Quantitative Testing of Probabilistic Phase Unwrapping Methods

by

Jodi Moran

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, August 2001

Abstract

The reconstruction of a phase surface from the observed principal values is required for a number of applications, including synthetic aperture radar (SAR) and magnetic resonance imaging (MRI). However, the process of reconstruction, called "phase unwrapping", is an ill-posed problem. One class of phase-unwrapping algorithms uses smoothness prior models to remedy this situation. We categorize this class of algorithms according to the type of prior model used. Motivated by this categorization, we propose that phase-unwrapping algorithms be tested by generating phase surfaces from the prior models, and then quantifying the deviation of each reconstructed surface from the corresponding original surface. Finally, we present results of the new testing method on a selection of phase-unwrapping algorithms, including a new algorithm.

Acknowledgements

Thanks to my readers Richard Mann and Paul Fieguth for their time and comments, to Elizabeth Lam for reading early versions and for helpful discussions, to Nemanja Petrovic and Andreas Loebel for the use of their software, to Anitha Kannan for sharing computer time and advice, to Steve Engels and Paul Nijjar for commiserating with me, and to the rest of the Probabilistic Modeling, Inference, and Learning Group for teaching me so much.

A special thank you to my supervisor Brendan Frey for his limitless patience and constant encouragement.

Finally, thank you to my dear parents and my one and only sister, who have always and will always provide me with all the support I could wish for.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are several different applications in which measurements are taken by aiming a signal at the object to be observed. The signal is reflected or refracted by the object and the changed signal is then received by measurement equipment. In these applications, the phase of the signal is related to the quantity we wish to measure. For example, in synthetic aperture radar (SAR) interferometry, terrain elevation measurements are reconstructed from radar signal phase [GP98]. In magnetic resonance imaging (MRI), the water/fat separation in tissue can be determined from the signal phase [GP98]. Optical interferometry can measure the vibrations of a surface using laser phase [NPD00]. There are many other similar applications.

However, in all of these problems the process of extracting the phase from the measured signal "wraps" the absolute phase so that all the recovered phase values fall within a single wavelength. This wrapped phase cannot be used for measurements. Thus a key step to the processing of the signal's information is to recover the absolute phase from the wrapped values, a process known as "phase unwrapping".

Unfortunately, phase unwrapping in its most general form is an ill-posed problem, so it cannot be solved without the addition of further information of some kind. Some phase-unwrapping algorithms, designed to be used for a particular application, use information that relates to that application. However, since there are such a variety of phase-unwrapping applications, it is useful to have algorithms that use only a general idea of the "smoothness" of the unwrapped phase to solve the problem.

To test a general phase-unwrapping algorithm, it is not satisfactory to run the algorithm on hand-selected data, either from real-world applications or from phase surfaces constructed by the tester, because hand-selected data cannot represent the full range of potential surfaces. Moreover, real-world data lacks a known solution to which the algorithm's reconstruction can be compared.

However, we find that phase-unwrapping algorithms that do not include application-specific data nearly always make use of one of two models for the original phase surface. We therefore suggest that these two models be used to generate surfaces with which we can test phase-unwrapping algorithms. Because the surfaces are simulated and not real, the ground truth is known, so we can quantify the discrepancy between the reconstructed phase and the true phase. Because the surfaces are generated randomly, they can represent the full range of phase unwrapping problems that satisfy the two models.

To introduce this quantitative testing method, we organize this paper as follows.

In Chapter 2 we formulate the phase unwrapping problem in the most general terms possible, note that it is an ill-posed problem, and discuss the type of information that can be included to make the problem well-posed.

In Chapter 3 we cast the phase-unwrapping problem as a probabilistic problem and discuss the models that are used by general algorithms.

In Chapter 4 we introduce a new method for testing general phase-unwrapping algorithms, describe some specific phase-unwrapping algorithms including a new algorithm for phase unwrapping, and present the results of our testing methods on these algorithms.

In Chapter 5 we summarize our conclusions, list the contributions of this paper, and give some suggestions for future work.

Chapter 2

The phase unwrapping problem

For many applications it is necessary to know the real-valued phase of a signal, since this quantity is related to some phenomenon of interest. However, it is usually the case that measurements can extract only the "principal values" of the signal phase, that is, values within a single wavelength. Therefore it is necessary to attempt to reconstruct the phase from its principal values, a process known as "phase unwrapping".

Mathematically, we can express the phase-unwrapping problem as follows. The "absolute" phase is a continuous differentiable function $s : R \to \mathbb{R}$ where R is a two-dimensional region in $\mathbb{R}^2$. The process of measuring s adds noise $n : R \to \mathbb{R}$ to the true phase and then wraps it to within a single wavelength. Let $u = s + n$ represent the noisy true phase. For simplicity, and without loss of generality, we take the unit of measurement to be the wavelength, and we shift the wrapping interval to zero. Then $w : R \to [0, 1)$ given by $w = u \mod 1$ is the wrapped noisy phase. Note that if we take the noise n to be continuous and differentiable, then u will be continuous and differentiable. However, even if u is continuous, w is at most piecewise continuous.

So far we have defined our functions on a continuous domain. However in practice we can perform measurements only at discrete points within this domain. To simplify notation we will assume that R is a rectangular region $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ (if R is not rectangular we can cover it with a rectangular R' and extend the definitions of s and n to R' by giving them default values outside of R). Partition $[x_{\min}, x_{\max}]$ into $x_{\min} = x_1 < x_2 < \ldots < x_M = x_{\max}$ and $[y_{\min}, y_{\max}]$ into $y_{\min} = y_1 < y_2 < \ldots < y_N = y_{\max}$. Then the set of points $G = \{(x_i, y_j) : 1 \leq i \leq M, 1 \leq j \leq N\}$ forms a grid in R. We will assume that the measurements of s occur at the points of G (in practice, measurements are always performed along such a grid).

Most algorithms that perform phase unwrapping make use of the discrete partial derivatives of functions on R. To simplify the form of the discrete partial derivative expressions, we make two further assumptions. The first is that the points x_i are equally spaced, and likewise the points y_j are equally spaced. The second is that the spacing between points is equal in the x- and y- directions. We can express these two assumptions as $x_{i+1} - x_i = y_{j+1} - y_j = c$ for all $1 \leq i \leq M-1, 1 \leq j \leq N-1$. Having made these assumptions, we can without loss of generality take $c = 1$, so that the discrete partial derivatives take the form of differences. These assumptions are used only for calculating discrete approximations to partial derviatives, and therefore could be relaxed by replacing differences in the discussions below with the appropriate approximation.

Let $s_{ij} = s(x_i, y_j)$, $n_{ij} = n(x_i, y_j)$, $u_{ij} = u(x_i, y_j)$, and $w_{ij} = w(x_i, y_j)$. Then $u_{ij} = s_{ij} + n_{ij}$ and $w_{ij} = u_{ij} \mod 1$. Note that due to the latter expression, $u_{ij} - w_{ij}$ is an integer. Let $k_{ij} = u_{ij} - w_{ij} \in \mathbb{Z}$. We will use bold symbols to denote the vector or matrix of values at grid points: thus $\mathbf{s} = \{s_{ij}\}$, $\mathbf{u} = \{u_{ij}\}$, and so forth.

In summary, the discrete phase unwrapping problem is as follows: given a set of MN

noisy, wrapped phase measurements $\mathbf{w}$, unwrap the measurements and filter out the noise to obtain the true phase values $\mathbf{s}$. It is immediately clear that such a problem is not well-posed; even ignoring the effects of noise (setting $n = 0$), there are infinitely many choices for the set of $\mathbf{s}$ that satisfy $w_{ij} = s_{ij} \mod 1$. Each of these sets could be a measurement at discrete points of a continuous differentiable phase function s. In particular we could simply take $s_{ij} = w_{ij}$ everywhere. However, such a trivial solution is not useful for most applications. This is because although they are often not explicitly included in the problem statement, there are certain properties that make one solution preferable to another. To remove the ambiguity of the phase unwrapping problem we need to quantify what makes one solution more desirable than another.

To begin with, all phase-unwrapping algorithms arbitrarily fix one reference point as "zero". This is necessary because there is no reasonable way to prefer one absolute phase surface to another if they differ only by an integer constant, since the integer shift of the original phase is lost in the wrapping operation. Having fixed this reference point, some algorithms base their phase surface preference on additional information that is specific to a particular application. However, we shall be concerned only with algorithms whose formulation of preferable solutions is not specific to any particular application, but rather indicates only a preference for "smooth" phase functions.

Since even to reconstruct the noisy absolute phase from the wrapped phase is an ill-posed problem to which restrictions must be added, many phase-unwrapping algorithms do not attempt to recreate the absolute phase $\mathbf{s}$. Instead, these algorithms simply reconstruct the noisy phase $\mathbf{u}$ from the observations $\mathbf{w}$ (this can be followed with a separate "filtering" or "smoothing" algorithm that reduces noise). In the discussions that follow, algorithms that unwrap and remove noise at once are distinguished by the use of the variable $\mathbf{s}$ for the reconstructed points. Those algorithms that only unwrap but do not filter noise instead reconstruct $\mathbf{u}$.

2.1 Choosing an assumption

To see what type of extra assumption might be made in order to produce meaningful solutions to the phase unwrapping problem without including application-specfic data, in this section we discuss the most popular assumption and the reasoning behind its choice. Many phase-unwrapping algorithms, and in particular all of those discussed in Ghiglia and Pritt's seminal book [GP98], assume that most of the measurements of the signal from which the phase is extracted are not aliased. That is, the assumption is that the grid on which the measurements have been made is fine enough so that the local differences between the true phase values s_{ij} are nearly always less than half of the wavelength. We call this assumption Itoh's assumption, for reasons we discuss below. Mathematically, Itoh's assumption is that

$$|(\Delta_x s)_{ij}| < 1/2 \tag{2.1}$$

$$|(\Delta_y s)_{ij}| < 1/2 \tag{2.2}$$

for most i, j, where we define

$$(\Delta_x f)_{ij} = f_{(i+1)j} - f_{ij} \text{ for all } 1 \leq i \leq M-1, 1 \leq j \leq N \tag{2.3}$$

$$(\Delta_y f)_{ij} = f_{i(j+1)} - f_{ij} \text{ for all } 1 \leq i \leq M, 1 \leq j \leq N-1 \tag{2.4}$$

for any function f on our grid.

There are two reasons why most phase unwrapping strategies employ this assumption. The first is practical: it is thought [GP98] that most measurements for which phase unwrapping is needed are not aliased. The second reason is theoretical, and begins with an analysis of a one-dimensional phase unwrapping problem. Itoh [Ito82] showed that in a one-dimensional phase unwrapping problem, the (noisy) absolute phase can be uniquely reconstructed from the wrapped phase when the local absolute phase differences are less than half a wavelength. Mathematically, suppose that $g(x)$ is the wrapped version of a single-variable phase function $h(x)$. We are given the wrapped measurements $\{g_i = g(x_i)\}_{i=1}^{N}$ at unit-spaced points such that $g_i = h_i \mod 1$ for some unknown $\{h_i = h(x_i)\}_{i=1}^{N}$. For each $1 \leq i \leq N-1$, if

$$|h_{i+1} - h_i| < 1/2 \tag{2.5}$$

then [Ito82]

$$h_{i+1} - h_i = W(g_{i+1} - g_i) \tag{2.6}$$

where

$$W(x) = \left(\left(x + \tfrac{1}{2}\right) \mod 1\right) - \tfrac{1}{2} = x - \mathrm{rnd}(x) \tag{2.7}$$

This means that we can find $(h_{i+1} - h_i)$ for all $1 \leq i \leq N-1$. So, starting from an arbitrarily fixed reference point $h_1 = g_1$, we can find the absolute phase at each point by the recursive equation

$$h_{i+1} = h_i + W(g_{i+1} - g_i) \text{ for all } 1 \leq i \leq N-1. \tag{2.8}$$

The repeated application of (2.8) is known as Itoh's algorithm.

We can view Itoh's algorithm as an integration of h along its single dimension. Denote the derivative of h with respect to x as h_x. We can approximate this derivative on the interval $[x_i, x_{i+1}]$ by its value at the midpoint, which in turn is approximated by a first-order finite difference. Thus our estimate of the derivative h_x on the interval $[x_i, x_{i+1}]$ is given by

$$\begin{aligned} h_x(x) &\simeq (h_x)_{i+\frac{1}{2}} \\ &\simeq (h_{i+1} - h_i) \\ &= W(g_{i+1} - g_i) \end{aligned} \tag{2.9}$$

where the subscript $i + \frac{1}{2}$ indicates a value at the point midway between x_i and x_{i+1}. So Itoh's algorithm is a method of estimating h_{i+1} using the equation

$$h_{i+1} = h_i + \int_{x_i}^{x_{i+1}} h_x(x)dx \tag{2.10}$$

where the estimate of the derivative is calculated from the wrapped phase values. That is, Itoh's algorithm finds the phase at x_{i+1} by integrating an estimated derivative along the path joining x_i to x_{i+1}. Note that the estimate of the derivative is only reasonable when the assumption (2.5) holds.

