiresolution optimization. First, the DFT for a one-dimensional field, and then the general n-dimensional DFT are shown. Finally, it will be shown that, coupled with the conjugate gradient method, estimating the coefficients of the DFT of a distributed parameter is equivalent to directly estimating the parameter using the gradient filtered by the DFT.

2.3.1 For One-dimensional Field

A one-dimensional distributed parameter can be represented by an array of values of the parameter sampled at equal intervals. Let $\boldsymbol{r} = (r_0, r_1, \cdots, r_{N-1})$ be such an array that represents a one-dimensional distributed parameter sampled at N discretized points. Let each element r_j be a complex number for a general case.

The one-dimensional DFT transforms the N complex elements of the parameter $\boldsymbol{r}$ into another set of N complex numbers in the frequency domain called the Fourier coefficients. Let $\boldsymbol{R} = (R_0, R_1, \cdots, R_{N-1})$ be the Fourier coefficients of the parameter $\boldsymbol{r}$.

One definition of the one-dimensional DFT from $\boldsymbol{r}$ to $\boldsymbol{R}$ is the following [21] :

$$R_k = \frac{1}{N}\sum_{j=0}^{N-1} r_j e^{-\frac{i2\pi kj}{N}} \qquad (k = 0, \cdots, N-1) \tag{2.28}$$

The corresponding inverse DFT, which reconstructs the distributed parameter $\boldsymbol{r}$ from the Fourier coefficients $\boldsymbol{R}$, is defined by:

$$r_j = \sum_{k=0}^{N-1} R_k e^{\frac{i2\pi jk}{N}} \qquad (j = 0, \cdots, N-1) \tag{2.29}$$

With this set of definitions, the coefficient R_0 for the frequency zero is equal to the average of the parameter elements r_j.

There are other definitions of the DFT and the inverse DFT than the above. They differ in the constant before the summation $\sum$ and/or the sign of the exponent of e. Any such definition can be used as long as the transform and the inverse transform are self-consistent. The following definitions in which the inverse DFT, instead of the DFT, has the constant $1/N$ seem to be used in most of the literature, e.g. [19, 40].

$$R_k = \sum_{j=0}^{N-1} r_j e^{-\frac{i2\pi kj}{N}} \qquad (k = 0, \cdots, N-1) \tag{2.30}$$

$$r_j = \frac{1}{N}\sum_{k=0}^{N-1} R_k e^{\frac{i2\pi jk}{N}} \qquad (j = 0, \cdots, N-1) \tag{2.31}$$

The only difference from the previous definitions (2.28) and (2.29) is that the magnitudes of the Fourier coefficients $\boldsymbol{R}$ are greater by the factor of N.

The sign of the exponent of e can be reversed as the following [18, 30].

$$R_k = \sum_{j=0}^{N-1} r_j e^{\frac{i2\pi kj}{N}} \qquad (k = 0, \cdots, N-1) \tag{2.32}$$

$$r_j = \frac{1}{N}\sum_{k=0}^{N-1} R_k e^{-\frac{i2\pi jk}{N}} \qquad (j = 0, \cdots, N-1) \tag{2.33}$$

The effect is simply that the parameter distribution is looked at in the reverse order as shown below. Since the Fourier transform implicitly assumes periodicity, one more element r_N can be defined as $r_N = r_0$. The reversed array of $\{r_j\}$ is $\{r'_j\}$ defined by $r'_j = r_{N-j}$ for $j = 0, \cdots, N$. Then, using the relation:

$$e^{\frac{i2\pi kj}{N}} = e^{(\frac{i2\pi kj}{N} - i2\pi k)} = e^{-\frac{i2\pi k(N-j)}{N}}$$

we can rewrite (2.32) that is about the array $\{r_j\}$ into an expression about the reversed array $\{r'_j\}$ as follows:

$$\begin{aligned} R_k &= \sum_{j=0}^{N-1} r_j e^{\frac{i2\pi kj}{N}} \\ &= \sum_{j=0}^{N-1} r'_{N-j} e^{-\frac{i2\pi k(N-j)}{N}} \\ &= \sum_{h=1}^{N} r'_h e^{-\frac{i2\pi kh}{N}} \qquad (h \equiv N-j) \\ &= \sum_{h=0}^{N-1} r'_h e^{-\frac{i2\pi kh}{N}} \qquad (\text{since } r'_0 = r'_N) \end{aligned} \tag{2.34}$$

The result is exactly the DFT for the reversed array $\{r'_j\}$ as defined by (2.30). Therefore, flipping the sign of the exponent of e in the DFT definition simply has the effect of reading the array $\{r_j\}$ backward in the order: $r_N \to r_{N-1} \to \cdots \to r_1$.

Actually, we can define either one of the two transforms below as the DFT and the other as the inverse DFT between complex arrays $\{x_j\}$ and $\{y_k\}$ $(j, k = 0, \cdots, N-1)$ as long as real constants α and β have the relation: $\alpha\beta = 1/N$.

$$y_k = \alpha \sum_{j=0}^{N-1} x_j e^{-\frac{i2\pi kj}{N}} \qquad (k = 0, \cdots, N-1) \tag{2.35}$$

$$x_j = \beta \sum_{k=0}^{N-1} y_k e^{\frac{i2\pi jk}{N}} \qquad (j = 0, \cdots, N-1) \tag{2.36}$$

For example, if (2.35) is chosen as the DFT and (2.36) as the inverse DFT, the array $\{x_j\}$ becomes the distributed parameter $\boldsymbol{r}$ and the array $\{y_k\}$ becomes the Fourier coefficients $\boldsymbol{R}$.

In any definitions above, the Fourier coefficient of the lowest frequency (zero) is R_0. The coefficient of the highest frequency, called the *Nyquist critical frequency*, is $R_{\frac{N}{2}}$, for which $e^{-\frac{i2\pi kj}{N}} = e^{-i\pi j}$ (or $e^{\frac{i2\pi jk}{N}} = e^{i\pi k}$) in the DFT takes a value $+1$ and -1 alternately as j (or k) changes. The rest of the Fourier coefficients are in the order shown in Table 2.1.

Fourier coefficients	relative frequency
R_0	0
R_1 and R_{N-1}	1
R_2 and R_{N-2}	2
$\vdots$	$\vdots$
R_k and R_{N-k}	k
$\vdots$	$\vdots$
$R_{\frac{N}{2}-2}$ and $R_{\frac{N}{2}+2}$	$\frac{N}{2}-2$
$R_{\frac{N}{2}-1}$ and $R_{\frac{N}{2}+1}$	$\frac{N}{2}-1$
$R_{\frac{N}{2}}$	$\frac{N}{2}$

Table 2.1: Fourier coefficients and frequency

We have considered a general case in which the distributed parameter $\boldsymbol{r}$ is complex. However, the parameter very often has only real values, i.e. the imaginary part of each parameter element is zero. In this case, the following theorem holds.

Theorem 2.1: *If all the elements of the distributed parameter* $\boldsymbol{r} = (r_0, r_1, \cdots, r_{N-1})$ *are real, the corresponding Fourier coefficients* $\boldsymbol{R} = (R_0, R_1, \cdots, R_{N-1})$ *are such that* R_0 *and* $R_{\frac{N}{2}}$ *are real, and* R_k *and* R_{N-k} $(k = 1, \cdots, \frac{N}{2} - 1)$ *are the complex conjugate of each other.*

Proof: Let the DFT defined by (2.35). It would be similar if a different definition is used. The Fourier coefficients R_k and R_{N-k} are as follows:

$$\begin{aligned} R_k &= \alpha \sum_{j=0}^{N-1} r_j e^{-\frac{i2\pi kj}{N}} \\ &= \alpha \sum_{j=0}^{N-1} r_j \left(\cos\frac{2\pi kj}{N} - i \sin\frac{2\pi kj}{N}\right) \\ R_{N-k} &= \alpha \sum_{j=0}^{N-1} r_j e^{-\frac{i2\pi(N-k)j}{N}} \end{aligned}$$

$$
\begin{aligned}
&= \alpha \sum_{j=0}^{N-1} r_j e^{\frac{i2\pi kj}{N}} \\
&= \alpha \sum_{j=0}^{N-1} r_j \left(\cos \tfrac{2\pi kj}{N} + i \sin \tfrac{2\pi kj}{N}\right)
\end{aligned}
$$

Since all the parameter elements r_j $(j = 0, \cdots, N-1)$ are real,

$$
\begin{aligned}
\mathrm{Re}(R_k) &= \alpha \sum_{j=0}^{N-1} r_j \cos \tfrac{2\pi kj}{N} = \mathrm{Re}(R_{N-k}) \\
\mathrm{Im}(R_k) &= -\alpha \sum_{j=0}^{N-1} r_j \sin \tfrac{2\pi kj}{N} = -\mathrm{Im}(R_{N-k})
\end{aligned}
$$

Therefore,

$$R_k = \overline{R_{N-k}} \qquad (k = 1, \cdots, \frac{N}{2})$$

From this, $R_{\frac{N}{2}}$ is the complex conjugate of itself, so it must be real. Also, R_0 is real because:

$$R_0 = \alpha \sum_{j=0}^{N-1} r_j$$

■

Using the theorem above, we can obtain the following.

Theorem 2.2: *For an array of N real numbers $\boldsymbol{r} = (r_0, r_1, \cdots, r_{N-1})$, a pair of Fourier coefficients R_k and R_{N-k} $(k = 1, \cdots, \frac{N}{2}-1)$ form a real-valued cosine (or sine) wave of relative frequency k in the inverse DFT, determining the wave's amplitude and initial phase.*

Proof: Let the inverse DFT be defined by (2.36), that is:

$$r_j = \beta \sum_{k=0}^{N-1} R_k e^{\frac{i2\pi jk}{N}} \qquad (j = 0, \cdots, N-1)$$

(The proof goes similarly if (2.35) is taken as the inverse DFT.) This is equivalent to the following:

$$r_j = \beta \left(R_0 + R_{\frac{N}{2}}(-1)^j + \sum_{k=1}^{\frac{N}{2}-1} \left(R_k e^{\frac{i2\pi jk}{N}} + R_{N-k} e^{\frac{i2\pi j(N-k)}{N}} \right) \right) \tag{2.37}$$

Let us define a variable φ as follows:

$$\begin{cases} \cos\varphi &= \dfrac{\mathrm{Re}(R_k)}{|R_k|} \\ \sin\varphi &= \dfrac{\mathrm{Im}(R_k)}{|R_k|} \end{cases}$$

where $|R_k| \equiv \sqrt{(\mathrm{Re}(R_k))^2 + (\mathrm{Im}(R_k))^2}$. Using this φ and the fact $R_{N-k} = \overline{R_k}$ from Theorem 2.1 because $\boldsymbol{r}$ is real, we get the following:

$$\begin{aligned} R_k e^{\frac{i2\pi jk}{N}} + R_{N-k} e^{\frac{i2\pi j(N-k)}{N}} &= R_k e^{\frac{i2\pi jk}{N}} + R_{N-k} e^{-\frac{i2\pi jk}{N}} \\ &= R_k e^{\frac{i2\pi jk}{N}} + \overline{R_k}\, \overline{e^{\frac{i2\pi jk}{N}}} \\ &= R_k e^{\frac{i2\pi jk}{N}} + \overline{R_k e^{\frac{i2\pi jk}{N}}} \\ &= 2\mathrm{Re}\Big(R_k e^{\frac{i2\pi jk}{N}}\Big) \\ &= 2\mathrm{Re}\Big((\mathrm{Re}(R_k) + i\mathrm{Im}(R_k))(\cos\tfrac{2\pi jk}{N} + i\sin\tfrac{2\pi jk}{N})\Big) \\ &= 2\left(\mathrm{Re}(R_k)\cos\tfrac{2\pi jk}{N} - \mathrm{Im}(R_k)\sin\tfrac{2\pi jk}{N}\right) \\ &= 2|R_k|\left(\cos\varphi\cos\tfrac{2\pi jk}{N} - \sin\varphi\sin\tfrac{2\pi jk}{N}\right) \\ &= 2|R_k|\cos\left(\tfrac{2\pi jk}{N} + \varphi\right) \end{aligned}$$

Therefore, coupled with a real constant β in (2.37), the pair R_k and R_{N-k} constitute a real-valued cosine wave of relative frequency k whose amplitude is $2\beta|R_k|$ and initial phase (at $j = 0$) is φ. ■

The Fourier coefficient R_0, one of the two that are left out of Theorem 2.2, can be viewed as representing a real cosine wave of frequency zero, and the other one $R_{\frac{N}{2}}$ as a real cosine wave of the Nyquist critical frequency, whose initial phase is either 0 or π. So, Theorem 2.2 tells that estimating the Fourier coefficients $\boldsymbol{R}$ of the real distributed parameter $\boldsymbol{r}$ is equivalent to estimating the amplitude and the initial phase of the cosine wave at each frequency. Therefore, if the step scheme is employed, the multiresolution estimation with the Fourier transform goes as shown

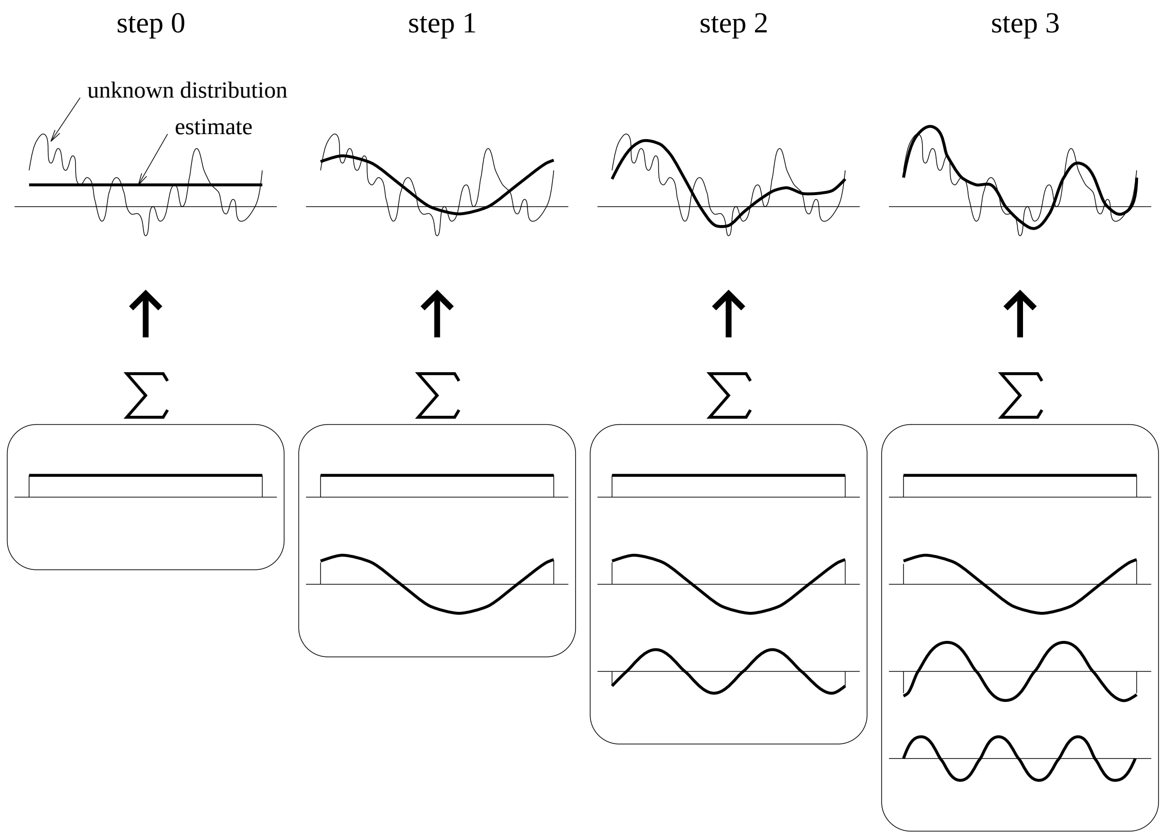


Figure 2.5: Multiresolution estimation with Fourier transform

in Figure 2.5. In each step, the amplitude and the initial phase of every cosine wave up to a certain frequency limit are estimated. If the weight scheme is taken, it simply estimates the amplitude and the initial phase of every cosine wave at all frequencies with certain weights. Since the initial phase of each cosine wave is allowed to take any value (except for the wave at the highest frequency), positions of the highs and the lows are flexible with the Fourier transform. This is one of the differences from wavelet transforms.

2.3.2 For n-dimensional Field

Now, let us consider the DFT for an n-dimensional field ($n \geq 1$). A parameter $\boldsymbol{r}$ distributed in an n-dimensional field can be described by an n-dimensional array $\{r_{j_1 j_2 \cdots j_n}\}$ where j_m is an integer from 0 to $N_m - 1$. The n-dimensional DFT is defined by combining n one-dimensional DFTs [30]. With real constants α and β having the relation $\alpha\beta = \frac{1}{N_1 N_2 \cdots N_n}$, the general forms of the n-dimensional DFT and the n-dimensional inverse DFT between two n-dimensional complex arrays $\{x_{j_1 \cdots j_n}\}$ and $\{y_{k_1 \cdots k_n}\}$ ($j_m, k_m = 0, \cdots, N_m - 1$) are the following:

$$y_{k_1 \cdots k_n} = \alpha \sum_{j_1=0}^{N_1-1} \cdots \sum_{j_n=0}^{N_n-1} x_{j_1 \cdots j_n} e^{-\frac{i2\pi k_1 j_1}{N_1}} \cdots e^{-\frac{i2\pi k_n j_n}{N_n}} \quad (k_m = 0, \cdots, N_m - 1) \tag{2.38}$$

$$x_{j_1 \cdots j_n} = \beta \sum_{k_1=0}^{N_1-1} \cdots \sum_{k_n=0}^{N_n-1} y_{k_1 \cdots k_n} e^{\frac{i2\pi j_1 k_1}{N_1}} \cdots e^{\frac{i2\pi j_n k_n}{N_n}} \quad (j_m = 0, \cdots, N_m - 1) \tag{2.39}$$

As in the one-dimensional case, one can pick either one of (2.38) and (2.39) as the n-dimensional DFT and the other as the corresponding n-dimensional inverse DFT.

The n-dimensional DFT and the inverse DFT, including the one-dimensional case, are efficiently computed by the n-dimensional *fast Fourier transform* (FFT) routines, if each N_m, the number of discretized points along the m-th axis, is a power of 2.

For simplicity, we write:

$$\boldsymbol{R} = \mathrm{DFT}(\boldsymbol{r}) \tag{2.40}$$

$$\boldsymbol{r} = \mathrm{DFT}^{-1}(\boldsymbol{R}) \tag{2.41}$$

as the DFT and the inverse DFT, respectively, between a distributed parameter $\boldsymbol{r}$ and its Fourier coefficients $\boldsymbol{R}$. Although the dimension of the DFT and the inverse DFT is not specified in this notation, it is assumed to be the same as the dimension of the field over which the parameter $\boldsymbol{r}$ is distributed.

Note that, from (2.38) and (2.39), both DFT() and $\mathrm{DFT}^{-1}()$ are linear such that

$$\mathrm{DFT}(c_1\boldsymbol{r}_1 + c_2\boldsymbol{r}_2) = c_1\mathrm{DFT}(\boldsymbol{r}_1) + c_2\mathrm{DFT}(\boldsymbol{r}_2) \tag{2.42}$$

$$\mathrm{DFT}^{-1}(c_1\boldsymbol{R}_1 + c_2\boldsymbol{R}_2) = c_1\mathrm{DFT}^{-1}(\boldsymbol{R}_1) + c_2\mathrm{DFT}^{-1}(\boldsymbol{R}_2) \tag{2.43}$$

with any complex numbers c_1 and c_2.

2.3.3 Filtering the Gradient

In Section 2.1.3, we considered the following two methods of multiresolution parameter estimation:

1. Indirect estimation of the parameter by estimating the coefficients of the transform.
2. Direct estimation of the parameter using the gradient filtered by the transform.

In this section, it is shown that the two methods above are actually equivalent if the transform is the DFT and a local search is performed by the conjugate gradient method.

First, we prove a lemma for proving the next theorem. For simplicity, the following notation is used:

$$W_N \equiv e^{\frac{i2\pi}{N}} \tag{2.44}$$

Lemma 2.1: *Let j and m be integers such that $0 \le j, m \le N-1$. Then,*

$$\sum_{k=0}^{N-1} W_N^{k(j-m)} = \begin{cases} N & \text{if } j = m \\ 0 & \text{if } j \ne m \end{cases} \tag{2.45}$$

Proof: If $j = m$,

$$\sum_{k=0}^{N-1} W_N^{k(j-m)} = \sum_{k=0}^{N-1} 1 = N$$

If $j \ne m$, $j - m$ is such that $j - m \ne 0$ and $-(N-1) \le j - m \le N - 1$, which yields the following:

$$W_N^{(j-m)} \equiv e^{\frac{i2\pi(j-m)}{N}} \ne 1$$

Therefore, we can use the sum of the geometric series:

$$\sum_{k=0}^{N-1} W_N^{k(j-m)} = \sum_{k=0}^{N-1} \left(W_N^{(j-m)}\right)^k = \frac{1-\left(W_N^{(j-m)}\right)^N}{1-W_N^{(j-m)}}$$

Since

$$\left(W_N^{(j-m)}\right)^N \equiv \left(e^{\frac{i2\pi(j-m)}{N}}\right)^N = e^{i2\pi(j-m)} = 1$$

we get:

$$\sum_{k=0}^{N-1} W_N^{k(j-m)} = 0 \tag{2.46}$$

■

Using this lemma, we prove the theorem below.

Theorem 2.3: *Let* $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ *and* $\boldsymbol{d}$ *be n-dimensional complex arrays in* $\mathcal{C}^{N_1\times\cdots\times N_n}$. *That is, for example,* $\boldsymbol{a}$ *consists of* $N_1 N_2 \cdots N_n$ *elements* $a_{k_1 k_2 \cdots k_n} (k_m = 0, \cdots, N_m - 1)$. *Let* $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ *and* $\boldsymbol{D}$ *be the coefficient arrays of the n-dimensional DFTs of* $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ *and* $\boldsymbol{d}$, *respectively.* $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D} \in \mathcal{C}^{N_1\times\cdots\times N_n})$ *Then, the following relation for inner products holds:*

$$\frac{\boldsymbol{A}\cdot\boldsymbol{B}}{\boldsymbol{C}\cdot\boldsymbol{D}} = \frac{\boldsymbol{a}\cdot\boldsymbol{b}}{\boldsymbol{c}\cdot\boldsymbol{d}} \tag{2.47}$$

Namely, ratios of inner products are unchanged by the DFT or the inverse DFT.

Proof: Let us take the following definition of the inner product of two arrays:

$$\boldsymbol{x}\cdot\boldsymbol{y} \triangleq \sum_{k_1=0}^{N_1-1} \cdots \sum_{k_n=0}^{N_n-1} \overline{x}_{k_1\cdots k_n}\, y_{k_1\cdots k_n} \tag{2.48}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}^{N_1\times\cdots\times N_n}$ and $\overline{z}$ is the complex conjugate of z. The proof goes similarly if a different definition is employed.

Let the n-dimensional DFT be defined by (2.38). That is, the Fourier coefficient array $\boldsymbol{A}$, for example, is defined as the following:

$$A_{k_1\cdots k_n} = \alpha \sum_{j_1=0}^{N_1-1} \cdots \sum_{j_n=0}^{N_n-1} a_{j_1\cdots j_n} W_{N_1}^{-k_1 j_1} \cdots W_{N_n}^{-k_n j_n} \qquad (k_m = 0, \cdots, N_m - 1) \tag{2.49}$$

where α is a real constant. The proof goes similarly if the n-dimensional DFT is defined by (2.39). Then, the inner product of two DFT coefficient arrays, e.g. $\boldsymbol{A}$ and $\boldsymbol{B}$, can be calculated as follows:

$$\begin{aligned}
\boldsymbol{A}\cdot\boldsymbol{B} &\equiv \sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1}\overline{A}_{k_1\cdots k_n}B_{k_1\cdots k_n}\\
&= \sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1}\left[\overline{\left(\alpha\sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}a_{j_1\cdots j_n}W_{N_1}^{-k_1j_1}\cdots W_{N_n}^{-k_nj_n}\right)}\right.\\
&\qquad\left.\cdot\left(\alpha\sum_{m_1=0}^{N_1-1}\cdots\sum_{m_n=0}^{N_n-1}b_{m_1\cdots m_n}W_{N_1}^{-k_1m_1}\cdots W_{N_n}^{-k_nm_n}\right)\right]\\
&= \alpha^2\sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1}\left[\left(\sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\overline{a}_{j_1\cdots j_n}W_{N_1}^{k_1j_1}\cdots W_{N_n}^{k_nj_n}\right)\right.\\
&\qquad\left.\cdot\left(\sum_{m_1=0}^{N_1-1}\cdots\sum_{m_n=0}^{N_n-1}b_{m_1\cdots m_n}W_{N_1}^{-k_1m_1}\cdots W_{N_n}^{-k_nm_n}\right)\right] \qquad (2.50)
\end{aligned}$$

A product of summations can be converted to summations of products as follows:

$$\left(\sum_{j_1}\cdots\sum_{j_n}x_{j_1\cdots j_n}\right)\left(\sum_{m_1}\cdots\sum_{m_n}y_{m_1\cdots m_n}\right)=\sum_{j_1}\cdots\sum_{j_n}\sum_{m_1}\cdots\sum_{m_n}x_{j_1\cdots j_n}y_{m_1\cdots m_n}$$

Therefore,

$$\begin{aligned}
(2.50) &= \alpha^2\sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1}\sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\sum_{m_1=0}^{N_1-1}\cdots\sum_{m_n=0}^{N_n-1}\\
&\qquad \overline{a}_{j_1\cdots j_n}b_{m_1\cdots m_n}W_{N_1}^{k_1j_1}\cdots W_{N_n}^{k_nj_n}W_{N_1}^{-k_1m_1}\cdots W_{N_n}^{-k_nm_n}\\
&= \alpha^2\sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1}\sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\sum_{m_1=0}^{N_1-1}\cdots\sum_{m_n=0}^{N_n-1}\\
&\qquad \overline{a}_{j_1\cdots j_n}b_{m_1\cdots m_n}W_{N_1}^{k_1(j_1-m_1)}\cdots W_{N_n}^{k_n(j_n-m_n)}\\
&= \alpha^2\sum_{j_1=0}^{N_1-1}\sum_{m_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\sum_{m_n=0}^{N_n-1}\\
&\qquad \overline{a}_{j_1\cdots j_n}b_{m_1\cdots m_n}\left(\sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1}W_{N_1}^{k_1(j_1-m_1)}\cdots W_{N_n}^{k_n(j_n-m_n)}\right)\\
&= \alpha^2\sum_{j_1=0}^{N_1-1}\sum_{m_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\sum_{m_n=0}^{N_n-1}\\
&\qquad \overline{a}_{j_1\cdots j_n}b_{m_1\cdots m_n}\left(\sum_{k_1=0}^{N_1-1}W_{N_1}^{k_1(j_1-m_1)}\right)\cdots\left(\sum_{k_n=0}^{N_n-1}W_{N_n}^{k_n(j_n-m_n)}\right) \qquad (2.51)
\end{aligned}$$

Lemma 2.1 tells that the following conversion is valid:

$$\sum_j \sum_m x_{\ldots jm \ldots} \sum_{k=0}^{N-1} W_N^{k(j-m)} = N \sum_j x_{\ldots jj \ldots}$$

Applying this relation to (2.51) gives

$$\begin{aligned} (2.51) \quad &= \quad \alpha^2 N_1 \cdots N_n \sum_{j_1=0}^{N_1-1} \cdots \sum_{j_n=0}^{N_n-1} \overline{a}_{j_1 \cdots j_n} b_{j_1 \cdots j_n} \\ &\equiv \quad \alpha^2 N_1 \cdots N_n \boldsymbol{a} \cdot \boldsymbol{b} \end{aligned} \tag{2.52}$$

Connecting (2.50) through (2.52) yields

$$\boldsymbol{A} \cdot \boldsymbol{B} = \alpha^2 N_1 \cdots N_n \boldsymbol{a} \cdot \boldsymbol{b} \tag{2.53}$$

Similarly, we get

$$\boldsymbol{C} \cdot \boldsymbol{D} = \alpha^2 N_1 \cdots N_n \boldsymbol{c} \cdot \boldsymbol{d} \tag{2.54}$$

Therefore,

$$\frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\boldsymbol{C} \cdot \boldsymbol{D}} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\boldsymbol{c} \cdot \boldsymbol{d}} \tag{2.55}$$

■

Now that the tools are ready, let us consider the equivalence of the two kinds of procedures introduced in Section 2.1.3. One procedure indirectly estimates the desired parameter by estimating the coefficients of the transform, and the other directly estimates the parameter by using the gradient filtered by the transform. Let us name the former $\mathbf{COEF}_{\mathrm{DFT}}$ and the latter $\mathbf{FILT}_{\mathrm{DFT}}$, which are described below.

$\mathbf{COEF}_{\mathrm{DFT}}$ **:** Estimate the Fourier coefficients $\boldsymbol{R}$ to indirectly estimate the real parameter $\boldsymbol{r} = \mathrm{DFT}^{-1}(\boldsymbol{R})$ by using the weighted gradient $\nabla_{\boldsymbol{R}}^{\mathcal{W}} H$ defined as follows:

$$\nabla_{\boldsymbol{R}}^{\mathcal{W}} H \triangleq \mathcal{W}(\nabla_{\boldsymbol{R}} H) \tag{2.56}$$

FILT$_{\text{DFT}}$ **:** Directly estimate the real parameter $\boldsymbol{r}$ using the frequency-filtered gradient $\nabla^{\text{flt}}_{\boldsymbol{r}} H$ defined as follows:

$$\nabla^{\text{flt}}_{\boldsymbol{r}} H \triangleq \text{DFT}^{-1}(\mathcal{W}(\text{DFT}(\nabla_{\boldsymbol{r}} H))) \tag{2.57}$$

In (2.56) and (2.57), $\mathcal{W}(\boldsymbol{X})$ multiplies each element of $\boldsymbol{X}$ by a certain real weight determined by the element's associated frequency. This $\mathcal{W}()$ is essential for controlling the resolution. In the step scheme, each weight is either 1 or 0 as a low-pass filter. In the weight scheme, the weights form a smooth attenuator.

In (2.57), the real-valued gradient $\nabla_{\boldsymbol{r}} H$ is transformed to the Fourier coefficient space, multiplied by certain real weights determined by frequency, and then transformed back to the parameter space to make the filtered gradient $\nabla^{\text{flt}}_{\boldsymbol{r}} H$. Since, in $\mathcal{W}()$, the transform coefficients that represent the same frequency are multiplied by the same real weight, the filtered gradient $\nabla^{\text{flt}}_{\boldsymbol{r}} H$ remains real.

Note that, as shown later by (3.19) in page 71, there is the following relation between the gradients in the two domains:

$$\nabla_{\boldsymbol{R}} H = \text{DFT}^{-1}(\nabla_{\boldsymbol{r}} H) \tag{2.58}$$

Theorem 2.4: COEF$_{\text{DFT}}$ *and* **FILT**$_{\text{DFT}}$ *produce identical estimates* $\boldsymbol{r}_j$ *of the real-valued parameter* $\boldsymbol{r}$ *on every local-search iteration* $j \geq 0$*, provided that (1) a local search is performed by the same type of the conjugate gradient method, i.e. either the Fletcher-Reeves or the Polak-Ribiere shown in (2.26), (2) they start with the same initial guess* $\boldsymbol{r}_0$*, (3) the same* $\mathcal{W}()$ *is employed, and (4) each line minimization is exact and no numerical errors are involved.*

Proof: First, we clarify some relations we are going to use. The weighting process $\mathcal{W}()$ multiplies real weights. Therefore, it is transparent in taking complex conjugate as follows:

$$\overline{\mathcal{W}(\boldsymbol{X})} = \mathcal{W}(\overline{\boldsymbol{X}}) \tag{2.59}$$

From the definitions (2.38) and (2.39), if $\boldsymbol{Y}$ is real, the complex conjugate of the inverse DFT of $\boldsymbol{Y}$ is equal to the DFT of $\boldsymbol{Y}$ multiplied by a positive real constant as follows:

$$\overline{\mathrm{DFT}^{-1}(\boldsymbol{Y})} = \gamma\mathrm{DFT}(\boldsymbol{Y}) \qquad \text{for real } \boldsymbol{Y} \tag{2.60}$$

where γ is a positive real constant. For example, $\gamma = \beta/\alpha$, if we define the DFT as (2.38) and the inverse DFT as (2.39).

We use the following notation.

In $\mathbf{COEF}_{\mathrm{DFT}}$

$\boldsymbol{R}^C$	:	the estimate of the Fourier coefficients
$\boldsymbol{d}^C$	:	the line search direction in the Fourier coefficient space
$\boldsymbol{r}^C$	:	the estimate of the real parameter $\left(\boldsymbol{r}^C \equiv \mathrm{DFT}^{-1}(\boldsymbol{R}^C)\right)$

In $\mathbf{FILT}_{\mathrm{DFT}}$

$\boldsymbol{r}^F$	:	the estimate of the real parameter
$\boldsymbol{d}^F$	:	the line search direction in the parameter space

If a subscript is attached to any of these, e.g. $\boldsymbol{R}^C_j$, then it means the value on the j-th line minimization. Also, we define $\boldsymbol{g}^C_j$ and $\boldsymbol{g}^F_j$ that are used as $\boldsymbol{g}_j$ of Section 2.2 as follows:

$$\begin{aligned} \boldsymbol{g}^C_j \triangleq \overline{\nabla^{\mathcal{W}}_{\boldsymbol{R}^C} H(\boldsymbol{R}^C_j)} &\equiv \overline{\mathcal{W}(\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}^C_j))} \\ &= \mathcal{W}(\overline{\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}^C_j)}) \end{aligned} \tag{2.61}$$

$$\boldsymbol{g}^F_j \triangleq \nabla^{\text{flt}}_{\boldsymbol{r}^F} H(\boldsymbol{r}^F_j) \equiv \mathrm{DFT}^{-1}(\mathcal{W}(\mathrm{DFT}(\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}^F_j)))) \tag{2.62}$$

where $\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}^C_j)$ is the complex gradient with respect to the Fourier coefficients $\boldsymbol{R}^C$ and $\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}^F_j)$ is the real gradient with respect to the parameter estimate $\boldsymbol{r}^F$.

We prove $\boldsymbol{r}^C_j = \boldsymbol{r}^F_j$ along with $\boldsymbol{d}^C_j = \gamma\mathrm{DFT}(\boldsymbol{d}^F_j)$ for all $j \geq 0$ by induction on j, the number of line minimizations.

1. When $j = 0$, the two procedures start with the same initial guess of the parameter:

$$\boldsymbol{r}_0^C = \boldsymbol{r}_0^F \tag{2.63}$$

and the line-search directions $\boldsymbol{d}_0^C$ and $\boldsymbol{d}_0^F$ are as follows:

$$\begin{aligned} \boldsymbol{d}_0^C &= -\boldsymbol{g}_0^C && (2.64) \\ \boldsymbol{d}_0^F &= -\boldsymbol{g}_0^F && (2.65) \end{aligned}$$

There is the following relation between $\boldsymbol{g}_0^C$ and $\boldsymbol{g}_0^F$:

$$\begin{aligned} \boldsymbol{g}_0^C &\equiv \mathcal{W}(\overline{\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}_0^C)}) \\ &= \mathcal{W}\left(\overline{\mathrm{DFT}^{-1}\left(\nabla_{\boldsymbol{r}^C} H(\boldsymbol{r}_0^C)\right)}\right) && \text{(use (2.58))} \\ &= \gamma\mathcal{W}\left(\mathrm{DFT}\left(\nabla_{\boldsymbol{r}^C} H(\boldsymbol{r}_0^C)\right)\right) && \text{(use (2.60))} \\ &= \gamma\mathcal{W}\left(\mathrm{DFT}\left(\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}_0^F)\right)\right) && \text{(since } \boldsymbol{r}_0^C = \boldsymbol{r}_0^F\text{)} \\ &= \gamma\mathrm{DFT}(\boldsymbol{g}_0^F) && \text{(use (2.62))} \qquad (2.66) \end{aligned}$$

Putting (2.64) and (2.65) into (2.66), we get:

$$\boldsymbol{d}_0^C = \gamma\mathrm{DFT}(\boldsymbol{d}_0^F) \tag{2.67}$$

2. Suppose the following relations hold for a certain $j \geq 0$:

$$\begin{aligned} \boldsymbol{r}_j^C &= \boldsymbol{r}_j^F && (2.68) \\ \boldsymbol{d}_j^C &= \gamma\mathrm{DFT}(\boldsymbol{d}_j^F) && (2.69) \end{aligned}$$

From (2.23) in page 21, $\boldsymbol{R}_{j+1}^C$ and $\boldsymbol{r}_{j+1}^F$ are determined by line minimization in the directions $\boldsymbol{d}_j^C$ and $\boldsymbol{d}_j^F$, respectively, as follows:

$$\begin{aligned} \boldsymbol{R}_{j+1}^C &= \boldsymbol{R}_j^C + \sigma_j^C \boldsymbol{d}_j^C && (2.70) \\ \boldsymbol{r}_{j+1}^F &= \boldsymbol{r}_j^F + \sigma_j^F \boldsymbol{d}_j^F && (2.71) \end{aligned}$$

Using (2.70),

$$\begin{aligned}\boldsymbol{r}^C_{j+1} &\equiv \mathrm{DFT}^{-1}(\boldsymbol{R}^C_{j+1}) \\ &= \mathrm{DFT}^{-1}(\boldsymbol{R}^C_j + \sigma^C_j \boldsymbol{d}^C_j) \\ &= \mathrm{DFT}^{-1}(\boldsymbol{R}^C_j) + \sigma^C_j \mathrm{DFT}^{-1}(\boldsymbol{d}^C_j) \\ &\equiv \boldsymbol{r}^C_j + \sigma^C_j \mathrm{DFT}^{-1}(\boldsymbol{d}^C_j) \qquad (2.72)\end{aligned}$$

From (2.69),

$$\mathrm{DFT}^{-1}(\boldsymbol{d}^C_j) = \gamma \boldsymbol{d}^F_j \qquad (2.73)$$

With this and (2.68), (2.72) becomes the following:

$$\boldsymbol{r}^C_{j+1} = \boldsymbol{r}^F_j + \gamma\sigma^C_j \boldsymbol{d}^F_j \qquad (2.74)$$

Comparing the right-hand sides of (2.74) and (2.71), we can see that both are the same line minimization in the direction $\boldsymbol{d}^F_j$ from $\boldsymbol{r}^F_j$. Therefore, they reach the same point, which results in $\sigma^F_j = \gamma\sigma^C_j$. Thus, the left-hand sides of (2.74) and (2.71) are equal to each other:

$$\boldsymbol{r}^C_{j+1} = \boldsymbol{r}^F_{j+1} \qquad (2.75)$$

From (2.24) in page 21, the next search directions $\boldsymbol{d}^C_{j+1}$ and $\boldsymbol{d}^F_{j+1}$ are determined as follows:

$$\begin{aligned}\boldsymbol{d}^C_{j+1} &= -\boldsymbol{g}^C_{j+1} + \tau^C_{j+1}\boldsymbol{d}^C_j \qquad (2.76) \\ \boldsymbol{d}^F_{j+1} &= -\boldsymbol{g}^F_{j+1} + \tau^F_{j+1}\boldsymbol{d}^F_j \qquad (2.77)\end{aligned}$$

where real coefficients τ^C_{j+1} and τ^F_{j+1} are calculated by either the Fletcher-Reeves formula or the Polak-Ribiere formula shown in (2.26), page 21. With (2.75), the following is obtained in a manner similar to (2.66).

$$\begin{aligned}\boldsymbol{g}^C_{j+1} &\equiv \mathcal{W}(\overline{\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}^C_{j+1})}) \\ &= \gamma\mathcal{W}\left(\mathrm{DFT}\left(\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}^F_{j+1})\right)\right) \\ &= \gamma\mathrm{DFT}(\boldsymbol{g}^F_{j+1}) \qquad (2.78)\end{aligned}$$

Similarly, with (2.68), we get:

$$\boldsymbol{g}_j^C = \gamma \mathrm{DFT}(\boldsymbol{g}_j^F) \tag{2.79}$$

Using (2.78), (2.79), and Theorem 2.3 in page 31, we get the following relations. If the Fletcher-Reeves formula is used,

$$\begin{aligned} \tau_{j+1}^C &= \frac{\boldsymbol{g}_{j+1}^C \cdot \boldsymbol{g}_{j+1}^C}{\boldsymbol{g}_j^C \cdot \boldsymbol{g}_j^C} \\ &= \frac{\mathrm{DFT}(\boldsymbol{g}_{j+1}^F) \cdot \mathrm{DFT}(\boldsymbol{g}_{j+1}^F)}{\mathrm{DFT}(\boldsymbol{g}_j^F) \cdot \mathrm{DFT}(\boldsymbol{g}_j^F)} \\ &= \frac{\boldsymbol{g}_{j+1}^F \cdot \boldsymbol{g}_{j+1}^F}{\boldsymbol{g}_j^F \cdot \boldsymbol{g}_j^F} \\ &= \tau_{j+1}^F \end{aligned} \tag{2.80}$$

If the Polak-Ribiere formula is used,

$$\begin{aligned} \tau_{j+1}^C &= \frac{\mathrm{Re}\Big((\boldsymbol{g}_{j+1}^C - \boldsymbol{g}_j^C) \cdot \boldsymbol{g}_{j+1}^C\Big)}{\boldsymbol{g}_j^C \cdot \boldsymbol{g}_j^C} \\ &= \frac{\mathrm{Re}\Big((\mathrm{DFT}(\boldsymbol{g}_{j+1}^F) - \mathrm{DFT}(\boldsymbol{g}_j^F)) \cdot \mathrm{DFT}(\boldsymbol{g}_{j+1}^F)\Big)}{\mathrm{DFT}(\boldsymbol{g}_j^F) \cdot \mathrm{DFT}(\boldsymbol{g}_j^F)} \\ &= \mathrm{Re}\left(\frac{\mathrm{DFT}(\boldsymbol{g}_{j+1}^F - \boldsymbol{g}_j^F) \cdot \mathrm{DFT}(\boldsymbol{g}_{j+1}^F)}{\mathrm{DFT}(\boldsymbol{g}_j^F) \cdot \mathrm{DFT}(\boldsymbol{g}_j^F)}\right) \\ &= \mathrm{Re}\left(\frac{(\boldsymbol{g}_{j+1}^F - \boldsymbol{g}_j^F) \cdot \boldsymbol{g}_{j+1}^F}{\boldsymbol{g}_j^F \cdot \boldsymbol{g}_j^F}\right) \\ &= \mathrm{Re}\Big(\tau_{j+1}^F\Big) \\ &= \tau_{j+1}^F \end{aligned} \tag{2.81}$$

We have established $\tau_{j+1}^C = \tau_{j+1}^F$ with both formulas. Using this relation along with (2.69), (2.76), (2.77) and (2.78), we get:

$$\begin{aligned} \boldsymbol{d}_{j+1}^C &= -\boldsymbol{g}_{j+1}^C + \tau_{j+1}^C \boldsymbol{d}_j^C \\ &= -\gamma \mathrm{DFT}(\boldsymbol{g}_{j+1}^F) + \tau_{j+1}^F \gamma \mathrm{DFT}(\boldsymbol{d}_j^F) \\ &= \gamma \mathrm{DFT}(-\boldsymbol{g}_{j+1}^F + \tau_{j+1}^F \boldsymbol{d}_j^F) \\ &= \gamma \mathrm{DFT}(\boldsymbol{d}_{j+1}^F) \end{aligned} \tag{2.82}$$

Therefore, as (2.75) and (2.82) show, if $\boldsymbol{r}_j^C = \boldsymbol{r}_j^F$ and $\boldsymbol{d}_j^C = \gamma\mathrm{DFT}(\boldsymbol{d}_j^F)$ hold for a certain $j \geq 0$, these relations also hold for $j+1$.

From 1. and 2. above, $\boldsymbol{r}_j^C = \boldsymbol{r}_j^F$ holds for all $j \geq 0$. ■

2.4 The Haar Wavelet Transform

This section introduces the *Haar wavelet transform* as another transform for multiresolution optimization. First, the Haar wavelet transform for a one-dimensional field, and then the transform for a general n-dimensional field are shown. Finally, it will be shown that, coupled with the conjugate gradient method, estimating the coefficients of the Haar wavelet transform of a distributed parameter is equivalent to directly estimating the parameter using the gradient filtered by the Haar wavelet transform.

2.4.1 For One-dimensional Field

The Haar wavelet transform uses the *Haar basis* (or the *Haar scaling function*) $\phi(x)$ and the *Haar wavelet* $\psi(x)$ defined as the following:

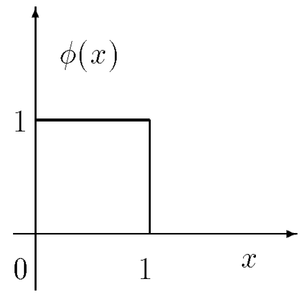


Figure 2.6: The Haar basis

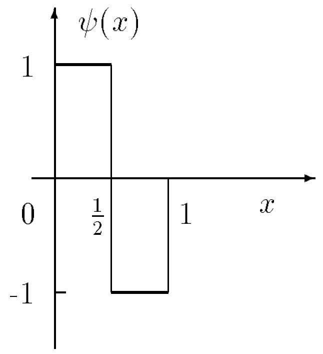


Figure 2.7: The Haar wavelet

$$\phi(x) = \begin{cases} 1 & \text{if } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.83}$$

$$\psi(x) = \begin{cases} 1 & \text{if } 0 \le x < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.84}$$

We define functions $\phi_j^M(x)$ and $\psi_j^M(x)$ from $\phi(x)$ and $\psi(x)$ as follows:

$$\phi_j^M(x) \triangleq \phi(2^M x - j) \tag{2.85}$$

$$\psi_j^M(x) \triangleq \psi(2^M x - j) \tag{2.86}$$

From (2.83) and (2.84), we can easily see:

$$\phi_j^M(x) = \begin{cases} 1 & \text{if } \frac{j}{2^M} \le x < \frac{j+1}{2^M} \\ 0 & \text{otherwise} \end{cases} \tag{2.87}$$

$$\psi_j^M(x) = \begin{cases} 1 & \text{if } \frac{2j}{2^{M+1}} \le x < \frac{2j+1}{2^{M+1}} \\ -1 & \text{if } \frac{2j+1}{2^{M+1}} \le x < \frac{2j+2}{2^{M+1}} \\ 0 & \text{otherwise} \end{cases} \tag{2.88}$$

As M increases by 1, the *scale* of $\phi_j^M(x)$ and $\psi_j^M(x)$ is halved. From (2.87) and (2.88), we get the *two-scale relations* of $\phi_j^M(x)$ and $\psi_j^M(x)$ as follows:

$$\phi_j^{M-1}(x) = \phi_{2j}^M(x) + \phi_{2j+1}^M(x) \tag{2.89}$$

$$\psi_j^{M-1}(x) = \phi_{2j}^M(x) - \phi_{2j+1}^M(x) \tag{2.90}$$

Inverting these relations yields:

$$\phi_{2j}^M(x) = \frac{1}{2}\left(\phi_j^{M-1}(x) + \psi_j^{M-1}(x)\right) \tag{2.91}$$

$$\phi_{2j+1}^M(x) = \frac{1}{2}\left(\phi_j^{M-1}(x) - \psi_j^{M-1}(x)\right) \tag{2.92}$$

Now, let $r(x)$ be a one-dimensional function defined in $0 \le x < 1$, and let $r^{M,0}(x)$ be the function representing $r(x)$ in resolution 2^M. We can express $r^{M,0}(x)$ with the scaling function $\phi_j^M(x)$ as follows:

$$r^{M,0}(x) = \sum_{j=0}^{2^M-1} R_j^{M,0}\phi_j^M(x) \tag{2.93}$$

With the relations (2.91) and (2.92), (2.93) becomes the following:

$$\begin{aligned}
&r^{M,0}(x)\\
&= \sum_{j=0}^{2^{M-1}-1}\left(R_{2j}^{M,0}\phi_{2j}^M(x) + R_{2j+1}^{M,0}\phi_{2j+1}^M(x)\right)\\
&= \sum_{j=0}^{2^{M-1}-1}\left(R_{2j}^{M,0}\cdot\frac{1}{2}(\phi_j^{M-1}(x)+\psi_j^{M-1}(x)) + R_{2j+1}^{M,0}\cdot\frac{1}{2}(\phi_j^{M-1}(x)-\psi_j^{M-1}(x))\right)\\
&= \sum_{j=0}^{2^{M-1}-1}\left(\frac{1}{2}(R_{2j}^{M,0}+R_{2j+1}^{M,0})\phi_j^{M-1}(x) + \frac{1}{2}(R_{2j}^{M,0}-R_{2j+1}^{M,0})\psi_j^{M-1}(x)\right) \qquad (2.94)
\end{aligned}$$

Let us define another function $r^{M,1}(x)$ as follows:

$$r^{M,1}(x) = \sum_{j=0}^{2^M-1} R_j^{M,1}\psi_j^M(x) \tag{2.95}$$

From the definitions (2.93) and (2.95),

$$r^{M-1,0}(x) + r^{M-1,1}(x) = \sum_{j=0}^{2^{M-1}-1}\left(R_j^{M-1,0}\phi_j^{M-1}(x) + R_j^{M-1,1}\psi_j^{M-1}(x)\right) \tag{2.96}$$

Therefore, comparing (2.94) and (2.96), we get the relation

$$r^{M,0}(x) = r^{M-1,0}(x) + r^{M-1,1}(x) \tag{2.97}$$

with

$$R_j^{M-1,0} = \frac{1}{2}(R_{2j}^{M,0} + R_{2j+1}^{M,0}) \qquad (0 \le j \le 2^{M-1}-1) \tag{2.98}$$

$$R_j^{M-1,1} = \frac{1}{2}(R_{2j}^{M,0} - R_{2j+1}^{M,0}) \qquad (0 \le j \le 2^{M-1}-1) \tag{2.99}$$

In other words, the function $r^{M-1,1}(x)$ is the difference between $r^{M,0}(x)$ and $r^{M-1,0}(x)$ which are the representations of $r(x)$ in the two resolutions. Inverting the relations (2.98) and (2.99) yields

$$R_{2j}^{M,0} = R_j^{M-1,0} + R_j^{M-1,1} \qquad (0 \le j \le 2^{M-1}-1) \tag{2.100}$$

$$R_{2j+1}^{M,0} = R_j^{M-1,0} - R_j^{M-1,1} \qquad (0 \le j \le 2^{M-1}-1) \tag{2.101}$$

The relations (2.98) and (2.99) can be combined into one equation as follows:

$$R_j^{M-1,s} = \frac{1}{2} \sum_{i \in \{2j, 2j+1\}} (-1)^{is} R_i^{M,0} \qquad (s \in \{0,1\},\ 0 \le j \le 2^{M-1}-1) \qquad (2.102)$$

This is the two-scale decomposition relation for decomposing the fine representation $R_i^{M,0}$ into the coarse representation $R_j^{M-1,0}$ and the difference information $R_j^{M-1,1}$. Similarly, the relations (2.100) and (2.101) can be combined into the following:

$$R_i^{M,0} = \sum_{s \in \{0,1\}} (-1)^{is} R_j^{M-1,s} \qquad (i \in \{2j, 2j+1\},\ 0 \le j \le 2^{M-1}-1) \qquad (2.103)$$

This is the two-scale reconstruction relation for reconstructing the fine representation $R_i^{M,0}$ from the coarse representation $R_j^{M-1,0}$ and the difference information $R_j^{M-1,1}$.

The two-scale decomposition and reconstruction relations (2.102) and (2.103) have factors $\frac{1}{2}$ and 1, respectively, in front of the summations, but these factors can be different with different definitions of $\phi_j^M(x)$ and $\psi_j^M(x)$. The general formulas can be written the following way. The general two-scale decomposition relation is

$$R_j^{M-1,s} = \alpha \sum_{i \in \{2j, 2j+1\}} (-1)^{is} R_i^{M,0} \qquad (s \in \{0,1\},\ 0 \le j \le 2^{M-1}-1) \qquad (2.104)$$

where α is a real constant. The general two-scale reconstruction relation is

$$R_i^{M,0} = \beta \sum_{s \in \{0,1\}} (-1)^{is} R_j^{M-1,s} \qquad (i \in \{2j, 2j+1\},\ 0 \le j \le 2^{M-1}-1) \qquad (2.105)$$

where β is a real constant. We show below that (2.104) and (2.105) are self-consistent as long as α and β have the relation: $\alpha\beta = 2^{-1}$ for the one-dimensional case.

We defined $r^{M,0}(x)$ in (2.93) as the function representing $r(x)$ in resolution 2^M. Here, we assume that the highest resolution for such representation is 2^{M_0} with some integer $M_0 \ge 0$. Then, we newly define $\phi_j^M(x)$ and $\psi_j^M(x)$ as follows:

$$\phi_j^M(x) \triangleq \beta^{M_0-M} \phi(2^M x - j) \qquad (2.106)$$

$$\psi_j^M(x) \triangleq \beta^{M_0-M} \psi(2^M x - j) \qquad (2.107)$$

These are designed so that, at the highest resolution 2^{M_0}, the factor β^{M_0-M} becomes unity and the array $\boldsymbol{R}^{M_0,0} = \{R_0^{M_0,0}, R_1^{M_0,0}, \cdots, R_{2^{M_0}-1}^{M_0,0}\}$ defined by (2.93) are actually the sampled values of $r(x)$ at 2^{M_0} points of equal intervals. The two-scale relations based on these new definitions are:

$$\phi_j^{M-1}(x) = \beta(\phi_{2j}^M(x) + \phi_{2j+1}^M(x)) \tag{2.108}$$

$$\psi_j^{M-1}(x) = \beta(\phi_{2j}^M(x) - \phi_{2j+1}^M(x)) \tag{2.109}$$

and

$$\phi_{2j}^M(x) = \frac{1}{2\beta}\left(\phi_j^{M-1}(x) + \psi_j^{M-1}(x)\right) \tag{2.110}$$

$$\phi_{2j+1}^M(x) = \frac{1}{2\beta}\left(\phi_j^{M-1}(x) - \psi_j^{M-1}(x)\right) \tag{2.111}$$

Then, $r^{M,0}(x)$ $(M \leq M_0)$ becomes the following (cf. (2.94)):

$$\begin{aligned}
&r^{M,0}(x) \\
&= \sum_{j=0}^{2^M-1} R_j^{M,0}\phi_j^M(x) \\
&= \sum_{j=0}^{2^{M-1}-1} \left(R_{2j}^{M,0}\phi_{2j}^M(x) + R_{2j+1}^{M,0}\phi_{2j+1}^M(x)\right) \\
&= \sum_{j=0}^{2^{M-1}-1} \left(R_{2j}^{M,0}\cdot\frac{1}{2\beta}(\phi_j^{M-1}(x) + \psi_j^{M-1}(x)) + R_{2j+1}^{M,0}\cdot\frac{1}{2\beta}(\phi_j^{M-1}(x) - \psi_j^{M-1}(x))\right) \\
&= \sum_{j=0}^{2^{M-1}-1} \left(\frac{1}{2\beta}(R_{2j}^{M,0} + R_{2j+1}^{M,0})\phi_j^{M-1}(x) + \frac{1}{2\beta}(R_{2j}^{M,0} - R_{2j+1}^{M,0})\psi_j^{M-1}(x)\right)
\end{aligned} \tag{2.112}$$

Comparing (2.112) with (2.96) yields

$$R_j^{M-1,0} = \frac{1}{2\beta}(R_{2j}^{M,0} + R_{2j+1}^{M,0}) \qquad (0 \leq j \leq 2^{M-1}-1) \tag{2.113}$$

$$R_j^{M-1,1} = \frac{1}{2\beta}(R_{2j}^{M,0} - R_{2j+1}^{M,0}) \qquad (0 \leq j \leq 2^{M-1}-1) \tag{2.114}$$

in order to have the relation (2.97). From (2.113) and (2.114), we get

$$R_{2j}^{M,0} = \beta(R_j^{M-1,0} + R_j^{M-1,1}) \qquad (0 \leq j \leq 2^{M-1}-1) \tag{2.115}$$

$$R_{2j+1}^{M,0} = \beta(R_j^{M-1,0} - R_j^{M-1,1}) \qquad (0 \leq j \leq 2^{M-1}-1) \tag{2.116}$$

$$\begin{tikzcd}
\boldsymbol{R}^{M_0,0} \arrow[r, "\mathcal{H}_{M_0-1}"] \arrow[dr] & \boldsymbol{R}^{M_0-1,0} \arrow[r, "\mathcal{H}_{M_0-2}"] \arrow[dr] & \cdots \arrow[r, "\mathcal{H}_1"] \arrow[dr] & \boldsymbol{R}^{1,0} \arrow[r, "\mathcal{H}_0"] \arrow[dr] & \boldsymbol{R}^{0,0} \\
& \boldsymbol{R}^{M_0-1,1} & & \boldsymbol{R}^{1,1} & \boldsymbol{R}^{0,1}
\end{tikzcd}$$

Figure 2.8: One-dimensional Haar wavelet transform

$$\begin{tikzcd}
\boldsymbol{R}^{M_0,0} & \boldsymbol{R}^{M_0-1,0} \arrow[l, "\mathcal{H}^{-1}_{M_0-1}"'] & \cdots \arrow[l, "\mathcal{H}^{-1}_{M_0-2}"'] & \boldsymbol{R}^{1,0} \arrow[l, "\mathcal{H}^{-1}_{1}"'] & \boldsymbol{R}^{0,0} \arrow[l, "\mathcal{H}^{-1}_{0}"'] \\
& \boldsymbol{R}^{M_0-1,1} \arrow[ul] & {} \arrow[ul] & \boldsymbol{R}^{1,1} \arrow[ul] & \boldsymbol{R}^{0,1} \arrow[ul]
\end{tikzcd}$$

Figure 2.9: One-dimensional inverse Haar wavelet transform

The two-scale decomposition relations (2.113) and (2.114) and the two-scale reconstruction relations (2.115) and (2.116) become (2.104) and (2.105), respectively, with $\alpha\beta = 2^{-1}$.

Let us define $\mathcal{H}_{M-1}$ as the two-scale decomposition process defined by (2.104), and its inverse $\mathcal{H}^{-1}_{M-1}$ as the two-scale reconstruction process defined by (2.105). Namely,

$$\boldsymbol{R}^{M,0} \quad \underset{\mathcal{H}^{-1}_{M-1}}{\overset{\mathcal{H}_{M-1}}{\rightleftarrows}} \quad \boldsymbol{R}^{M-1,0}, \boldsymbol{R}^{M-1,1} \tag{2.117}$$

Then, the one-dimensional Haar wavelet transform is calculated as shown in Figure 2.8. The initial coefficient array $\boldsymbol{R}^{M_0,0}$ is set to the one-dimensional distributed parameter $\boldsymbol{r} \equiv \{r_0, r_1, \cdots, r_{(2^{M_0}-1)}\}$, thus $\boldsymbol{R}^{M_0,0} = \boldsymbol{r}$. In each step, the array $\boldsymbol{R}^{M,0}$ with 2^M elements is decomposed by $\mathcal{H}_{M-1}$ into the two arrays $\boldsymbol{R}^{M-1,0}$ and $\boldsymbol{R}^{M-1,1}$ each of which has 2^{M-1} elements. The array $\boldsymbol{R}^{M-1,0}$, the coarse representation of $\boldsymbol{r}$, is further decomposed in the next step, and the difference information $\boldsymbol{R}^{M-1,1}$ is kept as a part of the final array $\boldsymbol{R}$. The decomposition process continues until $\boldsymbol{R}^{0,0}$ and

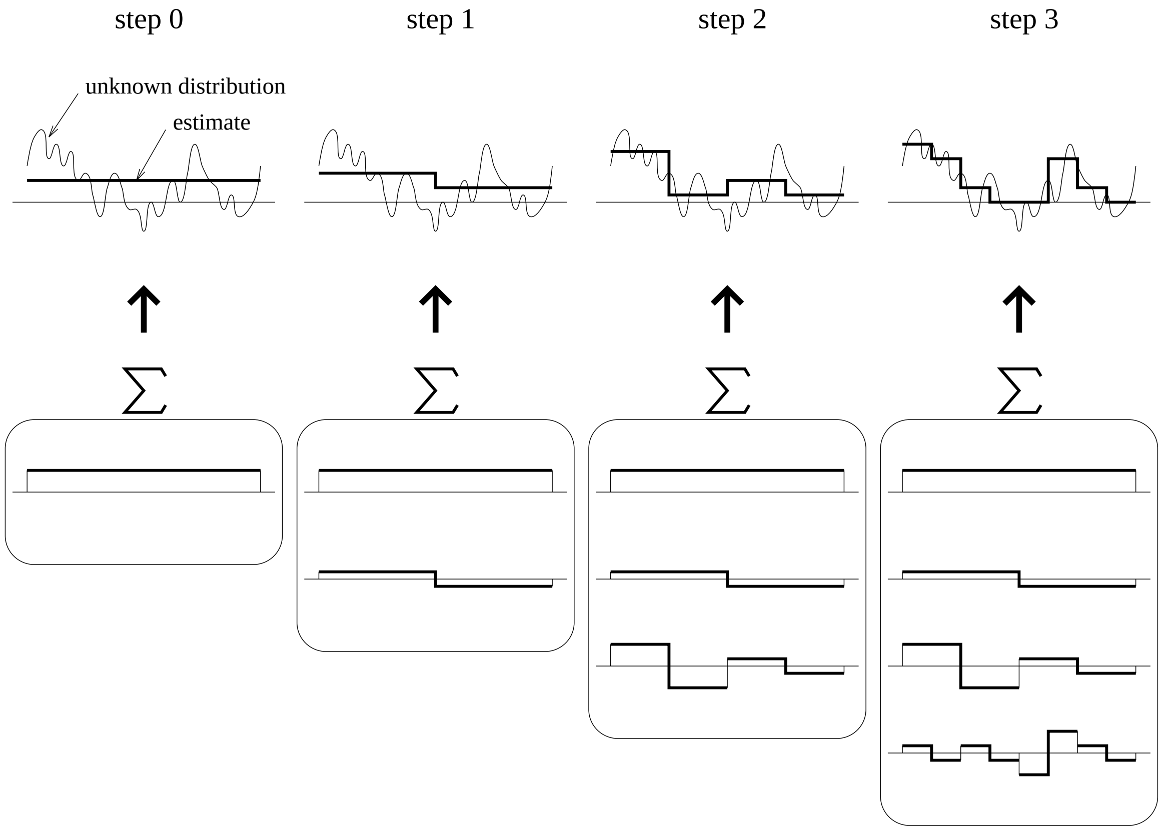


Figure 2.10: Multiresolution estimation with Haar wavelet transform

$\boldsymbol{R}^{0,1}$ are produced.

The final array $\boldsymbol{R}$, which we call the coefficients of the Haar wavelet transform, consists of $\boldsymbol{R}^{0,0}$ and all the difference information $\{\boldsymbol{R}^{M,1}\}_{0 \leq M \leq M_0 - 1}$. Thus,

$$\boldsymbol{R} \triangleq \left\{ \boldsymbol{R}^{0,0}, \{\boldsymbol{R}^{M,1}\}_{0 \leq M \leq M_0 - 1} \right\} \tag{2.118}$$

Figure 2.9 is the one-dimensional inverse Haar wavelet transform that is the reverse process of Figure 2.8.

Figure 2.10 shows the step scheme of multiresolution estimation with the one-dimensional Haar wavelet transform. In step 0, the array $\boldsymbol{R}^{0,0}$, which has only one element, is estimated. In step 1, both array $\boldsymbol{R}^{0,0}$ and array $\boldsymbol{R}^{0,1}$ are estimated. In this figure, it might look unnecessary to include the first array $\boldsymbol{R}^{0,0}$ in the search in step 1 because it is already estimated in step 0. In reality, however, it is necessary because the mathematical model on which the cost is calculated is a nonlinear process. In step 2, $\boldsymbol{R}^{0,0}, \boldsymbol{R}^{0,1}$ and $\boldsymbol{R}^{1,1}$ are estimated, and so on. In general, step M estimates $\boldsymbol{R}^{0,0}$ and the difference information $\boldsymbol{R}^{0,1}, \cdots, \boldsymbol{R}^{M-1,1}$ to reconstruct the parameter distribution in resolution 2^M.

2.4.2 For n-dimensional Field

Although Daubechies [6] shows construction of the two-dimensional wavelet transform, higher-dimensional wavelet transforms are referred to as simply "analogous". In this section, we define a general n-dimensional Haar wavelet transform "analogous" to her two-dimensional wavelet structure.

It is assumed that the parameter $\boldsymbol{r}$ has the same number N of real elements in each of the n coordinates, thus $\boldsymbol{r} \in \mathcal{R}^{N^n}$, and that $N = 2^{M_0}$ with some integer $M_0 \geq 0$. It turns out that the transform coefficient array $\boldsymbol{R}$ fits in the same shape as the parameter $\boldsymbol{r}$, so $\boldsymbol{R} \in \mathcal{R}^{N^n}$. Therefore, both $\boldsymbol{r}$ and $\boldsymbol{R}$ can be viewed as an n-dimensional cube with edges of length $N = 2^{M_0}$.

An n-dimensional field has n coordinates $(x_1, \cdots, x_n)$, and each coordinate can take either the one-dimensional scaling function $\phi()$ (2.83) to produce the "average" or the one-dimensional wavelet $\psi()$ (2.84) to produce the difference information at a time. Hence, there are 2^n ways of obtaining information at each resolution. We create one scaling function $\Phi(x_1, \cdots, x_n)$ and $2^n - 1$ wavelets $\Psi^{s_1 \cdots s_n}(x_1, \cdots, x_n)$ for the n-dimensional Haar wavelet transform as follows:

$$\Phi(x_1, \cdots, x_n) \triangleq \prod_{t=1}^{n} \phi(x_t) \tag{2.119}$$

$$\Psi^{s_1 \cdots s_n}(x_1, \cdots, x_n) \triangleq \prod_{t=1}^{n} \left(\delta_{s_t,0}\phi(x_t) + \delta_{s_t,1}\psi(x_t)\right) \tag{2.120}$$

where $s_1 \cdots s_n$ for the wavelet $\Psi^{s_1 \cdots s_n}(x_1, \cdots, x_n)$ is such that $s_m \in \{0,1\}$ $(m = 1, \cdots, n)$, $s_1 \cdots s_n \neq 0 \cdots 0$(all zero), and $\delta_{i,j}$ is Kronecker's delta such that $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$. In this notation, $s_1 \cdots s_n$ can be viewed as n "switches": if $s_m = 0$, the scaling function $\phi(x_m)$ is taken, and if $s_m = 1$, the wavelet $\psi(x_m)$ is taken.

For the one-dimensional Haar wavelet transform, we defined $\boldsymbol{R}^{M,0}$ as the coefficient array that corresponds to $\phi()$, and $\boldsymbol{R}^{M,1}$ as the coefficient array that corresponds to $\psi()$. In an analogous manner, for the n-dimensional Haar wavelet transform, we define $\boldsymbol{R}^{M,0}$ as the coefficient array that corresponds to $\Phi()$, and $\boldsymbol{R}^{M,s_1 \cdots s_n}$ $(s_1 \cdots s_n \neq 0 \cdots 0)$ as the coefficient array that corresponds to $\Psi^{s_1 \cdots s_n}()$. For convenience, we allow $s_1 \cdots s_n$ to be all zeros when appropriate and define $\boldsymbol{R}^{M,0 \cdots 0} \equiv \boldsymbol{R}^{M,0}$. Then, the two-scale decomposition relation is defined as

$$R^{M-1,s_1 \cdots s_n}_{i_1,\cdots,i_n} \triangleq \alpha \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2i_m, 2i_m+1\}}} (-1)^{\sum_{t=1}^n j_t s_t} R^{M,0}_{j_1,\cdots,j_n} \qquad (0 \leq i_m \leq 2^{M-1} - 1) \quad (2.121)$$

in which α is a real constant. The two-scale reconstruction relation is defined as

$$R^{M,0}_{j_1,\cdots,j_n} = \beta \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n j_t s_t} R^{M-1,s_1 \cdots s_n}_{i_1,\cdots,i_n} \qquad (j_m \in \{2i_m, 2i_m + 1\}) \qquad (2.122)$$

in which β is a real constant.

Let $\mathcal{H}_{M-1}$ denote the two-scale decomposition process defined by (2.121) and $\mathcal{H}^{-1}_{M-1}$ denote the two-scale reconstruction process defined by (2.122). Namely,

$$\boldsymbol{R}^{M,0} \ \underset{\mathcal{H}^{-1}_{M-1}}{\overset{\mathcal{H}_{M-1}}{\rightleftarrows}} \ \boldsymbol{R}^{M-1,0}, \{\boldsymbol{R}^{M-1,s_1 \cdots s_n}\}_{\substack{s_m \in \{0,1\} \\ s_1 \cdots s_n \neq 0 \cdots 0}} \qquad (2.123)$$

The array on the left-hand side of (2.123) has $(2^M)^n = 2^{nM}$ elements. Each of the 2^n arrays on the right-hand side has $2^{n(M-1)}$ elements, the total being $2^n \cdot 2^{n(M-1)} = 2^{nM}$.

$$\begin{tikzcd}
\boldsymbol{R}^{M_0,0} \arrow[r, "\mathcal{H}_{M_0-1}"] \arrow[dr] & \boldsymbol{R}^{M_0-1,0} \arrow[r, "\mathcal{H}_{M_0-2}"] \arrow[dr] & \cdots \arrow[r, "\mathcal{H}_1"] \arrow[dr] & \boldsymbol{R}^{1,0} \arrow[r, "\mathcal{H}_0"] \arrow[dr] & \boldsymbol{R}^{0,0} \\
& \{\boldsymbol{R}^{M_0-1,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} & & \{\boldsymbol{R}^{1,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} & \{\boldsymbol{R}^{0,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}}
\end{tikzcd}$$

Figure 2.11: n-dimensional Haar wavelet transform

$$\begin{tikzcd}
\boldsymbol{R}^{M_0,0} & \boldsymbol{R}^{M_0-1,0} \arrow[l, "\mathcal{H}^{-1}_{M_0-1}"'] & \cdots \arrow[l, "\mathcal{H}^{-1}_{M_0-2}"'] & \boldsymbol{R}^{1,0} \arrow[l, "\mathcal{H}^{-1}_1"'] & \boldsymbol{R}^{0,0} \arrow[l, "\mathcal{H}^{-1}_0"'] \\
& \{\boldsymbol{R}^{M_0-1,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \arrow[ul] & {} \arrow[ul] & \{\boldsymbol{R}^{1,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \arrow[ul] & \{\boldsymbol{R}^{0,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \arrow[ul]
\end{tikzcd}$$

Figure 2.12: n-dimensional inverse Haar wavelet transform

Therefore, the amount of data is conserved through the two-scale decomposition and reconstruction processes of (2.123).

Figure 2.11 shows the calculation of the n-dimensional Haar wavelet transform. The initial array $\boldsymbol{R}^{M_0,0}$ is set to the n-dimensional distributed parameter $\boldsymbol{r}$. In each step, $\mathcal{H}_{M-1}$ decomposes $\boldsymbol{R}^{M,0}$ into $\boldsymbol{R}^{M-1,0}$, which is the coarse representation of $\boldsymbol{r}$, and 2^n-1 arrays $\{\boldsymbol{R}^{M-1,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}}$, which have the difference information between $\boldsymbol{R}^{M,0}$ and $\boldsymbol{R}^{M-1,0}$. The final array $\boldsymbol{R}$, which we call the coefficients of the Haar wavelet transform, consists of $\boldsymbol{R}^{0,0}$ and all the difference information $\{\boldsymbol{R}^{M,s_1\cdots s_n}\}_{\substack{0\le M\le M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}}$. Thus,

$$\boldsymbol{R} \triangleq \left\{ \boldsymbol{R}^{0,0},\ \{\boldsymbol{R}^{M,s_1\cdots s_n}\}_{\substack{0\le M\le M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \tag{2.124}$$

The n-dimensional inverse Haar wavelet transform is the reconstruction process shown in Figure 2.12.

As the following theorem shows, the two-scale decomposition and reconstruction in (2.123), and as a result the transform and the inverse transform in Figures 2.11 and 2.12, are inverse to each other, as long as real constants α and β in (2.121) and (2.122) have the relation: $\alpha\beta = 2^{-n}$.