The analogue of Itoh's algorithm in two dimensions is to estimate the absolute phase *gradient* from the wrapped phase differences, and then to use the estimated gradient to reconstruct the absolute phase. Since for the phase unwrapping problem $w_{ij} = u_{ij} \mod 1$, by analogy with the one-dimensional case we can write

$$(\Delta_d u)_{ij} = W((\Delta_d w)_{ij}) \tag{2.11}$$

whenever

$$|(\Delta_d u)_{ij}| < 1/2 \tag{2.12}$$

for each of $d = x, y$. Note that we cannot say anything about the true phase surface points $\mathbf{s}$ due to the addition of noise, so the assumption required is actually stronger than the form of Itoh's assumption in (2.1)-(2.2). Let us suppose that (2.12) holds for $d = x, y$ at nearly every applicable i and j. It is then reasonable to approximate $(\Delta_d u)_{ij}$ by $W((\Delta_d w)_{ij})$ on the grid. Using a standard finite-difference technique, we approximate the partial derivative function u_x between the points (x_i, y_j) and (x_{i+1}, y_j) by the difference $(\Delta_x u)_{ij}$. Likewise u_y between (x_i, y_j) and (x_{i+1}, y_j) is approximated by $(\Delta_y u)_{ij}$. Thus the local approximations are

$$\begin{aligned} u_d(x, y) &\simeq (\Delta_d u)_{ij} && (2.13) \\ &\simeq W((\Delta_d w)_{ij}) && (2.14) \end{aligned}$$

for each of $d = x, y$, at each applicable i and j. Having an estimate of the gradient of the absolute phase, the pair of two-dimensional equations that correspond to (2.10) are

$$u_{(i+1)j} = u_{ij} + \int_{(x_i,y_j)}^{(x_{i+1},y_j)} u_x(x, y)dx \tag{2.15}$$

and

$$u_{i(j+1)} = u_{ij} + \int_{(x_i,y_j)}^{(x_i,y_{j+1})} u_y(x, y)dy. \tag{2.16}$$

in which we use the local estimates of $u_x(x, y)$ and $u_y(x, y)$ given by (2.14).

Notice, however, that there are two possible ways to obtain an estimate for $u_{(i+1)(j+1)}$ given an estimate for u_{ij} and the corresponding gradient estimates. We can first apply (2.15) and then apply (2.16), or we can apply (2.16) and then (2.15). As far as we have seen, there is no reason that the two different estimates obtained by these two different methods should be the same. Extending this to the general case, there are many possible orders of application of (2.15) and (2.16) between two points (x_i, y_j) and (x_k, y_l). Each of these orders represents a different path of integration through the domain. Presumably each one of these different paths could lead to a different estimated value for u_{kl}. This problem is not present in the one-dimensional version of Itoh's algorithm, since in one dimension there is only one possible path between two points.

Vector calculus theory [Ste99] tells us that if we use the true values of $u_x(x, y)$ and $u_y(x, y)$ in (2.15) and (2.16), then the integrals along any path between (x_i, y_j) and (x_k, y_l) must be the same. This means that we can choose *any* path between (x_i, y_j) and (x_k, y_l) to find u_{kl} given u_{ij} and the gradient – the integral is then said to be *path-independent*. However, we do not know the gradient of the absolute phase; we can only estimate it from the wrapped phase differences. Since the values we use in (2.15) and (2.16) are only estimates and not the true absolute phase gradient, it may indeed be the case that different paths of integration between two points produce differing estimates.

Fortunately it is not required that the estimate of the absolute phase gradient used in equations (2.15) and (2.16) be equal to the true absolute phase gradient in order for integrals on the grid to be path-independent. It is enough that the following condition be satisfied: that

$$\oint_C \widehat{\nabla u} = 0 \tag{2.17}$$

for any closed curve C in the domain, where $\widehat{\nabla u}$ is our estimate of the absolute phase gradient. When (2.17) is satisfied, then $\widehat{\nabla u}$ is in fact the gradient of some single-valued differentiable function, and so its integral must be path-independent. Using a standard finite-difference approximation, the discrete analogue of this condition is that

$$\sum_{k=1}^{K-1} \big(\delta(i_{k+1}, i_k + 1)(\Delta_x u)_{i_k j_k} + \delta(i_{k+1}, i_k - 1)(-(\Delta_x u)_{i_{k-1} j_k}) \\ + \delta(j_{k+1}, j_k + 1)(\Delta_y u)_{i_k j_k} + \delta(j_{k+1}, j_k - 1)(-(\Delta_y u)_{i_k j_{k-1}})\big) = 0 \quad (2.18)$$

where δ is the discrete delta function, $i_K = i_1$, $j_K = j_1$, and for each $1 \leq k \leq K - 1$ either $i_{k+1} = i_k$ and $j_{k+1} = j_k \pm 1$, or $i_{k+1} = i_k \pm 1$ and $j_{k+1} = j_k$. The condition (2.18) states that the sum of the discrete estimated derivatives around a discrete closed path in the domain must be zero. This condition can be reduced to a simpler form by noting that any closed curve in a grid can be expressed as a sum of two-by-two closed curves. If the sum of the derivatives around a closed curve in the grid is *not* zero, the sum of derivatives around at least one of the two-by-two closed curves that comprise it must also be nonzero. Therefore it is sufficient to check that the sum of the partial derivatives around each of the two-by-two closed curves in the grid is zero. So a necessary and sufficient condition for path-independent integrals in the grid is that

$$(\Delta_y u)_{ij} + (\Delta_x u)_{i(j+1)} - (\Delta_y u)_{(i+1)j} - (\Delta_x u)_{ij} = 0 \quad (2.19)$$

for all $1 \leq i \leq M-1, 1 \leq j \leq N-1$. Note that the step of discretizing the derivative cannot possibly produce path-dependent integrals, since if we know the true finite differences, (2.19) is always equal to zero. Path-dependent integrals can therefore only introduced by approximating the finite differences. When we use Itoh's assumption, we approximate $(\Delta_d u)_{ij}$ by $W((\Delta_d w)_{ij})$. If Itoh's assumption (2.12) holds at a particular i, j, these two are equal, so (2.19) cannot be violated. Therefore a violation of (2.19) on a particular two-by-two square in the grid must be caused by a violation of (2.12) somewhere in that square. We call squares that violate (2.19) under the approximation $\widehat{(\Delta_d u)}_{ij} = W((\Delta_d w)_{ij})$

residues. Thus each residue in a grid indicates a location at which Itoh's assumption is violated (though not *all* violations may be so indicated). It is possible to make use of this information to help guide the phase unwrapping process.

Itoh's assumption is essentially a method of enforcing smoothness in the solution of the phase unwrapping problem. There are many notions of smoothness that could be used in place of these assumptions. However, these conditions are particularly useful theoretically because whenever they hold, the absolute phase can be exactly reconstructed, and because we can identify some locations at which the assumption is violated. Practically, this assumption makes sense because we often expect that the originating signal is not aliased.

Chapter 3

A probabilistic framework for phase unwrapping

We have seen that the phase unwrapping problem is ill-posed, and therefore we must include additional information to obtain a meaningful solution. We have discussed as an example one possible assumption and the reasoning behind this choice. However, having selected an assumption that reflects the desired properties of a good solution, we need some way of making use of this information in our algorithm. In this chapter we introduce a natural and structured method for combining our additional assumptions with the relationship between the observed data and the unknown solution.

Phase unwrapping is the process of making a "best guess" for the unknown and unobservable absolute phase $\mathbf{s}$ using our measurements $\mathbf{w}$. Under the Bayesian framework, the true conditional distribution $p^{R}_{\mathbf{s}|\mathbf{w}}$ of the unknown absolute phase given the observed wrapped phase captures all of the information necessary to make such a choice, regardless of the criterion we use to decide whether our choice is "best". Of course, there is no way to determine the true distribution $p^R_{\mathbf{s}|\mathbf{w}}$. Instead, we choose a model $p_{\mathbf{s}|\mathbf{w}}$ of this distribution, and so the quality of our decisions is limited by the quality of our model. Therefore we can view the phase unwrapping problem as consisting of two parts: finding a reasonable model $p_{\mathbf{s}|\mathbf{w}}$, and and then making a choice of $\mathbf{s}$ based on this distribution.

For most problems, including the phase unwrapping problem, it is often easier to decompose the desired distribution $p_{\mathbf{s}|\mathbf{w}}$ using Bayes' theorem

$$p_{\mathbf{s}|\mathbf{w}} = \frac{p_{\mathbf{w}|\mathbf{s}} p_{\mathbf{s}}}{p_{\mathbf{w}}} \tag{3.1}$$

where we can find $p_{\mathbf{w}}$ by

$$p_{\mathbf{w}} = \int_{\mathbf{s}} p_{\mathbf{w}|\mathbf{s}} p_{\mathbf{s}}, \tag{3.2}$$

and then to model the distributions $p_{\mathbf{w}|\mathbf{s}}$ and $p_{\mathbf{s}}$. This is because $p_{\mathbf{w}|\mathbf{s}}$ and $p_{\mathbf{s}}$ have natural interpretations related to the problem, and so it is generally easier to decide on a reasonable model for each of these distributions. In particular, Bayes' theorem can be interpreted as a way of modifying our original beliefs about the unknown phenomenon, using the new information introduced by observing the measurable data, to obtain a new belief about the unknown phenomenon. The original beliefs are represented by $p_{\mathbf{s}}$, the modifying influence by $p_{\mathbf{w}|\mathbf{s}}$ and the new beliefs by $p_{\mathbf{s}|\mathbf{w}}$.

Using this interpretation, the conditional distribution $p_{\mathbf{w}|\mathbf{s}}$ of the observed data given the unobserved phenomenon is known as the "likelihood", and is indicative of the way in which the data $\mathbf{w}$ are generated from the phenomenon $\mathbf{s}$. For the phase unwrapping problem, we saw in Chapter 2 that the measurement process adds noise to $\mathbf{s}$ and then wraps the noisy version of $\mathbf{s}$ into $[0, 1)$ to obtain $\mathbf{w}$. The likelihood needs to model this process.

The distribution $p_{\mathbf{s}}$ of the unknown quantity is called the "prior" distribution. In general it expresses an *a priori* preference for certain estimates of $\mathbf{s}$ over others. This is particularly important in the context of the phase unwrapping problem, since in the absence of such a preference the phase unwrapping problem admits infinitely many solutions. The prior distribution we choose must express our prior knowledge of the problem; that is, what it is that makes one solution more meaningful than another. In the context of phase unwrapping it is usually desirable that the original function s is "smooth" in some sense, so the prior distribution usually prefers smooth surfaces.

Viewing the phase unwrapping problem from this probabilistic perspective, we need to perform the following steps. First, we must choose a prior probability model that expresses the desired properties of the solution. Second, we must choose a likelihood model that expresses the way in which the wrapped phase is generated from the unwrapped phase. Third, we must calculate $p_{\mathbf{s}|\mathbf{w}}$, the "posterior" distribution, through Bayes' theorem. Finally, we can decide on an estimate of the absolute phase using the posterior distribution. This framework provides not only an estimate of the unwrapped phase, but also a way to judge the quality of the estimate, in the form of the posterior distribution.

3.1 Estimating the absolute phase from the posterior

There are several ways of choosing an estimate of $\mathbf{s}$ knowing $p_{\mathbf{s}|\mathbf{w}}$. One common method [NPD00] is to choose a mode of the posterior distribution, that is, to find

$$\boldsymbol{s}^* = \arg\max_{\mathbf{s}} p_{\mathbf{s}|\mathbf{w}}(\mathbf{s}, \mathbf{w}). \tag{3.3}$$

The estimate $\mathbf{s}^*$ found using such a procedure is known as the maximum *a posteriori* (MAP) estimate, since we are choosing a value at which the posterior distribution has a maximum. It can be shown to be a Bayes-optimal estimate with respect to a delta loss function [BS94]. Other commonly used estimates are the mean or median of the posterior. However, the MAP estimate has the advantage that the estimation takes the form of an optimization of the posterior, which is computationally easier than the calculation of the expectation of a loss function with respect to the posterior required by Bayes-optimal estimates in general.

Since for an MAP estimate we are concerned only with a maximization with respect to the unknown variables $\mathbf{s}$, we need only know $p_{\mathbf{s}|\mathbf{w}}$ up to additive and multiplicative constants that do not depend on $\mathbf{s}$. In particular, since $p_{\mathbf{w}}$ does not depend on $\mathbf{s}$, we can write

$$\begin{aligned}
\mathbf{s}^* &= \arg\max_{\mathbf{s}} p_{\mathbf{s}|\mathbf{w}}(\mathbf{s},\mathbf{w}) && (3.4)\\
&= \arg\max_{\mathbf{s}} p_{\mathbf{s}|\mathbf{w}}(\mathbf{s},\mathbf{w}) p_{\mathbf{w}}(\mathbf{w}) && (3.5)\\
&= \arg\max_{\mathbf{s}} p_{\mathbf{s},\mathbf{w}}(\mathbf{s},\mathbf{w}) && (3.6)\\
&= \arg\max_{\mathbf{s}} p_{\mathbf{w}|\mathbf{s}}(\mathbf{s},\mathbf{w}) p_{\mathbf{s}}(\mathbf{s}) && (3.7)
\end{aligned}$$

Due to (3.7) it is not necessary to actually calculate the integral in (3.2) to find the MAP estimate. We need merely multiply the prior and the likelihood to obtain the joint distribution of $\mathbf{s}$ and $\mathbf{w}$. In fact, some algorithms do not directly model the prior distribution and likelihood, but instead arrive at the joint distribution through other means. These algorithms then make use of (3.6).

It is often possible to simplify the optimization process by transforming the problem to the logarithmic domain. Since the logarithm is an increasing function, we will have

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} \ \log\big(p_{\mathbf{w}|\mathbf{s}}(\mathbf{s},\mathbf{w})p_{\mathbf{s}}(\mathbf{s})\big) \tag{3.8}$$

$$= \arg\max_{\mathbf{s}} \ \big(\log p_{\mathbf{w}|\mathbf{s}}(\mathbf{s},\mathbf{w}) + \log p_{\mathbf{s}}(\mathbf{s})\big) \tag{3.9}$$

In fact, in many cases the prior distribution and the likelihood are of the form

$$p_{\mathbf{s}}(\mathbf{s}) = \exp\left(-f_{\mathbf{s}}(\mathbf{s})\right) \tag{3.10}$$

$$p_{\mathbf{w}|\mathbf{s}}(\mathbf{s},\mathbf{w}) = \exp\left(-f_{\mathbf{w},\mathbf{s}}(\mathbf{s},\mathbf{w})\right), \tag{3.11}$$

for some functions $f_{\mathbf{s}}$, $f_{\mathbf{w},\mathbf{s}}$, in which case the optimization becomes

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \ \left(f_{\mathbf{w},\mathbf{s}}(\mathbf{s},\mathbf{w}) + f_{\mathbf{s}}(\mathbf{s})\right) \tag{3.12}$$

Placing the optimization in the form (3.12) often makes the optimization process more efficient, but there is a second reason for making such a transformation. The above shows that any phase-unwrapping algorithm that is an optimization of the form in (3.12) can be viewed as a probabilistic algorithm in which the likelihood is given by (3.11) and the prior is given by (3.10). Therefore we can compare any such "optimization" phase-unwrapping algorithm to any "probabilistic" algorithms within the same probabilistic framework. This framework allows us to see what assumptions are implicitly made, in the form of the prior distribution and likelihood, about the nature of the desired solution and the nature of the relationship between the observed data and the generating process respectively.

3.2 Estimating differences in place of point values

So far we have expressed the phase unwrapping problem as the problem of estimating the point absolute phase values $\mathbf{s}$ given the point wrapped absolute phase values $\mathbf{w}$. However, it is possible to change the variable of estimation in several ways. One method is to view the phase unwrapping problem as the problem of recreating a surface from its gradient field. As we saw in Section 2.1, if we know the true discrete gradient field of a surface we can reconstruct it up to a constant (which is the best we can ever hope to do for the phase unwrapping problem). We do not know the true gradient field of the absolute phase surface, but we can attempt to estimate it from the wrapped phase gradients. Let $\boldsymbol{\Delta f}$ represent the set of all x- and y-direction differences of the function f, that is

$$\begin{aligned}\boldsymbol{\Delta f} =& \{(\Delta_x f)_{ij} \mid 1 \leq i \leq M-1, 1 \leq j \leq N\} \\ & \cup \{(\Delta_y f)_{ij} \mid 1 \leq i \leq M, 1 \leq j \leq N-1\}\end{aligned} \tag{3.13}$$

for any function f on the region R in question. Then the problem is to find $p_{\boldsymbol{\Delta s}|\boldsymbol{\Delta w}}$ and then make a choice of $\boldsymbol{\Delta s}$ based on this distribution. To make use of Bayes' theorem, we choose a prior distribution $p_{\boldsymbol{\Delta s}}$ and a likelihood $p_{\boldsymbol{\Delta w}|\boldsymbol{\Delta s}}$.

However, we need to be careful when making our choice of $\boldsymbol{\Delta s}$ based on $p_{\boldsymbol{\Delta s}|\boldsymbol{\Delta w}}$. Since according to vector calculus [Ste99] the set of possible gradient fields is

$$C = \left\{\boldsymbol{\Delta s} \in \mathbb{R}^{MN-M-N} \middle| (\Delta_y s)_{ij} + (\Delta_x s)_{i(j+1)} - (\Delta_y s)_{(i+1)j} - (\Delta_x s)_{ij} = 0 \text{ for each } i, j\right\}, \tag{3.14}$$

that is, the set of conservative vector fields, the MAP estimate should be calculated as

$$\boldsymbol{\Delta s}^* = \underset{\boldsymbol{\Delta s} \in C}{\arg\max}\, p_{\boldsymbol{\Delta s}|\boldsymbol{\Delta w}}(\boldsymbol{\Delta s}, \boldsymbol{\Delta w}). \tag{3.15}$$

This estimate is in fact equivalent[1] to the MAP estimate of $\mathbf{s}$ produced by setting $p_{\mathbf{s}}(\mathbf{s}) = p_{\boldsymbol{\Delta s}}(\boldsymbol{\Delta s})$ and $p_{\mathbf{w}|\mathbf{s}}(\mathbf{w}, \mathbf{s}) = p_{\boldsymbol{\Delta w}|\boldsymbol{\Delta s}}(\boldsymbol{\Delta w}, \boldsymbol{\Delta s})$. However, since constrained optimization is

[1]In the sense that $\boldsymbol{\Delta s}^*$ can be obtained by taking the differences of $\mathbf{s}^*$, and $\mathbf{s}^*$ can be obtained up to a constant by integrating $\boldsymbol{\Delta s}^*$.

generally less efficient than unconstrained optimization, it would be preferable to avoid restricting the optimization to C. This can be done by injecting a preference for conservative vector fields into the prior distribution $p_{\boldsymbol{\Delta s}}$. For example, we can multiply the prior distribution by

$$c(\boldsymbol{\Delta s}) = \prod \delta\big((\Delta_y s)_{ij} + (\Delta_x s)_{i(j+1)} - (\Delta_y s)_{(i+1)j} - (\Delta_x s)_{ij}, 0\big), \tag{3.16}$$

which puts all of the probability mass on conservative vector fields, and therefore is essentially the same as performing the constrained optimization. Alternatively we can multiply the prior distribution by

$$c(\boldsymbol{\Delta s}) = \prod \exp\left(-\frac{1}{2\sigma_c^2}((\Delta_y s)_{ij} + (\Delta_x s)_{i(j+1)} - (\Delta_y s)_{(i+1)j} - (\Delta_x s)_{ij})^2\right), \tag{3.17}$$

which places higher mass on conservative vector fields and lower mass on nonconservative vector fields. The degree of the preference is controlled by σ_c. The “relaxed” curl constraint (3.17) is not guaranteed to produce conservative vector fields as estimates, but (depending on σ_c) will do so often enough to make it useful. Its advantage over the “strict” curl constraint (3.16) is that it may make the optimization more efficient.

3.3 Prior models

In the rest of this chapter we discuss some models that have been applied to the phase unwrapping problem. We begin by a discussion of prior distribution models. As mentioned above, a prior distribution for the phase unwrapping problem needs to quantify what makes one solution preferable to another. It is possible to do this via an application-specific prior model of the unwrapped phase [CF00, CZ01]. However, we restrict our discussion to models that do not include application-specific data, but rather indicate only a preference for "smooth" solutions. Solving ill-posed problems by adding smoothness constraints is a well-established technique known as *regularization.* Both of the prior models discussed below have their origins in the application of regularization to surface reconstruction in computer vision [Sze90].

3.3.1 The first-order prior

The first measure of smoothness we consider is the squared first derivatives of the phase function s [FKMM00, SGPV99, AFKM01, KFPM01, FKP01]. That is, the smaller $s_x^2 + s_y^2$, the "smoother" the function s. In discrete terms this corresponds to preferring small values of

$$\sum (s_{(i+1)j} - s_{ij})^2 + \sum (s_{i(j+1)} - s_{ij})^2 \tag{3.18}$$

Following the discussion in Section 3.1 above, we write this as an exponential-form prior distribution over the absolute phase point values as

$$p_{\mathbf{s}}(\mathbf{s}) \propto \prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{(i+1)j} - s_{ij})^2\right) \prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{i(j+1)} - s_{ij})^2\right) \tag{3.19}$$

We can instead write this as a prior over the absolute phase differences

$$p_{\boldsymbol{\Delta s}}(\boldsymbol{\Delta s}) \propto \prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_x s)_{ij})^2\right) \prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_y s)_{ij})^2\right) \tag{3.20}$$

The first-order prior, also known as the "membrane" model, is not very satisfactory as a smoothness criterion, since it does not satisfy the intuitive notion of what it means to be smooth. Rather it prefers solutions that change locally by small amounts, which does not prevent solutions from being jagged. However it is still commonly used as a prior for phase unwrapping because of its simple form and its relation to Itoh's assumption.

3.3.2 The second-order prior

A more intuitive notion of smoothness uses the second derivatives of the absolute phase function s [MR95, SGPV99, GNPS98]. Accordingly smoother functions have smaller sums of squared second derivatives $s_{xx}^2 + s_{xy}^2 + s_{yx}^2 + s_{yy}^2$. In discrete terms, we prefer solutions with smaller sums

$$\begin{aligned} &\sum((\Delta_x s)_{(i+1)j} - (\Delta_x s)_{ij})^2 + \sum((\Delta_y s)_{i(j+1)} - (\Delta_y s)_{ij})^2 \\ &+ \sum((\Delta_x s)_{i(j+1)} - (\Delta_x s)_{ij})^2 + \sum((\Delta_y s)_{(i+1)j} - (\Delta_y s)_{ij})^2 \end{aligned} \tag{3.21}$$

Again, we write a prior distribution over the absolute phase point values as an exponential form of this summation

$$\begin{aligned} p_{\mathbf{s}}(\mathbf{s}) \propto &\prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{(i+2)j} - 2s_{(i+1)j} + s_{ij})^2\right) \\ &\prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{i(j+2)} - 2s_{i(j+1)} + s_{ij})^2\right) \\ &\prod \exp\left(-\frac{1}{\sigma_p^2}(s_{(i+1)(j+1)} - s_{(i+1)j} - s_{i(j+1)} + s_{ij})^2\right) \end{aligned} \tag{3.22}$$

or a prior distribution over the absolute phase differences as

$$\begin{aligned} p_{\boldsymbol{\Delta s}}(\boldsymbol{\Delta s}) \propto &\prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_x s)_{(i+1)j} - (\Delta_x s)_{ij})^2\right) \\ &\prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_y s)_{i(j+1)} - (\Delta_y s)_{ij})^2\right) \\ &\prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_x s)_{i(j+1)} - (\Delta_x s)_{ij})^2\right) \\ &\prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_y s)_{(i+1)j} - (\Delta_y s)_{ij})^2\right) \end{aligned} \tag{3.23}$$

This prior, also known as the "thin plate" model, is more satisfactory as a measure of smoothness than the first-order model, because it prefers solutions in which adjacent differences have similar values. This corresponds more closely to the intuitive notion of smoothness.

3.4 Likelihood models

The likelihood needs to model the process by which the wrapped phase is generated from the unwrapped phase. As with prior distributions, it is possible to use an application-specific likelihood model [PRS00]. However, we discuss only likelihood models that do not relate specifically to particular applications, but model the transformation of the unwrapped phase into the wrapped phase more generally.

3.4.1 The L^p-norm likelihood

The L^p-norm likelihood model [Cos96, Cos98, Fly97, GP98, GZW88, MR95] makes use of Itoh's assumption. Recall from Section 2.1 that wherever

$$|(\Delta_d s)_{ij}| < 1/2, \tag{3.24}$$

we will have

$$(\Delta_d s)_{ij} = W((\Delta_d w)_{ij}) \tag{3.25}$$

where $d = x, y$. If we assume that Itoh's assumption (3.24) holds nearly everywhere in R, it is reasonable to attempt to match the unwrapped phase differences to the transformed wrapped-phase differences. The measure that is used to evaluate the quality of the match is the Euclidean L^p norm, which is defined as

$$n_p(x, y) = |x - y|^p \tag{3.26}$$

for $p > 0$ and as

$$n_p(x,y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases} \tag{3.27}$$

for $p = 0$. Thus the L^p-norm model requires the minimization of

$$\sum n_p((\Delta_x s)_{ij}, W((\Delta_x s)_{ij})) + \sum n_p((\Delta_y s)_{ij}, W((\Delta_y s)_{ij})) \tag{3.28}$$

with respect to either the unwrapped phase values $\mathbf{s}$ or the unwrapped phase differences $\boldsymbol{\Delta s}$. We can view the L^p-norm model from a probabilistic perspective by setting it into the form described in Section 3.1. That is, we can define a likelihood relating the absolute phase and wrapped phase point values by

$$\begin{aligned} p_{\mathbf{w}|\mathbf{s}}(\mathbf{w},\mathbf{s}) \propto \prod \exp\left(-\frac{1}{2\sigma_l^2} n_p(s_{(i+1)j} - s_{ij}, W(w_{(i+1)j} - w_{ij}))\right) \\ \prod \exp\left(-\frac{1}{2\sigma_l^2} n_p(s_{i(j+1)} - s_{ij}, W(w_{i(j+1)} - w_{ij}))\right) \end{aligned} \tag{3.29}$$

or a likelihood relating the absolute phase and wrapped phase differences by

$$\begin{aligned} p_{\boldsymbol{\Delta w}|\boldsymbol{\Delta s}}(\boldsymbol{\Delta w},\boldsymbol{\Delta s}) \propto \prod \exp\left(-\frac{1}{2\sigma_l^2} n_p((\Delta_x s)_{ij}, W((\Delta_x w)_{ij}))\right) \\ \prod \exp\left(-\frac{1}{2\sigma_l^2} n_p((\Delta_y s)_{ij}, W((\Delta_y w)_{ij}))\right) \end{aligned} \tag{3.30}$$

Several algorithms maximize the L^p-norm likelihood without first multiplying it by a prior distribution [Cos96, Cos98, Fly97, GP98, GZW88]. This can be viewed as setting the joint distribution $p_{\mathbf{s},\mathbf{w}}$ or $p_{\boldsymbol{\Delta s},\boldsymbol{\Delta w}}$ equal to the exponential form described above and then finding the MAP estimate, or instead as finding a maximum likelihood (ML) estimate.