Theorem 2.5: *The two-scale Haar reconstruction defined by (2.122) and the two-scale Haar decomposition defined by (2.121) are inverse to each other with $\alpha\beta = 2^{-n}$.*

Proof: Let $\boldsymbol{R}^{M,0}$ be decomposed by (2.121) and reconstructed back by (2.122). Let $\boldsymbol{Q}^{M,0}$ denote the reconstructed array. In other words, we examine the relation between $\boldsymbol{R}^{M,0}$ and $\boldsymbol{Q}^{M,0}$ defined by

$$\boldsymbol{Q}^{M,0} = \mathcal{H}_{M-1}^{-1}(\mathcal{H}_{M-1}(\boldsymbol{R}^{M,0})) \tag{2.125}$$

From (2.121), $\boldsymbol{R}^{M,0}$ is decomposed as

$$R_{i_1,\cdots,i_n}^{M-1,s_1\cdots s_n} \triangleq \alpha \sum_{\substack{k_1,\cdots,k_n \\ k_m \in \{2i_m, 2i_m+1\}}} (-1)^{\sum_{t=1}^n k_t s_t} R_{k_1,\cdots,k_n}^{M,0} \qquad (0 \le i_m \le 2^{M-1}-1) \tag{2.126}$$

From (2.122), the reconstruction of $\boldsymbol{Q}^{M,0}$ is performed as

$$Q_{j_1,\cdots,j_n}^{M,0} = \beta \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n j_t s_t} R_{i_1,\cdots,i_n}^{M-1,s_1\cdots s_n} \qquad (j_m \in \{2i_m, 2i_m+1\}) \tag{2.127}$$

Substituting (2.126) into (2.127) yields the following ($j_m \in \{2i_m, 2i_m+1\}$):

$$\begin{aligned} Q_{j_1,\cdots,j_n}^{M,0} &= \beta \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n j_t s_t} \left(\alpha \sum_{\substack{k_1,\cdots,k_n \\ k_m \in \{2i_m, 2i_m+1\}}} (-1)^{\sum_{h=1}^n k_h s_h} R_{k_1,\cdots,k_n}^{M,0} \right) \\ &= \alpha\beta \sum_{\substack{k_1,\cdots,k_n \\ k_m \in \{2i_m, 2i_m+1\}}} R_{k_1,\cdots,k_n}^{M,0} \left(\sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n (j_t+k_t)s_t} \right) \end{aligned} \tag{2.128}$$

If $j_a \neq k_a$ for some integer a ($1 \le a \le n$), then $(j_a + k_a)$ is odd since $j_a, k_a \in \{2i_a, 2i_a+1\}$, and the following results.

$$\begin{aligned}
&\sum_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} && (2.129)\\
&= \sum_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} \left((-1)^{(j_a+k_a)s_a} \cdot (-1)^{\sum_{\substack{t=1\\t\neq a}}^{n}(j_t+k_t)s_t} \right)\\
&= \sum_{\substack{s_1,\cdots,s_{a-1},s_{a+1},\cdots,s_n\\ s_m\in\{0,1\}}} \left(\sum_{s_a\in\{0,1\}} (-1)^{(j_a+k_a)s_a} \cdot (-1)^{\sum_{\substack{t=1\\t\neq a}}^{n}(j_t+k_t)s_t} \right)\\
&= \sum_{\substack{s_1,\cdots,s_{a-1},s_{a+1},\cdots,s_n\\ s_m\in\{0,1\}}} \left((-1)^{\sum_{\substack{t=1\\t\neq a}}^{n}(j_t+k_t)s_t} - (-1)^{\sum_{\substack{t=1\\t\neq a}}^{n}(j_t+k_t)s_t} \right)\\
&= 0 && (2.130)
\end{aligned}$$

Therefore,

$$\sum_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} = \begin{cases} 2^n & \text{if } j_t = k_t \text{ for all } t \\ 0 & \text{otherwise} \end{cases} \tag{2.131}$$

Applying this to (2.128), we get the following:

$$Q^{M,0}_{j_1,\cdots,j_n} = 2^n \alpha\beta R^{M,0}_{j_1,\cdots,j_n} \tag{2.132}$$

Therefore, with $\alpha\beta = 2^{-n}$, the reconstructed array $\boldsymbol{Q}^{M,0}$ is equal to the original array $\boldsymbol{R}^{M,0}$, that is:

$$\boldsymbol{R}^{M,0} = \mathcal{H}^{-1}_{M-1}(\mathcal{H}_{M-1}(\boldsymbol{R}^{M,0})) \tag{2.133}$$

for any $\boldsymbol{R}^{M,0}$.

Now, let us consider the opposite way. Take any n-dimensional array $\boldsymbol{S} \in \mathcal{R}^{2^{nM}}$. The array $\boldsymbol{S}$ can be partitioned into 2^n arrays each of which has $2^{n(M-1)}$ elements, since the total number of elements is $2^n \cdot 2^{n(M-1)} = 2^{nM}$. Let each "small" array be $\boldsymbol{R}^{M-1,s_1\cdots s_n}$ such that

$$\boldsymbol{S} \equiv \{\boldsymbol{R}^{M-1,s_1\cdots s_n}\}_{s_m\in\{0,1\}} \tag{2.134}$$

Now we calculate a set of arrays $\{\boldsymbol{T}^{M-1,s_1\cdots s_n}\}_{s_m\in\{0,1\}}$ as follows:

$$\{\boldsymbol{T}^{M-1,s_1\cdots s_n}\}_{s_m\in\{0,1\}} = \mathcal{H}_{M-1}(\mathcal{H}^{-1}_{M-1}(\{\boldsymbol{R}^{M-1,s_1\cdots s_n}\}_{s_m\in\{0,1\}})) \tag{2.135}$$

From the definition of the two-scale reconstruction (2.122),

$$R^{M,0}_{j_1,\cdots,j_n} = \beta \sum_{\substack{d_1,\cdots,d_n \\ d_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n j_t d_t} R^{M-1,d_1\cdots d_n}_{k_1,\cdots,k_n} \qquad (j_m \in \{2k_m, 2k_m+1\}) \tag{2.136}$$

From the definition of the two-scale decomposition (2.121),

$$T^{M-1,s_1\cdots s_n}_{k_1,\cdots,k_n} = \alpha \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} (-1)^{\sum_{t=1}^n j_t s_t} R^{M,0}_{j_1,\cdots,j_n} \qquad (0 \le k_m \le 2^{M-1}-1) \tag{2.137}$$

Substituting (2.136) into (2.137) yields the following:

$$\begin{aligned} T^{M-1,s_1\cdots s_n}_{k_1,\cdots,k_n} &= \alpha \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} (-1)^{\sum_{t=1}^n j_t s_t} \left(\beta \sum_{\substack{d_1,\cdots,d_n \\ d_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n j_t d_t} R^{M-1,d_1\cdots d_n}_{k_1,\cdots,k_n} \right) \\ &= \alpha\beta \sum_{\substack{d_1,\cdots,d_n \\ d_m \in \{0,1\}}} R^{M-1,d_1\cdots d_n}_{k_1,\cdots,k_n} \left(\sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} (-1)^{\sum_{t=1}^n j_t (s_t+d_t)} \right) \end{aligned} \tag{2.138}$$

If $s_a \neq d_a$ for some integer a $(1 \le a \le n)$, then $(s_a + d_a) = 1$ since $s_a, d_a \in \{0,1\}$, and the following results:

$$\sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} (-1)^{\sum_{t=1}^n j_t (s_t+d_t)} \tag{2.139}$$

$$\begin{aligned} &= \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} \left((-1)^{j_a(s_a+d_a)} \cdot (-1)^{\sum_{\substack{t=1 \\ t\neq a}}^n j_t(s_t+d_t)} \right) \\ &= \sum_{\substack{j_1,\cdots,j_{a-1},j_{a+1},\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} \left(\sum_{j_a \in \{2k_a, 2k_a+1\}} (-1)^{j_a} \cdot (-1)^{\sum_{\substack{t=1 \\ t\neq a}}^n j_t(s_t+d_t)} \right) \\ &= \sum_{\substack{j_1,\cdots,j_{a-1},j_{a+1},\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} \left((-1)^{\sum_{\substack{t=1 \\ t\neq a}}^n j_t(s_t+d_t)} - (-1)^{\sum_{\substack{t=1 \\ t\neq a}}^n j_t(s_t+d_t)} \right) \\ &= 0 \end{aligned} \tag{2.140}$$

Therefore,

$$\sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2k_m, 2k_m+1\}}} (-1)^{\sum_{t=1}^n j_t (s_t+d_t)} = \begin{cases} 2^n & \text{if } s_t = d_t \text{ for all } t \\ 0 & \text{otherwise} \end{cases} \tag{2.141}$$

Applying this to (2.138) yields:

$$T_{k_1,\cdots,k_n}^{M-1,s_1\cdots s_n} = 2^n \alpha\beta R_{k_1,\cdots,k_n}^{M-1,s_1\cdots s_n} \tag{2.142}$$

With $\alpha\beta = 2^{-n}$, we get:

$$\boldsymbol{T}^{M-1,s_1\cdots s_n} = \boldsymbol{R}^{M-1,s_1\cdots s_n} \tag{2.143}$$

for all $s_m \in \{0,1\}$. Then, from (2.134) and (2.135), we get:

$$\boldsymbol{S} = \mathcal{H}_{M-1}(\mathcal{H}_{M-1}^{-1}(\boldsymbol{S})) \tag{2.144}$$

Therefore, the two-scale decomposition $\mathcal{H}_{M-1}()$ and the two-scale reconstruction $\mathcal{H}_{M-1}^{-1}()$ are inverse to each other. ■

Regardless of the dimension of the field, we use the following notation for the Haar wavelet transform and the inverse transform between a distributed parameter $\boldsymbol{r}$ and its Haar wavelet coefficients $\boldsymbol{R}$.

$$\boldsymbol{R} = \mathrm{Haar}(\boldsymbol{r}) \tag{2.145}$$

$$\boldsymbol{r} = \mathrm{Haar}^{-1}(\boldsymbol{R}) \tag{2.146}$$

The dimension of the transforms Haar() and Haar^{-1}() is assumed to be the same as that of the field over which the parameter $\boldsymbol{r}$ is distributed.

As seen from (2.121) and (2.122), Haar() and Haar^{-1}() are linear operators such that

$$\mathrm{Haar}(c_1\boldsymbol{r}_1 + c_2\boldsymbol{r}_2) = c_1\mathrm{Haar}(\boldsymbol{r}_1) + c_2\mathrm{Haar}(\boldsymbol{r}_2) \tag{2.147}$$

$$\mathrm{Haar}^{-1}(c_1\boldsymbol{R}_1 + c_2\boldsymbol{R}_2) = c_1\mathrm{Haar}^{-1}(\boldsymbol{R}_1) + c_2\mathrm{Haar}^{-1}(\boldsymbol{R}_2) \tag{2.148}$$

with any real numbers c_1 and c_2.

2.4.3 Filtering the Gradient

In this section, we prove that the two methods of estimation considered in Section 2.1.3, namely the indirect estimation of the parameter by estimating the coefficients of the transform and the direct estimation of the parameter using the gradient filtered by the transform, are actually equivalent with the Haar wavelet transform. We start with a lemma used for proving the next theorem.

Lemma 2.2: *Let* $\boldsymbol{A}^{M,0}$ *and* $\boldsymbol{B}^{M,0}$ *be n-dimensional real arrays each of which is decomposed by the two-scale decomposition* $\mathcal{H}_{M-1}$ *defined by (2.123) and (2.121). Then,*

$$\mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}) \cdot \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}) = 2^n \alpha^2 \boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0}$$

Proof:

$$\begin{aligned}
&\mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}) \cdot \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}) \\
&= \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} \boldsymbol{A}^{M-1,s_1\cdots s_n} \cdot \boldsymbol{B}^{M-1,s_1\cdots s_n} \\
&= \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} \sum_{\substack{i_1,\cdots,i_n \\ 0\le i_m<2^{M-1}}} A^{M-1,s_1\cdots s_n}_{i_1,\cdots,i_n} B^{M-1,s_1\cdots s_n}_{i_1,\cdots,i_n} \\
&= \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} \sum_{\substack{i_1,\cdots,i_n \\ 0\le i_m<2^{M-1}}} \left(\alpha \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2i_m, 2i_m+1\}}} (-1)^{\sum_{t=1}^n j_t s_t} A^{M,0}_{j_1,\cdots,j_n} \right) \\
&\qquad \cdot \left(\alpha \sum_{\substack{k_1,\cdots,k_n \\ k_m \in \{2i_m, 2i_m+1\}}} (-1)^{\sum_{t=1}^n k_t s_t} B^{M,0}_{k_1,\cdots,k_n} \right) \\
&= \alpha^2 \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} \sum_{\substack{i_1,\cdots,i_n \\ 0\le i_m<2^{M-1}}} \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2i_m, 2i_m+1\}}} \sum_{\substack{k_1,\cdots,k_n \\ k_m \in \{2i_m, 2i_m+1\}}} (-1)^{\sum_{t=1}^n (j_t+k_t)s_t} A^{M,0}_{j_1,\cdots,j_n} B^{M,0}_{k_1,\cdots,k_n} \\
&= \alpha^2 \sum_{\substack{i_1,\cdots,i_n \\ 0\le i_m<2^{M-1}}} \sum_{\substack{j_1,\cdots,j_n \\ j_m \in \{2i_m, 2i_m+1\}}} \sum_{\substack{k_1,\cdots,k_n \\ k_m \in \{2i_m, 2i_m+1\}}} A^{M,0}_{j_1,\cdots,j_n} B^{M,0}_{k_1,\cdots,k_n} \sum_{\substack{s_1,\cdots,s_n \\ s_m \in \{0,1\}}} (-1)^{\sum_{t=1}^n (j_t+k_t)s_t}
\end{aligned} \tag{2.149}$$

The last summation $\sum\limits_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t}$ is calculated as follows. When $j_t = k_t$ for all t $(1 \le t \le n)$, all $(j_t + k_t)$ are even, and therefore $(-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} = 1$ regardless of whether each s_t is 0 or 1. Therefore, $\sum\limits_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} = 2^n$.

When $j_t \neq k_t$ for some t, suppose $t = h$ is such t, namely $j_h \neq k_h$. The summation can be rewritten with a summation with respect to all s_m $(1 \le m \le n)$ except s_h and a summation with respect to s_h as follows:

$$\sum_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} = \sum_{\substack{s_1,\cdots,s_{h-1},s_{h+1},\cdots,s_n\\ s_m\in\{0,1\}}} \left(\sum_{s_h=0}^{1} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} \right) \tag{2.150}$$

Since $j_h, k_h \in \{2i_m, 2i_m + 1\}$, $(j_h + k_h)$ is odd from $j_h \neq k_h$. So, $(j_h + k_h)s_h$ is even when $s_h = 0$ and odd when $s_h = 1$. Hence,

$$\sum_{s_h=0}^{1} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} = 0 \tag{2.151}$$

Therefore, $\sum\limits_{\substack{s_1,\cdots,s_n\\ s_m\in\{0,1\}}} (-1)^{\sum_{t=1}^{n}(j_t+k_t)s_t} = 0$ when $j_t \neq k_t$ for some t.

From these results, we have to count only the cases where $j_t = k_t$ for all t in (2.149) as follows:

$$\begin{aligned}
(2.149) &= 2^n\alpha^2 \sum_{\substack{i_1,\cdots,i_n\\ 0\le i_m<2^{M-1}}} \sum_{\substack{j_1,\cdots,j_n\\ j_m\in\{2i_m,2i_m+1\}}} A^{M,0}_{j_1,\cdots,j_n} B^{M,0}_{j_1,\cdots,j_n} \\
&= 2^n\alpha^2 \sum_{\substack{i_1,\cdots,i_n\\ 0\le i_m<2^{M}}} A^{M,0}_{i_1,\cdots,i_n} B^{M,0}_{i_1,\cdots,i_n} \\
&\equiv 2^n\alpha^2 \boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0}
\end{aligned} \tag{2.152}$$

Connecting (2.149) and (2.152) gives

$$\mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}) \cdot \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}) = 2^n\alpha^2 \boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0} \tag{2.153}$$

■

Theorem 2.6: *Let $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$ be n-dimensional real arrays in $\mathcal{R}^{N^n}$, where $N = 2^{M_0}$ for a certain integer $M_0 \geq 0$. That is, $\boldsymbol{a}$ for example has N^n elements each of which is $a_{k_1 k_2 \cdots k_n} (k_j = 0, \cdots, N-1)$. Let $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ be the coefficient arrays of the n-dimensional Haar wavelet transform of $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$, respectively. Then, if $\alpha = \beta = 2^{-\frac{n}{2}}$, the following relation of inner products holds:*

$$\frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\boldsymbol{C} \cdot \boldsymbol{D}} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\boldsymbol{c} \cdot \boldsymbol{d}} \tag{2.154}$$

In other words, ratios of inner products are unchanged by the Haar wavelet transform and its inverse transform.

Proof: Let $\boldsymbol{A'}_M$ $(M_0 - 1 \geq M \geq 0)$ denote the collection of the coefficient arrays that result after a series of two-scale decomposition steps $\mathcal{H}_{M_0-1}, \mathcal{H}_{M_0-2}, \cdots, \mathcal{H}_M$. So, $\boldsymbol{A'}_M$ consists of $\boldsymbol{A}^{M,0}$ and the collection of $\{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{s_m \in \{0,1\} \\ s_1 \cdots s_n \neq 0 \cdots 0}}$ from $K = M_0 - 1$ to $K = M$ as follows:

$$\boldsymbol{A'}_M \triangleq \left\{ \boldsymbol{A}^{M,0}, \quad \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M \leq K \leq M_0 - 1 \\ s_m \in \{0,1\} \\ s_1 \cdots s_n \neq 0 \cdots 0}} \right\} \tag{2.155}$$

In particular, $\boldsymbol{A'}_0$ is the collection of arrays after all the decomposition steps, which is the result of the Haar wavelet transform. Thus, $\boldsymbol{A'}_0 \equiv \boldsymbol{A}$. We perform the proof by induction on M for the following:

$$\frac{\boldsymbol{A'}_M \cdot \boldsymbol{B'}_M}{\boldsymbol{C'}_M \cdot \boldsymbol{D'}_M} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\boldsymbol{c} \cdot \boldsymbol{d}} \tag{2.156}$$

1. When $M = M_0 - 1$, $\boldsymbol{A'}_{M_0-1}$ is the resulting arrays after the first two-scale decomposition $\mathcal{H}_{M_0-1}$. Thus, using Lemma 2.2, we get the following:

$$\begin{aligned} \boldsymbol{A'}_{M_0-1} \cdot \boldsymbol{B'}_{M_0-1} &= \mathcal{H}_{M_0-1}(\boldsymbol{A}^{M_0,0}) \cdot \mathcal{H}_{M_0-1}(\boldsymbol{B}^{M_0,0}) \\ &= 2^n \alpha^2 \boldsymbol{A}^{M_0,0} \cdot \boldsymbol{B}^{M_0,0} \end{aligned} \tag{2.157}$$

   Since the initial coefficients are set such that $\boldsymbol{A}^{M_0,0} = \boldsymbol{a}$ and $\boldsymbol{B}^{M_0,0} = \boldsymbol{b}$, the following results:

$$\boldsymbol{A'}_{M_0-1} \cdot \boldsymbol{B'}_{M_0-1} = 2^n \alpha^2 \boldsymbol{a} \cdot \boldsymbol{b} \tag{2.158}$$

The same relation applies to $\boldsymbol{c}$ and $\boldsymbol{d}$. Thus, (2.156) holds for $M = M_0 - 1$.

2. Suppose (2.156) holds for a certain M ($M_0 - 1 \geq M \geq 0$). Referring to the definition (2.155), $\boldsymbol{A'}_{M-1}$ can be rewritten as follows:

$$\begin{aligned}
\boldsymbol{A'}_{M-1} &\equiv \left\{ \boldsymbol{A}^{M-1,0},\ \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M-1\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \\
&= \left\{ \boldsymbol{A}^{M-1,0},\ \{\boldsymbol{A}^{M-1,s_1\cdots s_n}\}_{\substack{s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}},\ \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \\
&= \left\{ \mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}),\ \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \qquad (2.159)
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\boldsymbol{A'}_{M-1} \cdot \boldsymbol{B'}_{M-1} \\
&= \left\{ \mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}),\ \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \\
&\quad \cdot \left\{ \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}),\ \{\boldsymbol{B}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \\
&= \mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}) \cdot \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}) \\
&\quad + \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \cdot \{\boldsymbol{B}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \\
&= \mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}) \cdot \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}) - \boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0} \\
&\quad + \left\{ \boldsymbol{A}^{M,0},\ \{\boldsymbol{A}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \cdot \left\{ \boldsymbol{B}^{M,0},\ \{\boldsymbol{B}^{K,s_1\cdots s_n}\}_{\substack{M\leq K\leq M_0-1\\ s_m\in\{0,1\}\\ s_1\cdots s_n\neq 0\cdots 0}} \right\} \\
&= \mathcal{H}_{M-1}(\boldsymbol{A}^{M,0}) \cdot \mathcal{H}_{M-1}(\boldsymbol{B}^{M,0}) - \boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0} + \boldsymbol{A'}_M \cdot \boldsymbol{B'}_M \\
&= (2^n\alpha^2 - 1)\boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0} + \boldsymbol{A'}_M \cdot \boldsymbol{B'}_M \qquad \text{(from Lemma 2.2)} \qquad (2.160)
\end{aligned}$$

Hence, with the similar relation for $\boldsymbol{C'}_{M-1} \cdot \boldsymbol{D'}_{M-1}$, we get:

$$\begin{aligned}\frac{\boldsymbol{A'}_{M-1} \cdot \boldsymbol{B'}_{M-1}}{\boldsymbol{C'}_{M-1} \cdot \boldsymbol{D'}_{M-1}} &= \frac{(2^n\alpha^2 - 1)\boldsymbol{A}^{M,0} \cdot \boldsymbol{B}^{M,0} + \boldsymbol{A'}_M \cdot \boldsymbol{B'}_M}{(2^n\alpha^2 - 1)\boldsymbol{C}^{M,0} \cdot \boldsymbol{D}^{M,0} + \boldsymbol{C'}_M \cdot \boldsymbol{D'}_M} \\ &= \frac{\boldsymbol{A'}_M \cdot \boldsymbol{B'}_M}{\boldsymbol{C'}_M \cdot \boldsymbol{D'}_M} \qquad (\text{with } \alpha = 2^{-\frac{n}{2}}) \\ &= \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\boldsymbol{c} \cdot \boldsymbol{d}} \qquad (\text{from the inductive hypothesis}) \end{aligned} \tag{2.161}$$

From 1. and 2. above, (2.156) holds for all M ($M_0 - 1 \geq M \geq 0$). In particular, (2.156) with $M = 0$ is equivalent to

$$\frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\boldsymbol{C} \cdot \boldsymbol{D}} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\boldsymbol{c} \cdot \boldsymbol{d}} \tag{2.162}$$

■

Now, let us consider the equivalence of the two kinds of procedures introduced in Section 2.1.3 with the Haar wavelet transform. One procedure indirectly estimates the desired parameter by estimating the coefficients of the transform, and the other procedure directly estimates the parameter by using the gradient filtered by the transform. We call the former $\mathbf{COEF}_{\text{Haar}}$ and the latter $\mathbf{FILT}_{\text{Haar}}$, which are described below.

$\mathbf{COEF}_{\text{Haar}}$ **:** Estimate the Haar wavelet coefficients $\boldsymbol{R}$ to indirectly estimate the parameter $\boldsymbol{r} = \text{Haar}^{-1}(\boldsymbol{R})$ by using the weighted gradient $\nabla^{\mathcal{W}}_{\boldsymbol{R}} H$ defined as follows:

$$\nabla^{\mathcal{W}}_{\boldsymbol{R}} H \triangleq \mathcal{W}(\nabla_{\boldsymbol{R}} H) \tag{2.163}$$

$\mathbf{FILT}_{\text{Haar}}$ **:** Directly estimate the parameter $\boldsymbol{r}$ using the scale-filtered gradient $\nabla^{\text{flt}}_{\boldsymbol{r}} H$ defined as follows:

$$\nabla^{\text{flt}}_{\boldsymbol{r}} H \triangleq \text{Haar}^{-1}(\mathcal{W}(\text{Haar}(\nabla_{\boldsymbol{r}} H))) \tag{2.164}$$

In (2.163) and (2.164), $\mathcal{W}(\boldsymbol{X})$ multiplies each element of $\boldsymbol{X}$ by a certain real weight determined by the element's associated scale. This $\mathcal{W}()$ is essential for controlling the resolution. In the step scheme, each weight is either 1 or 0 as a low-pass filter. In the weight scheme, the weights form a smooth attenuator.

In (2.164), the gradient $\nabla_{\boldsymbol{r}} H$ is transformed to the Haar wavelet coefficient space, multiplied by certain real weights determined by scale, and then transformed back to the parameter space to make the filtered gradient $\nabla_{\boldsymbol{r}}^{\mathrm{flt}} H$.

Note that, as shown later by (3.24), page 73, there is the following relation between the gradients in the two domains:

$$\nabla_{\boldsymbol{R}} H = \mathrm{Haar}(\nabla_{\boldsymbol{r}} H) \tag{2.165}$$

Theorem 2.7: $\mathbf{COEF}_{\mathrm{Haar}}$ *and* $\mathbf{FILT}_{\mathrm{Haar}}$ *produce identical estimates* $\boldsymbol{r}_j$ *of the real-valued parameter* $\boldsymbol{r}$ *on every local-search iteration* $j \geq 0$, *provided that (1) a local search is performed by the same type of the conjugate gradient method, i.e. either the Fletcher-Reeves or the Polak-Ribiere shown in (2.13), (2) they start with the same initial guess* $\boldsymbol{r}_0$, *(3) the same* $\mathcal{W}()$ *is employed, (4)* $\alpha = \beta = 2^{-\frac{n}{2}}$ *for* Haar() *and* Haar^{-1}, *and (5) each line minimization is exact and no numerical errors are involved.*

Proof: We use the following notation.

In $\mathbf{COEF}_{\mathrm{Haar}}$

$\boldsymbol{R}^C$	:	the estimate of the Haar wavelet coefficients
$\boldsymbol{d}^C$	:	the line search direction in the Haar wavelet coefficient space
$\boldsymbol{r}^C$	:	the estimate of the parameter $\left(\boldsymbol{r}^C \equiv \mathrm{Haar}^{-1}(\boldsymbol{R}^C)\right)$

In $\mathbf{FILT}_{\mathrm{Haar}}$

$\boldsymbol{r}^F$	:	the estimate of the parameter
$\boldsymbol{d}^F$	:	the line search direction in the parameter space

If a subscript is attached to any of these, e.g. $\boldsymbol{R}_j^C$, it means the value on the j-th line minimization. Also, we define $\boldsymbol{g}_j^C$ and $\boldsymbol{g}_j^F$ that are used as $\boldsymbol{g}_j$ of Section 2.2 as follows:

$$\boldsymbol{g}_j^C \triangleq \nabla^{\mathcal{W}}_{\boldsymbol{R}^C} H(\boldsymbol{R}_j^C) \equiv \mathcal{W}(\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}_j^C)) \tag{2.166}$$

$$\boldsymbol{g}_j^F \triangleq \nabla^{\text{flt}}_{\boldsymbol{r}^F} H(\boldsymbol{r}_j^F) \equiv \text{Haar}^{-1}(\mathcal{W}(\text{Haar}(\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}_j^F)))) \tag{2.167}$$

where $\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}_j^C)$ is the gradient with respect to the Haar wavelet coefficients $\boldsymbol{R}^C$ and $\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}_j^F)$ is the gradient with respect to the parameter estimate $\boldsymbol{r}^F$.

We prove $\boldsymbol{r}_j^C = \boldsymbol{r}_j^F$ along with $\boldsymbol{d}_j^C = \text{Haar}(\boldsymbol{d}_j^F)$ for all $j \geq 0$ by induction on j, the number of line minimizations.

1. When $j = 0$, the two procedures start with the same initial guess of the parameter:

$$\boldsymbol{r}_0^C = \boldsymbol{r}_0^F \tag{2.168}$$

and the line-search directions $\boldsymbol{d}_0^C$ and $\boldsymbol{d}_0^F$ are as follows:

$$\boldsymbol{d}_0^C = -\boldsymbol{g}_0^C \tag{2.169}$$

$$\boldsymbol{d}_0^F = -\boldsymbol{g}_0^F \tag{2.170}$$

There is the following relation between $\boldsymbol{g}_0^C$ and $\boldsymbol{g}_0^F$:

$$\begin{aligned}
\boldsymbol{g}_0^C &\equiv \mathcal{W}(\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}_0^C)) \\
&= \mathcal{W}\left(\text{Haar}\left(\nabla_{\boldsymbol{r}^C} H(\boldsymbol{r}_0^C)\right)\right) \quad \text{(use (2.165))} \\
&= \mathcal{W}\left(\text{Haar}\left(\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}_0^F)\right)\right) \quad \text{(since } \boldsymbol{r}_0^C = \boldsymbol{r}_0^F\text{)} \\
&= \text{Haar}(\boldsymbol{g}_0^F) \quad \text{(use (2.167))}
\end{aligned} \tag{2.171}$$

Putting (2.169) and (2.170) into (2.171), we get

$$\boldsymbol{d}_0^C = \text{Haar}(\boldsymbol{d}_0^F) \tag{2.172}$$

2. Suppose the following relations hold for a certain $j \geq 0$:

$$\boldsymbol{r}_j^C = \boldsymbol{r}_j^F \tag{2.173}$$

$$\boldsymbol{d}_j^C = \text{Haar}(\boldsymbol{d}_j^F) \tag{2.174}$$

From (2.10) in page 18, $\boldsymbol{R}_{j+1}^C$ and $\boldsymbol{r}_{j+1}^F$ are determined by line minimization in the directions $\boldsymbol{d}_j^C$ and $\boldsymbol{d}_j^F$, respectively, as follows:

$$\boldsymbol{R}_{j+1}^C = \boldsymbol{R}_j^C + \sigma_j^C \boldsymbol{d}_j^C \tag{2.175}$$

$$\boldsymbol{r}_{j+1}^F = \boldsymbol{r}_j^F + \sigma_j^F \boldsymbol{d}_j^F \tag{2.176}$$

Using (2.175),

$$\begin{aligned} \boldsymbol{r}_{j+1}^C &\equiv \text{Haar}^{-1}(\boldsymbol{R}_{j+1}^C) \\ &= \text{Haar}^{-1}(\boldsymbol{R}_j^C + \sigma_j^C \boldsymbol{d}_j^C) \\ &= \text{Haar}^{-1}(\boldsymbol{R}_j^C) + \sigma_j^C \text{Haar}^{-1}(\boldsymbol{d}_j^C) \\ &\equiv \boldsymbol{r}_j^C + \sigma_j^C \text{Haar}^{-1}(\boldsymbol{d}_j^C) \end{aligned} \tag{2.177}$$

From (2.174),

$$\text{Haar}^{-1}(\boldsymbol{d}_j^C) = \boldsymbol{d}_j^F \tag{2.178}$$

With this and (2.173), (2.177) becomes the following:

$$\boldsymbol{r}_{j+1}^C = \boldsymbol{r}_j^F + \sigma_j^C \boldsymbol{d}_j^F \tag{2.179}$$

Comparing the right-hand sides of (2.179) and (2.176), we can see that both are the same line minimization in the direction $\boldsymbol{d}_j^F$ from $\boldsymbol{r}_j^F$. Therefore, they reach the same point, which results in $\sigma_j^F = \sigma_j^C$. Thus, the left-hand sides of (2.179) and (2.176) are equal to each other:

$$\boldsymbol{r}_{j+1}^C = \boldsymbol{r}_{j+1}^F \tag{2.180}$$

From (2.12) in page 18, the next search directions $\boldsymbol{d}^C_{j+1}$ and $\boldsymbol{d}^F_{j+1}$ are determined as follows:

$$\begin{aligned}\boldsymbol{d}^C_{j+1} &= -\boldsymbol{g}^C_{j+1} + \tau^C_{j+1}\boldsymbol{d}^C_j &\quad (2.181)\\ \boldsymbol{d}^F_{j+1} &= -\boldsymbol{g}^F_{j+1} + \tau^F_{j+1}\boldsymbol{d}^F_j &\quad (2.182)\end{aligned}$$

where real coefficients τ^C_{j+1} and τ^F_{j+1} are calculated by either the Fletcher-Reeves formula or the Polak-Ribiere formula shown in (2.13), page 18. The following is obtained with (2.180) in a manner similar to (2.171).

$$\begin{aligned}\boldsymbol{g}^C_{j+1} &\equiv \mathcal{W}(\nabla_{\boldsymbol{R}^C} H(\boldsymbol{R}^C_{j+1}))\\ &= \mathcal{W}\left(\text{Haar}\left(\nabla_{\boldsymbol{r}^F} H(\boldsymbol{r}^F_{j+1})\right)\right)\\ &= \text{Haar}(\boldsymbol{g}^F_{j+1}) &\quad (2.183)\end{aligned}$$

Similarly, with (2.173), we get:

$$\boldsymbol{g}^C_j = \text{Haar}(\boldsymbol{g}^F_j) \quad (2.184)$$

Using (2.183), (2.184), and Theorem 2.6 in page 55, we get the following relations. If the Fletcher-Reeves formula is used,

$$\begin{aligned}\tau^C_{j+1} &= \frac{\boldsymbol{g}^C_{j+1}\cdot\boldsymbol{g}^C_{j+1}}{\boldsymbol{g}^C_j\cdot\boldsymbol{g}^C_j}\\ &= \frac{\text{Haar}(\boldsymbol{g}^F_{j+1})\cdot\text{Haar}(\boldsymbol{g}^F_{j+1})}{\text{Haar}(\boldsymbol{g}^F_j)\cdot\text{Haar}(\boldsymbol{g}^F_j)}\\ &= \frac{\boldsymbol{g}^F_{j+1}\cdot\boldsymbol{g}^F_{j+1}}{\boldsymbol{g}^F_j\cdot\boldsymbol{g}^F_j}\\ &= \tau^F_{j+1} &\quad (2.185)\end{aligned}$$

If the Polak-Ribiere formula is used,

$$\begin{aligned}\tau^C_{j+1} &= \frac{(\boldsymbol{g}^C_{j+1}-\boldsymbol{g}^C_j)\cdot\boldsymbol{g}^C_{j+1}}{\boldsymbol{g}^C_j\cdot\boldsymbol{g}^C_j}\\ &= \frac{(\text{Haar}(\boldsymbol{g}^F_{j+1})-\text{Haar}(\boldsymbol{g}^F_j))\cdot\text{Haar}(\boldsymbol{g}^F_{j+1})}{\text{Haar}(\boldsymbol{g}^F_j)\cdot\text{Haar}(\boldsymbol{g}^F_j)}\end{aligned}$$

$$\begin{aligned}
&= \frac{\mathrm{Haar}(\boldsymbol{g}^F_{j+1} - \boldsymbol{g}^F_j) \cdot \mathrm{Haar}(\boldsymbol{g}^F_{j+1})}{\mathrm{Haar}(\boldsymbol{g}^F_j) \cdot \mathrm{Haar}(\boldsymbol{g}^F_j)} \\
&= \frac{(\boldsymbol{g}^F_{j+1} - \boldsymbol{g}^F_j) \cdot \boldsymbol{g}^F_{j+1}}{\boldsymbol{g}^F_j \cdot \boldsymbol{g}^F_j} \\
&= \tau^F_{j+1} \qquad (2.186)
\end{aligned}$$

We have established $\tau^C_{j+1} = \tau^F_{j+1}$ for both formulas. Using this relation along with (2.174), (2.181), (2.182) and (2.183), we get:

$$\begin{aligned}
\boldsymbol{d}^C_{j+1} &= -\boldsymbol{g}^C_{j+1} + \tau^C_{j+1}\boldsymbol{d}^C_j \\
&= -\mathrm{Haar}(\boldsymbol{g}^F_{j+1}) + \tau^F_{j+1}\mathrm{Haar}(\boldsymbol{d}^F_j) \\
&= \mathrm{Haar}(-\boldsymbol{g}^F_{j+1} + \tau^F_{j+1}\boldsymbol{d}^F_j) \\
&= \mathrm{Haar}(\boldsymbol{d}^F_{j+1}) \qquad (2.187)
\end{aligned}$$

Therefore, as (2.180) and (2.187) show, if $\boldsymbol{r}^C_j = \boldsymbol{r}^F_j$ and $\boldsymbol{d}^C_j = \mathrm{Haar}(\boldsymbol{d}^F_j)$ hold for a certain $j \geq 0$, these relations also hold for $j+1$.

From 1. and 2. above, $\boldsymbol{r}^C_j = \boldsymbol{r}^F_j$ holds for all $j \geq 0$. ■

3. Calculation of the Gradient

The conjugate gradient method needs the gradient of the cost with respect to the distributed parameter to be estimated. Each element of the gradient is the partial derivative of the cost with respect to an element of the distributed parameter. If we use a simple numerical differentiation to get each of them, we have to solve the forward problem on two slightly different values of a parameter element to get *one* partial derivative, and the computation time would be of the order of the number of parameter elements multiplied by the computation time for the forward solution. It is prohibitive to calculate the gradient this way if there are many parameter elements.

Fortunately, we can utilize the so-called *back propagation* to calculate the gradient very efficiently. Its computation time is only of the same order of the forward calculation time.

3.1 The Back-Propagation Algorithm

The *back-propagation* algorithm [14, 15, 33, 32] is a scheme for solving learning problems of artificial neural networks where a cost function that represents learning errors is to be minimized. Although the scheme is sometimes viewed as including the gradient descent method, which is the simplest method of minimization, the essence of back propagation is its efficient way of calculating the gradient of the cost function. So it can be employed to calculate gradients for other methods of minimization such as the conjugate gradient method.

Back propagation is simply a hierarchical application of the chain rule for partial differentiation. The chain rule for partial differentiation is the following: if you want the partial derivative of a certain function H with respect to a variable x, find all the intermediate functions y_j that directly depend on x, and then calculate as follows:

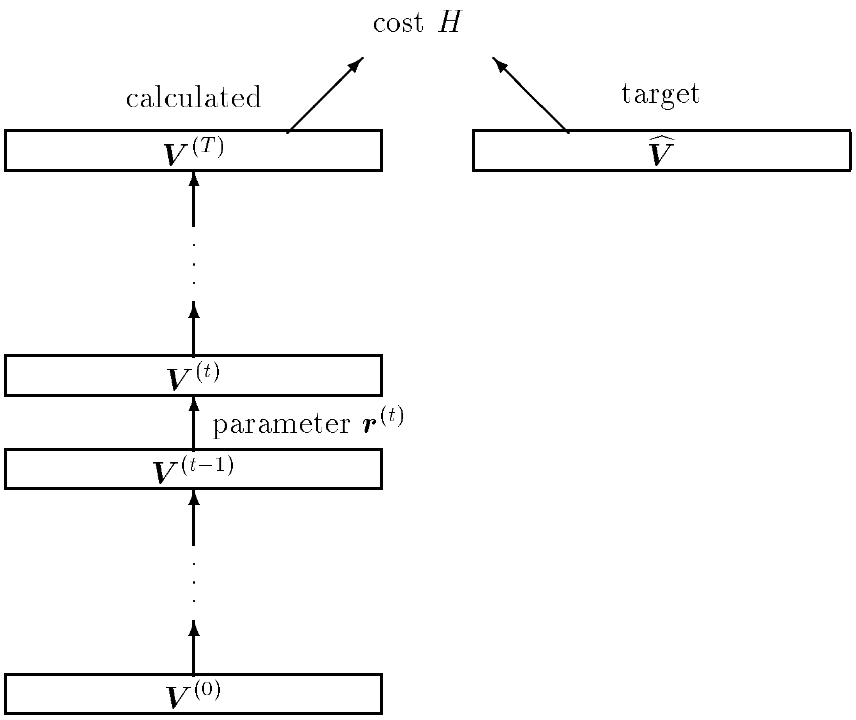


Figure 3.1: Layers of forward calculation

$$\frac{\partial H}{\partial x} = \sum_j \frac{\partial H}{\partial y_j} \frac{\partial y_j}{\partial x} \tag{3.1}$$

In most cases, it is easy to obtain the analytic form of $\frac{\partial y_j}{\partial x}$ because y_j directly depends on x. Therefore, if we know the value of each $\frac{\partial H}{\partial y_j}$, we can easily calculate the desired derivative by (3.1). Back propagation obtains the value of each $\frac{\partial H}{\partial y_j}$ by repeatedly applying (3.1) in a hierarchical manner. We present the details below.