The L^p-norm likelihood model is actually a family of models, since its effects are different depending on the choice of p. For $p = 0$, the likelihood attempts to match the absolute phase differences with the transformed wrapped phase differences at as many locations as possible. At places where they do not match, the amount of separation does not matter. The L^0 model is sometimes considered to be the best of the L^p models. However, the maximization of the L^0 model has been shown to be NP-hard [CZ00]. Still, some "branch cut" minimization algorithms like Goldstein's algorithm [GZW88], which can be viewed as this type of algorithm since the reconstructed phase differences match the transformed wrapped phase differences everywhere except across the branch cuts, approximate the L^0 model reasonably well and produce good results [CZ00].

Using the L^1-norm likelihood model is an attractive option, since the global maximum of this model can be found reasonably efficiently using either network programming techniques [Cos96, Cos98] or using Flynn's algorithm [Fly97]. These algorithms combine the L^1 model with the functional transformation F_1 below to change the variable of estimation. The L^1 model has been found to produce good results in practice [CZ00].

Finding the global maximum of the L^2 likelihood model is extremely efficient, since it can be done for example through a cosine transform [GP98]. However, L^2 solutions tend to "smooth" the surface too much, so estimates based on an L^2 model are not as good as L^0 and L^1 estimates [CZ00]. For $p > 2$ this problem increases, so L^p-norm likelihood models with $p > 2$ are not generally used [GP98].

3.4.2 "Functional" likelihoods

"Functional" likelihoods arise when we use relationships between the wrapped and unwrapped phase to write some of our variables as functions of other variables. Instead of using Bayes' theorem, we change the variable of estimation to find the joint probability of the observed and unobserved variables. The idea is to use the relations

$$w_{ij} = u_{ij} \mod 1 \tag{3.31}$$

$$u_{ij} = w_{ij} + k_{ij} \tag{3.32}$$

from Chapter 2 to transform the noisy absolute phase point values $\mathbf{u}$ to the wrapped phase values $\mathbf{w}$ and the integer shifts $\mathbf{k}$; or similarly to transform absolute phase differences to wrapped phase differences and integer shift differences. Essentially, we are ignoring the stochastic nature of the relationship between the absolute phase and the wrapped phase, and instead treating it as a deterministic relationship.

From probability theory, we know that if A is a random variable and $B = f(A)$ where f is reasonably well-behaved[2], then B is a random variable and

$$\Pr(B \in S) = \Pr(f(A) \in S) = \Pr(A \in f^{-1}(S)) \tag{3.33}$$

where $f^{-1}(S) = \{a | f(a) \in S\}$ is the inverse image of S [Pfe90]. The transformations that we will look at involve a map $f : \mathbb{R} \to \mathbb{Z} \times [c, d)$ such that f is a bijection and the inverse function of f is $f^{-1}(y, z) = y + z$. Let X be a real-valued random variable and let $(Y, Z) = f(X)$. Then Y, Z are random variables. If $S = \{y\} \times [z_1, z_2]$, then $f^{-1}(S) = [y + z_1, y + z_2]$, so from (3.33)

$$\begin{aligned}\Pr(Y = y, z_1 \leq Z \leq z_2) &= \Pr(y + z_1 \leq X \leq y + z_2) && (3.34)\\ &= \int_{y+z_1}^{y+z_2} p_X(x)dx && (3.35)\\ &= \int_{z_1}^{z_2} p_X(y + z)dz && (3.36)\end{aligned}$$

[2] f must be a Borel function.

where the last line is a standard change of variable. But by definition,

$$\Pr(Y = y, z_1 \leq Z \leq z_2) = \int_{z_1}^{z_2} p_{Y,Z}(y, z) dz \tag{3.37}$$

therefore

$$p_{Y,Z}(y, z) = p_X(y + z) \tag{3.38}$$

Thus we have shown that when the random variables X, Y, Z are related by a function f with the properties defined above, then the probability densities of X and Y, Z are related by (3.38). If we think of p_X as a prior distribution, and choose f so that one of the variables Y, Z is observed and the other is unobserved, then applying the function f transforms the prior into a joint distribution that we can use to find an estimate of the unobserved variable using (3.6). In phase unwrapping, the wrapped phase is observed, so we need a function that satisfies the above properties while transforming the absolute phase into the wrapped phase (or a function of the wrapped phase) and some other unobserved variable. We consider two such functions.

The first mapping [GNPS98, SGPV99, Cos96, Cos98, Fly97] we consider is used to transform differences. It is defined as $F_1 : \mathbb{R} \to \mathbb{Z} \times [-\frac{1}{2}, \frac{1}{2})$ given by

$$F_1(x) = (\mathrm{rnd}(x), W(x)) \tag{3.39}$$

where $\mathrm{rnd}(x) = \lfloor x + \frac{1}{2} \rfloor$ rounds its argument to the nearest integer, and $W(x)$ is the wrapping operation of (2.7). Note that $x = W(x) + \text{rnd}(x)$. Then

$$
\begin{aligned}
F_1^{-1}(y,z) &= \{x \mid \text{rnd}(x) = y \text{ and } W(x) = z\} && (3.40)\\
&= \{x \mid \text{rnd}(x) = y\} \cap \{x \mid W(x) = z\} && (3.41)\\
&= \left[y - \frac{1}{2}, y + \frac{1}{2}\right) \cap \{z + n \mid n \in \mathbb{Z}\} && (3.42)\\
&= y + z && (3.43)
\end{aligned}
$$

So F_1 is a bijection and the inverse function of F_1 is $F_1^{-1}(y,z) = y + z$. Applying F_1 to $(\Delta_d u)_{ij}$ gives $(\kappa_{dij}, \omega_{dij})$ where $\omega_{dij} = W((\Delta_d u)_{ij}) = W((\Delta_d w)_{ij})$ and $\kappa_{dij} = \text{rnd}((\Delta_d u)_{ij})$. Thus by (3.38) we can write

$$
p_{\boldsymbol{\kappa},\boldsymbol{\omega}}(\boldsymbol{\kappa}, \boldsymbol{\omega}) = p_{\boldsymbol{\Delta u}}\Big((\Delta_d u)_{ij} = \kappa_{dij} + \omega_{dij}\Big) \qquad (3.44)
$$

Thus applying F_1 transforms a prior distribution over $\boldsymbol{\Delta u}$ to a joint distribution over $\boldsymbol{\kappa}$ and $\boldsymbol{\omega}$. Since we know $\boldsymbol{\omega}$, we can use this joint distribution in (3.6) to find a MAP estimate of $\boldsymbol{\kappa}$. Then we simply apply the inverse of F_1 to each pair $\kappa_{dij}, \omega_{dij}$ to obtain an estimate of the noisy absolute phase differences.

The second mapping [AFKM01, KFPM01, FKP01] we consider transforms point values rather than differences. $F_2 : \mathbb{R} \to \mathbb{Z} \times [0, 1)$ given by

$$
F_2(x) = (\lfloor x \rfloor, x \mod 1) \qquad (3.45)
$$

Then

$$
\begin{aligned}
F_2^{-1}(y,z) &= \{x \mid \lfloor x \rfloor = y \text{ and } x \mod 1 = z\} && (3.46)\\
&= \{x \mid \lfloor x \rfloor = y\} \cap \{x \mid x \mod 1 = z\} && (3.47)\\
&= [y, y+1) \cap \{z + n \mid n \in \mathbb{Z}\} && (3.48)\\
&= y + z && (3.49)
\end{aligned}
$$

So F_2 is a bijection and the inverse function of F_2 is $F_2^{-1}(y, z) = y + z$. Applying F_2 to u_{ij} gives (k_{ij}, w_{ij}). Thus by (3.38) we can write

$$p_{\mathbf{k},\mathbf{w}}(\mathbf{k}, \mathbf{w}) = p_{\mathbf{u}}\Big(u_{ij} = k_{ij} + w_{ij}\Big) \tag{3.50}$$

Again, applying F_2 transforms a prior distribution over $\mathbf{u}$ to a joint distribution over the unknown $\mathbf{k}$ and the known $\mathbf{w}$, so we can use this distribution in (3.6) to find a MAP estimate of $\mathbf{k}$. Then we simply apply the inverse of F_2 to each pair k_{ij}, w_{ij} to obtain an estimate of the noisy absolute phase point values.

Both F_1 and F_2 achieve the similar goal of decomposing the absolute phase into two parts, an integer part and a "fractional" part. These two "functional" likelihoods differ mainly because F_1 is used to transform differences, while F_2 is used to transform point values.

3.4.3 Gaussian likelihood

Another possible likelihood arises from modelling the noise as Gaussian [FKMM00]. Recall from Chapter 2 that

$$s_{ij} + n_{ij} = w_{ij} + k_{ij} \tag{3.51}$$

for each i, j, where n_{ij} is the noise, w_{ij} is the observed wrapped phase, and k_{ij} is an integer shift. If we model the noise as a Gaussian random variable with mean zero and variance σ_l^2, then

$$p_{w_{ij},k_{ij}|s_{ij}}(w_{ij}, k_{ij}, s_{ij}) \propto \exp\left(-\frac{1}{2\sigma_l^2}(w_{ij} + k_{ij} - s_{ij})^2\right) \tag{3.52}$$

Marginalizing out k_{ij} gives

$$p_{w_{ij}|s_{ij}}(w_{ij}, s_{ij}) \propto \sum_{k_{ij}\in\mathbb{Z}} \exp\left(-\frac{1}{2\sigma_l^2}(w_{ij} + k_{ij} - s_{ij})\right) \tag{3.53}$$

Assuming that the w_{ij} are independent, we get a likelihood of the form

$$p_{\mathbf{w}|\mathbf{s}}(\mathbf{w}, \mathbf{s}) \propto \prod_{ij} \sum_{k_{ij}\in\mathbb{Z}} \exp\left(-\frac{1}{2\sigma_l^2}(w_{ij} + k_{ij} - s_{ij})\right) \tag{3.54}$$

This likelihood seems to model the generating process well, but it may be too complex to admit efficient optimization.

3.5 Summary

Clearly there are many possible ways of combining the prior and likelihood models that we have discussed. In Table 3.1 we summarize the combinations that have been applied to the phase unwrapping problem. Two types of prior model, a first-order model and a second-order model, have been commonly used ("-" in the table indicates that no prior model was used; i.e. that the likelihood took the place of the joint). Several likelihood models have been used. In addition, some algorithms estimate absolute phase differences, in which case a curl constraint, either "strict" or "relaxed", must be added to the problem. When absolute phase point values are estimated, no curl constraint is necessary.

We have seen that a probabilistic framework provides a structured way of combining our prior beliefs about phase surfaces with the actual observed wrapped phase. There are many possible variations within this structure, including the choice of prior and likelihood models, and the choice of estimation variable. Because of this, nearly every phase unwrapping algorithm can be cast into this framework. This provides us with a way of comparing algorithms that are seemingly very different.

Table 3.1: Summary of phase unwrapping models.

Prior	Likelihood	PV/Diff	Curl Constraint	Reference
-	L^0	differences	strict	[GZW88]
-	$L^1 + F_1$	differences	strict	[Cos96, Cos98]
-	$L^1 + F_1$	point values	N/A	[Fly97]
-	L^2	point values	N/A	[GP98]
1st-order	gaussian	point values	N/A	[FKMM00]
1st-order/2nd-order	F_1	differences	relaxed	[SGPV99]
1st-order	F_2	differences	relaxed	[AFKM01]
1st-order	F_2	differences	strict	[KFPM01, FKP01]
2nd-order	L^2	point values	N/A	[MR95]
2nd-order	F_1	differences	relaxed	[GNPS98]

Moreover, when viewed within this framework, there are relatively few different types of prior and likelihood model that have been used by general phase-unwrapping algorithms. The fact that phase-unwrapping algorithms can be categorized in this manner suggests the testing method we develop in the next chapter.

Chapter 4

Testing phase unwrapping methods

In order to be able to choose a phase unwrapping method, we need to judge the success of phase unwrapping algorithms in an unbiased way. However, there is little consensus regarding which method should be used to evaluate phase unwrapping algorithms.

One possibility is to test the algorithms on real-world problems. Of course, it is usually the case that these problems consist merely of noisy wrapped phase values and have no "known" solution. This means that the results can only be compared to the results of other algorithms. The comparison is judged by visual inspection or other *ad hoc* methods which are not trustworthy as measures of success. In a few cases [CZ00, CZ01] real-world problems have been tested that have "known" solutions through other means of measurement. However, since the other means also may introduce error, it is difficult to determine which of the reconstructions is responsible for discrepancies between them. Moreover, secondary means of measurement may not be available for all applications of phase unwrapping.

Another possibility is to test the algorithms on simulated data. In this case, we begin with a "known" surface, so the problems of judging success are eliminated. However, we are left with the question of how to choose the simulated datasets. Many authors hand-construct datasets in order to produce particular features that introduce complexity into the phase unwrapping process. However, such "toy" surfaces may be biased toward some algorithms and against others, and in any case, likely do not represent the full range of phase unwrapping features. Sometimes an application-specific probability model is used to generate simulated datasets. This approach is promising since it provides "ground truth" without the bias introduced by hand-constructed data, and it would be ideal for phase-unwrapping algorithms that are specific to that application. However, unless we use models of many different phase-unwrapping applications, this method does not give an overall picture of how well a non-specific algorithm performs.

4.1 A new method for testing

In Chapter 3 we saw that many general phase-unwrapping algorithms rely on one of the three following assumptions about the nature of the original phase surface:

1. That it is modelled well by

$$p_{\mathbf{s}}(\mathbf{s}) \propto \prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{(i+1)j} - s_{ij})^2\right) \prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{i(j+1)} - s_{ij})^2\right) \tag{4.1}$$

2. That it is modelled well by

$$p_{\mathbf{s}}(\mathbf{s}) \propto \prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{(i+2)j} - 2s_{(i+1)j} + s_{ij})^2\right) \\ \prod \exp\left(-\frac{1}{2\sigma_p^2}(s_{i(j+2)} - 2s_{i(j+1)} + s_{ij})^2\right) \\ \prod \exp\left(-\frac{1}{\sigma_p^2}(s_{(i+1)(j+1)} - s_{(i+1)j} - s_{i(j+1)} + s_{ij})^2\right) \tag{4.2}$$

3. That $|(\Delta_d s)_{ij}| < \frac{1}{2}$ for nearly all i, j (algorithms that use an L^p-norm likelihood are based on this assumption).

Note that (1) and (3) above are related: both concern first-order differences. With an appropriately chosen σ_p, (1) could indeed be said to incorporate (3), since (1) will require "nearly all" first-order differences to be less than $\frac{1}{2}$ (e.g. with $\sigma_p = \frac{1}{2}$ about 68% of differences will be less than $\frac{1}{2}$; with $\sigma_p = \frac{1}{4}$ more than 95% will be).

Since general phase-unwrapping algorithms resolve the ambiguity of the phase-unwrapping problem by assuming either (1) or (2) as a prior model for the absolute phase, it makes sense that we can judge an algorithm based on how it performs on surfaces drawn from these models. Therefore we propose that phase-unwrapping algorithms can be well tested by evaluating their performance on data that is created by drawing surfaces from the distributions (4.1) and (4.2) and then wrapping each surface into $[0, 1)$.

Drawing the surfaces from these distributions provides us with a "ground truth" surface, so no *ad hoc* methods need be used to evaluate the quality of the unwrapping; rather we can define an error metric between the estimate provided by an algorithm and the true surface. It also eliminates the problem of bias introduced by hand-selecting toy problems for testing, since the surfaces are generated randomly. Finally, by drawing enough surfaces from these distributions, we will encounter every possible feature that can be described by either (4.1) or (4.2), rather than just the types of features that are specific to one type of phase-unwrapping application.

In summary, we suggest that phase-unwrapping algorithms should be tested by the following procedure:

1. Draw a random surface $\mathbf{s}$ from either (4.1) or (4.2).
2. Calculate the wrapped phase by $w_{ij} = (s_{ij} + n_{ij}) \mod 1$, using some noise model.
3. Use the algorithm to find an estimate $\widehat{\mathbf{s}}$ of $\mathbf{s}$.
4. Compare the estimate to the original surface using some error measure.

We have applied this method to several phase-unwrapping algorithms. The first three, Costantini's algorithm, Flynn's algorithm, and the least-squares algorithm, are all L^p-norm maximum likelihood methods; and represent the best of the general phase-unwrapping algorithms presented by Ghiglia and Pritt [GP98]. The fourth method, the Frey-Koetter algorithm [KFPM01, FKP01], is a recent method that shows potential to improve on these three methods. Finally, we apply this testing procedure to a new algorithm that combines a first-order prior and a functional likelihood. In the next section we describe the details of each of these algorithms, and in Section 4.3 we describe the details of the experimental method and present results.

4.2 Tested algorithms

4.2.1 Goldstein's algorithm

We mentioned in Section 3.4.1 that maximizing the L^0-norm likelihood model is NP-hard. However there are algorithms that approximately maximize this likelihood, usually using some heuristic methods. Goldstein's algorithm [GZW88] approximately maximizes the L^0 likelihood by growing an approximately minimum set of "branch cuts". The process is based on Itoh's assumption. Recall from Section 2.1 that in any area of the grid in which there are no residues, the set of transformed wrapped phase differences $\{W((\Delta_x w)_{ij}), W((\Delta_x w)_{ij})\}$ is a gradient field, by definition of a residue. If Itoh's assumption holds on the area, this gradient field is actually the gradient field of the noisy absolute phase $\mathbf{u}$. Thus it is reasonable to try to divide the grid into areas without residues and then integrate each of these areas using Itoh's algorithm.

This is not quite satisfactory since we need an estimate of $\mathbf{u}$ at *all* of the grid points, including ones that are part of residues. But it is actually sufficient to prevent integration paths from travelling completely around groups of unbalanced residues [GP98], where we call a group of residues balanced if the sum of the residues is zero. This avoids inconsistent results by eliminating some of the paths whenever there are multiple paths that result in different point estimates of $\mathbf{u}$.

Goldstein's algorithm joins residues with "branch cuts" in such a way that each disjoint branch cut is balanced. The integration that follows is prevented from crossing these branch cuts, and thus integration paths can only encircle groups of balanced residues. Since the integration does not cross over branch cuts, the gradient of the resulting estimate will not match the transformed wrapped phase differences wherever a branch cut passes through the grid. Thus the length of the branch cuts is related to the L^0 norm.

To approximately minimize the length of the branch cuts (and thus the L^0 norm), Goldstein's algorithm uses heuristic methods to "grow" a branch cut outward from an initial residue. The branch cut is extended to nearby residues, or to the border of the grid, until the group of connected residues is balanced. Because of the way that the branch cuts are grown, it is possible for sections of the grid to be completely isolated from other sections. If this happens there is no way to relate the phase in one section to another section. This often results in large discrepancies between the original surface and the surface reconstructed by Goldstein's algorithm.

Our implementation follows the description of the algorithm in [GP98].

4.2.2 Costantini's algorithm

Costantini's algorithm [Cos96, Cos98] exactly optimizes the L^1-norm likelihood model through a transformation into a "minimum cost flow" problem on a graph. To maximize the L^1 likelihood, we must minimize

$$\sum |(\Delta_x s)_{ij} - W((\Delta_x w)_{ij})| + \sum |(\Delta_y s)_{ij} - W((\Delta_y w)_{ij})| \tag{4.3}$$

with respect to either the point values $\mathbf{s}$ or the differences $\boldsymbol{\Delta s}$. In order to be able to see the minimum-cost-flow structure, Costantini first applies the transformation F_1 (3.39) to (4.3) to obtain

$$\sum |\kappa_{xij} + \omega_{xij} - W((\Delta_x w)_{ij})| + \sum |\kappa_{yij} + \omega_{yij} - W((\Delta_y w)_{ij})| \tag{4.4}$$

$$= \sum |\kappa_{xij}| + \sum |\kappa_{yij}| \tag{4.5}$$

which we minimize with respect to $\boldsymbol{\kappa}$. Since this means we are dealing in differences, a curl constraint is required. Under F_1 a strict curl constraint

$$(\Delta_y s)_{ij} + (\Delta_x s)_{i(j+1)} - (\Delta_y s)_{(i+1)j} - (\Delta_x s)_{ij} = 0 \tag{4.6}$$

becomes

$$\kappa_{yij} + \omega_{yij} + \kappa_{xi(j+1)} + \omega_{xi(j+1)} - \kappa_{y(i+1)j} - \omega_{y(i+1)j} - \kappa_{xij} - \omega_{xij} = 0, \tag{4.7}$$

or

$$\kappa_{yij} + \kappa_{xi(j+1)} - \kappa_{y(i+1)j} - \kappa_{xij} = -\omega_{yij} - \omega_{xi(j+1)} + \omega_{y(i+1)j} + \omega_{xij}. \tag{4.8}$$

Now the graph problem structure can be seen through the change of variables

$$e^+_{xij} = \max(0, \kappa_{xij}) \tag{4.9}$$

$$e^-_{xij} = -\min(0, \kappa_{xij}) \tag{4.10}$$

$$e^+_{yij} = \max(0, \kappa_{yij}) \tag{4.11}$$

$$e^-_{yij} = -\min(0, \kappa_{yij}) \tag{4.12}$$

so that

$$\kappa_{xij} = e^+_{xij} - e^-_{xij} \tag{4.13}$$

$$|\kappa_{xij}| = e^+_{xij} + e^-_{xij} \tag{4.14}$$

$$\kappa_{yij} = e^+_{yij} - e^-_{yij} \tag{4.15}$$

$$|\kappa_{yij}| = e^+_{yij} + e^-_{yij}. \tag{4.16}$$

Thus our problem becomes

$$\min_{\mathbf{v}^+,\mathbf{v}^-} \sum e^+_{xij} + e^-_{xij} + \sum e^+_{yij} + e^-_{yij} \tag{4.17}$$

subject to

$$e^+_{yij} - e^-_{yij} + e^+_{xi(j+1)} - e^-_{xi(j+1)} - (e^+_{y(i+1)j} - e^-_{yij}) - (e^+_{xij} + e^-_{xij}) = \\ - \omega_{yij} - \omega_{xi(j+1)} + \omega_{y(i+1)j} + \omega_{xij} \quad (4.18)$$

for each i, j, and subject to $e^+_{dij} \geq 0$, $e^-_{dij} \geq 0$ for $d = x, y$ and all i, j. This is a minimum cost flow problem on a graph with one vertex v_{ij} for each constraint (4.18), plus an extra "earth" node. The edges of the graph are formed by connecting each vertex v_{ij} to each of its neighbours $v_{(i-1)j}, v_{(i+1)j}, v_{i(j-1)}, v_{i(j+1)}$ by a directed edge to and a directed edge from that neighbour (subscripts out of range refer instead to the "earth" node). The variables e^+_{xij}, e^-_{xij} represent the flow on the edge from $v_{i(j-1)}$ to v_{ij} and from v_{ij} to $v_{i(j-1)}$ respectively, while the variables e^+_{yij}, e^-_{yij} represent the flow from v_{ij} to $v_{(i-1)j}$ and from $v_{(i-1)j}$ to v_{ij} respectively.

Due to this formulation of the problem as a minimum cost flow on a graph, we can use standard minimum cost flow techniques to find the exact minimization very efficiently. Our implementation uses the minimum cost flow solver MCFv1.2, written by Andreas Loebel [Loe00].

4.2.3 The least-squares algorithm

The least-squares algorithm exactly maximizes the L^2-norm likelihood. This is equivalent to minimizing

$$\sum |(\Delta_x s)_{ij} - W((\Delta_x w)_{ij})|^2 + \sum |(\Delta_y s)_{ij} - W((\Delta_y w)_{ij})|^2 \quad (4.19)$$

with respect to either the point values $\mathbf{s}$ or the differences $\boldsymbol{\Delta s}$. It can be shown [GP98] that the minimization of (4.19) with respect to the point values $\mathbf{s}$ is equivalent to solving the discrete Poisson equations

$$(\Delta_x s)_{(i+1)j} - (\Delta_x s)_{ij} + (\Delta_y s)_{i(j+1)} - (\Delta_y s)_{ij} = \\ W((\Delta_x w)_{(i+1)j}) - W((\Delta_x w)_{ij}) + W((\Delta_y w)_{i(j+1)}) - W((\Delta_y w)_{ij}) \quad (4.20)$$

with appropriate (Neumann) boundary conditions. The discrete Poisson equations can in turn be solved very efficiently using either fast fourier transforms or discrete cosine transforms [GP98]. We use an implementation of the discrete cosine transform method written by Nemanja Petrovic.