Suppose we have T hierarchical layers of calculation for solving a forward problem as shown in Figure 3.1. In the t-th layer, a set of variables $\boldsymbol{V}^{(t)}$ is calculated from the variables in the $(t-1)$-th layer $\boldsymbol{V}^{(t-1)}$ with a distributed parameter $\boldsymbol{r}^{(t)}$ $(1 \leq t \leq T)$. In the case of an artificial neural network, $\boldsymbol{V}^{(t)}$ would be a set of the outputs of neurons in the t-th layer and $\boldsymbol{r}^{(t)}$ would be the connection weights from the $(t-1)$-th layer to the t-th layer. In the case of a mathematical model, such as a set of partial differential equations, that is solved by an iteration method, "layer t" denotes the t-th iteration of the forward calculation. Although the parameter $\boldsymbol{r}$ may be constant through all the forward calculation layers, it is treated here as independent in each layer t.

Let $V_j^{(t)}$ denote each element of $\boldsymbol{V}^{(t)}$ and $r_k^{(t)}$ each element of $\boldsymbol{r}^{(t)}$. If $\boldsymbol{V}^{(t)}$ and $\boldsymbol{r}^{(t)}$ are multi-dimensional, this notation of elements can be viewed as the j-th and k-th elements in a certain one-dimensional array representation of the multi-dimensional objects.

The *cost* H is calculated by comparing $\boldsymbol{V}^{(T)}$, the variables on the top layer T, with some target values $\widehat{\boldsymbol{V}}$. Let us assume that there are μ_0 sets of conditions on which the closeness of the two is to be evaluated, and the cost is the sum of the closeness for all the sets of conditions $\mu = 0, \cdots, \mu_0 - 1$ as follows:

$$H \triangleq \sum_{\mu=0}^{\mu_0-1} \sum_j f(V_j^{(T)}\Big|_\mu, \widehat{V}_j\Big|_\mu) \tag{3.2}$$

where $f()$ is a certain function and $x|_\mu$ is the value of x on the condition μ.

We want to calculate the gradient of the cost H with respect to each parameter $\boldsymbol{r}^{(t)}$ in the t-th layer, which is a set of partial derivatives:

$$\nabla_{\boldsymbol{r}^{(t)}} H \equiv \left\{ \frac{\partial H}{\partial r_j^{(t)}} \right\}_{\text{all } j} \tag{3.3}$$

Each of the partial derivatives is calculated by the chain rule as follows:

$$\frac{\partial H}{\partial r_j^{(t)}} = \sum_{\mu=0}^{\mu_0-1} \sum_k \left.\frac{\partial H}{\partial V_k^{(t)}}\right|_\mu \left.\frac{\partial V_k^{(t)}}{\partial r_j^{(t)}}\right|_\mu \tag{3.4}$$

Let us assume that $\left.\dfrac{\partial V_k^{(t)}}{\partial r_j^{(t)}}\right|_\mu$ in (3.4) is available in analytic form. This is usually the case, because the relation between $V_k^{(t)}$ and $r_j^{(t)}$ is described by the given mathematical model. Then, calculating (3.4) reduces to calculating $\left.\dfrac{\partial H}{\partial V_k^{(t)}}\right|_\mu$, which is performed from the top layer ($t = T$) toward the bottom layer in the following way.

If $t = T$, $V_k^{(t)}$ is on the top layer and therefore directly related to the cost H as defined in (3.2). From that definition we get:

$$\left.\frac{\partial H}{\partial V_k^{(T)}}\right|_\mu = f'(\left.V_k^{(T)}\right|_\mu, \left.\widehat{V}_k\right|_\mu) \tag{3.5}$$

where $f'()$ is the derivative of the function $f()$. For example, if we set $f(x, \hat{x}) \triangleq \frac{1}{2}(x - \hat{x})^2$ in (3.2) and define the cost H as

$$H \triangleq \frac{1}{2} \sum_{\mu=0}^{\mu_0 - 1} \sum_j (\left.V_j^{(T)}\right|_\mu - \left.\widehat{V}_j\right|_\mu)^2 \tag{3.6}$$

then,

$$\left.\frac{\partial H}{\partial V_k^{(T)}}\right|_\mu = \left.V_k^{(T)}\right|_\mu - \left.\widehat{V}_k\right|_\mu \tag{3.7}$$

If $t < T$, $V_k^{(t)}$ is not directly related to the cost H, hence the derivative $\left.\frac{\partial H}{\partial V_k^{(t)}}\right|_\mu$ is not immediately available. However, $V_k^{(t)}$ is used to get $V_j^{(t+1)}$ on the layer above it in the forward calculation and their relation is defined by the given model. From that relation, $\frac{\partial V_j^{(t+1)}}{\partial V_k^{(t)}}$ is obtained in analytic form. Therefore, using the chain rule again, the derivative of H with respect to $V_k^{(t)}$ is calculated from the derivatives with respect to $V_k^{(t+1)}$ as follows:

$$\left.\frac{\partial H}{\partial V_k^{(t)}}\right|_\mu = \sum_j \left.\frac{\partial H}{\partial V_j^{(t+1)}}\right|_\mu \left.\frac{\partial V_j^{(t+1)}}{\partial V_k^{(t)}}\right|_\mu \tag{3.8}$$

In other words, the derivatives $\frac{\partial H}{\partial V_j^{(t)}}$ are "back-propagated" from the top layer $t = T$ through the bottom layer $t = 1$.

With all these derivatives ready, we can calculate all the elements of the desired gradient by (3.4). Let us summarize the process of calculation:

1. Put $\frac{\partial H}{\partial V_k^{(T)}}$ on the top layer into analytic form by differentiating H.
2. Put $\frac{\partial V_k^{(t)}}{\partial r_j^{(t)}}$ into analytic form by differentiating the given mathematical model.

3. Calculate each value of $\left.\frac{\partial H}{\partial V_k^{(T)}}\right|_{\mu}$ on the top layer by putting the value of $V_k^{(T)}$ on condition μ into the analytic form obtained in 1.
4. Calculate each value of $\left.\frac{\partial H}{\partial V_k^{(t)}}\right|_{\mu}$ on the lower layers by back-propagating as in (3.8).
5. Calculate each value of $\left.\frac{\partial V_k^{(t)}}{\partial r_j^{(t)}}\right|_{\mu}$ on all the layers by putting the value of $\left.V_k^{(t)}\right|_{\mu}$ into the analytic form obtained in 2.
6. Put $\left.\frac{\partial V_k^{(t)}}{\partial r_j^{(t)}}\right|_{\mu}$ calculated in 5. and $\left.\frac{\partial H}{\partial V_k^{(t)}}\right|_{\mu}$ calculated in 3. and 4. into (3.4) to obtain the gradient.

As seen in the summary above, all the values $\left.V_k^{(t)}\right|_{\mu}$ have to be stored during the forward calculation on condition μ to be used by the back-propagation process. The time complexity of back propagation is of the same order of the computation time required for the forward solution, because back propagation requires only one sweep of calculation from the top layer to the bottom layer.

Literature of artificial neural networks usually defines and uses the "delta value" δ as the partial derivative of the cost with respect to the *input* of a function representing $V_j^{(t)}$. This is fine with most neural networks, because such a function usually has only one input, e.g. the sum of several signals. However, this assumption may not work in the case of parameter estimation, because the variable $V_j^{(t)}$ can be a function of two or more inputs [1]. Viewing back propagation in a general manner as above eliminates such unnecessary restrictions and allows us to use the technique for general parameter estimation problems.

3.2 When the Parameter is Constant

In the previous section, we treated the distributed parameter as independent in each layer. But, in many problems, the parameter to be estimated should be constant through all the forward calculation layers. We consider such a case here.

The gradient to be obtained is with respect to some constant distributed parameter $\boldsymbol{r} = \{r_j\}$ as follows (cf. (3.3)):

$$\nabla_{\boldsymbol{r}} H \equiv \left\{ \frac{\partial H}{\partial r_j} \right\}_{\text{all } j} \tag{3.9}$$

We can view the case in the following way: each layer's distributed parameter $\boldsymbol{r}^{(t)}$ is independent per se of the other layers' parameters as in the previous section, but the parameters of all the layers *happen* to have the same distribution $\boldsymbol{r}$ such that:

$$\boldsymbol{r}^{(t)} = \boldsymbol{r} \qquad (1 \leq t \leq T) \tag{3.10}$$

In other words, each element $r_j^{(t)}$ is a function of r_j and that function happens to be the identity function. Then,

$$\frac{\partial r_j^{(t)}}{\partial r_j} = 1 \tag{3.11}$$

Using this relation and the chain rule, we can get the desired partial derivative as follows:

$$\begin{aligned} \frac{\partial H}{\partial r_j} &= \sum_{t=1}^{T} \frac{\partial H}{\partial r_j^{(t)}} \frac{\partial r_j^{(t)}}{\partial r_j} \\ &= \sum_{t=1}^{T} \frac{\partial H}{\partial r_j^{(t)}} \end{aligned} \tag{3.12}$$

Intuitively, this relation is obvious because $\frac{\partial H}{\partial r_j}$ is the effect of changing r_j by a unit amount, which is equivalent to the effect of changing all $r_j^{(t)}$ from $t = 1$ to $t = T$ by the same amount, which in turn is the sum of all the partial derivatives $\frac{\partial H}{\partial r_j^{(t)}}$.

Therefore, we can get the gradient (3.9) by first calculating all the partial derivatives $\frac{\partial H}{\partial r_j^{(t)}}$ using the method in Section 3.1 and then summing them up as in (3.12).

3.3 For Converging Forward Calculation

In Section 3.1 we assumed that the number of layers T is fixed. However, if the forward problem is solved in such a way that the iteration continues until $\boldsymbol{V}^{(t)}$ converges within some tolerance, the number of layers T can be very large. Then, it may be impossible to store all the values $\boldsymbol{V}^{(t)}$ from $t = 1$ to $t = T$ as required for back propagation. We consider such a problem in this section. In this case, the distributed parameter should be constant throughout the forward iterations, because otherwise "convergence" would be meaningless.

The key to solving this problem is the fact that a converged solution does *not* depend on the initial values of the variables. At least we assume so within a certain range of values, since otherwise there would be no reason to employ such convergence calculation. This means the same solution will result from any initial values if they are within some distance from the solution. Then, we can certainly reach the same solution if we use the solution itself as the initial values! We can view it in a different way: if we start the forward iteration with some initial values and let it continue even after it converges, the converged values will stay the same (within some tolerance) throughout the further iterations.

Based on this observation, let us *imagine* the case in which the forward iteration starts with its solution as the initial values and continues running forever. After an *infinite* number of iterations, we stop the calculation and use back propagation to obtain the gradient. Let the final layer of calculation be numbered T as in the previous sections, and we have an infinite number of layers from $t = -\infty$ to $t = T$. Then, each element of the gradient can be calculated the same way as in (3.12) as follows:

$$\frac{\partial H}{\partial r_j} = \sum_{t=-\infty}^{T} \frac{\partial H}{\partial r_j^{(t)}} \tag{3.13}$$

This infinite series has to be convergent for the gradient to exist. Therefore, in actual calculation, we can start summing $\dfrac{\partial H}{\partial r_j^{(t)}}$ at $t = T$ toward the negative t direction and stop when the change of the summation becomes smaller than a certain threshold.

In this method, one does not have to actually perform an infinite number of iterations in the forward calculation. We only imagined such calculation to derive the method of getting the gradient. The actual process is the following:

1. Run the forward calculation to get the converged forward solution $\boldsymbol{V}^{\text{conv}}$.
2. Use $\boldsymbol{V}^{\text{conv}}$ in place of $\boldsymbol{V}^{(t)}$ for every layer t in back propagation described in Section 3.1.
3. Sum $\dfrac{\partial H}{\partial r_j^{(t)}}$ for layer t from the top layer downwards as in (3.13) until the summation converges.

This method has to store only one set of variables $\boldsymbol{V}^{\text{conv}}$. Therefore, it is extremely memory-efficient compared with the ordinary way that must store $\boldsymbol{V}^{(t)}$ of all the layers.

3.4 The Gradient for Fourier Coefficients

When a local search method such as the conjugate gradient method is to estimate the Fourier coefficients $\boldsymbol{R}$ which are complex in general, it needs $\overline{\nabla_{\boldsymbol{R}} H}$, the complex conjugate of the gradient, as shown in Section 2.2.2. This $\overline{\nabla_{\boldsymbol{R}} H}$ is obtained simply by flipping the signs of the imaginary parts of the gradient $\nabla_{\boldsymbol{R}} H$. Therefore, in this section, we consider how to calculate the gradient with respect to the Fourier coefficients, assuming that we already have the gradient with respect to the distributed parameter to be estimated. Namely, we consider the relation between the gradient $\nabla_{\boldsymbol{r}} H$ with respect to a distributed parameter $\boldsymbol{r}$ and the gradient $\nabla_{\boldsymbol{R}} H$ with respect to the Fourier coefficients $\boldsymbol{R} = \text{DFT}(\boldsymbol{r})$.

Let us take (2.38) and (2.39) as the definitions of the n-dimensional DFT and the inverse DFT, respectively. Namely, we have the following relations between the n-dimensional distributed parameter $\boldsymbol{r} = \{r_{j_1\cdots j_n}\}_{0\le j_m\le N_m-1}$ and its Fourier coefficients $\boldsymbol{R} = \{R_{k_1\cdots k_n}\}_{0\le k_m\le N_m-1}$:

$$\begin{aligned} R_{k_1\cdots k_n} &= \alpha \sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1} r_{j_1\cdots j_n} e^{-\frac{i2\pi k_1 j_1}{N_1}}\cdots e^{-\frac{i2\pi k_n j_n}{N_n}} \qquad (3.14)\\ &\quad (k_m = 0,\cdots,N_m-1)\\ r_{j_1\cdots j_n} &= \beta \sum_{k_1=0}^{N_1-1}\cdots\sum_{k_n=0}^{N_n-1} R_{k_1\cdots k_n} e^{\frac{i2\pi j_1 k_1}{N_1}}\cdots e^{\frac{i2\pi j_n k_n}{N_n}} \qquad (3.15)\\ &\quad (j_m = 0,\cdots,N_m-1)\end{aligned}$$

where α and β are real constants such that $\alpha\beta = \frac{1}{N_1N_2\cdots N_n}$. Differentiating (3.15) with respect to a Fourier coefficient $R_{k_1\cdots k_n}$ gives

$$\frac{\partial r_{j_1\cdots j_N}}{\partial R_{k_1\cdots k_n}} = \beta e^{\frac{i2\pi j_1 k_1}{N_1}}\cdots e^{\frac{i2\pi j_n k_n}{N_n}} \qquad (3.16)$$

Each Fourier coefficient $R_{k_1\cdots k_n}$ is related to all the parameter elements $r_{j_1\cdots j_N}$. Therefore, using the chain rule for partial differentiation and (3.16), an element of the gradient $\nabla_{\boldsymbol{R}} H$ is expressed as follows:

$$\begin{aligned}\frac{\partial H}{\partial R_{k_1\cdots k_n}} &= \sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\frac{\partial H}{\partial r_{j_1\cdots j_n}}\frac{\partial r_{j_1\cdots j_n}}{\partial R_{k_1\cdots k_n}}\\ &= \beta\sum_{j_1=0}^{N_1-1}\cdots\sum_{j_n=0}^{N_n-1}\frac{\partial H}{\partial r_{j_1\cdots j_n}} e^{\frac{i2\pi j_1 k_1}{N_1}}\cdots e^{\frac{i2\pi j_n k_n}{N_n}} \qquad (3.17)\end{aligned}$$

The resulting expression is exactly the inverse DFT of the derivative $\dfrac{\partial H}{\partial r_{j_1\cdots j_n}}$. So, we can write

$$\frac{\partial H}{\partial \boldsymbol{R}} = \mathrm{DFT}^{-1}\left(\frac{\partial H}{\partial \boldsymbol{r}}\right) \qquad (3.18)$$

or using the gradient notation,

$$\nabla_{\boldsymbol{R}} H = \mathrm{DFT}^{-1}(\nabla_{\boldsymbol{r}} H) \qquad (3.19)$$

Conversely,

$$\nabla_{\boldsymbol{r}} H = \mathrm{DFT}(\nabla_{\boldsymbol{R}} H) \tag{3.20}$$

It is interesting to see that these relations of gradients are the opposite of the relations of points in the two spaces shown in (2.40) and (2.41), page 29.

3.5 The Gradient for Haar Wavelet Coefficients

In this section, we consider the relation between the gradient $\nabla_{\boldsymbol{r}} H$ with respect to the distributed parameter $\boldsymbol{r}$ and the gradient $\nabla_{\boldsymbol{R}} H$ with respect to the Haar wavelet coefficients $\boldsymbol{R}$.

As the two-scale reconstruction relation (2.122) in page 47 shows, $\boldsymbol{R}^{M,0}$, i.e. the representation of $\boldsymbol{r}$ in the M-th resolution from the coarsest (0-th), and $\boldsymbol{R}^{M-1,s_1\cdots s_n}$, i.e. the Haar wavelet coefficients at the $(M-1)$-th resolution, are directly related to each other. Let us apply the chain rule for partial differentiation to them as the following:

$$\frac{\partial H}{\partial R^{M-1,s_1\cdots s_n}_{i_1,\cdots,i_n}} = \sum_{\substack{j_1,\cdots,j_n \\ j_m\in\{2i_m,2i_m+1\}}} \frac{\partial H}{\partial R^{M,0}_{j_1,\cdots,j_n}} \frac{\partial R^{M,0}_{j_1,\cdots,j_n}}{\partial R^{M-1,s_1\cdots s_n}_{i_1,\cdots,i_n}} \tag{3.21}$$

From (2.122), we get:

$$\frac{\partial R^{M,0}_{j_1,\cdots,j_n}}{\partial R^{M-1,s_1\cdots s_n}_{i_1,\cdots,i_n}} = \beta(-1)^{\sum_{t=1}^{n} j_t s_t} \qquad (j_m \in \{2i_m, 2i_m+1\}) \tag{3.22}$$

Substituting (3.22) into (3.21) gives:

$$\frac{\partial H}{\partial R^{M-1,s_1\cdots s_n}_{i_1,\cdots,i_n}} = \beta \sum_{\substack{j_1,\cdots,j_n \\ j_m\in\{2i_m,2i_m+1\}}} (-1)^{\sum_{t=1}^{n} j_t s_t} \frac{\partial H}{\partial R^{M,0}_{j_1,\cdots,j_n}} \tag{3.23}$$

If we choose $\beta = \alpha = 2^{-\frac{n}{2}}$, (3.23) is exactly the two-scale decomposition relation for $\dfrac{\partial H}{\partial \boldsymbol{R}^{M,0}}$ (cf. (2.121), page 47). Therefore, if $\alpha = \beta = 2^{-\frac{n}{2}}$, we have the following relations between the two gradients:

$$\nabla_{\boldsymbol{R}} H = \mathrm{Haar}(\nabla_{\boldsymbol{r}} H) \tag{3.24}$$

$$\nabla_{\boldsymbol{r}} H = \mathrm{Haar}^{-1}(\nabla_{\boldsymbol{R}} H) \tag{3.25}$$

4. Application: Electrical Impedance Tomography

In this chapter, we evaluate the multiresolution parameter estimation methods developed in the previous chapters in simulation of the so-called *electrical impedance tomography* [16, 17, 22, 24, 26, 28, 29, 38, 41, 42, 44, 45]. It is a highly nonlinear, ill-posed problem and therefore challenging to an estimation algorithm. We will see how well the multiresolution methods perform on this problem compared with the conventional single-resolution estimation.

4.1 Model

Electrical impedance tomography (EIT) is the problem of reconstructing the internal resistivity distribution of an object by injecting electrical current and measuring the relations between the current and the voltage on the exterior of the object. If there is no source or sink of electrical current in the interior of the object, the current is conserved everywhere inside the object. This yields the following elliptic partial differential equation that sets the divergence of current density to be zero:

$$\mathrm{div}\left(\frac{\mathrm{grad}V}{r}\right) \equiv \nabla \cdot (\frac{1}{r}\nabla V) = 0 \tag{4.1}$$

r : resistivity (e.g. in $[\Omega \cdot m]$)

V : voltage (e.g. in $[V]$)

This is the equation that applies to every internal point of the object. In the two-dimensional case, (4.1) becomes the following:

$$\frac{\partial}{\partial x}(\frac{1}{r}\frac{\partial V}{\partial x}) + \frac{\partial}{\partial y}(\frac{1}{r}\frac{\partial V}{\partial y}) = 0 \tag{4.2}$$

Let us consider a two-dimensional rectangular field and let (i, j) denote the grid point at i-th x position and j-th y position. (4.2) is converted into a difference equation in the finite difference method (FDM). The current density in x direction $\frac{1}{r}\frac{\partial V}{\partial x}$ between grid points (i, j) and $(i+1, j)$ is converted into the following difference expression by using the average r of the two.

$$\frac{1}{r}\frac{\partial V}{\partial x} \cong \frac{2}{r_{i+1,j} + r_{i,j}} \cdot \frac{V_{i+1,j} - V_{i,j}}{h_x} \tag{4.3}$$

where h_x is the grid spacing in x direction. If the grid point (i, j) is on the boundary and the grid point $(i+1, j)$ is outside the boundary, the current density between the two points is determined by the injected current $J_{i,j}$ (e.g. in $[A]$) as the boundary condition:

$$\frac{1}{r}\frac{\partial V}{\partial x} \cong \frac{J_{i,j}}{h_y} \tag{4.4}$$

where h_y is the grid spacing in y direction. The right-hand side of (4.4) includes unit depth in the implicit z direction such as $\frac{J_{i,j}}{h_y \cdot 1}$, so its dimension is consistent with that of the left-hand side. Here, the sign of the injected current $J_{i,j}$ is defined to be positive if the current flows into the object.

In order to express equations for both internal grid points and those on the boundary at the same time, I will use the following notation:

$$\left\{ \begin{array}{c} \textit{expression } A \\ \textit{expression } B \end{array} \right\} \tag{4.5}$$

where *expression A* is taken whenever A is valid (i.e. no points in *expression A* are outside the boundary), otherwise *expression B* is taken (i.e. if one or more points in *expression A* are outside the boundary). Using this notation, (4.3) and (4.4) are combined together as follows:

$$\frac{1}{r}\frac{\partial V}{\partial x} \cong \left\{ \begin{array}{c} \frac{2}{h_x}\frac{V_{i+1,j} - V_{i,j}}{r_{i+1,j} + r_{i,j}} \\ \frac{J_{i,j}}{h_y} \end{array} \right\} \tag{4.6}$$

Similarly, the current density between $(i-1,j)$ and (i,j) is:

$$\frac{1}{r}\frac{\partial V}{\partial x} \cong \left\{ \begin{array}{c} \frac{2}{h_x}\frac{V_{i,j}-V_{i-1,j}}{r_{i,j}+r_{i-1,j}} \\ -\frac{J_{i,j}}{h_y} \end{array} \right\} \tag{4.7}$$

From (4.6) and (4.7), the finite difference approximation below is obtained:

$$\frac{\partial}{\partial x}\left(\frac{1}{r}\frac{\partial V}{\partial x}\right) \cong \frac{1}{h_x}\left(\left\{ \begin{array}{c} \frac{2}{h_x}\frac{V_{i+1,j}-V_{i,j}}{r_{i+1,j}+r_{i,j}} \\ \frac{J_{i,j}}{h_y} \end{array} \right\} - \left\{ \begin{array}{c} \frac{2}{h_x}\frac{V_{i,j}-V_{i-1,j}}{r_{i,j}+r_{i-1,j}} \\ -\frac{J_{i,j}}{h_y} \end{array} \right\} \right) \tag{4.8}$$

Expressions for y direction are made similarly. Then, (4.2), multiplied by grid spacings $h_x h_y$, is converted into the following finite difference form:

$$\left\{ \begin{array}{c} \frac{2h_y}{h_x}\frac{V_{i+1,j}-V_{i,j}}{r_{i+1,j}+r_{i,j}} \\ J_{i,j} \end{array} \right\} + \left\{ \begin{array}{c} \frac{2h_y}{h_x}\frac{V_{i-1,j}-V_{i,j}}{r_{i-1,j}+r_{i,j}} \\ J_{i,j} \end{array} \right\} + \left\{ \begin{array}{c} \frac{2h_x}{h_y}\frac{V_{i,j+1}-V_{i,j}}{r_{i,j+1}+r_{i,j}} \\ J_{i,j} \end{array} \right\} + \left\{ \begin{array}{c} \frac{2h_x}{h_y}\frac{V_{i,j-1}-V_{i,j}}{r_{i,j-1}+r_{i,j}} \\ J_{i,j} \end{array} \right\} = 0 \tag{4.9}$$

If (i,j) is a corner of the boundary, the injected current $J_{i,j}$ is taken twice in (4.9). To compensate for this, we assume from now on that the values of $J_{i,j}$ at the corners are halved before this equation and those below are used.

4.2 Forward Solution

Numerical methods to solve elliptic difference equations such as (4.9) are classified into *direct methods* and *iterative methods* [9, 13, 27, 30, 36]. Direct methods directly solve the finite difference equations of the matrix form $\boldsymbol{Av} = \boldsymbol{b}$ with respect to the column vector $\boldsymbol{v}$. A serious problem with direct methods is that a large amount of memory is required. For example, in a two-dimensional problem with $N_x \times N_y$ grid points, the size of the matrix $\boldsymbol{A}$ is (N_xN_y) by (N_xN_y). Therefore, if all of its elements are to be stored, it takes storage of size $O((N_xN_y)^2)$. Although there are methods that utilize the fact that the matrix $\boldsymbol{A}$ is a sparse matrix whose elements are non-zero only in a band $2N_x+1$ elements wide, their storage requirements are still large, e.g.

$O(N_x^2 N_y)$ with a typical method for sparse matrices [27]. For this reason, we choose an iterative method here that only needs memory of size $O(N_x N_y)$.

A two-dimensional difference equation in general regarding a variable V can be written in the following form:

$$a^C V_{i,j} + a^E V_{i+1,j} + a^W V_{i-1,j} + a^S V_{i,j+1} + a^N V_{i,j-1} = b_{i,j} \tag{4.10}$$

where a^C is the coefficient for the center (i,j) of the five points, a^E for the east neighbor $(i+1,j)$, a^W for the west neighbor $(i-1,j)$, a^S for the south neighbor $(i,j+1)$, and a^N for the north neighbor $(i,j-1)$.

In the *Jacobi-iterative* method, (4.10) is iteratively solved with respect to $V_{i,j}$ as the following:

$$V_{i,j}^{(t)} = \frac{1}{a^C}\left[b_{i,j} - \left(a^E V_{i+1,j}^{(t-1)} + a^W V_{i-1,j}^{(t-1)} + a^S V_{i,j+1}^{(t-1)} + a^N V_{i,j-1}^{(t-1)}\right)\right] \tag{4.11}$$

where $V^{(t)}$ denotes V at the t-th iteration. This method has the so-called *two-cyclic property*, which means there are two independent series of simultaneous computation. The Jacobi-iterative method is not often used in practice, since its convergence is slow.

The *Successive Over-Relaxation* (SOR) method is the following:

$$V_{i,j}^{(t)} = \omega\frac{1}{a^C}\left[b_{i,j} - \left(a^E V_{i+1,j}^{(t-1)} + a^W V_{i-1,j}^{(t)} + a^S V_{i,j+1}^{(t-1)} + a^N V_{i,j-1}^{(t)}\right)\right] + (1-\omega)V_{i,j}^{(t-1)} \tag{4.12}$$

where ω is the over-relaxation parameter $(1 < \omega < 2)$ for accelerating convergence. In (4.12), it is assumed that the grid points are calculated in the increasing order of i and j. Thus, the point (i,j) is calculated on the new values of points $(i-1,j)$ and $(i,j-1)$. Because of this, the SOR does not have the two-cyclic property of the Jacobi-iterative method. However, this updating scheme also makes it difficult to vectorize/parallelize the SOR.

More suitable for massively parallel computers and vectorized supercomputers than the SOR is the *Extrapolated Jacobi-iterative* (EJ) method [27] as follows:

$$V_{i,j}^{(t)} = \theta \frac{1}{a^C} \left[b_{i,j} - \left(a^E V_{i+1,j}^{(t-1)} + a^W V_{i-1,j}^{(t-1)} + a^S V_{i,j+1}^{(t-1)} + a^N V_{i,j-1}^{(t-1)} \right) \right] + (1-\theta) V_{i,j}^{(t-2)} \tag{4.13}$$

where θ is the extrapolation parameter $(1 < \theta < 2)$. Although, due to the two-cyclic property, the EJ method needs twice as many iterations for convergence as the SOR does, the method is easily parallelized/vectorized.

We employ the EJ method because it is suitable for parallel processing and easy to implement. Let us first trace how a difference equation is converted into the EJ formula. The difference equation (4.10) is solved with respect to $V_{i,j}$ as follows:

$$V_{i,j} = \frac{1}{a^C} \left[b_{i,j} - \left(a^E V_{i+1,j} + a^W V_{i-1,j} + a^S V_{i,j+1} + a^N V_{i,j-1} \right) \right] \tag{4.14}$$

If this equation is converted into an iteration formula as it is, it would become the Jacobi-iterative method (4.11). We use a new variable $X_{i,j}^{(t)}$ in place of $V_{i,j}^{(t)}$ in (4.11) to avoid confusion as follows:

$$X_{i,j}^{(t)} = \frac{1}{a^C} \left[b_{i,j} - \left(a^E V_{i+1,j}^{(t-1)} + a^W V_{i-1,j}^{(t-1)} + a^S V_{i,j+1}^{(t-1)} + a^N V_{i,j-1}^{(t-1)} \right) \right] \tag{4.15}$$

Then, extrapolating $V_{i,j}^{(t)}$ using this $X_{i,j}^{(t)}$ and the two-iteration old value $V_{i,j}^{(t-2)}$ yields:

$$V_{i,j}^{(t)} = \theta X_{i,j}^{(t)} + (1-\theta) V_{i,j}^{(t-2)} \tag{4.16}$$

which is equivalent to the EJ formula in (4.13).

Now, according to the observation above, we convert our difference equation (4.9) into the EJ formula. For simplicity, we use the following notation:

$$\left.\begin{array}{lcll} E_{i,j} & \triangleq & \frac{h_y}{h_x}\frac{2}{r_{i+1,j}+r_{i,j}} & (= W_{i+1,j}) \\ W_{i,j} & \triangleq & \frac{h_y}{h_x}\frac{2}{r_{i-1,j}+r_{i,j}} & (= E_{i-1,j}) \\ S_{i,j} & \triangleq & \frac{h_x}{h_y}\frac{2}{r_{i,j+1}+r_{i,j}} & (= N_{i,j+1}) \\ N_{i,j} & \triangleq & \frac{h_x}{h_y}\frac{2}{r_{i,j-1}+r_{i,j}} & (= S_{i,j-1}) \\ C_{i,j} & \triangleq & \multicolumn{2}{l}{\left\{\begin{array}{c} E_{i,j} \\ 0 \end{array}\right\} + \left\{\begin{array}{c} W_{i,j} \\ 0 \end{array}\right\} + \left\{\begin{array}{c} S_{i,j} \\ 0 \end{array}\right\} + \left\{\begin{array}{c} N_{i,j} \\ 0 \end{array}\right\}} \end{array}\right\} \tag{4.17}$$

Then, the difference equation (4.9) is rewritten as below:

$$\begin{aligned} &\left\{\begin{array}{c} E_{i,j}(V_{i+1,j} - V_{i,j}) \\ J_{i,j} \end{array}\right\} + \left\{\begin{array}{c} W_{i,j}(V_{i-1,j} - V_{i,j}) \\ J_{i,j} \end{array}\right\} \\ + &\left\{\begin{array}{c} S_{i,j}(V_{i,j+1} - V_{i,j}) \\ J_{i,j} \end{array}\right\} + \left\{\begin{array}{c} N_{i,j}(V_{i,j-1} - V_{i,j}) \\ J_{i,j} \end{array}\right\} = 0 \end{aligned} \tag{4.18}$$

which is equivalent to:

$$C_{i,j}V_{i,j} = \left\{\begin{array}{c} E_{i,j}V_{i+1,j} \\ J_{i,j} \end{array}\right\} + \left\{\begin{array}{c} W_{i,j}V_{i-1,j} \\ J_{i,j} \end{array}\right\} + \left\{\begin{array}{c} S_{i,j}V_{i,j+1} \\ J_{i,j} \end{array}\right\} + \left\{\begin{array}{c} N_{i,j}V_{i,j-1} \\ J_{i,j} \end{array}\right\} \tag{4.19}$$

This is solved with respect to $V_{i,j}$ as follows:

$$V_{i,j} = \left\{\begin{array}{c} \frac{E_{i,j}V_{i+1,j}}{C_{i,j}} \\ \frac{J_{i,j}}{C_{i,j}} \end{array}\right\} + \left\{\begin{array}{c} \frac{W_{i,j}V_{i-1,j}}{C_{i,j}} \\ \frac{J_{i,j}}{C_{i,j}} \end{array}\right\} + \left\{\begin{array}{c} \frac{S_{i,j}V_{i,j+1}}{C_{i,j}} \\ \frac{J_{i,j}}{C_{i,j}} \end{array}\right\} + \left\{\begin{array}{c} \frac{N_{i,j}V_{i,j-1}}{C_{i,j}} \\ \frac{J_{i,j}}{C_{i,j}} \end{array}\right\} \tag{4.20}$$

We change this equation into an iteration form and use a new variable $X_{i,j}$ in place of $V_{i,j}$ (cf. (4.15)):

$$X_{i,j}^{(t)} = \left\{\begin{array}{c} \frac{E_{i,j}^{(t)}V_{i+1,j}^{(t-1)}}{C_{i,j}^{(t)}} \\ \frac{J_{i,j}}{C_{i,j}^{(t)}} \end{array}\right\} + \left\{\begin{array}{c} \frac{W_{i,j}^{(t)}V_{i-1,j}^{(t-1)}}{C_{i,j}^{(t)}} \\ \frac{J_{i,j}}{C_{i,j}^{(t)}} \end{array}\right\} + \left\{\begin{array}{c} \frac{S_{i,j}^{(t)}V_{i,j+1}^{(t-1)}}{C_{i,j}^{(t)}} \\ \frac{J_{i,j}}{C_{i,j}^{(t)}} \end{array}\right\} + \left\{\begin{array}{c} \frac{N_{i,j}^{(t)}V_{i,j-1}^{(t-1)}}{C_{i,j}^{(t)}} \\ \frac{J_{i,j}}{C_{i,j}^{(t)}} \end{array}\right\} \tag{4.21}$$

Using the notation below,

$$w_{E_{i,j}}^{(t)} \triangleq \frac{E_{i,j}^{(t)}}{C_{i,j}^{(t)}} \qquad w_{W_{i,j}}^{(t)} \triangleq \frac{W_{i,j}^{(t)}}{C_{i,j}^{(t)}} \qquad w_{S_{i,j}}^{(t)} \triangleq \frac{S_{i,j}^{(t)}}{C_{i,j}^{(t)}} \qquad w_{N_{i,j}}^{(t)} \triangleq \frac{N_{i,j}^{(t)}}{C_{i,j}^{(t)}} \tag{4.22}$$

$$w^{(t)}_{JE_{i,j}} = w^{(t)}_{JW_{i,j}} = w^{(t)}_{JS_{i,j}} = w^{(t)}_{JN_{i,j}} \triangleq \frac{1}{C^{(t)}_{i,j}} \tag{4.23}$$

we can simplify (4.21) to the following:

$$X^{(t)}_{i,j} = \left\{ \begin{array}{c} w^{(t)}_{E_{i,j}} V^{(t-1)}_{i+1,j} \\ w^{(t)}_{JE_{i,j}} J_{i,j} \end{array} \right\} + \left\{ \begin{array}{c} w^{(t)}_{W_{i,j}} V^{(t-1)}_{i-1,j} \\ w^{(t)}_{JW_{i,j}} J_{i,j} \end{array} \right\} + \left\{ \begin{array}{c} w^{(t)}_{S_{i,j}} V^{(t-1)}_{i,j+1} \\ w^{(t)}_{JS_{i,j}} J_{i,j} \end{array} \right\} + \left\{ \begin{array}{c} w^{(t)}_{N_{i,j}} V^{(t-1)}_{i,j-1} \\ w^{(t)}_{JN_{i,j}} J_{i,j} \end{array} \right\} \tag{4.24}$$

The newly defined variables $w^{(t)}_{\dots}$ above can be viewed as "connection weights" as if in a neural network from $V^{(t-1)}_{\dots}$ or $J_{i,j}$ to $X^{(t)}_{i,j}$. Finally, we get the EJ formula similar to (4.16) as follows:

$$V^{(t)}_{i,j} = \begin{cases} \theta X^{(t)}_{i,j} + (1-\theta) V^{(t-2)}_{i,j} & \text{if } (i,j) \neq (i_G, j_G) \\ 0 & \text{if } (i,j) = (i_G, j_G) \end{cases} \tag{4.25}$$

where (i_G, j_G) is the ground position at which the voltage must be always zero. The forward calculation repeats (4.25) until the voltage solution $V^{(t)}_{i,j}$ converges.