4.2.4 The Frey-Koetter algorithm

The model used by Frey-Koetter algorithm [KFPM01, FKP01] combines the first-order prior (3.19) with a strict curl constraint (3.16) and the "functional liklihoood" F_2 (3.45). Thus the joint distribution is

$$\begin{aligned} p_{\boldsymbol{\Delta k},\boldsymbol{\Delta w}}(\boldsymbol{\Delta k},\boldsymbol{\Delta w}) \propto & \prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_x w)_{ij} + (\Delta_x k)_{ij})^2\right) \\ & \prod \exp\left(-\frac{1}{2\sigma_p^2}((\Delta_x w)_{ij} + (\Delta_x k)_{ij})^2\right) \\ & \prod \delta\big((\Delta_y k)_{ij} + (\Delta_x k)_{i(j+1)} - (\Delta_y k)_{(i+1)j} - (\Delta_x k)_{ij}, 0\big) \end{aligned} \quad (4.21)$$

There are no w_{ij} in the curl constraint here, since

$$\begin{aligned} & (\Delta_y k)_{ij} + (\Delta_y w)_{ij} + (\Delta_x k)_{i(j+1)} + (\Delta_x w)_{i(j+1)} \\ & \quad - (\Delta_y k)_{(i+1)j} - (\Delta_y w)_{(i+1)j} - (\Delta_x k)_{ij} - (\Delta_x w)_{ij} \quad (4.22) \\ & = \big((\Delta_y k)_{ij} + (\Delta_x k)_{i(j+1)} - (\Delta_y k)_{(i+1)j} - (\Delta_x k)_{ij}\big) \\ & \quad + \big((\Delta_y w)_{ij} + (\Delta_x w)_{i(j+1)} - (\Delta_y w)_{(i+1)j} - (\Delta_x w)_{ij}\big) \quad (4.23) \\ & = (\Delta_y k)_{ij} + (\Delta_x k)_{i(j+1)} - (\Delta_y k)_{(i+1)j} - (\Delta_x k)_{ij} \quad (4.24) \end{aligned}$$

because the curl of the observed wrapped phase $\mathbf{w}$ must be zero.

To maximize the joint distribution (4.21) directly is difficult because of its form, so instead the authors first find an approximation $q_{\boldsymbol{\Delta k}|\boldsymbol{\Delta w}}$ to the posterior $p_{\boldsymbol{\Delta k}|\boldsymbol{\Delta w}}$ and then use the approximation in the maximization.

The posterior is approximated using "loopy" probability propagation, an extension of probability propagation to models that contain cycles in their graphical representation. Probability propagation itself is a technique that was designed to perform exact inference in models that whose graphical representation is a tree. However, when it is applied to models with cycles there is no guarantee on the quality of the approximation (in fact, the algorithm may not "converge" to a single distribution at all). However, for some models, "loopy" probability propagation has proven very efficient at obtaining approximate values close to the true posterior probabilities.

The form of the approximation $q_{\boldsymbol{\Delta k}|\boldsymbol{\Delta w}}$ treats each $(\Delta_d k)_{ij}$ as independent, so once the algorithm has completed we can find the MAP estimate of $\boldsymbol{\Delta k}^*$ from the approximation by

$$(\Delta_d k)^*_{ij} = \arg\max_{(\Delta_d k)_{ij}} q_{(\Delta_d k)_{ij}|\boldsymbol{\Delta w}}((\Delta_d k)_{ij}, \boldsymbol{\Delta w}) \tag{4.25}$$

for each i, j. Then we reconstruct an estimate of the absolute phase gradient field by adding $(\Delta_d k)^*_{ij}$ to $(\Delta_d w)_{ij}$ at each i, j.

In order to obtain the absolute phase point values from the estimate of its gradient field, we need to integrate it. However, because we only obtain an approximation of the posterior, the strict curl constraint may not be satisfied by the gradient field estimate. If the constraint is satisfied by the gradient estimate, the estimate is integrated directly. If it is not satisfied, the Frey-Koetter algorithm projects the gradient estimate onto the space

C of conservative vector fields (3.14), and then integrates the result.

We use an implementation of the Frey-Koetter algorithm written by Brendan Frey. The value σ_p required by the algorithm is automatically estimated from the wrapped phase variance. To make the computation possible, it is necessary to restrict the range of each $(\Delta_d k)_{ij}$. This implementation restricts the range to $\{-1, 0, 1\}$. For our tests, we try at first 50 iterations. If the vector field returned is non-conservative, we try 50 more iterations. If the estimate is still non-conservative after these 100 iterations, it is projected onto the space of conservative vector fields and the result integrated.

4.2.5 A new algorithm for phase unwrapping

Our model is a combination of the first-order prior (3.19) with the functional likelihood (3.45). Combining the two gives a joint distribution of the form

$$\begin{aligned} p_{\mathbf{k},\mathbf{w}}(\mathbf{k},\mathbf{w}) \propto \prod_{ij} & \exp\left(\frac{1}{2\sigma_p^2}(w_{(i+1)j} + k_{(i+1)j} - w_{ij} - k_{ij})^2\right) \\ & \cdot \exp\left(\frac{1}{2\sigma_p^2}(w_{i(j+1)} + k_{i(j+1)} - w_{ij} - k_{ij})^2\right) \end{aligned} \tag{4.26}$$

To maximize this joint distribution directly requires the solution of

$$\arg\min_{\mathbf{k}} \sum (w_{(i+1)j} + k_{(i+1)j} - w_{ij} - k_{ij})^2 + \sum (w_{i(j+1)} + k_{i(j+1)} - w_{ij} - k_{ij})^2 \tag{4.27}$$

This is a non-linear integer programming problem, which is very costly to solve in general. Therefore we choose instead to approximate the posterior distribution $p_{\mathbf{k}|\mathbf{w}}$ with a mean-field distribution

$$q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) = \prod_{ij} \rho_{ijk_{ij}} \tag{4.28}$$

where $\boldsymbol{\rho} = \{\rho_{ijl}\}$ are the parameters on which the $q_{\boldsymbol{\rho}}$-distribution depends. To make the approximation as tight as possible, we minimize a "distance" between the approximating distribution and the model distribution. The "distance" measure[1] we choose is the Kullback-Leibler (KL) divergence, which is commonly used to measure the amount of separation of two distributions [Fre98]. The KL divergence between $q_{\boldsymbol{\rho}}$ and $p_{\mathbf{k}|\mathbf{w}}$ is given by

$$D(q_{\boldsymbol{\rho}}||p_{\mathbf{k}|\mathbf{w}}) = \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log \frac{q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho})}{p_{\mathbf{k}|\mathbf{w}}(\mathbf{k}, \mathbf{w})} \tag{4.29}$$

We minimize (4.29) with respect to the parameters $\rho_{ijk_{ij}}$ of $q_{\boldsymbol{\rho}}$, with the constraint that $\sum_{k_{ij}} \rho_{ijk_{ij}} = 1$ for each i, j (since $\{\rho_{ijk_{ij}}\}_{k_{ij}\in\mathbb{Z}}$ must be a probability distribution). Thus the problem is to find

$$\begin{aligned}
\boldsymbol{\rho}^* &= \arg\min_{\boldsymbol{\rho}} D(q_{\boldsymbol{\rho}}||p_{\mathbf{k}|\mathbf{w}}) && (4.30)\\
&= \arg\min_{\boldsymbol{\rho}} D(q_{\boldsymbol{\rho}}||p_{\mathbf{k}|\mathbf{w}}) - \log p_{\mathbf{w}}(\mathbf{w}) && (4.31)\\
&= \arg\min_{\boldsymbol{\rho}} \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log \frac{q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho})}{p_{\mathbf{k}|\mathbf{w}}(\mathbf{k}, \mathbf{w})} - \log p_{\mathbf{w}}(\mathbf{w}) && (4.32)\\
&= \arg\min_{\boldsymbol{\rho}} \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log \frac{q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho})}{p_{\mathbf{k}|\mathbf{w}}(\mathbf{k}, \mathbf{w})} - \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log p_{\mathbf{w}}(\mathbf{w}) && (4.33)\\
&= \arg\min_{\boldsymbol{\rho}} \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log \frac{q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho})}{p_{\mathbf{k}|\mathbf{w}}(\mathbf{k}, \mathbf{w}) p_{\mathbf{w}}(\mathbf{w})} && (4.34)\\
&= \arg\min_{\boldsymbol{\rho}} \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log \frac{q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho})}{p_{\mathbf{k},\mathbf{w}}(\mathbf{k}, \mathbf{w})} && (4.35)\\
&= \arg\min_{\boldsymbol{\rho}} M(\boldsymbol{\rho}, \mathbf{k}, \mathbf{w}) && (4.36)
\end{aligned}$$

[1]KL-divergence is not a true distance since it is not symmetric.

where

$$M(\boldsymbol{\rho}, \mathbf{k}, \mathbf{w}) = \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log \frac{q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho})}{p_{\mathbf{k},\mathbf{w}}(\mathbf{k}, \mathbf{w})} \tag{4.37}$$

We minimize M via an iterative scheme by setting the partial derivatives to zero, and based on the resulting equations (see Appendix A) update one $\rho_{ijk_{ij}}$ while keeping all others fixed. Due to the form of the approximation $q_{\boldsymbol{\rho}}$, the update equation for $\rho_{ijk_{ij}}$ depends only on the distributions of its horizontally- and vertically-adjacent neighbours. This means that we can satisfy the constraint that $\sum_{k_{ij}} \rho_{ijk_{ij}} = 1$ for each i, j by calculating all $\rho_{ijk_{ij}}$ for a fixed i, j, and then normalizing the values over the range of k_{ij}. It also suggests a "checkerboard" update strategy where the distribution of every other point is calculated simultaneously, followed by the calculation of the distributions of the remaining points (different schedules are are also possible).

Once we have estimated $\boldsymbol{\rho}^*$, our approximation $q_{\boldsymbol{\rho}}$ to the posterior $p_{\mathbf{k}|\mathbf{w}}$ is fully determined. Since it is a discrete distribution, to choose the MAP estimate of $\mathbf{k}^*$ from this distribution simply means choosing

$$k^*_{ij} = \arg\max_{k_{ij}} \rho_{ijk_{ij}} \tag{4.38}$$

for each i, j. Then an estimate of the absolute phase is reconstructed by adding the integer shift k^*_{ij} to w_{ij} at each point.

For our tests we updated the value of each $\rho_{ijk_{ij}}$ 1000 times. We use a fixed value of $\sigma_p = 0.8$, chosen from trial runs on toy surfaces. For computation purposes, the range for each k_{ij} must be limited. Our tests restrict the k_{ij} values to fall between -5 and 5, a range wide enough so that the true k_{ij} values of the surfaces tested are likely to fall within it, while small enough so that the computation does not grow too costly.

4.3 Results

We have tested Goldstein's algorithm (GOLD), Costantini's algorithm (COST), the least-squares algorithm (LSQ), the Frey-Koetter algorithm (FK), and the new algorithm (NEW) described above using the method described in Section 4.1.

The first step is to sample a surface from (4.1) or (4.2). We do this via Gibbs sampling, a Markov chain Monte Carlo technique (see Appendix B). At each Gibbs sampling iteration we sample a single surface point s_{ij} from the conditional distribution $p_{s_{ij}|\{s_{kl}|(k,l)\neq(i,j)\}}$. If none of the conditional distributions are identically zero, and we repeat this procedure enough times, Markov chain theory tells us that eventually the set of points we have drawn is a sample from $p_{\mathbf{s}}(\mathbf{s})$. Because (4.1) is a Gaussian distribution over the first-order differences of $\mathbf{s}$, and (4.2) is a Gaussian distribution over the second-order differences of $\mathbf{s}$, it turns out that for both cases, the conditional distributions $p_{s_{ij}|\{s_{kl}|(k,l)\neq(i,j)\}}$ are Gaussian. This means that the conditional distributions are not identically zero, and so the conditions required for Gibbs sampling are satisfied. It also means that it is easy to perform the sampling required at each iteration of the Gibbs sampling algorithm.

To generate the surface samples, it is necessary to choose the size of the surface, the variance σ_p^2, and a number of iterations. All of the results we present are on a surface of size 100 by 100. Because of the way in which the surfaces are wrapped (see below), it is not necessary to choose different variances for the sampled surfaces. Thus the variance for all surfaces is set at 0.1. Finally, each surface is generated by sampling every point 5000 times, so that a total of $(5000)(100)(100) = 50\ 000\ 000$ Gibbs iterations are performed. Figure 4.1 shows an example of a first-order surface generated from (4.1) using this technique, and Figure 4.4 shows an example of a second-order surface generated from (4.2).

Once the phase surface is generated, we must wrap it into the interval $[0, 1)$. To simplify the experiment, we choose to ignore the effects of noise. Instead of varying the variance of the model from which the surface is generated, we produce a similar effect by varying the wavelengths used to wrap the surfaces. Accordingly we wrap the phase surface using 20 wavelengths. The wavelengths are equally spaced in the logarithmic domain from a minimum value equal to the variance of the surface to a maximum value large enough so that the entire surface lies within a single wavelength. This means that at the maximum value there are no wrappings, so all algorithms should reconstruct the surface exactly.