4.3 Obtaining the Gradient

We have to calculate $\nabla_{\boldsymbol{r}} H$, the gradient of the cost H with respect to the resistivity distribution $\boldsymbol{r}$. Since the forward calculation (4.25) is a convergence calculation, we follow the method developed in Section 3.3. Namely, each element of the gradient $\nabla_{\boldsymbol{r}} H$ is calculated as the sum of an infinite series of partial derivatives as follows:

$$\frac{\partial H}{\partial r_{i,j}} = \sum_{t=-\infty}^{T} \frac{\partial H}{\partial r^{(t)}_{i,j}} \tag{4.26}$$

The summation in (4.26) starts at the top layer $t = T$ and goes down toward the negative t direction until it converges. In this section, we obtain the form for calculating each $\dfrac{\partial H}{\partial r^{(t)}_{i,j}}$ of (4.26) by back propagation.

As shown in (4.24), $r^{(t)}_{i,j}$ directly affects $X^{(t)}_{i,j}$ and also its neighbors $X^{(t)}_{i+1,j}$, $X^{(t)}_{i-1,j}$, $X^{(t)}_{i,j+1}$ and $X^{(t)}_{i,j-1}$ through the "connection weights" $w^{(t)}_{\dots}$. So, using the chain rule for partial differentiation, we can write:

$$\begin{aligned}\frac{\partial H}{\partial r_{i,j}^{(t)}} &= \frac{\partial H}{\partial X_{i,j}^{(t)}}\frac{\partial X_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}} + \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i+1,j}^{(t)}}\frac{\partial X_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}} \\ 0\end{array}\right\} + \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i-1,j}^{(t)}}\frac{\partial X_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}} \\ 0\end{array}\right\} \\ &+ \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i,j+1}^{(t)}}\frac{\partial X_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}} \\ 0\end{array}\right\} + \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i,j-1}^{(t)}}\frac{\partial X_{i,j-1}^{(t)}}{\partial r_{i,j}^{(t)}} \\ 0\end{array}\right\}\end{aligned} \tag{4.27}$$

This way, calculating $\frac{\partial H}{\partial r_{i,j}^{(t)}}$ is reduced to calculating $\frac{\partial H}{\partial X_{i,j}^{(t)}}$, $\frac{\partial X_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}$, and the effects of $r_{i,j}^{(t)}$ on the neighbors $\frac{\partial X_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}$, $\frac{\partial X_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}$, etc.

Let us first get $\frac{\partial H}{\partial X_{i,j}^{(t)}}$. As shown in (4.25), only $V_{i,j}^{(t)}$ is directly affected by $X_{i,j}^{(t)}$. Therefore,

$$\frac{\partial H}{\partial X_{i,j}^{(t)}} = \frac{\partial H}{\partial V_{i,j}^{(t)}}\frac{\partial V_{i,j}^{(t)}}{\partial X_{i,j}^{(t)}} = \begin{cases}\theta\frac{\partial H}{\partial V_{i,j}^{(t)}} & \text{if } (i,j) \neq (i_G, j_G) \\ 0 & \text{if } (i,j) = (i_G, j_G)\end{cases} \tag{4.28}$$

We have to get $\frac{\partial H}{\partial V_{i,j}^{(t)}}$ for (4.28). $V_{i,j}^{(t)}$ directly affects not only $X_{\dots}^{(t+1)}$ of the neighbors (see (4.24)) but also two layers higher $V_{i,j}^{(t+2)}$ (see (4.25)). Therefore, using the chain rule again,

$$\begin{aligned}\frac{\partial H}{\partial V_{i,j}^{(t)}} &= \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i+1,j}^{(t+1)}}\frac{\partial X_{i+1,j}^{(t+1)}}{\partial V_{i,j}^{(t)}} \\ 0\end{array}\right\} + \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i-1,j}^{(t+1)}}\frac{\partial X_{i-1,j}^{(t+1)}}{\partial V_{i,j}^{(t)}} \\ 0\end{array}\right\} \\ &+ \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i,j+1}^{(t+1)}}\frac{\partial X_{i,j+1}^{(t+1)}}{\partial V_{i,j}^{(t)}} \\ 0\end{array}\right\} + \left\{\begin{array}{c}\frac{\partial H}{\partial X_{i,j-1}^{(t+1)}}\frac{\partial X_{i,j-1}^{(t+1)}}{\partial V_{i,j}^{(t)}} \\ 0\end{array}\right\} + \frac{\partial H}{\partial V_{i,j}^{(t+2)}}\frac{\partial V_{i,j}^{(t+2)}}{\partial V_{i,j}^{(t)}}\end{aligned} \tag{4.29}$$

If the layer $(t+2)$ does not exist, i.e., when $T \geq t \geq T-1$ (T: top layer), the last term of (4.29) is ignored. From (4.25),

$$\frac{\partial H}{\partial X_{\bullet}^{(t+1)}} = \frac{\partial H}{\partial V_{\bullet}^{(t+1)}}\frac{\partial V_{\bullet}^{(t+1)}}{\partial X_{\bullet}^{(t+1)}} = \theta\frac{\partial H}{\partial V_{\bullet}^{(t+1)}} \tag{4.30}$$

where $\bullet$ is one of the four positions: $(i+1,j)$, $(i-1,j)$, $(i,j+1)$, or $(i,j-1)$. From (4.24), with i and j changed appropriately, we get the following:

$$\frac{\partial X_{i+1,j}^{(t+1)}}{\partial V_{i,j}^{(t)}} = w_{W_{i+1,j}}^{(t+1)} \qquad \frac{\partial X_{i-1,j}^{(t+1)}}{\partial V_{i,j}^{(t)}} = w_{E_{i-1,j}}^{(t+1)} \qquad \frac{\partial X_{i,j+1}^{(t+1)}}{\partial V_{i,j}^{(t)}} = w_{N_{i,j+1}}^{(t+1)} \qquad \frac{\partial X_{i,j-1}^{(t+1)}}{\partial V_{i,j}^{(t)}} = w_{S_{i,j-1}}^{(t+1)} \tag{4.31}$$

And from (4.25),

$$\frac{\partial V_{i,j}^{(t+2)}}{\partial V_{i,j}^{(t)}} = 1-\theta \tag{4.32}$$

Substituting (4.30), (4.31) and (4.32) into (4.29) yields:

$$\begin{aligned}\frac{\partial H}{\partial V_{i,j}^{(t)}} &= \theta\left(\left\{\begin{array}{c} w_{W_{i+1,j}}^{(t+1)}\dfrac{\partial H}{\partial V_{i+1,j}^{(t+1)}} \\ 0 \end{array}\right\} + \left\{\begin{array}{c} w_{E_{i-1,j}}^{(t+1)}\dfrac{\partial H}{\partial V_{i-1,j}^{(t+1)}} \\ 0 \end{array}\right\}\right. \\ &+ \left.\left\{\begin{array}{c} w_{N_{i,j+1}}^{(t+1)}\dfrac{\partial H}{\partial V_{i,j+1}^{(t+1)}} \\ 0 \end{array}\right\} + \left\{\begin{array}{c} w_{S_{i,j-1}}^{(t+1)}\dfrac{\partial H}{\partial V_{i,j-1}^{(t+1)}} \\ 0 \end{array}\right\}\right) + (1-\theta)\frac{\partial H}{\partial V_{i,j}^{(t+2)}}\end{aligned} \tag{4.33}$$

The partial derivatives $\dfrac{\partial H}{\partial V_{i,j}^{(t)}}$ at $t = T$ (the top layer) are given by the cost function as shown in (3.5), and they are back-propagated to determine those in lower layers by (4.33).

The other partial derivatives $\dfrac{\partial X_{\cdots}^{(t)}}{\partial r_{i,j}^{(t)}}$ for (4.27) are obtained as follows. Since $r_{i,j}^{(t)}$ is involved only in the "connection weights" $w_{\cdots}^{(t)}$ and not in $V_{\cdots}^{(t-1)}$ nor $J_{i,j}$ in (4.24), we get the following:

$$\begin{aligned}\frac{\partial X_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}} &= \left\{\begin{array}{c} \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{E_{i,j}}^{(t)}}\dfrac{\partial w_{E_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{JE_{i,j}}^{(t)}}\dfrac{\partial w_{JE_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} + \left\{\begin{array}{c} \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{W_{i,j}}^{(t)}}\dfrac{\partial w_{W_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{JW_{i,j}}^{(t)}}\dfrac{\partial w_{JW_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \\ &+ \left\{\begin{array}{c} \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{S_{i,j}}^{(t)}}\dfrac{\partial w_{S_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{JS_{i,j}}^{(t)}}\dfrac{\partial w_{JS_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} + \left\{\begin{array}{c} \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{N_{i,j}}^{(t)}}\dfrac{\partial w_{N_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ \dfrac{\partial X_{i,j}^{(t)}}{\partial w_{JN_{i,j}}^{(t)}}\dfrac{\partial w_{JN_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \\ &= \left\{\begin{array}{c} V_{i+1,j}^{(t-1)}\dfrac{\partial w_{E_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j}\dfrac{\partial w_{JE_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} + \left\{\begin{array}{c} V_{i-1,j}^{(t-1)}\dfrac{\partial w_{W_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j}\dfrac{\partial w_{JW_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}\end{aligned}$$

$$+\left\{\begin{array}{c} V_{i,j+1}^{(t-1)}\dfrac{\partial w_{S_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j}\dfrac{\partial w_{JS_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i,j-1}^{(t-1)}\dfrac{\partial w_{N_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j}\dfrac{\partial w_{JN_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \tag{4.34}$$

The partial derivative $\dfrac{\partial X_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}$, which is the effect of $r_{i,j}^{(t)}$ on the east neighbor's $X^{(t)}$, is obtained by adding 1 to i for everything but $r^{(t)}$ in (4.34) as follows:

$$\begin{aligned}\frac{\partial X_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}} &= \left\{\begin{array}{c} V_{i+2,j}^{(t-1)}\dfrac{\partial w_{E_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i+1,j}\dfrac{\partial w_{JE_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i,j}^{(t-1)}\dfrac{\partial w_{W_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i+1,j}\dfrac{\partial w_{JW_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \\ &+\left\{\begin{array}{c} V_{i+1,j+1}^{(t-1)}\dfrac{\partial w_{S_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i+1,j}\dfrac{\partial w_{JS_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i+1,j-1}^{(t-1)}\dfrac{\partial w_{N_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i+1,j}\dfrac{\partial w_{JN_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}\end{aligned} \tag{4.35}$$

Similarly, the partial derivatives of $X^{(t)}$ of the west, south and north neighbors with respect to $r_{i,j}^{(t)}$ are obtained as below:

$$\begin{aligned}\frac{\partial X_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}} &= \left\{\begin{array}{c} V_{i,j}^{(t-1)}\dfrac{\partial w_{E_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i-1,j}\dfrac{\partial w_{JE_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i-2,j}^{(t-1)}\dfrac{\partial w_{W_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i-1,j}\dfrac{\partial w_{JW_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \\ &+\left\{\begin{array}{c} V_{i-1,j+1}^{(t-1)}\dfrac{\partial w_{S_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i-1,j}\dfrac{\partial w_{JS_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i-1,j-1}^{(t-1)}\dfrac{\partial w_{N_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i-1,j}\dfrac{\partial w_{JN_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}\end{aligned} \tag{4.36}$$

$$\begin{aligned}\frac{\partial X_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}} &= \left\{\begin{array}{c} V_{i+1,j+1}^{(t-1)}\dfrac{\partial w_{E_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j+1}\dfrac{\partial w_{JE_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i-1,j+1}^{(t-1)}\dfrac{\partial w_{W_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j+1}\dfrac{\partial w_{JW_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \\ &+\left\{\begin{array}{c} V_{i,j+2}^{(t-1)}\dfrac{\partial w_{S_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j+1}\dfrac{\partial w_{JS_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i,j}^{(t-1)}\dfrac{\partial w_{N_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j+1}\dfrac{\partial w_{JN_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}\end{aligned} \tag{4.37}$$

$$\frac{\partial X_{i,j-1}^{(t)}}{\partial r_{i,j}^{(t)}} = \left\{\begin{array}{c} V_{i+1,j-1}^{(t-1)}\dfrac{\partial w_{E_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j-1}\dfrac{\partial w_{JE_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i-1,j-1}^{(t-1)}\dfrac{\partial w_{W_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j-1}\dfrac{\partial w_{JW_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}$$

$$+\left\{\begin{array}{c} V_{i,j}^{(t-1)}\frac{\partial w_{S_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j-1}\frac{\partial w_{JS_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\}+\left\{\begin{array}{c} V_{i,j-2}^{(t-1)}\frac{\partial w_{N_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \\ J_{i,j-1}\frac{\partial w_{JN_{i,j-1}}^{(t)}}{\partial r_{i,j}^{(t)}} \end{array}\right\} \tag{4.38}$$

The partial derivatives $\frac{\partial w_{\cdots}^{(t)}}{\partial r_{i,j}^{(t)}}$ are obtained as follows:

$$\left.\begin{array}{lllll}
\frac{\partial w_{E_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{E_{i,j}^{(t)}}{C_{i,j}^{(t)}}\right) & = & \frac{1}{C_{i,j}^{(t)}}\left(\frac{\partial E_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{E_{i,j}^{(t)}}{C_{i,j}^{(t)}}\frac{\partial C_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{W_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{W_{i,j}^{(t)}}{C_{i,j}^{(t)}}\right) & = & \frac{1}{C_{i,j}^{(t)}}\left(\frac{\partial W_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{W_{i,j}^{(t)}}{C_{i,j}^{(t)}}\frac{\partial C_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{S_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{S_{i,j}^{(t)}}{C_{i,j}^{(t)}}\right) & = & \frac{1}{C_{i,j}^{(t)}}\left(\frac{\partial S_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{S_{i,j}^{(t)}}{C_{i,j}^{(t)}}\frac{\partial C_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{N_{i,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{N_{i,j}^{(t)}}{C_{i,j}^{(t)}}\right) & = & \frac{1}{C_{i,j}^{(t)}}\left(\frac{\partial N_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{N_{i,j}^{(t)}}{C_{i,j}^{(t)}}\frac{\partial C_{i,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right)
\end{array}\right\} \tag{4.39}$$

$$\left.\begin{array}{lllll}
\frac{\partial w_{E_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{E_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\right) & = & \frac{1}{C_{i+1,j}^{(t)}}\left(\frac{\partial E_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{E_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\frac{\partial C_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{W_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{W_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\right) & = & \frac{1}{C_{i+1,j}^{(t)}}\left(\frac{\partial W_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{W_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\frac{\partial C_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{S_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{S_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\right) & = & \frac{1}{C_{i+1,j}^{(t)}}\left(\frac{\partial S_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{S_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\frac{\partial C_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{N_{i+1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{N_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\right) & = & \frac{1}{C_{i+1,j}^{(t)}}\left(\frac{\partial N_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{N_{i+1,j}^{(t)}}{C_{i+1,j}^{(t)}}\frac{\partial C_{i+1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right)
\end{array}\right\} \tag{4.40}$$

$$\left.\begin{array}{lllll}
\frac{\partial w_{E_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{E_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\right) & = & \frac{1}{C_{i-1,j}^{(t)}}\left(\frac{\partial E_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{E_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\frac{\partial C_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{W_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{W_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\right) & = & \frac{1}{C_{i-1,j}^{(t)}}\left(\frac{\partial W_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{W_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\frac{\partial C_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{S_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{S_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\right) & = & \frac{1}{C_{i-1,j}^{(t)}}\left(\frac{\partial S_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{S_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\frac{\partial C_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{N_{i-1,j}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{N_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\right) & = & \frac{1}{C_{i-1,j}^{(t)}}\left(\frac{\partial N_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{N_{i-1,j}^{(t)}}{C_{i-1,j}^{(t)}}\frac{\partial C_{i-1,j}^{(t)}}{\partial r_{i,j}^{(t)}}\right)
\end{array}\right\} \tag{4.41}$$

$$\left.\begin{array}{lllll}
\frac{\partial w_{E_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{E_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\right) & = & \frac{1}{C_{i,j+1}^{(t)}}\left(\frac{\partial E_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{E_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\frac{\partial C_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{W_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{W_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\right) & = & \frac{1}{C_{i,j+1}^{(t)}}\left(\frac{\partial W_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{W_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\frac{\partial C_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{S_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{S_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\right) & = & \frac{1}{C_{i,j+1}^{(t)}}\left(\frac{\partial S_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{S_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\frac{\partial C_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}\right) \\
\frac{\partial w_{N_{i,j+1}}^{(t)}}{\partial r_{i,j}^{(t)}} & \equiv & \frac{\partial}{\partial r_{i,j}^{(t)}}\left(\frac{N_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\right) & = & \frac{1}{C_{i,j+1}^{(t)}}\left(\frac{\partial N_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}-\frac{N_{i,j+1}^{(t)}}{C_{i,j+1}^{(t)}}\frac{\partial C_{i,j+1}^{(t)}}{\partial r_{i,j}^{(t)}}\right)
\end{array}\right\} \tag{4.42}$$

$$\left.\begin{array}{lclcl}
\frac{\partial w^{(t)}_{E_{i,j-1}}}{\partial r^{(t)}_{i,j}} & \equiv & \frac{\partial}{\partial r^{(t)}_{i,j}}\left(\frac{E^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\right) & = & \frac{1}{C^{(t)}_{i,j-1}}\left(\frac{\partial E^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} - \frac{E^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\frac{\partial C^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}}\right) \\
\frac{\partial w^{(t)}_{W_{i,j-1}}}{\partial r^{(t)}_{i,j}} & \equiv & \frac{\partial}{\partial r^{(t)}_{i,j}}\left(\frac{W^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\right) & = & \frac{1}{C^{(t)}_{i,j-1}}\left(\frac{\partial W^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} - \frac{W^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\frac{\partial C^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}}\right) \\
\frac{\partial w^{(t)}_{S_{i,j-1}}}{\partial r^{(t)}_{i,j}} & \equiv & \frac{\partial}{\partial r^{(t)}_{i,j}}\left(\frac{S^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\right) & = & \frac{1}{C^{(t)}_{i,j-1}}\left(\frac{\partial S^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} - \frac{S^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\frac{\partial C^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}}\right) \\
\frac{\partial w^{(t)}_{N_{i,j-1}}}{\partial r^{(t)}_{i,j}} & \equiv & \frac{\partial}{\partial r^{(t)}_{i,j}}\left(\frac{N^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\right) & = & \frac{1}{C^{(t)}_{i,j-1}}\left(\frac{\partial N^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} - \frac{N^{(t)}_{i,j-1}}{C^{(t)}_{i,j-1}}\frac{\partial C^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}}\right)
\end{array}\right\} \quad (4.43)$$

From (4.17), we get the following:

$$\left.\begin{array}{ll}
\frac{\partial E^{(t)}_{i,j}}{\partial r^{(t)}_{i,j}} = -\frac{h_x}{2h_y}(E^{(t)}_{i,j})^2 & \frac{\partial W^{(t)}_{i,j}}{\partial r^{(t)}_{i,j}} = -\frac{h_x}{2h_y}(W^{(t)}_{i,j})^2 \\
\frac{\partial S^{(t)}_{i,j}}{\partial r^{(t)}_{i,j}} = -\frac{h_y}{2h_x}(S^{(t)}_{i,j})^2 & \frac{\partial N^{(t)}_{i,j}}{\partial r^{(t)}_{i,j}} = -\frac{h_y}{2h_x}(N^{(t)}_{i,j})^2
\end{array}\right\} \quad (4.44)$$

$$\frac{\partial C^{(t)}_{i,j}}{\partial r^{(t)}_{i,j}} = -\left\{\begin{array}{c}\frac{h_x}{2h_y}(E^{(t)}_{i,j})^2 \\ 0\end{array}\right\} - \left\{\begin{array}{c}\frac{h_x}{2h_y}(W^{(t)}_{i,j})^2 \\ 0\end{array}\right\} - \left\{\begin{array}{c}\frac{h_y}{2h_x}(S^{(t)}_{i,j})^2 \\ 0\end{array}\right\} - \left\{\begin{array}{c}\frac{h_y}{2h_x}(N^{(t)}_{i,j})^2 \\ 0\end{array}\right\} \quad (4.45)$$

$$\frac{\partial E^{(t)}_{i+1,j}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial W^{(t)}_{i+1,j}}{\partial r^{(t)}_{i,j}} = -\frac{h_x}{2h_y}(E^{(t)}_{i,j})^2 \qquad \frac{\partial S^{(t)}_{i+1,j}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial N^{(t)}_{i+1,j}}{\partial r^{(t)}_{i,j}} = 0 \quad (4.46)$$

$$\frac{\partial C^{(t)}_{i+1,j}}{\partial r^{(t)}_{i,j}} = -\left\{\begin{array}{c}\frac{h_x}{2h_y}(E^{(t)}_{i,j})^2 \\ 0\end{array}\right\} \quad (4.47)$$

$$\frac{\partial E^{(t)}_{i-1,j}}{\partial r^{(t)}_{i,j}} = -\frac{h_x}{2h_y}(W^{(t)}_{i,j})^2 \qquad \frac{\partial W^{(t)}_{i-1,j}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial S^{(t)}_{i-1,j}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial N^{(t)}_{i-1,j}}{\partial r^{(t)}_{i,j}} = 0 \quad (4.48)$$

$$\frac{\partial C^{(t)}_{i-1,j}}{\partial r^{(t)}_{i,j}} = -\left\{\begin{array}{c}\frac{h_x}{2h_y}(W^{(t)}_{i,j})^2 \\ 0\end{array}\right\} \quad (4.49)$$

$$\frac{\partial E^{(t)}_{i,j+1}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial W^{(t)}_{i,j+1}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial S^{(t)}_{i,j+1}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial N^{(t)}_{i,j+1}}{\partial r^{(t)}_{i,j}} = -\frac{h_y}{2h_x}(S^{(t)}_{i,j})^2 \quad (4.50)$$

$$\frac{\partial C^{(t)}_{i,j+1}}{\partial r^{(t)}_{i,j}} = -\left\{\begin{array}{c}\frac{h_y}{2h_x}(S^{(t)}_{i,j})^2 \\ 0\end{array}\right\} \quad (4.51)$$

$$\frac{\partial E^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial W^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} = 0 \qquad \frac{\partial S^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} = -\frac{h_y}{2h_x}(N^{(t)}_{i,j})^2 \qquad \frac{\partial N^{(t)}_{i,j-1}}{\partial r^{(t)}_{i,j}} = 0 \quad (4.52)$$

$$\frac{\partial C_{i,j-1}^{(t)}}{\partial r_{i,j}^{(t)}} = -\left\{ \begin{array}{c} \frac{h_y}{2h_x}(N_{i,j}^{(t)})^2 \\ 0 \end{array} \right\} \tag{4.53}$$

4.4 Implementation

The estimation methods for EIT (electrical impedance tomography) using the multiresolution optimization as well as the conventional single-resolution optimization were implemented in MPL (the MasPar Programming Language) on the MasPar MP-2204 massively parallel computer that has $64 \times 64 = 4096$ parallel processors.

The Polak-Ribiere formula shown in (2.13), page 18, and (2.26), page 21, is chosen for the conjugate gradient method. Back propagation is executed once per line minimization to obtain the gradient to determine the line search direction. The line minimization algorithm in the conjugate gradient method is based on so-called *Brent's method* [30] that uses the parabolic interpolation for searching a minimum whenever appropriate and the golden section search otherwise.

All the parameters and the variables are calculated in double precision. The stopping criterion for a forward voltage solution is set as small as 10^{-12} for the relative change to the maximum magnitude of voltage values in order to conduct the experiments as accurately as possible. The stopping criterion for the gradient calculation in the back-propagation algorithm is set to 10^{-10} for the relative change. The stopping criterion for line minimization is set to 10^{-3} for the bracket width relative to the initial bracket at the start of each line minimization. The extrapolation parameter θ in the EJ (extrapolated Jacobi-iterative) method (4.25) is chosen to be $\theta = 1.99$ based on the preliminary experiments.

During each line minimization, a number of distributions of the parameter have to be evaluated until a line minimum is reached. Each such evaluation involves the forward convergence calculation to obtain a voltage solution. To expedite the forward convergence calculation, the old voltage solutions are interpolated/extrapolated on the minimization line by a parabolic fit to obtain as good an initial voltage distribution as possible for the forward calculation. Still, each line minimization takes about 2 CPU-minutes on the MasPar MP-2204.

All the graphs of the experimental results in this chapter are processed by the tool "gnuplot" in UNIX. The pictures of the resistivity distributions are prepared by using MATLAB^{TM} of the MathWorks, Inc.

4.5 Experiments

4.5.1 Experimental Conditions

In EIT (electrical impedance tomography), measurement methods can be partitioned into two classes. One class injects electrical current to the medium and measures the resulting voltage on the boundary. The other class applies voltage and measures the resulting current on the boundary. The former (current injection) generally allows more accurate measurements than the latter (voltage input method) because the contact impedance between the electrodes and the object surface is negligible for a current source [41]. We choose here the current injection method used by most EIT research groups. In all of the following experiments, voltage/current measurements are computer-simulated ones.

If the medium is a disk and its resistivity distribution is homogeneous, injecting currents that vary as a cosine curve along the boundary gives a perfectly uniform current distribution and hence maximizes distinguishability [41]. In our case, although the medium is a square and the resistivity distribution is not homogeneous, cosine-varying currents are used as approximately best current patterns.

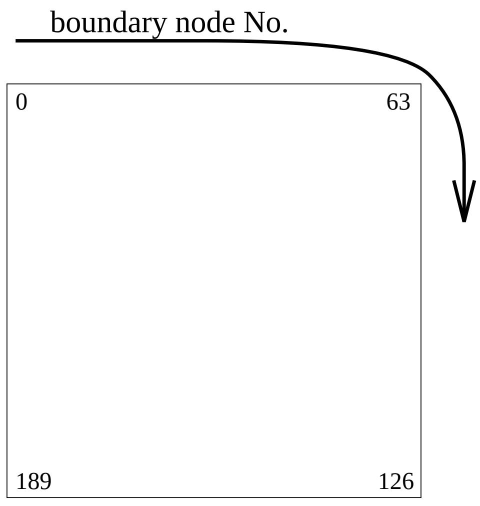


Figure 4.1: Node numbering

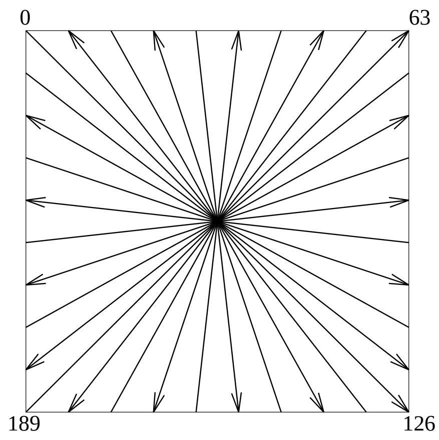


Figure 4.2: Current directions

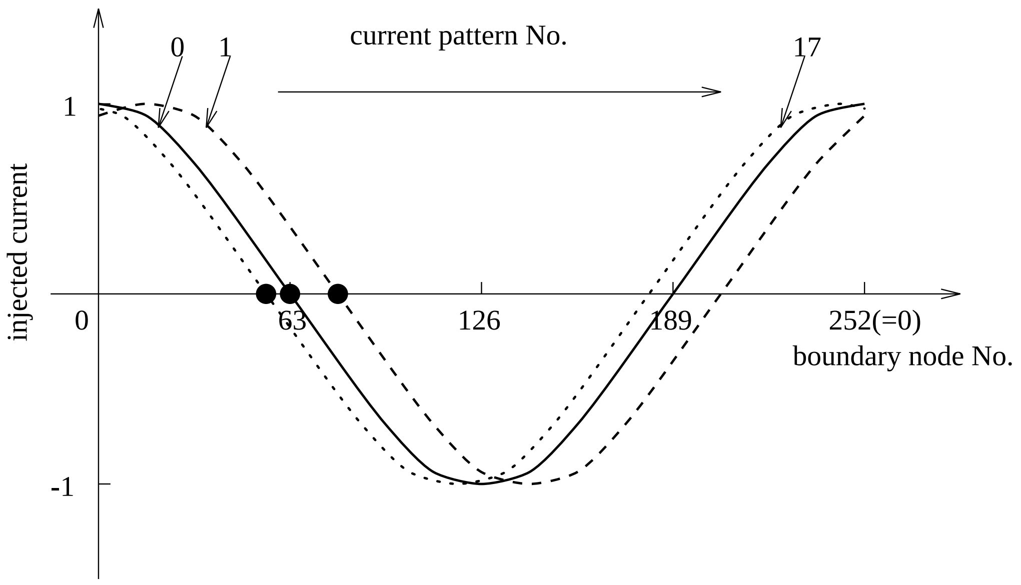


Figure 4.3: Injection current patterns

Since the field is a square with $64 \times 64 = 4096$ discretized nodes, the number of nodes along the boundary is $63 \times 4 = 252$. We assume that current injection and voltage measurement are performed on all the boundary nodes except one grounded node as the voltage reference. Therefore, one measurement obtains 251 voltage values, and determining 4096 values of the resistivity requires $4096/251 \approx 16.3$ or more measurements on different current patterns.

We choose 18 current patterns as follows. The boundary nodes are numbered b from 0 to 251 as shown in Figure 4.1. The peak current position b_{peak} for current pattern μ is determined as follows:

$$b_{\text{peak}} = \begin{cases} 14\mu & \text{if } 0 \leq \mu < 9 \\ 14\mu + 7 & \text{if } 9 \leq \mu < 18 \end{cases} \tag{4.54}$$

Each current pattern is determined so that the peak current 1 is assigned to the boundary node $b = b_{\text{peak}}$ and the currents at the rest of the nodes shape a cosine curve as the following:

$$J_b = \cos\left(2\pi \frac{(b - b_{\text{peak}})}{252}\right) \tag{4.55}$$

where J_b is the injection current at boundary node b. Figure 4.2 shows the current flow directions of all the 18 patterns, where the tail of each arrow is at the peak current node. The node 90 degrees clockwise from the peak-current node b_{peak} is grounded as the voltage reference (zero volt). Figure 4.3 shows the current patterns, in which the black dots show the grounded nodes.

Figures 4.4 and 4.5 show the two patterns of true resistivity distributions that are used for the experiments. In both patterns, the darkest regions are 0.2 in resistivity value, the lightest regions 1.0, and the background is 0.5. In Figure 4.4 which we call “Pattern A”, the upper and lower disks are 0.3, and the left and right disks 0.7. In “Pattern B”, Figure 4.5, all the disks are either 0.2 or 1.0. The same gray-level mapping will be used for displaying the estimated resistivity distributions in the experiments below.

The experiments are performed in two stages. The first stage simulates the actual measurements by solving the forward problem on the true resistivity distribution (either Pattern A or Pattern B) to obtain the voltage values on the boundary for

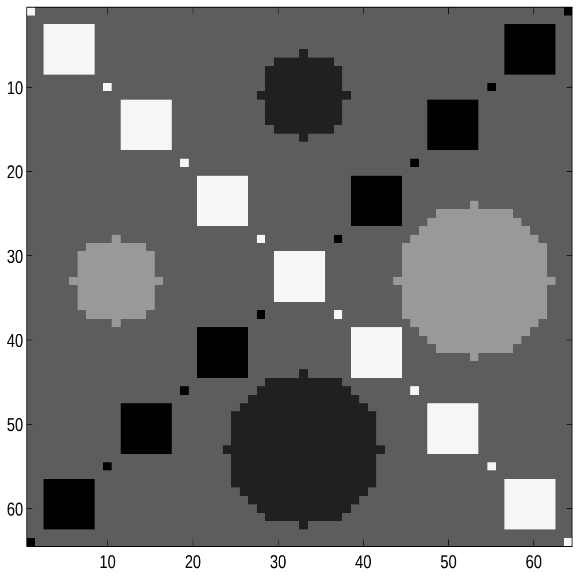


Figure 4.4: Pattern A

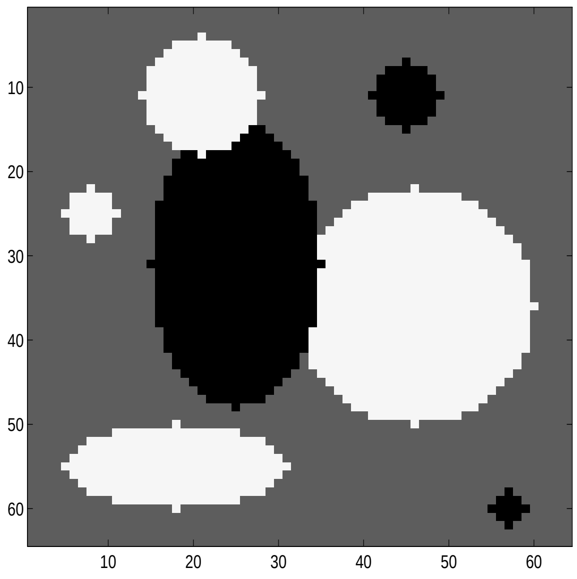


Figure 4.5: Pattern B

each of the 18 injection current patterns. Then, the second stage feeds the simulated voltage data to the estimation program to be evaluated and executes the program to reconstruct the true resistivity distribution.

The cost is calculated by (3.6) in page 66 only on the boundary nodes to simulate the actual EIT that can perform measurements only on the surface of the object. Thus, voltage values of the internal nodes are neglected. As a result, the derivatives $\left.\frac{\partial H}{\partial V_{\cdots}^{(T)}}\right|_{\mu}$ of (3.7) in page 66 are calculated only on the boundary nodes, and those derivatives of the internal nodes are always zero.

In the step scheme using the two-dimensional DFT, each of the transform coefficients has two frequencies associated with it; the frequency in the horizontal direction and the frequency in the vertical direction. In the experiments, the frequency limit in each resolution step is set such that both of the two frequency values have to be smaller than or equal to the limit in order for the coefficient to be activated in the search. In the weight scheme with the two-dimensional DFT, the frequency value used to determine the weight is the higher of the two frequencies associated with each transform coefficient.

4.5.2 Conventional Single-resolution Method

Figure 4.6 (a) shows the history of the cost in the estimation process. The horizontal axis is the number of line minimizations in the conjugate gradient method, and the vertical axis is the cost. Since the conjugate gradient method always proceeds in a cost-decreasing direction, the cost always goes down as more line minimizations are performed.

Figure 4.6 (b) shows the history of the distance to the true distribution. The distance is defined as follows:

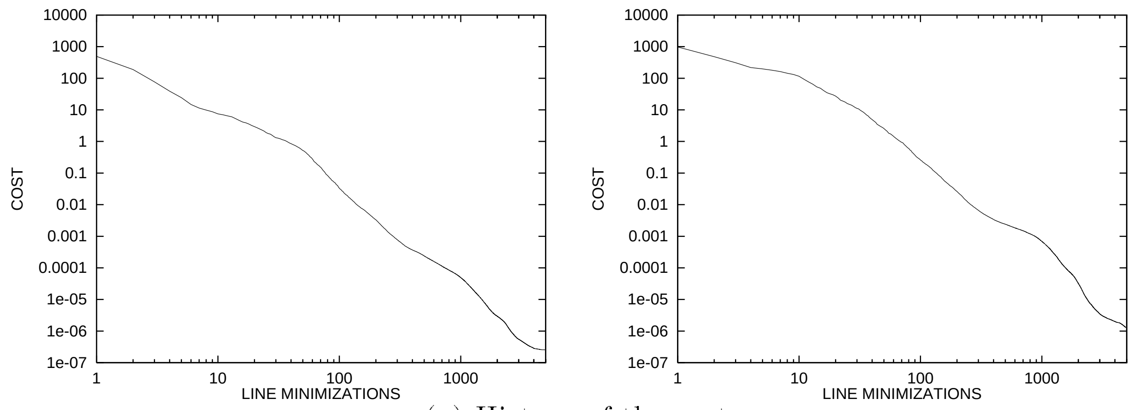
$$\text{distance} \triangleq \sqrt{\sum_{i=0}^{63}\sum_{j=0}^{63}(r_{i,j} - \hat{r}_{i,j})^2} \tag{4.56}$$

where $r_{i,j}$ is an element (i, j) of the estimated parameter distribution $\boldsymbol{r}$, and $\hat{r}_{i,j}$ is the corresponding element of the true parameter distribution $\hat{\boldsymbol{r}}$ that is either Pattern A or Pattern B of Figures 4.4 and 4.5. Of course, the estimation program cannot know the true distribution $\hat{\boldsymbol{r}}$. Hence, the distance is calculated outside the estimation process.

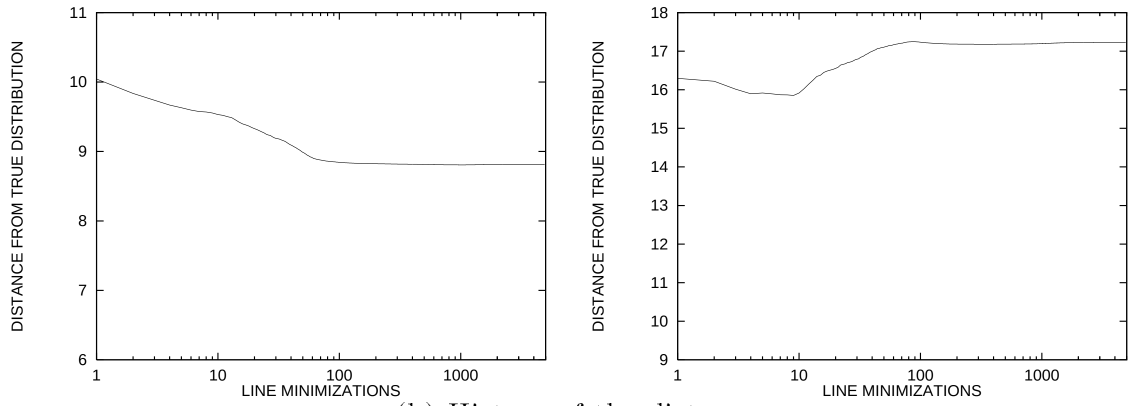
We can see in Figure 4.6 (b) that the progress in approaching the true distribution almost stops around 100 line minimizations for Pattern A, and the distance even increases for Pattern B. It is not actually caused by being caught in a local minimum, since the cost keeps decreasing as seen in Figure 4.6 (a). It is just that the search takes a path that rapidly decreases the cost but is not really directed toward the true minimum in this single-resolution estimation.

Figure 4.6 (c) shows the relation between the distance and the cost. The vertical axis is the cost raised to the power 0.05. The value 0.05 is chosen only so that the graph curves can be easily compared, and the number itself does not have any special

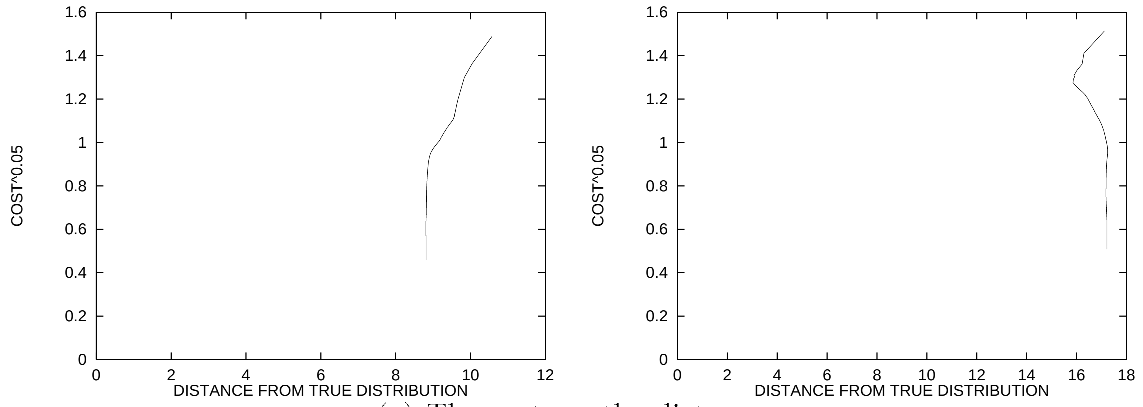
[Pattern A] **[Pattern B]**



(a) History of the cost

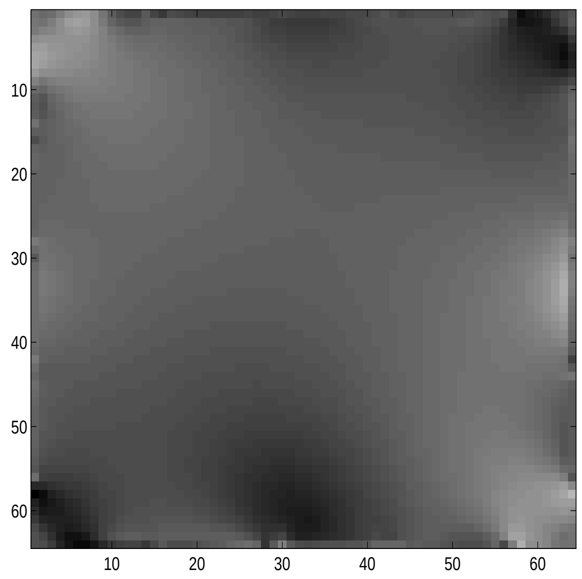


(b) History of the distance

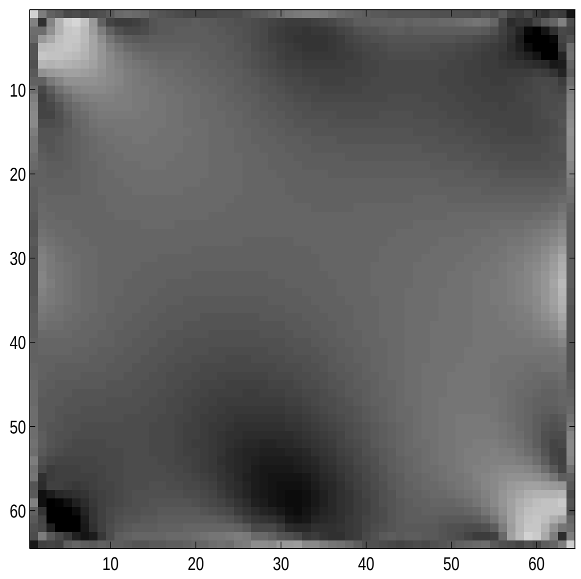


(c) The cost vs. the distance

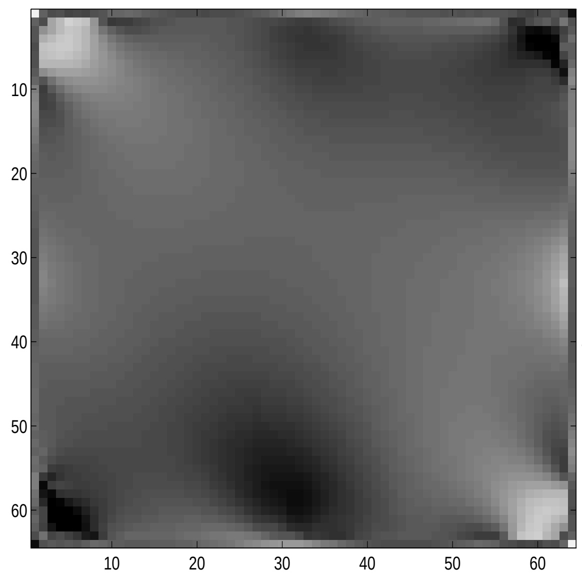
Figure 4.6: History of single-resolution estimation



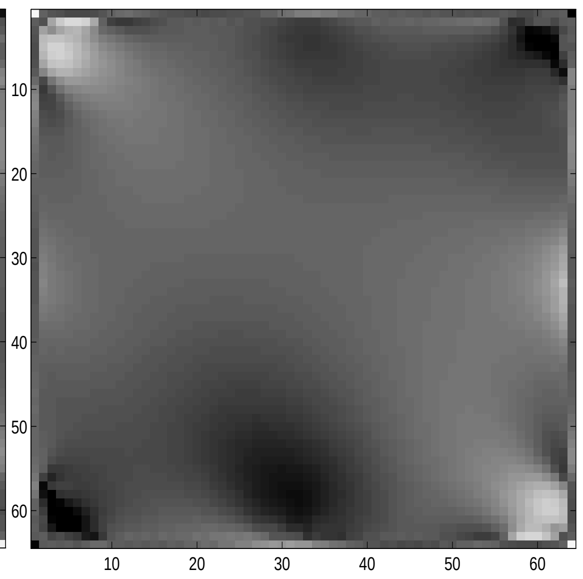
(a) after 20 line minimizations



(b) after 100 line minimizations

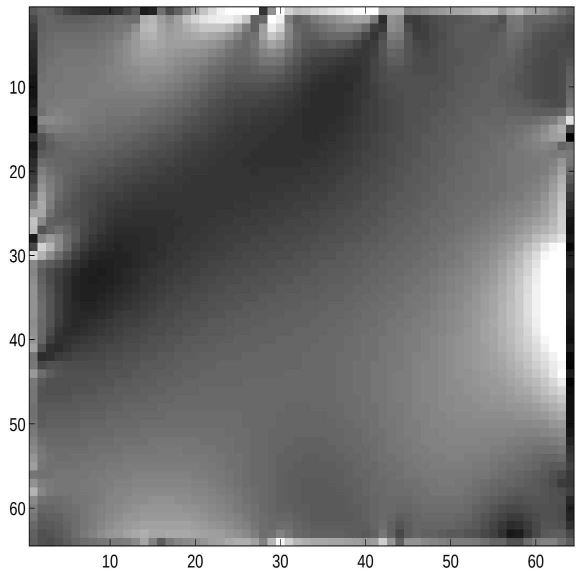


(c) after 500 line minimizations

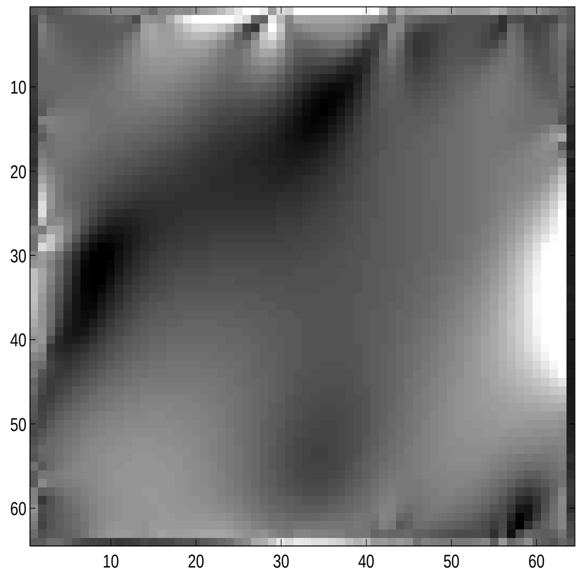


(d) after 5000 line minimizations

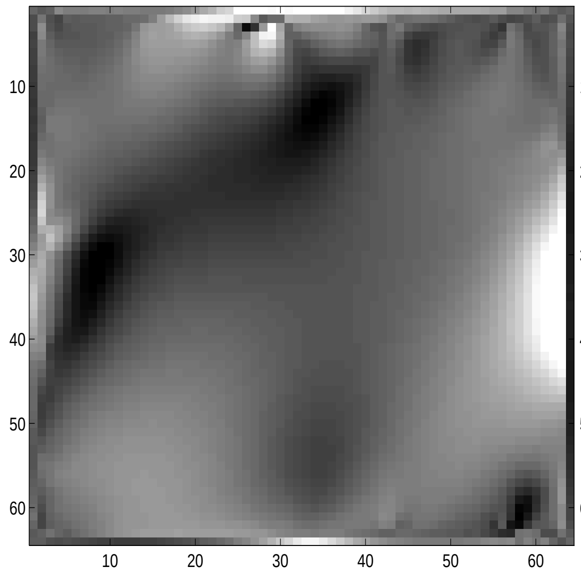
Figure 4.7: Single-resolution method (Pattern A)



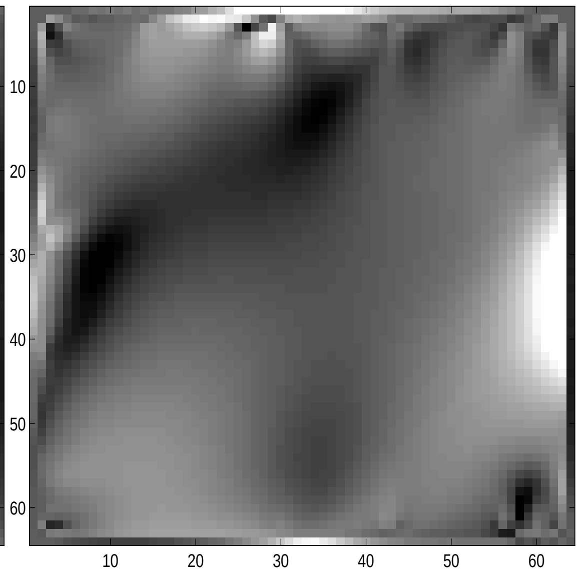
(a) after 20 line minimizations



(b) after 100 line minimizations



(c) after 500 line minimizations



(d) after 5000 line minimizations

Figure 4.8: Single-resolution method (Pattern B)

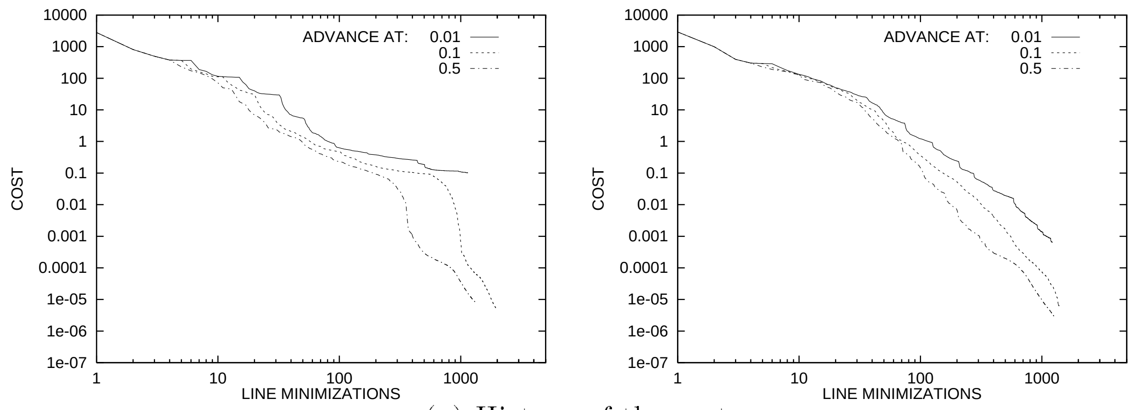
meaning. In these cost-versus-distance graphs, the search history starts at the upper right and goes toward the lower left if the distance decreases. If the search were perfect, it would reach the origin of the cost-versus-distance graph, because the cost should be zero with the distance being zero ($\boldsymbol{r} = \hat{\boldsymbol{r}}$) in these simulated experiments. In this sense, the more the graph is directed toward the origin, the better the search performance is. As seen in Figure 4.6 (c), the performance of the single-resolution estimation is poor, since the graph is going almost only downwards, not toward the origin, for both Pattern A and Pattern B. In other words, the single-resolution method is only good at reducing the cost, and poor at approaching the true distribution.

Figures 4.7 and 4.8 show the progress of the estimated resistivity distribution $\boldsymbol{r}$ for Pattern A and Pattern B, respectively. Comparing these pictures with the true distributions Figures 4.4 and 4.5, we can easily see that the estimation results are poor. For Pattern A (Figure 4.7), only regions near the external boundary are somewhat close to the true distribution, and the inner regions do not even show traces of the pattern. This is understandable because the measurements are performed only on the external boundary and thus the information about the inner part may be diluted in the measured data. The results for Pattern B (Figure 4.8) are even worse. The estimated pattern is not close to the true one at all.

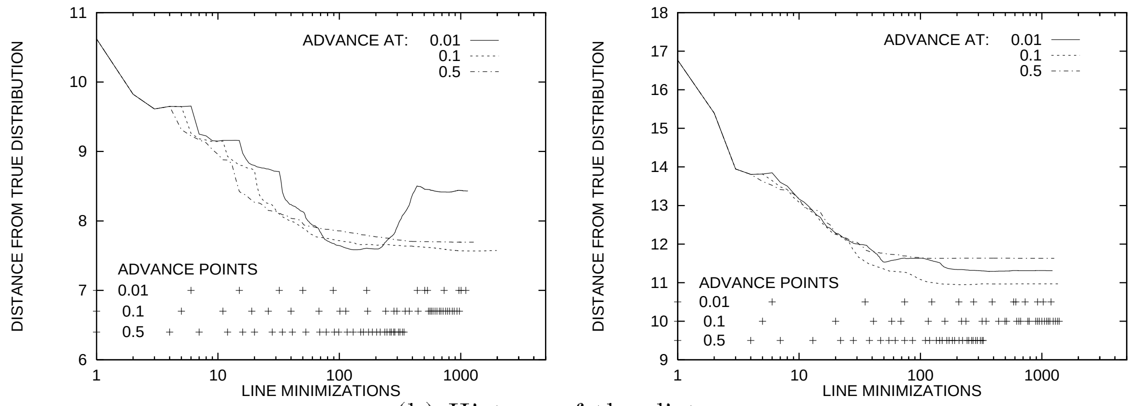
4.5.3 Step Scheme with the Fourier Transform

Figure 4.9 shows the estimation history of the step scheme that uses the Fourier transform. The numbers shown in the graphs are the values of C in the advancing criterion (2.7) in page 13. In (a), the cost goes down in a similar manner as in the single-resolution case Figure 4.6. For both Pattern A and Pattern B, the greater the advancing criterion constant is, the more quickly the cost decreases. This may be because the search can advance to a greater degree of freedom more quickly with

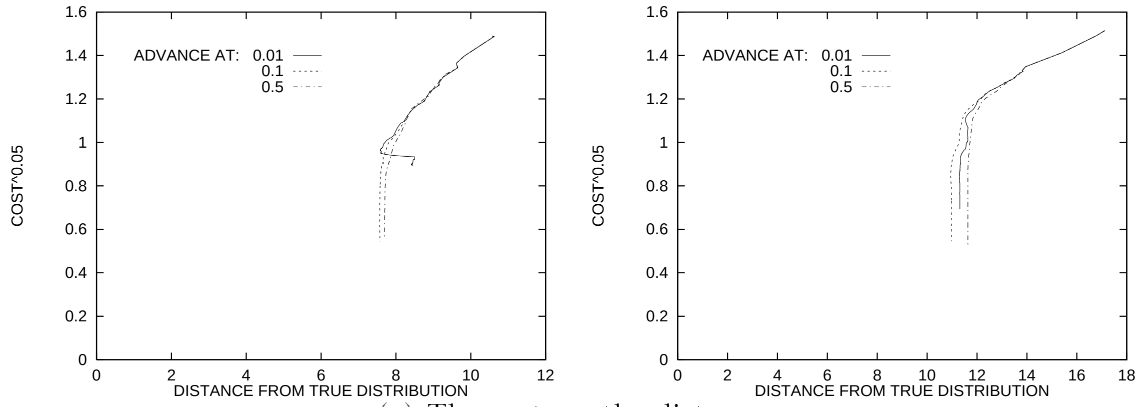
[Pattern A] **[Pattern B]**



(a) History of the cost

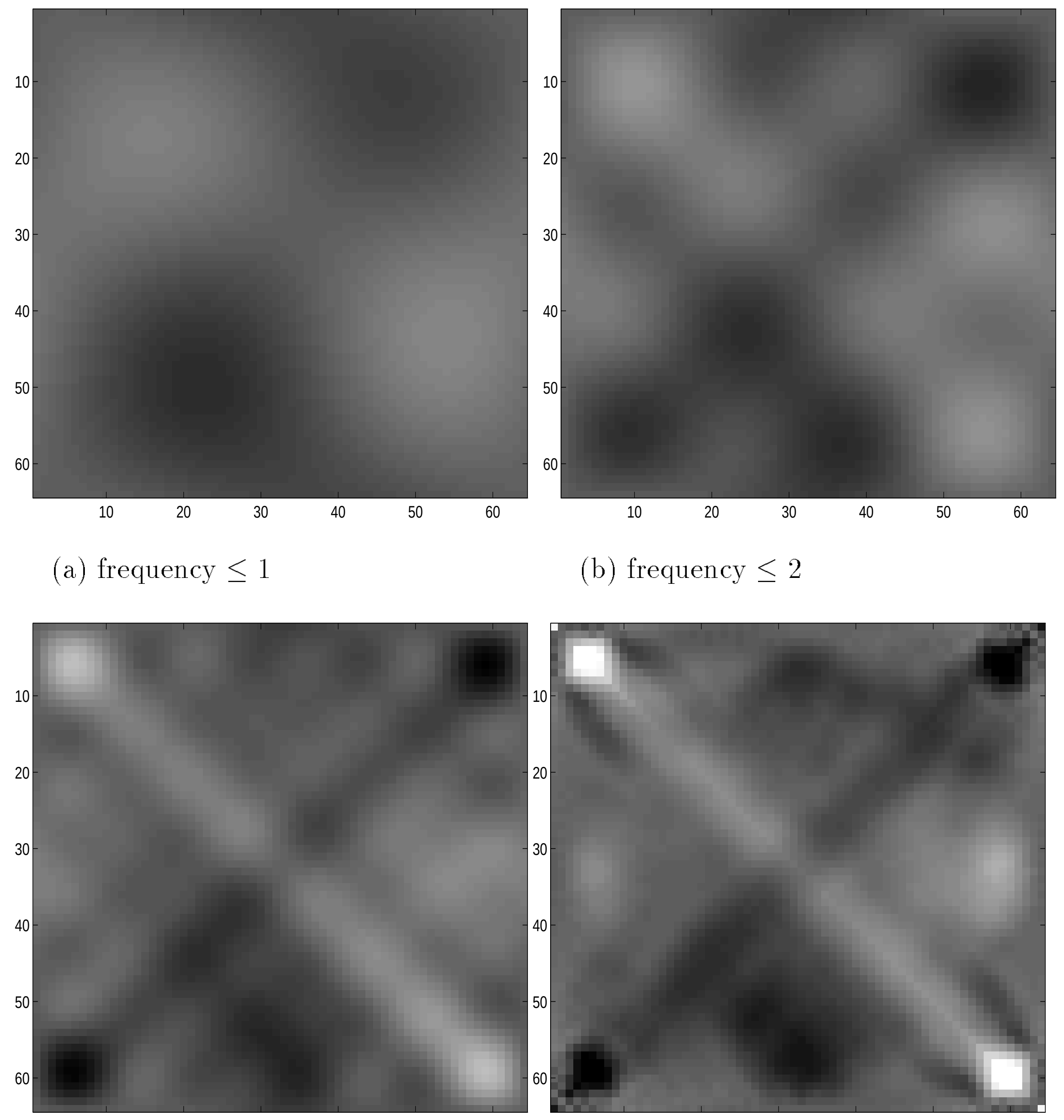


(b) History of the distance



(c) The cost vs. the distance

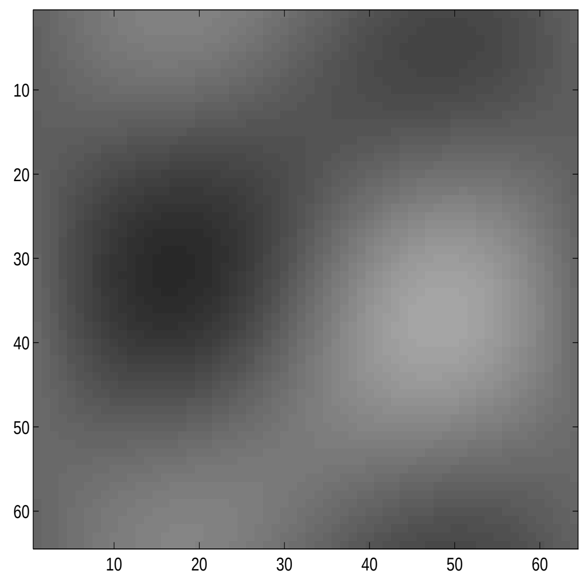
Figure 4.9: History of Fourier step scheme



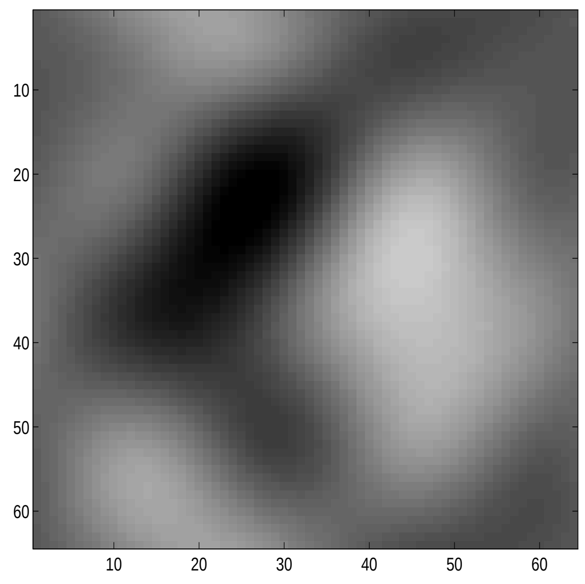
(a) frequency ≤ 1 (b) frequency ≤ 2

(c) frequency ≤ 4 (d) at 1000 line minimizations

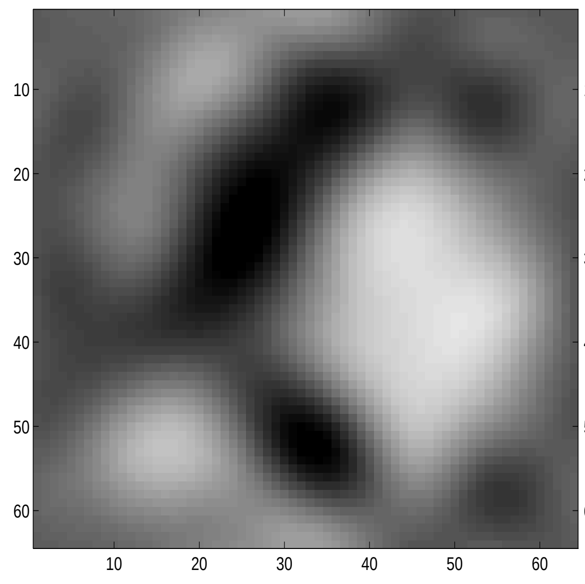
Figure 4.10: Fourier step scheme (Pattern A, advance=0.1)



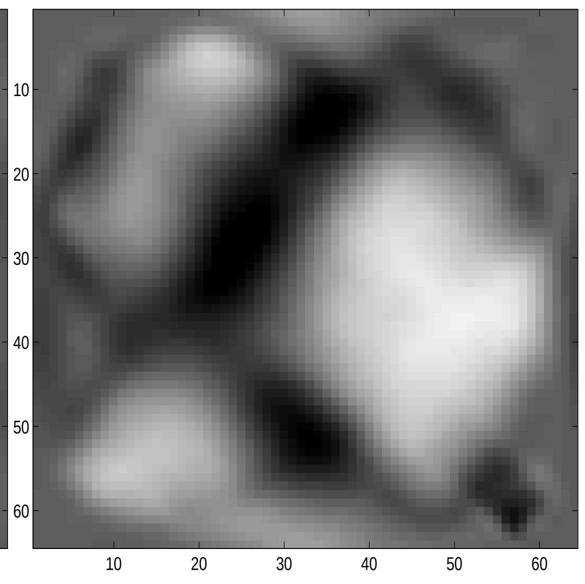
(a) frequency ≤ 1



(b) frequency ≤ 2



(c) frequency ≤ 4



(d) at 1000 line minimizations

Figure 4.11: Fourier step scheme (Pattern B, advance=0.1)

a greater advancing constant. The levels of the cost curves, especially for Pattern A, are a little higher than those of the single-resolution method after around 100 line minimizations. The distance curves in Figure 4.9 (b), however, are much lower than in the single-resolution method, which means the distributions estimated by the Fourier step scheme are much better than those by the conventional single-resolution method. The + marks in the (b) graphs show the positions of the advancing points on the horizontal axis. We can see the distance curves descend stepwise each time the advancing occurs. The greater the advance setting C is, the earlier the advancing occurs. Since the field has 64×64 discretized points, the highest frequency is 32, and there are 33 frequency steps from zero to the highest frequency (see Table 2.1, page 25). We can see that if the advancing constant is too small (0.01), the distance may increase at some point in the estimation, although the cost goes down smoothly. If the advancing constant is too large (0.5), the distance does not reduce well. In the cost-versus-distance graphs (c), the curves go farther toward the origin than in the single-resolution case.

Figure 4.10 shows the distributions estimated by the Fourier step scheme on Pattern A. The picture (a) is estimated with the frequency limit 1, (b) with 2, and (c) with 4, respectively. It is seen that as the frequency limit increases, the more detailed structure of the distribution shows up. The picture (c), which is only after 26 line minimizations performed, already shows the structure of the true distribution more clearly than the estimate of the single-resolution method after 5000 line minimizations (Figure 4.7, page 93). The picture (d) is the estimate at 1000 line minimizations with the frequency limit being 32, i.e., with the full frequency range. It is still vague compared with the true one (Figure 4.4, page 90), but is much better than the estimate of the single-resolution method (Figure 4.7).

Figure 4.11 shows the estimated distributions on Pattern B. The pictures (a), (b) and (c) have the same frequency limits as in Figure 4.10. The picture (d) is at 1000 line minimizations and its frequency limit is still 23. Although it is lower than the full frequency, the estimate is as good as the one with the full frequency, since the distance curve is almost flat around 1000 line minimizations as seen in Figure 4.9 (b). Comparing with the true distribution Figure 4.5 in page 90, we can see that some regions are wrong. For example, the dark regions around (horizontal,vertical)= $(35, 10)$ and $(35, 50)$ in the estimate should not be there. But overall, the estimate is much closer to the true distribution than the estimate by the single-resolution method, Figure 4.8, page 94.

4.5.4 Weight Scheme with the Fourier Transform

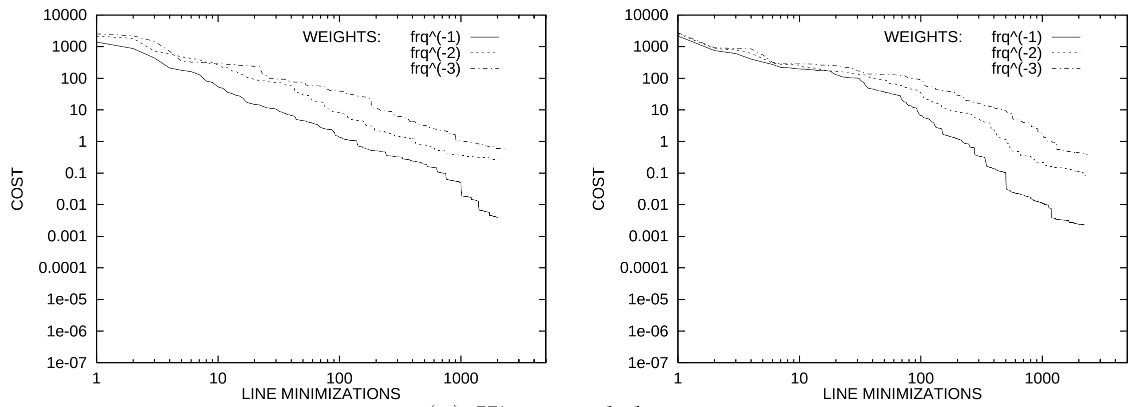
Figure 4.12 shows the estimation history of the weight scheme with the DFT. I used the following formula to determine the weight for each of the transform coefficients,

$$\text{weight} = \frac{1}{(\text{frequency})^{P_F}} \tag{4.57}$$

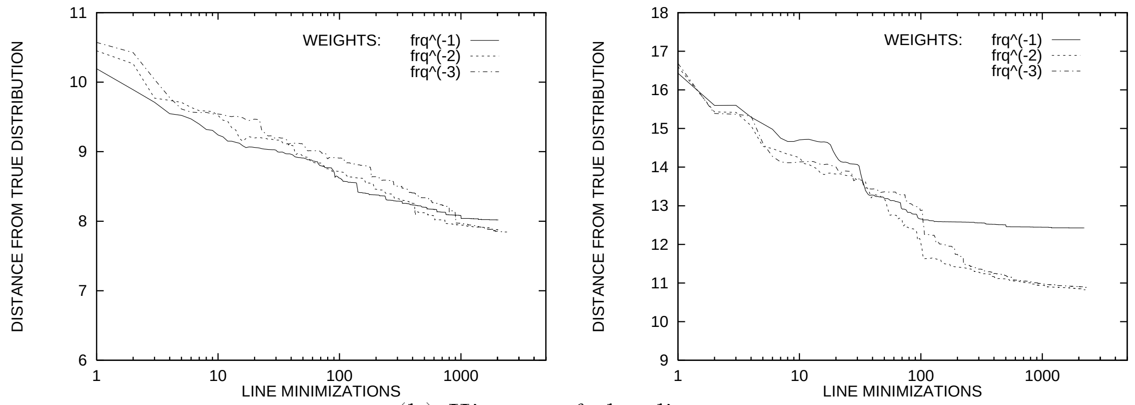
where "frequency" is the higher of the two frequencies associated with each coefficient as discussed in Section 4.5.1. The lowest frequency $= 0$ was changed to 0.5 for this formula. The real constant P_F can be any positive number for enhancing the relative sensitivity of the search to low-frequency information. The greater the value of P_F is, the more prioritized the low-frequency information is relative to the higher-frequency information. In this experiment, P_F was chosen to be 1, 2, or 3.