After reconstructing the absolute phase using a particular algorithm, we compare the resulting estimate to the original surface using two different error measures. The first is the mean squared error (MSE) between the *point values* of the original surface and the estimate, and the second is the MSE between the *first-order differences* of the original surface and the estimate. We use two measures for the comparison since the point-value measure may be biased toward algorithms that reconstruct point values (LSQ and NEW), while the difference measure may be biased toward algorithms that reconstruct differences (GOLD, COST, and FK).

As the wavelength decreases, more wrappings occur when the absolute phase is transformed into the wrapped phase, so the absolute phase is more difficult to reconstruct. On the other hand, as the wavelength increases, eventually the entire surface will lie within a single wavelength, so no wrappings occur. In this case the absolute phase should be easy to reconstruct (it is simply equal to the wrapped phase). Therefore we should expect that as the wavelength decreases, the mean squared error in both differences and point values should grow large, while as wavelength increases, the mean squared error should tend to zero.

Figure 4.2 shows, for each of five first-order surfaces and for each tested algorithm, the mean squared error of the point values of the reconstructed absolute phase versus the wavelength λ. Similarly, Figure 4.3 shows the mean squared error of the differences of the reconstructed phase for the same surfaces. Each of the curves in these figures have a "waterfall" shape that satisfies the expected properties described above. Both figures show that the Frey-Koetter algorithm performs the best on this type of surface, since its MSE is the first to fall as the wavelength increases, while at small wavelengths its MSE is competitive with the other algorithms. Costantini's algorithm is close behind. The MSE of the new algorithm and Goldstein's algorithm fall more quickly than that of the least-squares algorithm as the wavelength increases, but the MSE of the least-squares algorithm falls earlier than that of these two. Finally, we observe that Goldstein's algorithm has a very large MSE compared to the other algorithms when the wavelength is small. This is probably because at small wavelengths, residues will be close together, so branch cuts will join residues that should not be joined for a correct unwrapping.

Figure 4.5 shows, for each of five second-order surfaces and for each tested algorithm, the mean squared error of the point values of the reconstructed absolute phase versus the wavelength. Similarly, Figure 4.6 shows the mean squared error of the differences of the reconstructed phase for the same surfaces. Again we see the waterfall shape for each curve. The difference-measure graphs show much the same result as the difference-measure graphs for the first-order surfaces, except that the new algorithm performs consistently worse than the other four algorithms.

However, for second-order surfaces there are differences between the point-value-measure graphs and the difference-measure graphs. Notably, using the point-value measure all of the algorithms fare worse at small wavelengths than they do using the difference measure.