Figure 4.12 (a) shows the history of the cost. For both Pattern A and Pattern B, the cost descent is slower with a greater magnitude of the constant P_F. However, as Figure 4.12 (b) shows, the effect of the magnitude of P_F on the descent of the distance is different. Especially for Pattern B, the distance goes down faster with $P_F = 2$ or 3 than with $P_F = 1$. For both Pattern A and Pattern B, the distance descent is not as quick as in the Fourier step scheme Figure 4.9 (b), page 96. It is interesting, though,

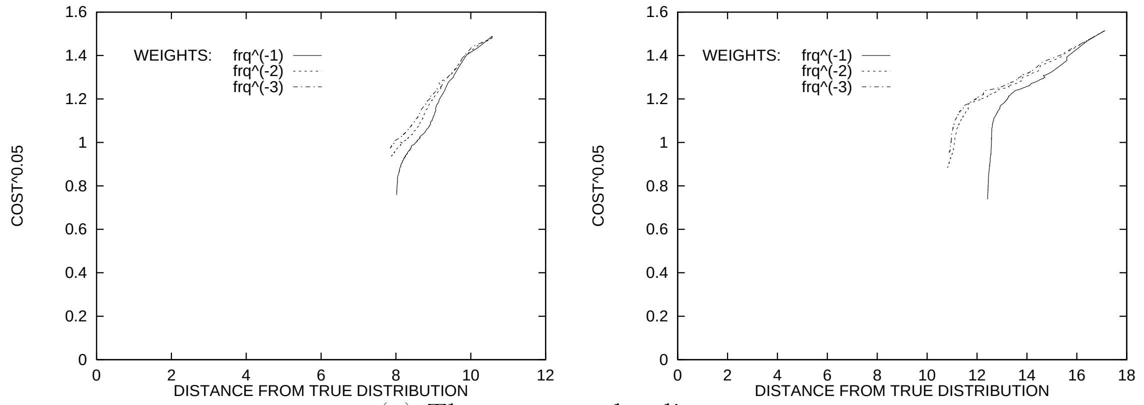
[Pattern A] **[Pattern B]**



(a) History of the cost

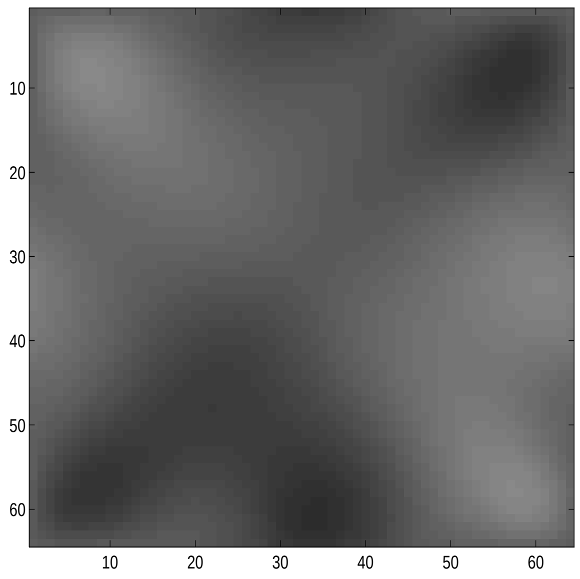


(b) History of the distance

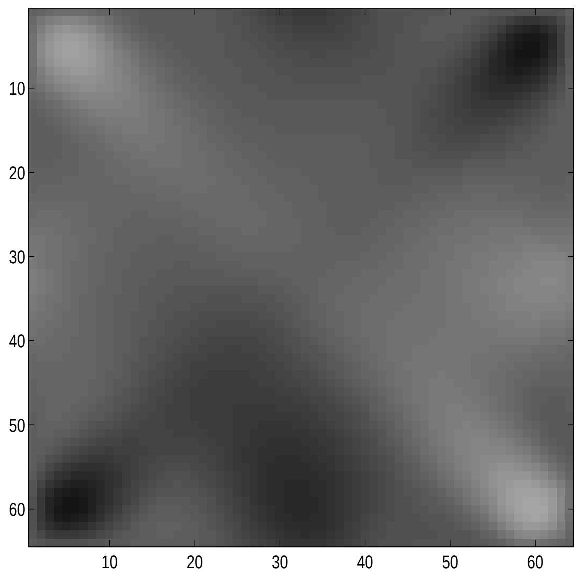


(c) The cost vs. the distance

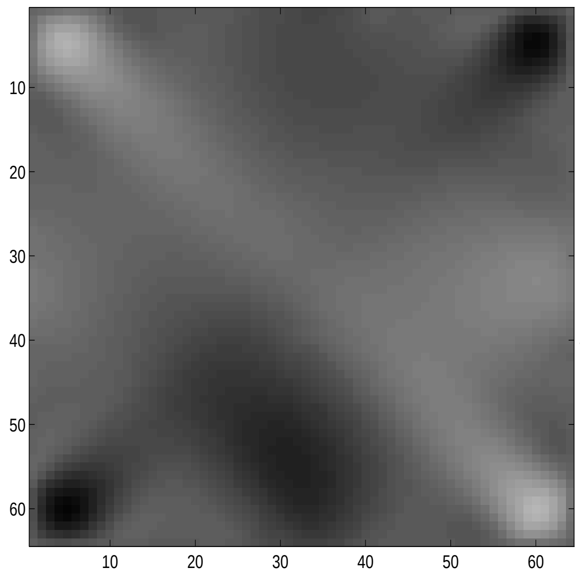
Figure 4.12: History of Fourier weight scheme



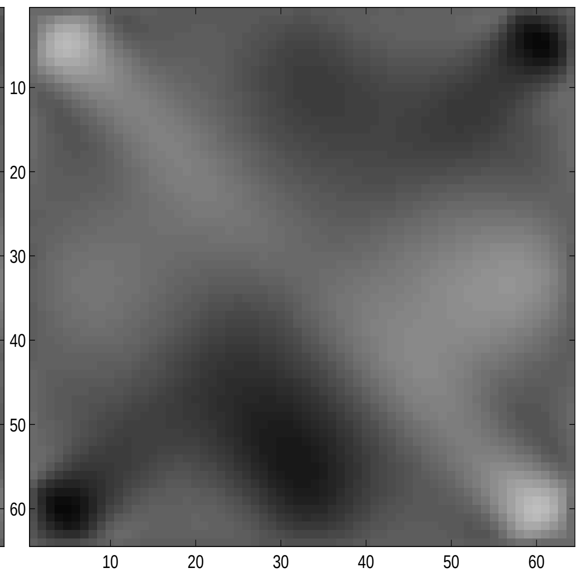
(a) after 30 line minimizations



(b) after 100 line minimizations

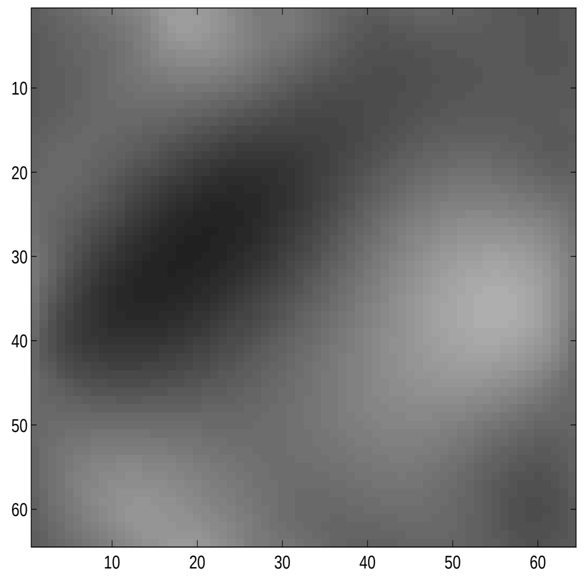


(c) after 300 line minimizations

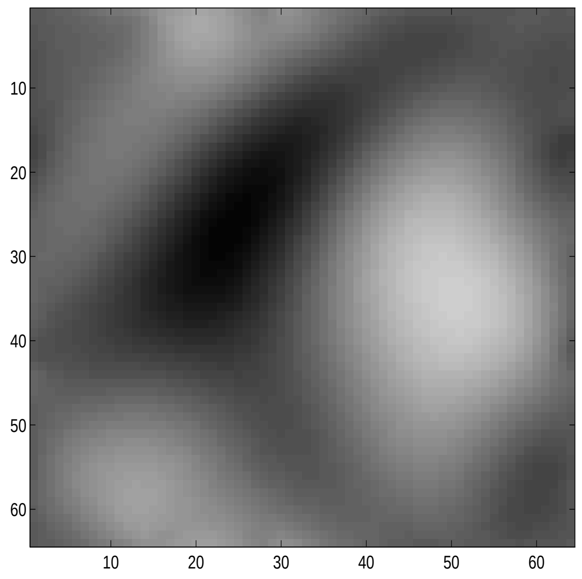


(d) after 1000 line minimizations

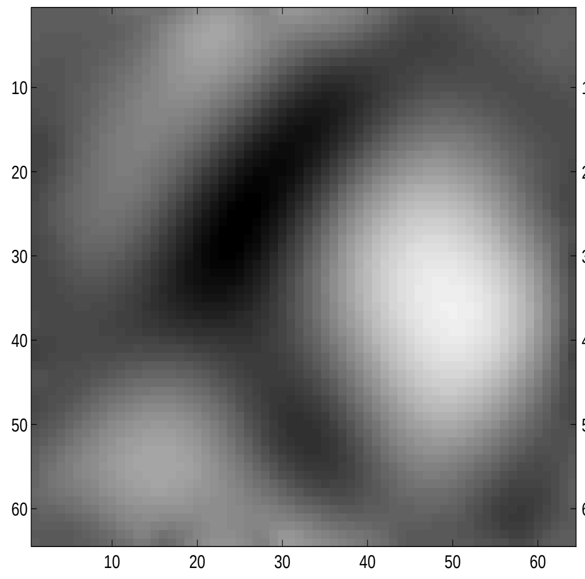
Figure 4.13: Fourier weight scheme (Pattern A, weight=freq^{-2})



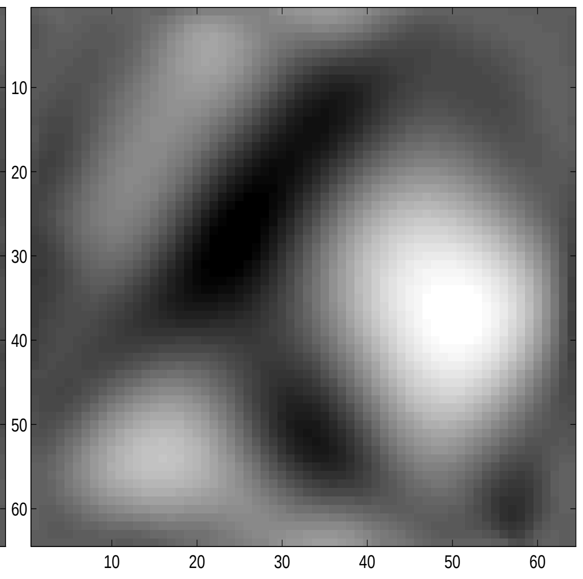
(a) after 30 line minimizations



(b) after 100 line minimizations



(c) after 300 line minimizations



(d) after 1000 line minimizations

Figure 4.14: Fourier weight scheme (Pattern B, weight=freq^{-2})

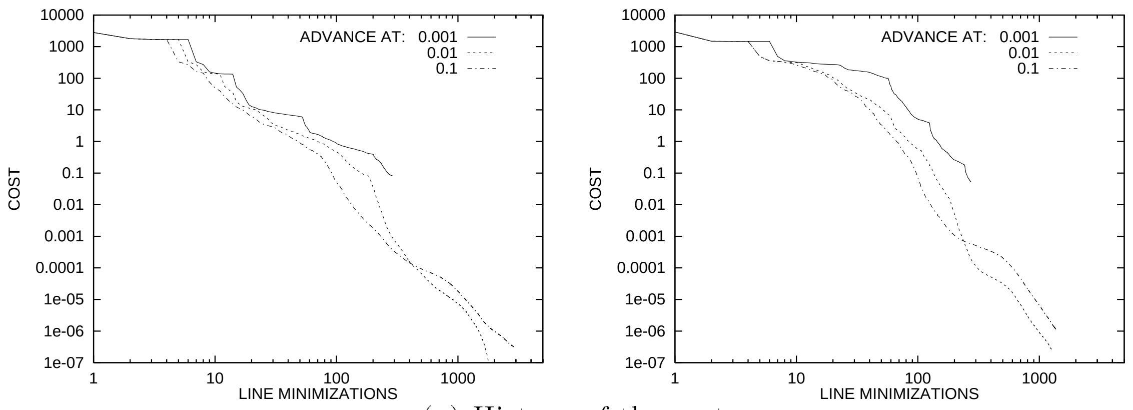
that the distance keeps reducing after 1000 line minimizations with $P_F = 2$ and 3, and for Pattern B it is getting even better than the smallest distance achieved with the Fourier step scheme. Figure 4.12 (c) shows this tendency more clearly. With P_F of a greater magnitude, the curve goes more toward the left, although the difference between $P_F = 2$ and $P_F = 3$ is slight.

Figure 4.13 shows the progress of the estimated distribution for Pattern A with $P_F = 2$. As the distance history showed, the progress speed is not as fast as in the Fourier step scheme. The estimated distribution looks somewhere between that by the single-resolution method and that by the Fourier step scheme. Figure 4.14 shows the estimates for Pattern B with the same $P_F = 2$. The estimate after 1000 line minimizations (d) is more blurred but has less wrong ghost image than the counterpart in the Fourier step scheme Figure 4.11 (d), page 98. In this sense, the Fourier weight scheme may be a slow-and-steady method, while the Fourier step scheme is rather quick but somewhat unstable.

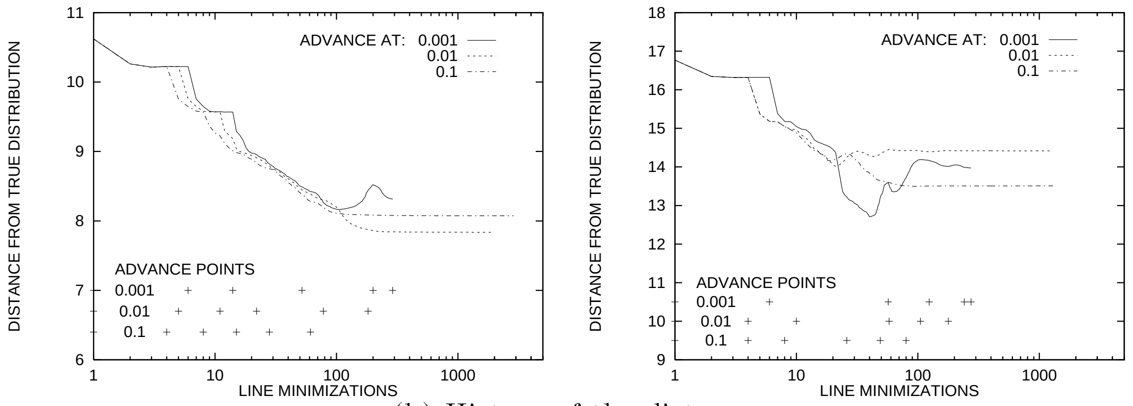
4.5.5 Step Scheme with the Haar Wavelet Transform

The estimation history with the Haar step scheme is shown in Figure 4.15. The settings are similar to those for the Fourier step scheme except that a smaller value of the constant for the advancing criterion seems suitable. The cost history (a) is not very different from that of the Fourier step scheme, Figure 4.9, page 96. The distance curves (b), however, are descending more slowly and their convergence levels are higher than with the Fourier step scheme. As will be discussed in Chapter 5, a probable cause for this is that there are much fewer resolution steps with the Haar wavelet transform than with the Fourier transform as the + marks show in the graphs. Accordingly, the cost-versus-distance curves (c) start descending straight down earlier than in the Fourier step scheme.

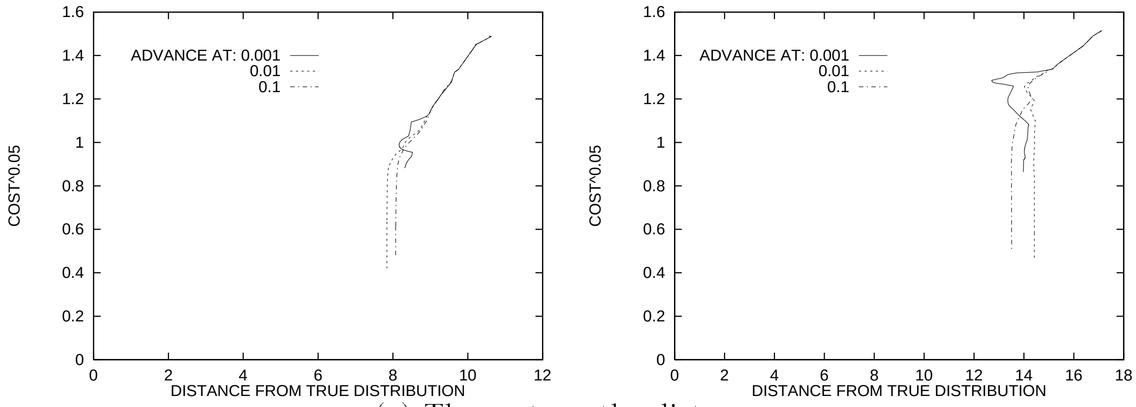
[Pattern A] **[Pattern B]**



(a) History of the cost

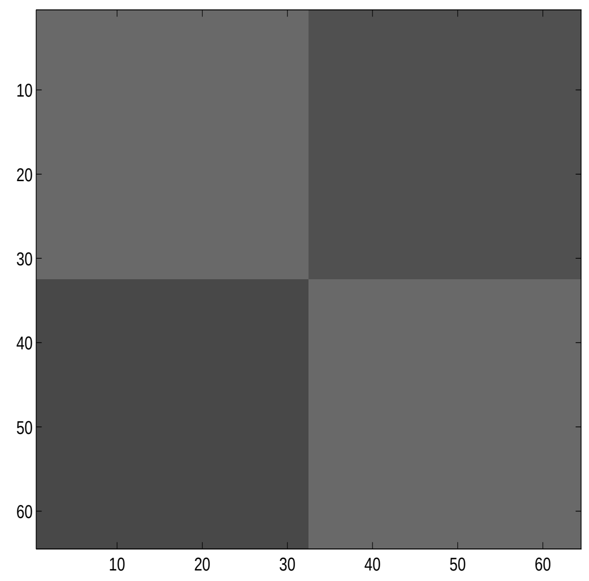


(b) History of the distance

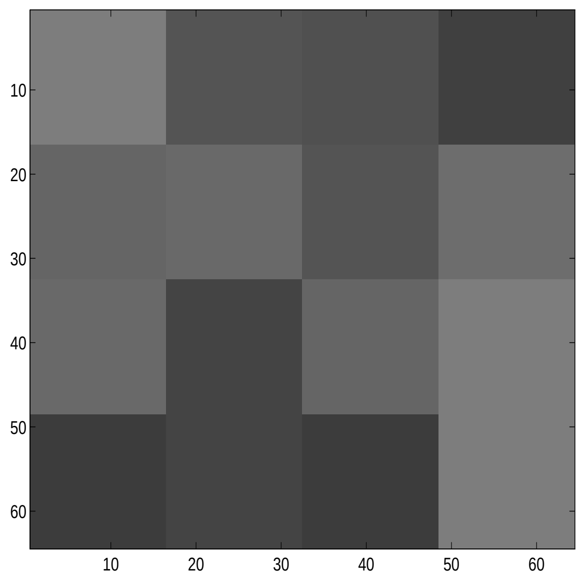


(c) The cost vs. the distance

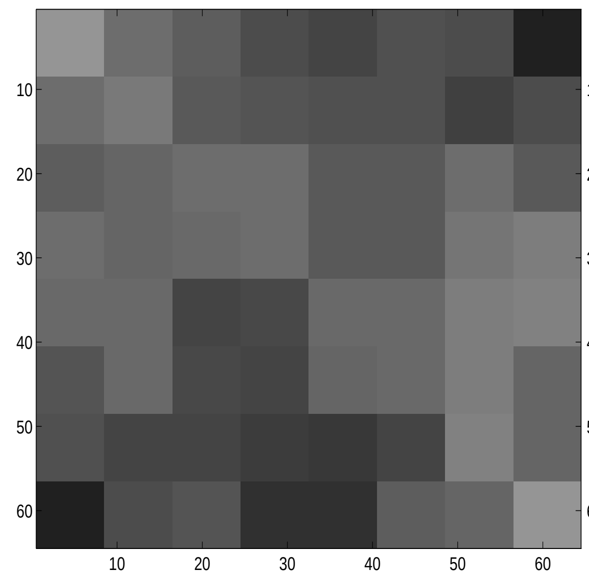
Figure 4.15: History of Haar step scheme



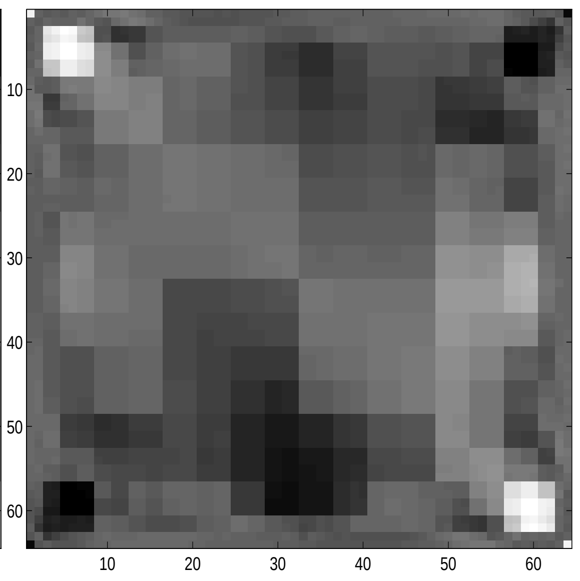
(a) resolution ≤ 2



(b) resolution ≤ 4

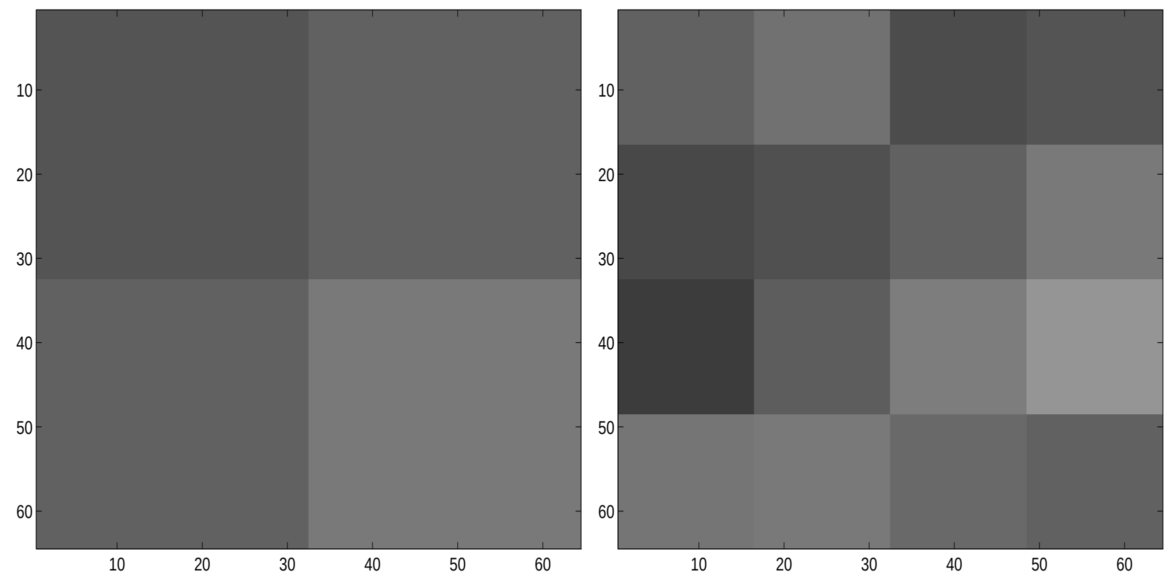


(c) resolution ≤ 8

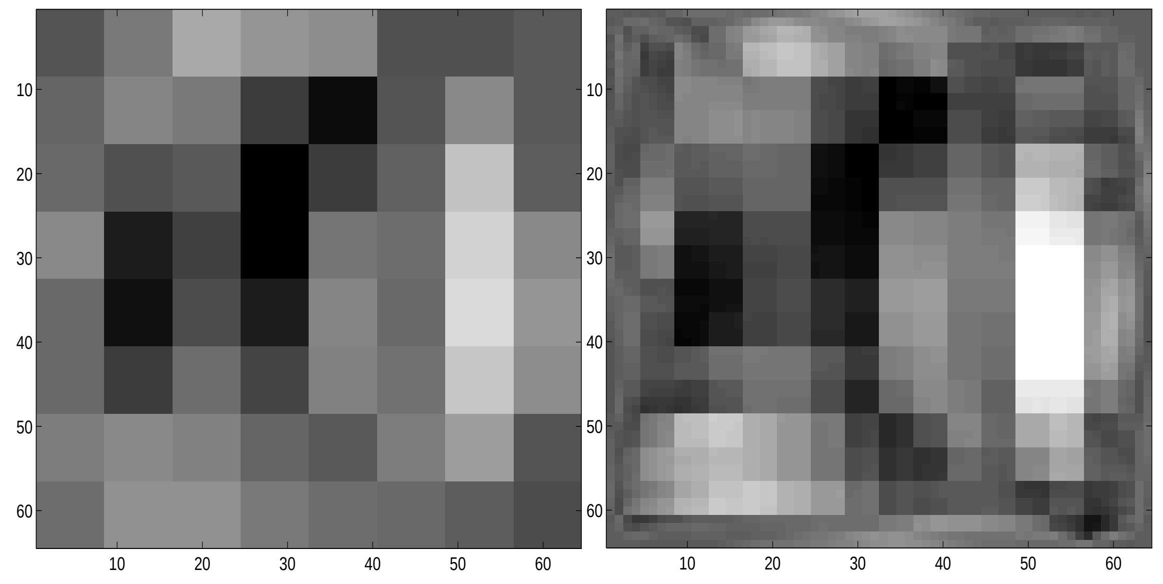


(d) at 1000 line minimizations

Figure 4.16: Haar step scheme (Pattern A, advance=0.01)



(a) resolution ≤ 2 (b) resolution ≤ 4



(c) resolution ≤ 8 (d) at 1000 line minimizations

Figure 4.17: Haar step scheme (Pattern B, advance=0.1)

Figures 4.16 and 4.17 show the estimation progress for Pattern A and Pattern B, respectively. As the resolution increases, the more details are estimated, but the overall quality is worse than in the Fourier step scheme, Figures 4.10 and 4.11, pages 97 and 98. The pictures (d) for both patterns are at 1000 line minimizations with the full resolution, but the patterns of earlier estimates still remain. The cause for this is probably twofold. One is that each wavelet coefficient influences only a certain local area. Therefore, if the search is insensitive to some coefficients above a certain resolution, the areas for which those coefficients are responsible remain unchanged, showing the old patterns. This does not happen with the Fourier transform, because each Fourier coefficient influences the whole field. The other probable cause is that the Haar wavelet is discontinuous. Hence, the borders of the local regions are clearly seen.

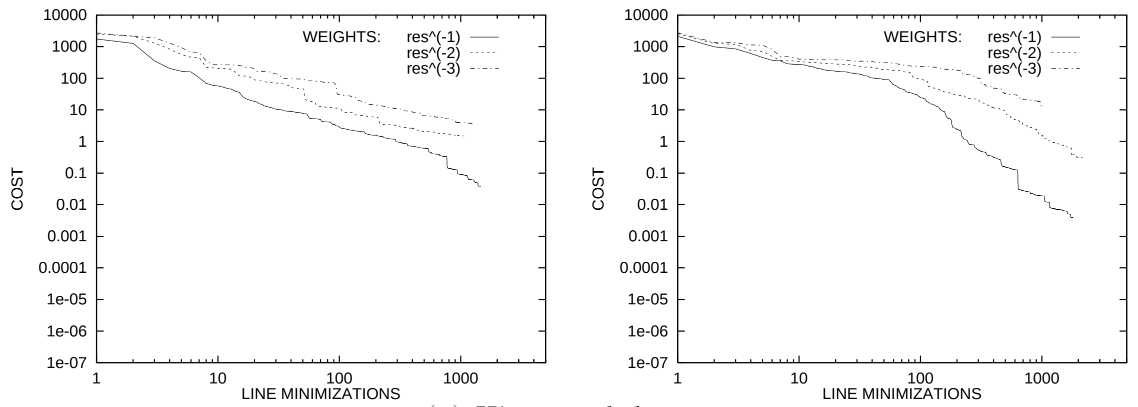
4.5.6 Weight Scheme with the Haar Wavelet Transform

Figure 4.18 shows the estimation history of the weight scheme using the Haar wavelet transform. The following formula similar to (4.57) in page 100 was used to determine the weight for each coefficient of the Haar wavelet transform.

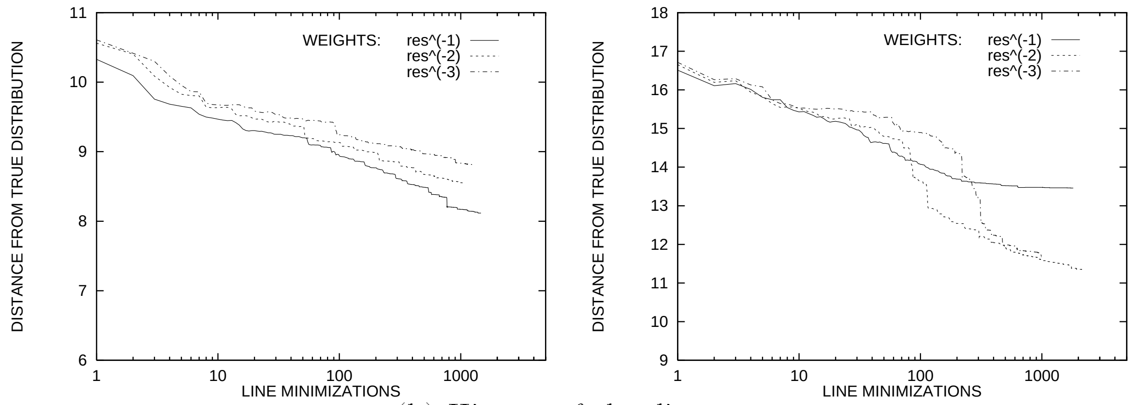
$$\text{weight} = \frac{1}{(\text{resolution})^{P_H}} \tag{4.58}$$

The real constant P_H was chosen to be 1, 2, or 3. In Figure 4.18 (a), the speed of the cost descent is slower than in the Haar step scheme, Figure 4.15, page 105. On the other hand, in Figure 4.18 (b), the distance for Pattern B after about 100 line minimizations is smaller than that in the Haar step scheme, although it is not as small as the distance in the final stage of the Fourier step scheme or the Fourier weight scheme. Accordingly, the cost-versus-distance curves in (c) are better than those of the Haar step scheme especially for Pattern B.

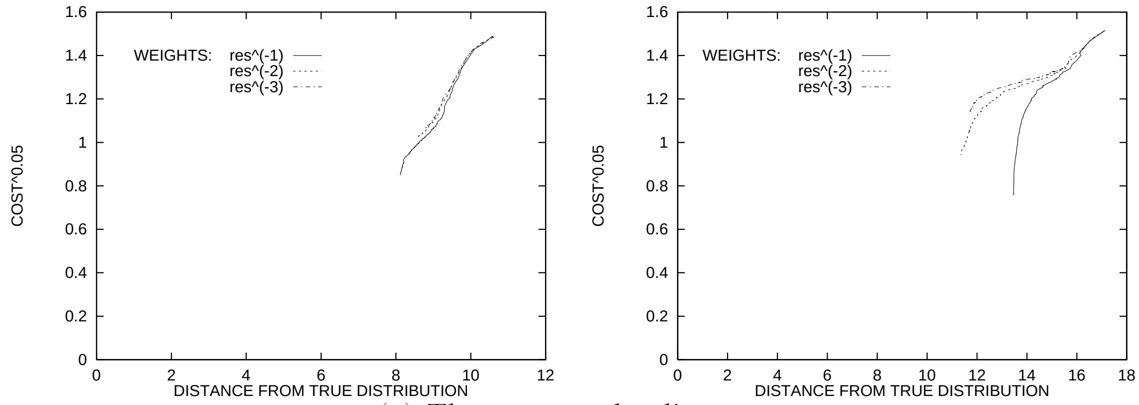
[Pattern A] **[Pattern B]**



(a) History of the cost

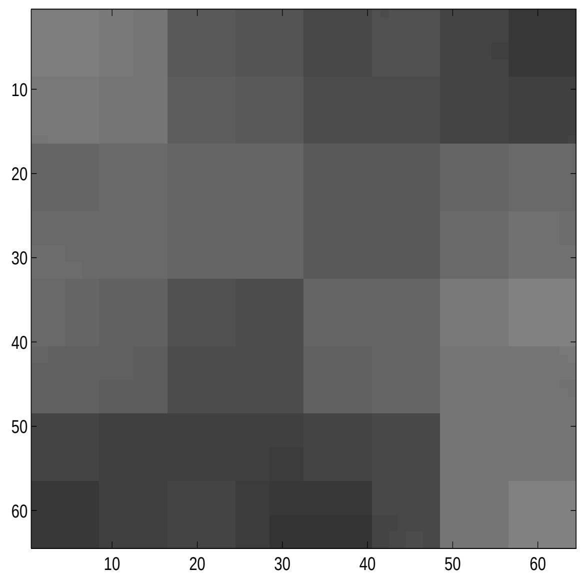


(b) History of the distance

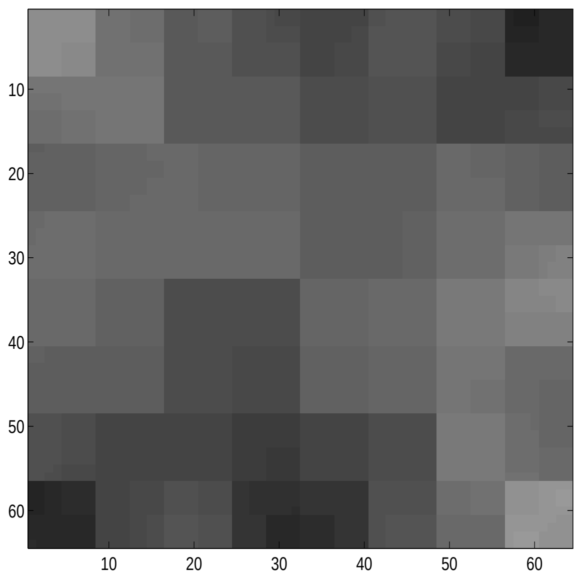


(c) The cost vs. the distance

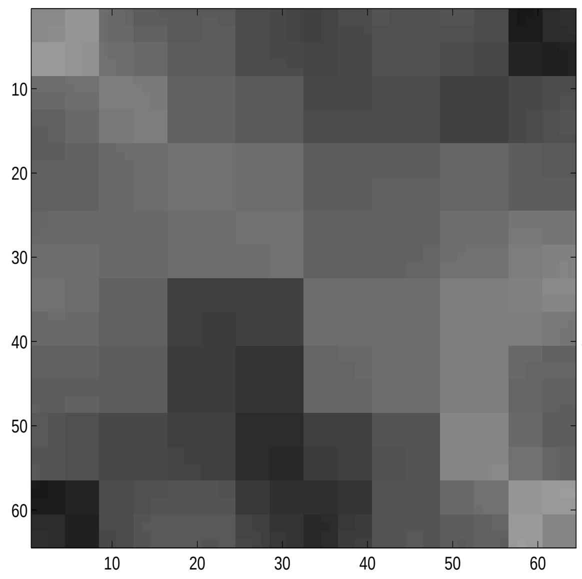
Figure 4.18: History of Haar weight scheme



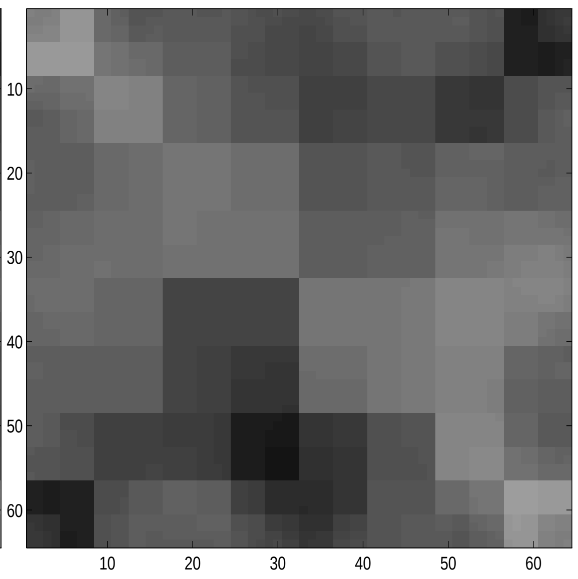
(a) after 30 line minimizations



(b) after 100 line minimizations

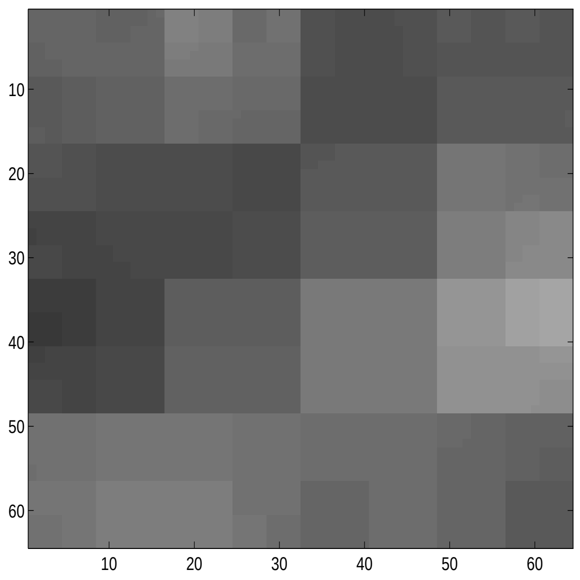


(c) after 300 line minimizations

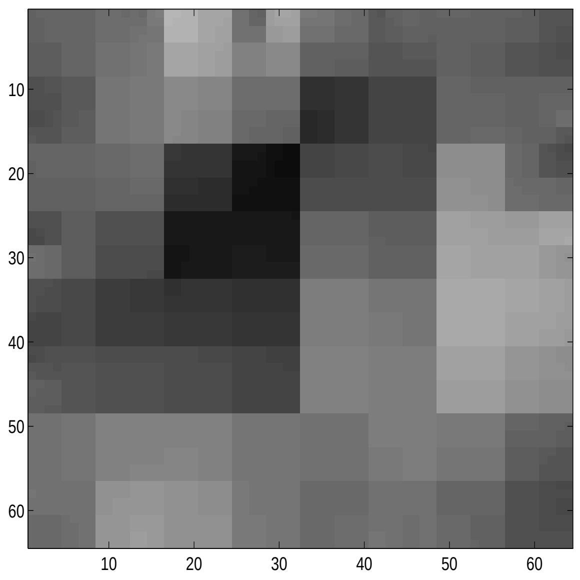


(d) after 1000 line minimizations

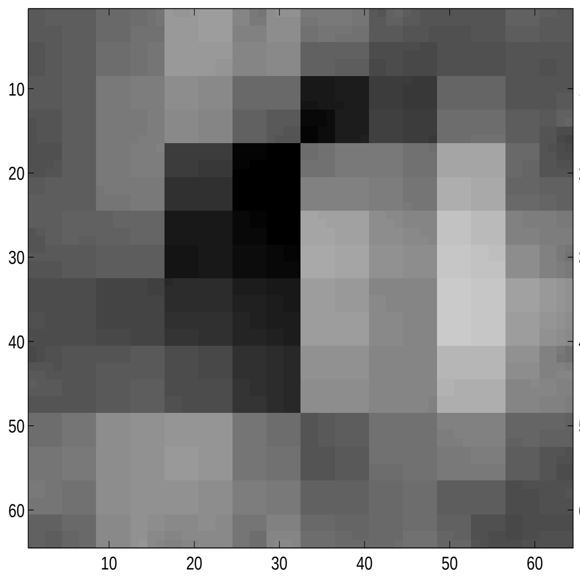
Figure 4.19: Haar weight scheme (Pattern A, weight=res^{-2})



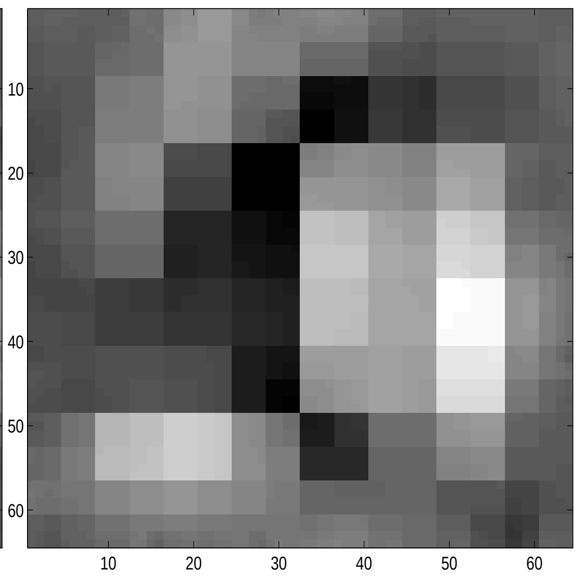
(a) after 30 line minimizations



(b) after 100 line minimizations



(c) after 300 line minimizations



(d) after 1000 line minimizations

Figure 4.20: Haar weight scheme (Pattern B, weight=res^{-2})

Figures 4.19 and 4.20 show the estimated distributions for Pattern A and Pattern B, respectively.

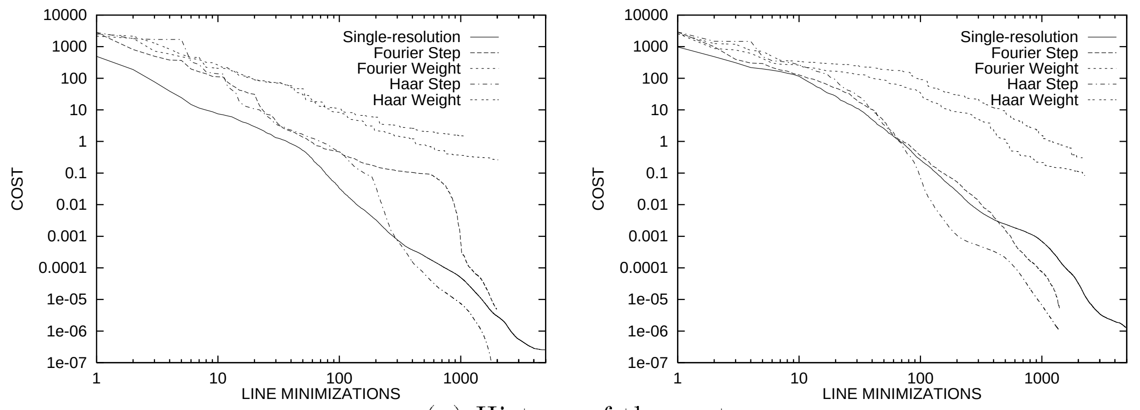
4.5.7 Comparison among Best Results

Figure 4.21 shows the history curves of all the estimation methods with the best advancing constants or weights that are previously shown. In the distance graphs (b), the methods are in the following order from the smallest distance at 1000 line minimizations for Pattern A: the Fourier step scheme, the Haar step scheme, the Fourier weight scheme, the Haar weight scheme, and the single-resolution method. For Pattern B, the order is the following: the Fourier step scheme and the Fourier weight scheme are about the same, and then the Haar weight scheme, the Haar step scheme, and finally the single-resolution method.

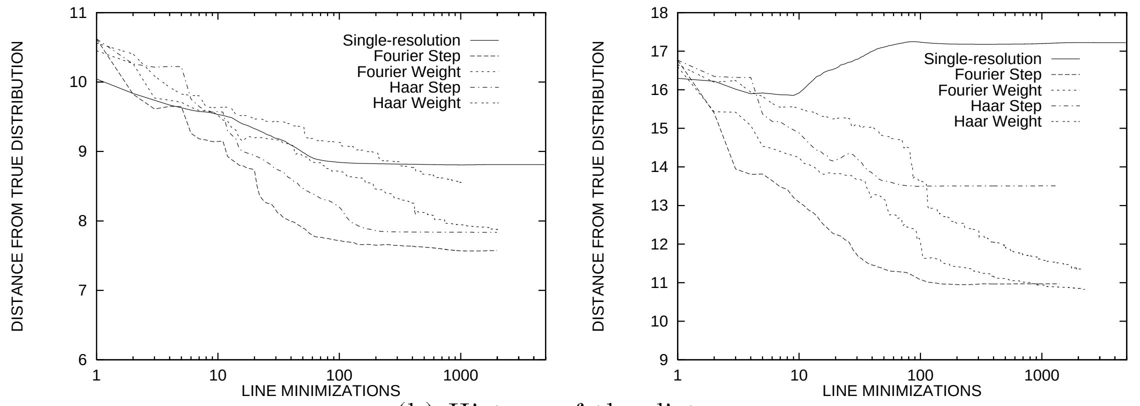
In the cost-versus-distance graphs Figure 4.21 (c), the methods are in the following order from the leftmost curve (the closest path to the global minimum) for Pattern A: the Fourier step scheme, the Fourier weight scheme, the Haar step scheme, the Haar weight scheme, and the single resolution. For Pattern B, the order is as follows: the Fourier weight scheme and the Fourier step scheme are about the same, then the Haar weight scheme, the Haar step scheme, and finally the single resolution.

Figure 4.22 shows the estimated distributions of all the methods at 1000 line minimizations for Pattern A. The settings are the same as in Figure 4.21. The estimate by the Fourier step scheme (c) looks closest to the true distribution (a), although it is still vague. Figure 4.23 shows the estimated distributions at 1000 line minimizations for Pattern B. The estimate by the Fourier step scheme (c) and also the one by the Fourier weight scheme (d) look closest to the true distribution (a). The estimate by the conventional single-resolution method (b) does not show the basic structure of the true distribution at all.

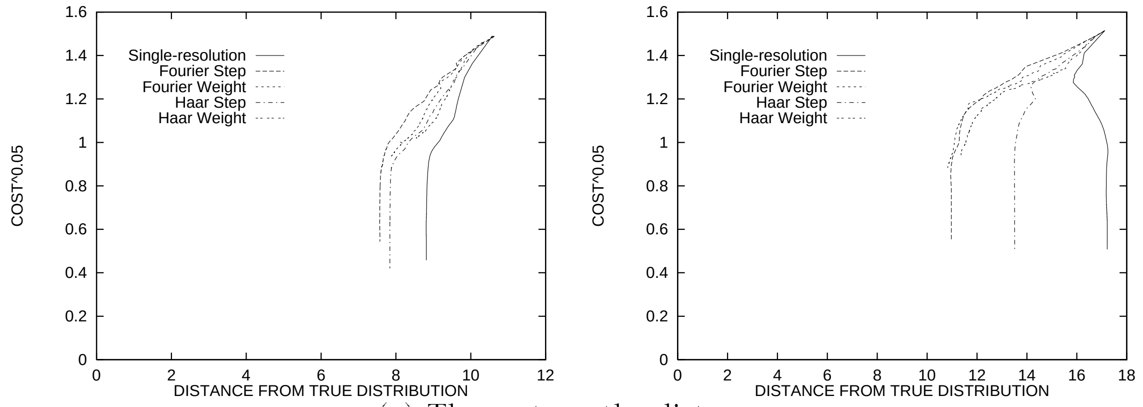
[Pattern A] **[Pattern B]**



(a) History of the cost



(b) History of the distance



(c) The cost vs. the distance

Figure 4.21: Comparison of history curves

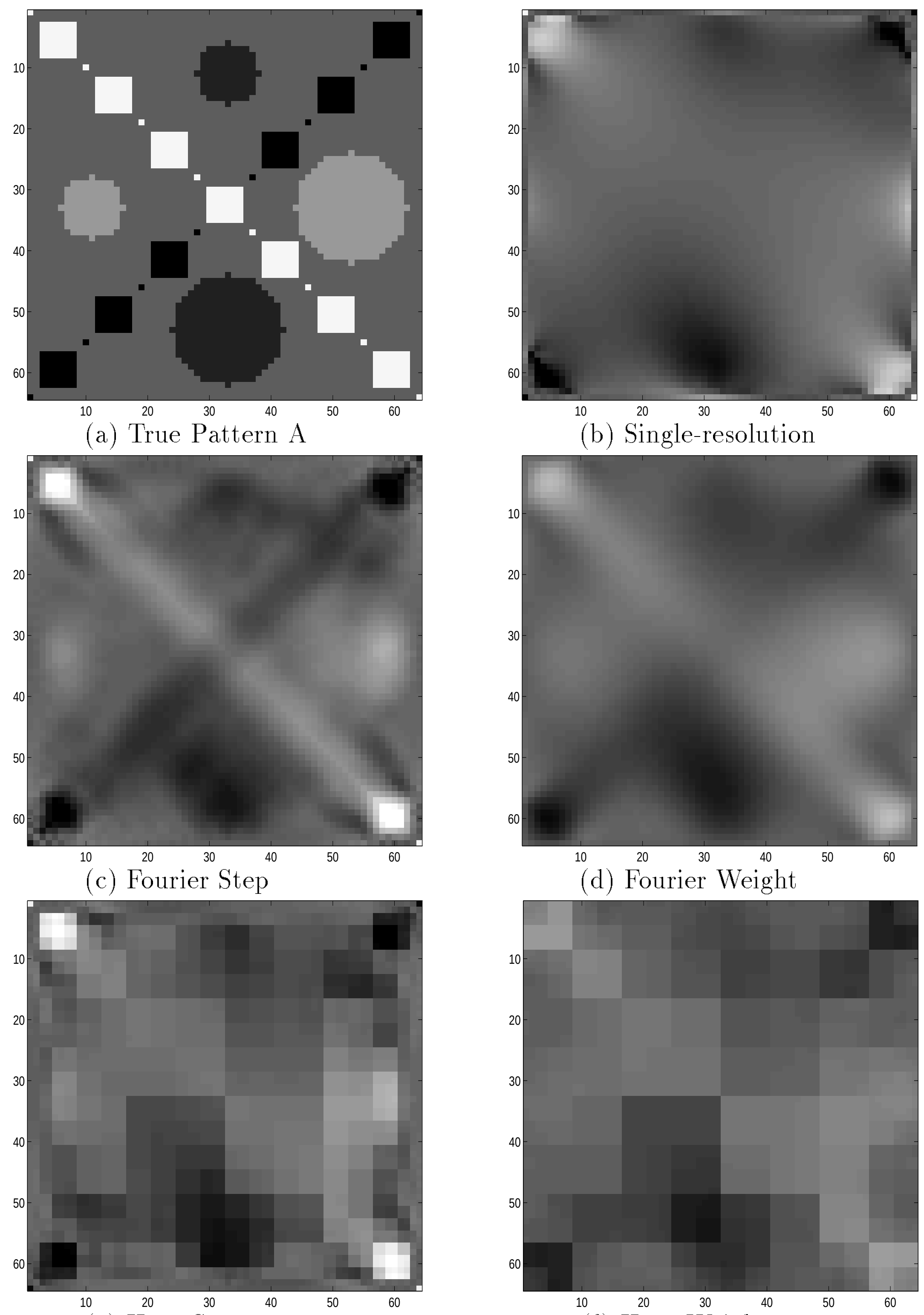

(a) True Pattern A (b) Single-resolution (c) Fourier Step (d) Fourier Weight (e) Haar Step (f) Haar Weight

Figure 4.22: Comparison for Pattern A

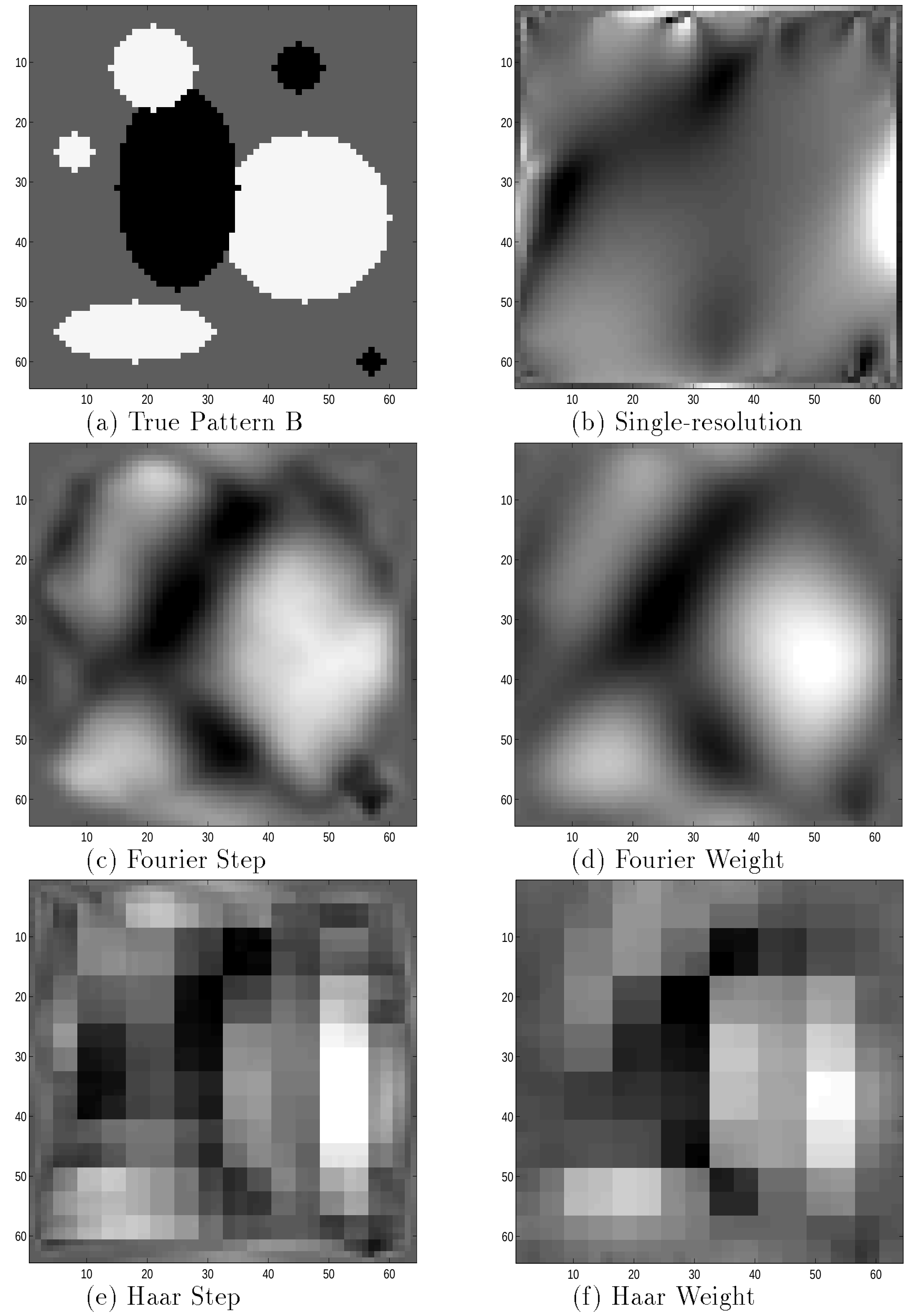


Figure 4.23: Comparison for Pattern B

5. Discussion

5.1 Multiresolution vs. Single-resolution

The cost-versus-distance graphs in the previous chapter show that the curves of the multiresolution estimation tend to proceed more toward the global minimum (at which both the cost and the distance to the true distribution are zero) than do the curves of the single-resolution estimation. Since the single-resolution optimization successfully decreases the cost but not the distance to the true distribution, the direction of the raw gradient may be very different from the direction toward the global minimum (the true distribution). Therefore, the shape of the cost surface is probably like a rain gutter shown in Figure 5.1 as a three-dimensional approximation. We can see, at least in the problem of electrical impedance tomography, that the single-resolution optimization goes down the hill almost straight to the bottom of the gutter and loses the height needed to slide down to the global minimum, while the multiresolution optimization

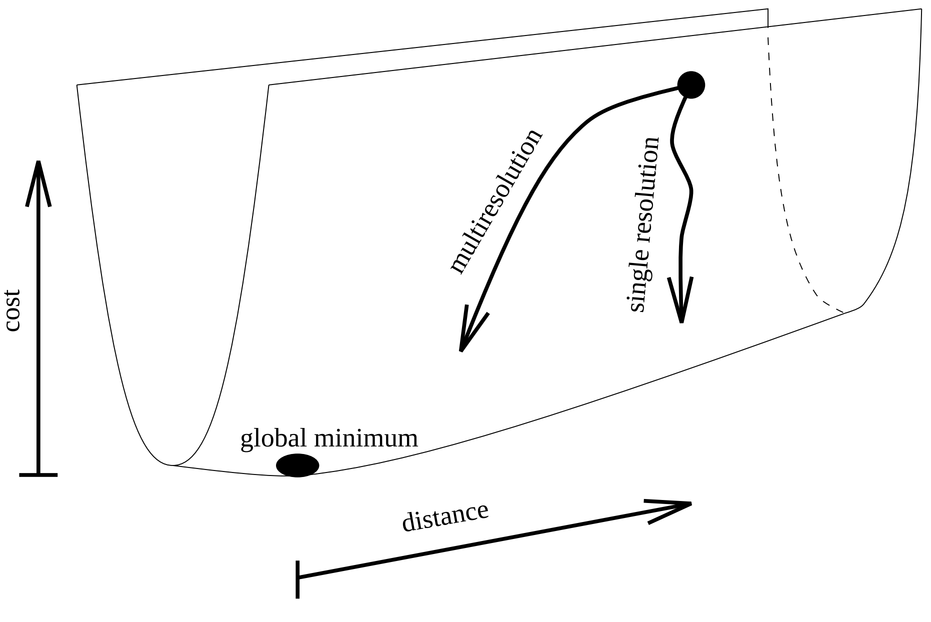


Figure 5.1: Search paths

leads closer to the global minimum. This fact tells that the low-resolution components of the gradient, which are enhanced in the multiresolution search, point more correctly to the global minimum than the high-resolution components do.

How to best utilize the multiresolution information, including the choice between the step scheme and the weight scheme, the advancing criterion for the step scheme, other types of filters for the gradient, etc., remains to be studied in the future.

5.2 Causes of Difficulties

The estimated resistivity distributions in the experiments of the previous chapter did not show enough details of the true distributions even with the multiresolution optimization. It might be because the image reconstruction of electrical impedance tomography is such a highly nonlinear, ill-posed problem [29] that even an excellent estimation algorithm has trouble with it. For example, two or more different resistivity distributions can produce very similar voltage distributions on the external boundary [41]. This means that there can be local minima that are almost as good as the global minimum in terms of the cost.

It is also possible that the existence of discontinuities in the true resistivity distributions (Pattern A and Pattern B) disturbed the estimation. Such "discontinuities" in a discretized field are not really discontinuous, because the grid width over which the value jumps is only finitely small. Besides, both the discrete Fourier transform and the discrete Haar wavelet transform can handle such a field with jumps flawlessly except for errors in the floating-point calculation. Nevertheless, such jumps might make the estimation difficult for the multiresolution optimization, since they contribute to larger magnitudes of the high-frequency components near the Nyquist critical frequency. Examining such effects of the frequency distributions on the search behavior can be an interesting and important topic for the future work.

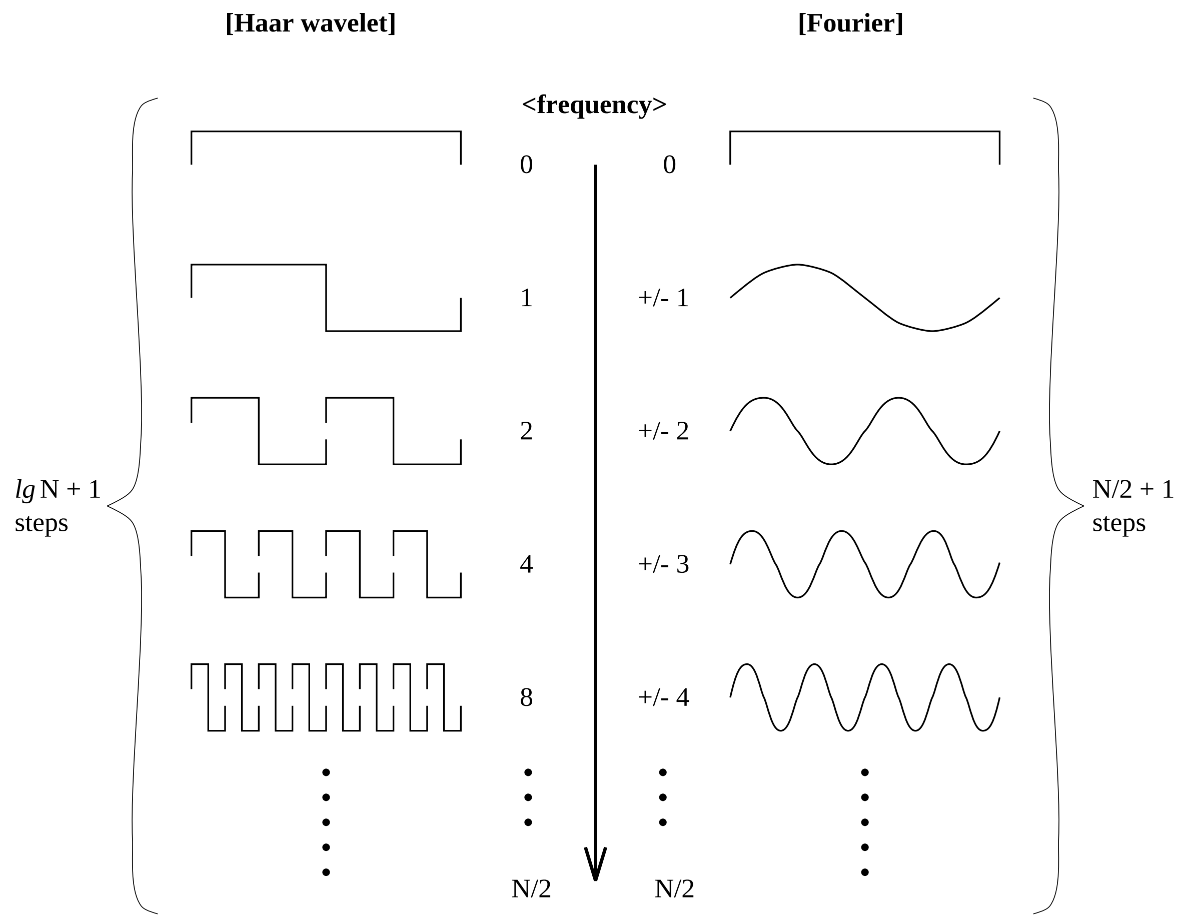


Figure 5.2: Wavelet transform vs. Fourier transform

5.3 Fourier Transform vs. Wavelet Transform

One of the fundamental differences between the Fourier transform and the wavelet transform lies in the way the frequency goes up as the resolution step proceeds. We call the number of pairs of the highs and the lows the “frequency” here, although each wave of the Haar wavelet transform is non-sinusoidal and hence has higher frequency components in itself. Figure 5.2 shows the difference between the two transforms in the one-dimensional case. In both transforms, the lowest frequency is zero and the highest is $N/2$, where N is the number of the discretized elements. In the wavelet transform, the frequency goes up exponentially as the resolution step proceeds, and the number of the corresponding coefficients is equal to the frequency. Thus, there are

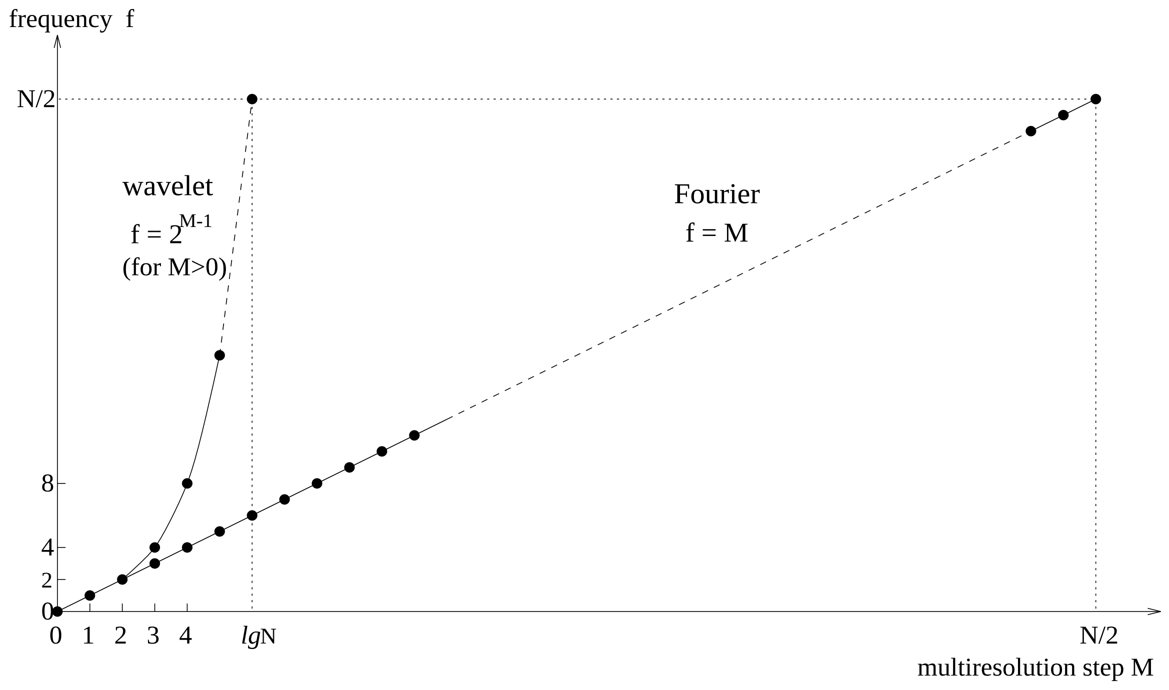


Figure 5.3: Multiresolution steps and frequency

$\lg N+1$ ($\lg \equiv \log_2$) resolution steps in the wavelet transform. In the Fourier transform, however, the frequency goes up linearly, and the number of the coefficients is two for each frequency (except for the lowest and the highest frequencies each of which has only one coefficient). As a result, the Fourier transform has $N/2+1$ resolution steps. Therefore, the relation between the frequency and the multiresolution step in the one-dimensional step scheme is as shown in Figure 5.3.

This means that the transition of the resolutions is smoother with the Fourier transform than with the wavelet transform. In another aspect, as the resolution step proceeds in the step scheme, exponentially many new coefficients are introduced in the search with the wavelet transform, whereas a constant number of (or linearly many in the two-dimensional case) new coefficients are introduced with the Fourier transform. As a result, the multiresolution optimization with the Fourier transform may proceed more smoothly than that with the wavelet transform. This may explain why the method with the Fourier transform performed better, if slightly, than the method with the wavelet transform in the experiments.

The wavelet transform is said to be better than the Fourier transform in applications other than parameter estimation, such as image compression [30]. The reason is that each wavelet coefficient represents localized information, while each Fourier coefficient represents information of the whole field. For the multiresolution estimation, however, the Fourier transform may be more suitable than the wavelet transform because of the smoother transition of resolutions.

5.4 Conditions for Multiresolution Optimization to Work

The original idea of multiresolution optimization is to consider a transform between the desired parameter $\boldsymbol{r}$ in the given field and the coefficients $\boldsymbol{R}$ of the transform in the frequency or scale domain, and then minimize the cost H with respect to the transform coefficients $\boldsymbol{R}$ rather than the parameter $\boldsymbol{r}$. The reason is that we can manipulate the coefficients $\boldsymbol{R}$ in a multiresolution manner because, in the $\boldsymbol{R}$ field, the information is inherently sorted out in frequency or scale, whereas we cannot perform such a manipulation in the original field of the parameter $\boldsymbol{r}$. Even in the method that directly optimizes the parameter $\boldsymbol{r}$ by using the filtered gradient, we can view the process in the above way, since, as we proved, the direct optimization using the filtered gradient is equivalent to the optimization of the transform coefficients.

Then, a serious question arises: "What if the parameters to be estimated are the transform coefficients $\boldsymbol{R}$ in the first place?" If we were to solve this estimation problem in another transformed domain, it would form an endless cycle. Let us take the Fourier transform as an example. As we saw in Section 2.3.1, the discrete Fourier transform and the inverse transform are virtually interchangeable. This means that, whether we use one or the other, the transformation process extracts and arranges the information in the virtually same manner. Therefore, if we were to estimate the Fourier coefficients $\boldsymbol{R}$ of some array $\boldsymbol{r}$, and we tried to solve the estimation problem by manipulating the Fourier coefficients $\boldsymbol{R}'$ of the desired Fourier coefficients $\boldsymbol{R}$, it would be the same as manipulating the array $\boldsymbol{r}$, because the "new" coefficients $\boldsymbol{R}' = \mathrm{DFT}(\boldsymbol{R})$ is virtually equivalent to the array $\boldsymbol{r} = \mathrm{DFT}^{-1}(\boldsymbol{R})$ in terms of the information contents. Thus, if we viewed the transform coefficients as the parameters to be estimated in the first place by the multiresolution optimization, it would form an endless cycle between the two domains, returning to the virtually same domain every two transforms.

It is obvious from the above argument that the multiresolution optimization does not always work. On the other hand, it is also obvious that it works in some cases as shown in the previous chapter. So, the next question is under what conditions the multiresolution optimization works. It should be a topic for the future work, since there is no clear answer to it yet. However, one possible answer may lie in the way the distributed parameter influences the cost as follows. Let us refer to Figure 2.1 in page 11. With most mathematical models described by partial differential equations for the forward problem $\boldsymbol{V} := G(\boldsymbol{r})$, each element of the distributed parameter $\boldsymbol{r}$ influences the elements of the variables $\boldsymbol{V}$ spatially close to it more than the elements of $\boldsymbol{V}$ far from it. Hence, each element of the parameter $\boldsymbol{r}$ influences the cost H somewhat locally, since the cost H is calculated on $\boldsymbol{V}$. On the other hand, the effect of the transform coefficients $\boldsymbol{R}$ on the cost H is not so local. In particular, with the Fourier transform, the effect is not local at all, since each element of $\boldsymbol{R}$ influences the whole field of the parameter $\boldsymbol{r}$ that leads to the cost H. Therefore, it might be correct to say that the multiresolution optimization works if the original distributed parameter to be estimated has somewhat local relationship with the cost to be minimized. We need to study further about this matter.

6. Conclusions

Multiresolution optimization methods for estimation of distributed parameters were developed in this dissertation. The basic idea was to give priority to low-resolution information over high-resolution information in the estimation process, assuming that the parameter distribution was continuous almost everywhere in the defined field. It was expected that the large-scale structure of the distribution would be found rapidly due to the enhanced low-resolution information, which would in turn help expedite the estimation of the details of the distribution.

The conjugate gradient method was employed for a local search, and the discrete Fourier transform and the Haar wavelet transform for a multidimensional field were used for manipulating the information in a multiresolution manner. Two schemes, the step scheme and the weight scheme, were devised as a way of manipulating the multiresolution information.

The original method was designed to indirectly estimate the desired parameter by estimating the transform coefficients with either the step scheme or the weight scheme. Later, it was proved that the same results can be obtained by directly estimating the desired parameter using the gradient that is filtered via the transform in either scheme.

The developed methods were evaluated in simulation of electrical impedance tomography. The multiresolution methods showed superior performance to the conventional single-resolution method in estimating the resistivity distributions for both of the two test patterns. The results of the multiresolution optimization with the Fourier transform were better than those with the Haar wavelet transform.

In addition to the multiresolution methods, efficient ways of calculating the gradient were developed. The framework was based on the general view of the back-propagation algorithm. In particular, an extremely memory-efficient method to obtain the gradient in a convergence-type forward problem was devised.

6.1 Future Work

The following subjects were proposed for the future work:

1. How to best utilize the multiresolution information, such as the choice between the step scheme and the weight scheme, the advancing criterion for the step scheme, other types of filters for the gradient, etc.
2. Effects of the frequency distribution of the parameter, especially of the high-end frequencies that correspond to discontinuities, on the search behavior of the multiresolution optimization.
3. The conditions for the multiresolution optimization to work, especially in terms of the relationship between the parameter to be estimated and the cost to be minimized.

References

[1] K. Amakawa and A. Pang. An inverse method of estimating parameter distributions based on a heart muscle model. In *Proceedings of Computers in Cardiology 1992*, pages 47–50. IEEE Computer Society Press, 1992.

[2] H.T. Banks and K. Kunisch. *Estimation Techniques for Distributed Parameter Systems.* Birkhäuser, 1989.

[3] M. Bertero and E.R. Pike, editors. *Inverse Problems in Scattering and Imaging: Proceedings of a NATO Advanced Research Workshop held at Cape Cod, USA, 14-19 April 1991.* Adam Hilger, 1992.

[4] C. Chavent and J. Liu. Multiscale parametrization for the estimation of a diffusion coefficient in elliptic and parabolic problems. In *Fifth IFAC Symposium on Control of Distributed Parameter Systems, Perpignan, France*, pages 193–202, 1989.

[5] C.K. Chui. *An Introduction to Wavelets.* Academic Press, 1992.

[6] I. Daubechies. *Ten Lectures on Wavelets.* Society for Industrial and Applied Mathematics, 1992.

[7] L. Davis, editor. *The Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[8] L. Davis and M. Steenstrup. Genetic algorithms and simulated annealing: An overview. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing.* Morgan Kaufmann Publishers, 1987.

[9] P. DuChateau and D. Zachmann. *Applied Partial Differential Equations.* Harper & Row, 1989.

[10] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[11] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

[12] F.A. Grünbaum. Diffuse tomography: the isotropic case. *Inverse Problems*, 8:409–419, 1992.

[13] C.A. Hall and T.A. Porsching. *Numerical Analysis of Partial Differential Equations.* Prentice Hall, 1990.

[14] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[15] G. Hinton. Connectionist learning procedures. Technical Report CMU-CS-87-115(version 2), Carnegie-Mellon University, December 1987.

[16] P. Hua, E.J. Woo, J.G. Webster, and W.J. Tompkins. Improved methods to determine optimal currents in electrical impedance tomography. *IEEE Transactions on Medical Imaging*, 11(4):488–495, 1992.

[17] D. Isaacson. Distinguishability of conductivities by electric current computed tomography. *IEEE Transactions on Medical Imaging*, MI-5(2):91–95, 1986.

[18] R.C. James. *Mathematics Dictionary.* Van Nostrand Reinhold, fifth edition, 1992.

[19] S.M. Kay. *Modern Spectral Estimation: Theory and Application.* Prentice Hall, 1988.

[20] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[21] P. Kraniauskas. *Transforms in Signals and Systems.* Addison-Wesley, 1992.

[22] G.A. Kyriacou, C.S. Koukourlis, and J.N. Sahalos. A reconstruction algorithm of electrical impedance tomography with optimal configuration of the driven electrodes. *IEEE Transactions on Medical Imaging*, 12(3):430–438, 1993.

[23] Jun Liu. A multiresolution method for distributed parameter estimation. *SIAM J. Sci. Comput.*, 14(2):389–405, 1993.

[24] A.K. Louis. Medical imaging: state of the art and future development. *Inverse Problems*, 8:709–738, 1992.

[25] D.G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing, second edition, 1984.

[26] T. Murai and Y. Kagawa. Electrical impedance computed tomography based on a finite element method. *IEEE Transactions on Biomedical Engineering*, BME-32(3):177–184, 1985.

[27] S. Nakamura. *Applied Numerical Methods with Software.* Prentice Hall, 1991.

[28] J.C. Newell, D.G. Gisser, and D. Isaacson. An electric current tomograph. *IEEE Transactions on Biomedical Engineering*, 35(10):828–833, 1988.

[29] K. Paulson, W. Lionheart, and M. Pidcock. Optimal experiments in electrical impedance tomography. *IEEE Transactions on Medical Imaging*, 12(4):681–686, 1993.

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

[31] O. Rioul and M. Vetterli. Wavelets and signal processing. *IEEE SP Magazine*, pages 14–38, October 1991.

[32] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA., 1986.

[33] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations. MIT Press, Cambridge, MA., 1986.

[34] P.C. Sabatier, editor. *Inverse Methods in Action: Proceedings of the Multicentennials Meeting on Inverse Problems, Montpellier, 1989.* Springer-Verlag, 1990.

[35] J.R. Singer, F.A. Grünbaum, P. Kohn, and J.P. Zubelli. Image reconstruction of the interior of bodies that diffuse radiation. *Science*, 248:990–993, 1990.

[36] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations.* Wadsworth & Brooks/Cole Advanced Books & Software, 1989.

[37] A. Tarantola. *Inverse Problem Theory.* Elsevier Science Publishers, 1987.

[38] M. Tasto and H. Schomberg. Object reconstruction from projections and some non-linear extensions. In C.H. Chen, editor, *Pattern Recognition and Signal Processing*, NATO Advanced Study Institutes Series, pages 485–503. Sijthoff & Noordhoff, 1978.

[39] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Publishing, 1986.

[40] J.S. Walker. *Fast Fourier Transforms.* CRC Press, 1991.

[41] J.G. Webster, editor. *Electrical Impedance Tomography.* The Adam Hilger Series on Biomedical Engineering. Adam Hilger, 1990.

[42] E.J. Woo, P. Hua, and J.G. Webster. A robust image reconstruction algorithm and its parallel implementation in electrical impedance tomography. *IEEE Transactions on Medical Imaging*, 12(2):137–146, 1993.

[43] A.H. Wright. Genetic algorithms for real parameter optimization. In J.E. Rawlins, editor, *Foundations of Genetic Algorithms.* Morgan Kaufmann Publishers, 1991.

[44] T.J. Yorkey, J.G. Webster, and W.J. Tompkins. Comparing reconstruction algorithms for electrical impedance tomography. *IEEE Transactions on Biomedical Engineering*, BME-34(11):843–852, 1987.

[45] T.J. Yorkey, J.G. Webster, and W.J. Tompkins. An improved perturbation technique for electrical impedance imaging with some criticisms. *IEEE Transactions on Biomedical Engineering*, BME-34(11):898–901, 1987.