Figure 4.1: Example of a generated first-order surface

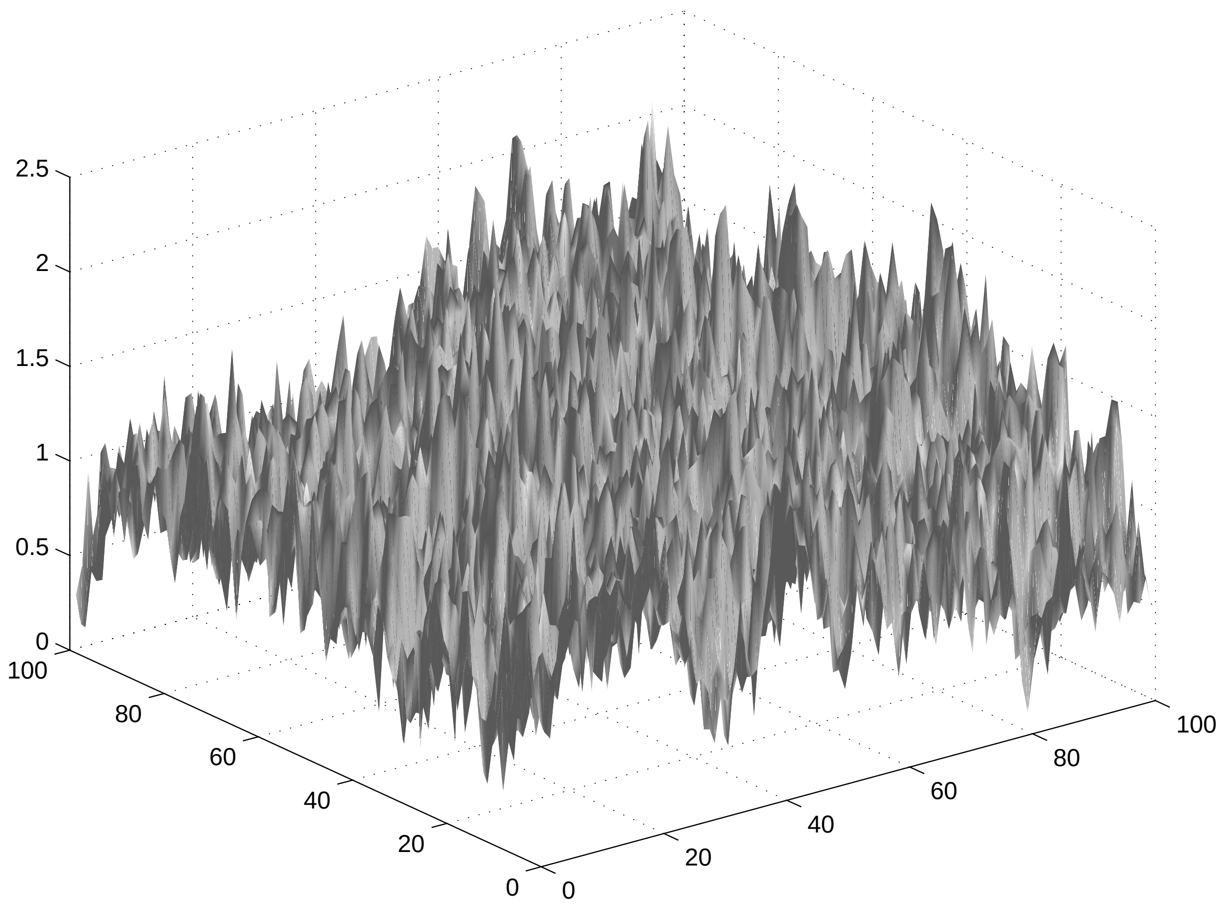


Figure 4.2: Point value MSE for five first-order surfaces

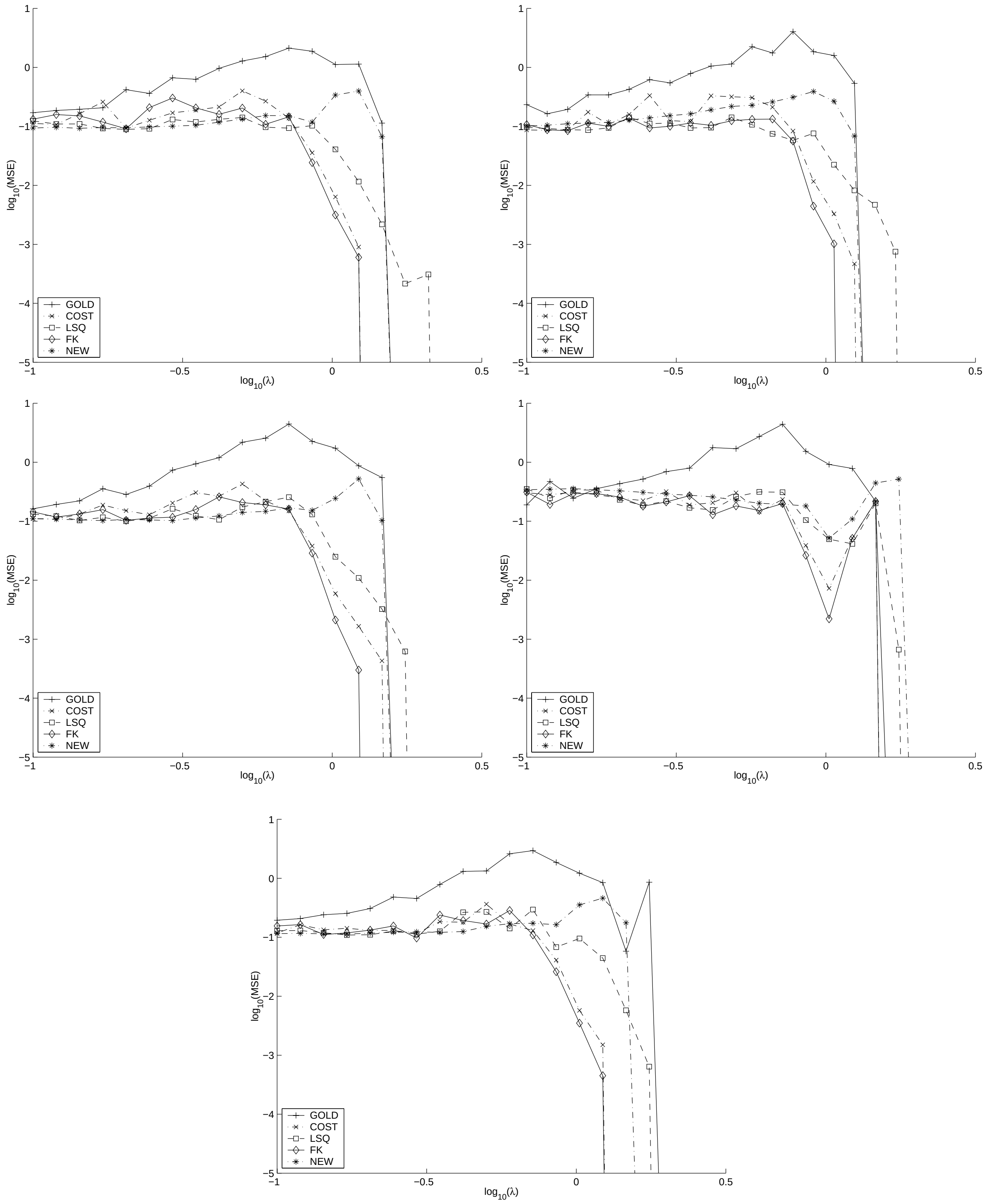


Figure 4.3: Difference MSE for five first-order surfaces

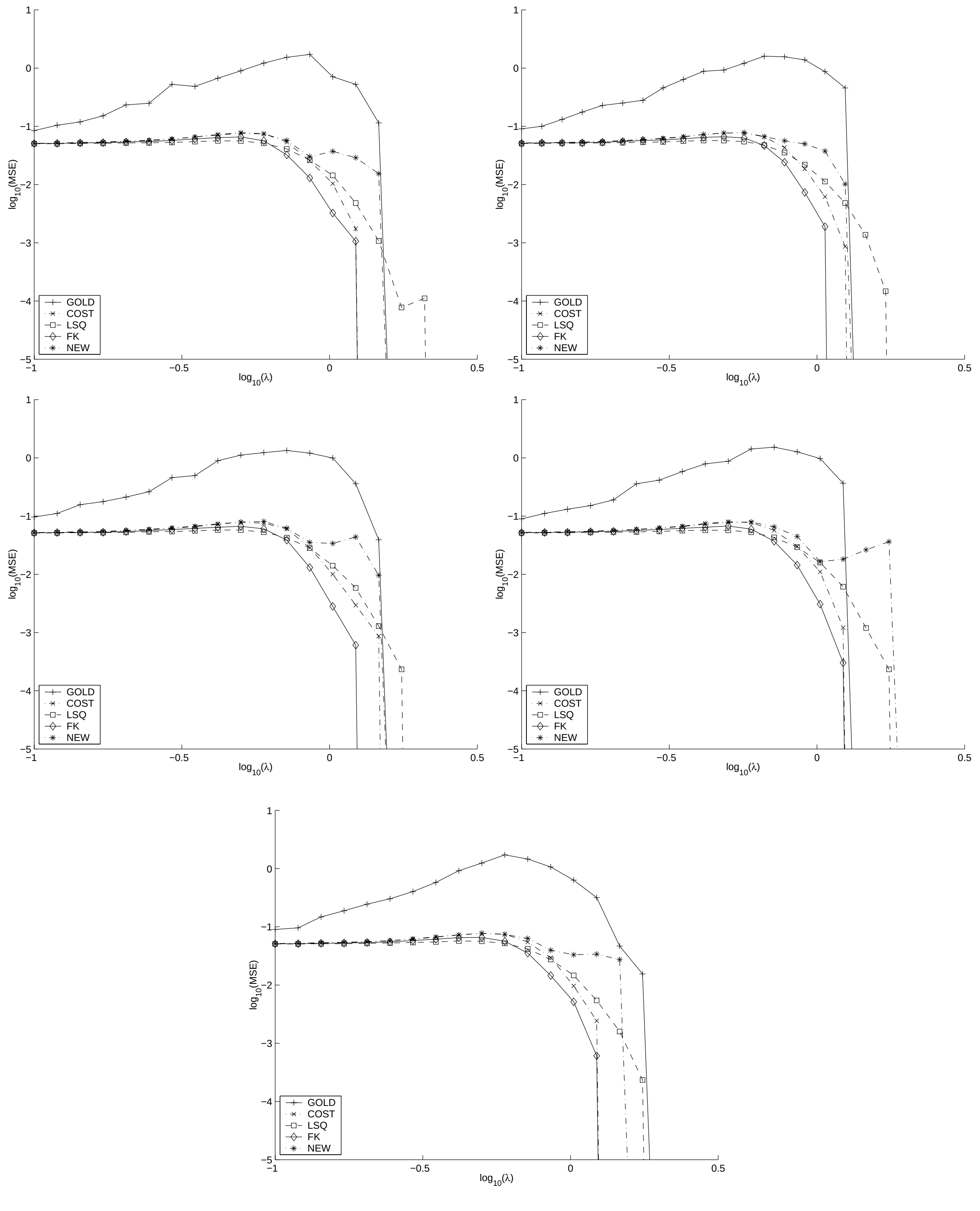


Figure 4.4: Example of a generated second-order surface

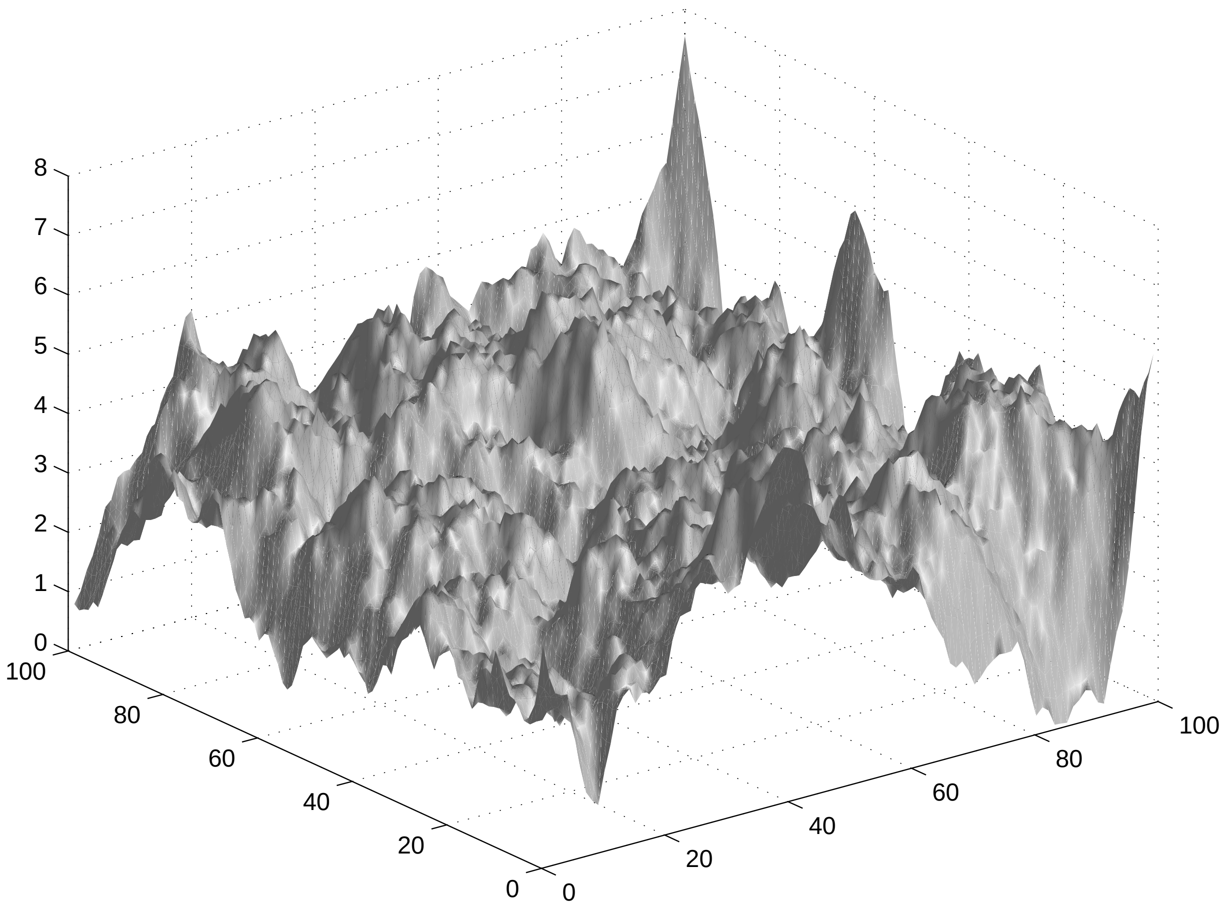


Figure 4.5: Point value MSE for five second-order surfaces

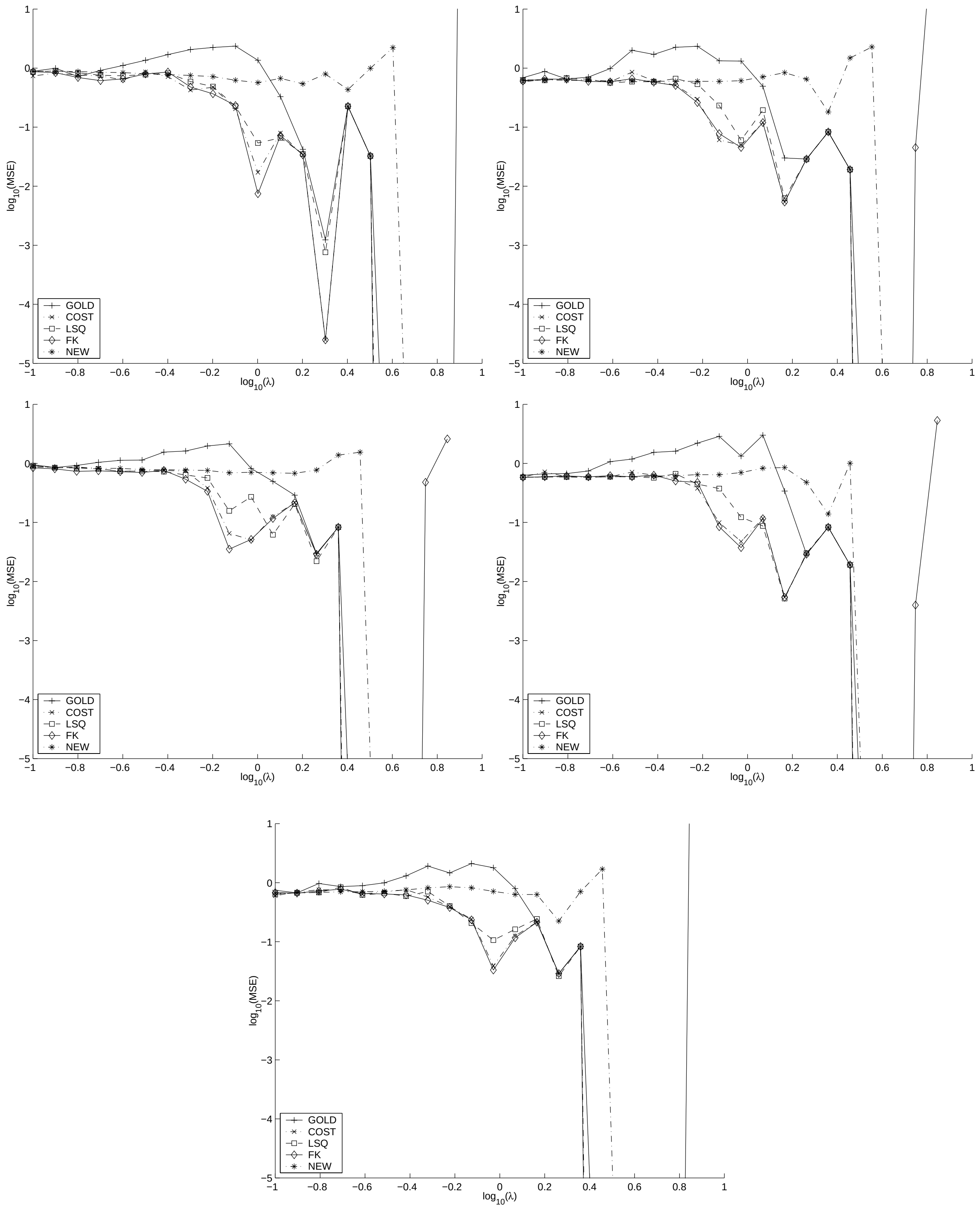
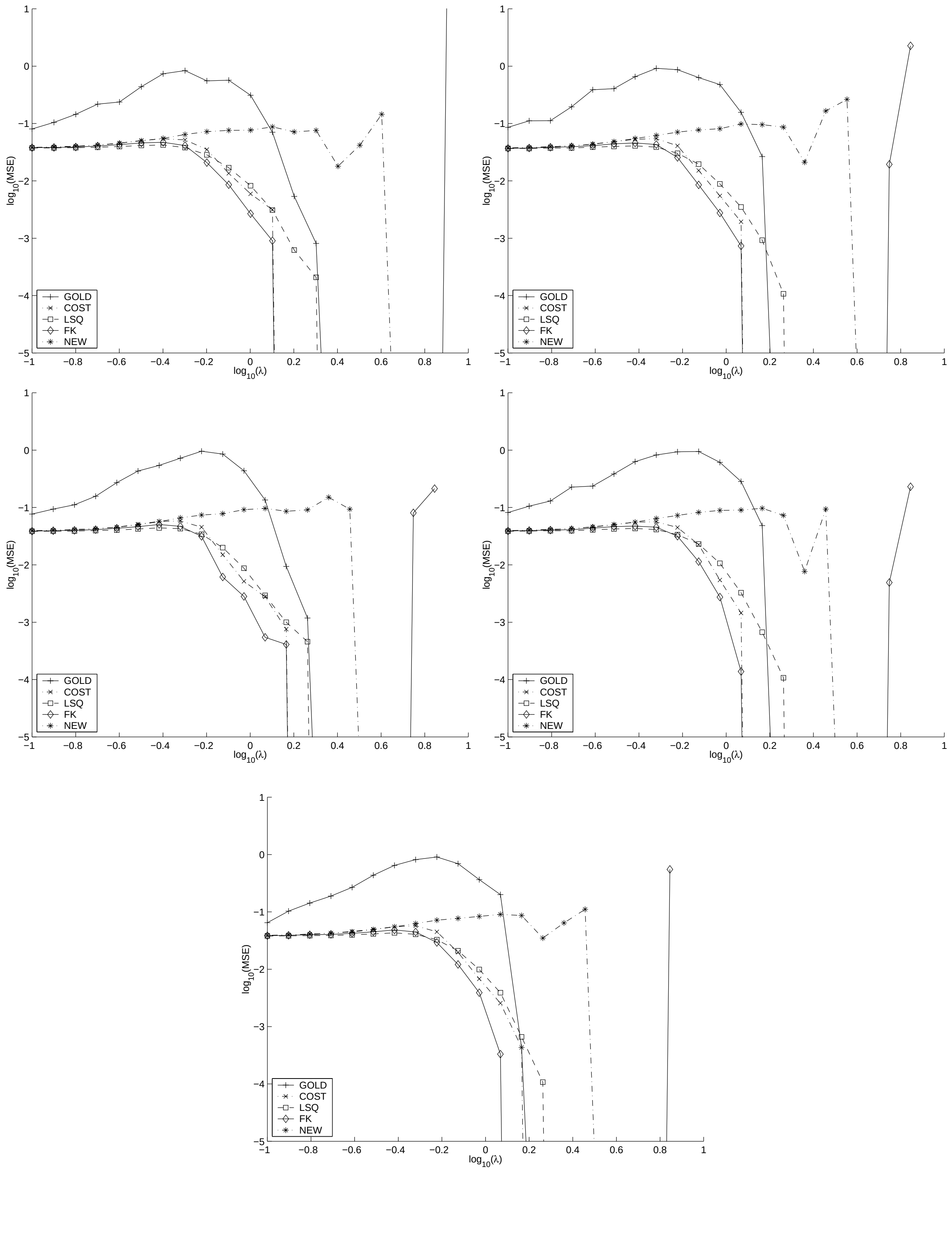


Figure 4.6: Difference MSE for five second-order surfaces



This was not the case for the first-order surfaces. The point-value measure graphs also show that the MSE of all of the algorithms, except the new algorithm, drop to zero at approximately the same wavelength, which is not true when the difference measure is applied. However, using either the point-value measure or the difference measure, the new algorithm performs consistently worse than the other algorithms, except at small wavelengths where Goldstein's algorithm again displays a large MSE.

There is one other notable artifact in the second-order surface graphs: the MSE of the Frey-Koetter algorithm rises sharply when very large wavelengths (compared to the surface height) are reached. This is due to a limitation of the implementation: when the variance σ_p^2 estimated from the wrapped phase is too small, underflow results, and the probability propagation becomes unreliable. This happens when large wavelengths are used because the wrapped phase is normalized to fall within $[0, 1)$.

The graphs of MSE versus wavelength that we produce by testing phase-unwrapping algorithms using this method allow us to quantitatively compare the performance of the tested algorithms. On first-order phase surfaces, the best algorithm is the Frey-Koetter algorithm, followed by Costantini's algorithm, the least-squares algorithm, the new algorithm, and Goldstein's algorithm. For second-order phase surfaces, the Frey-Koetter algorithm is again the best, followed by Costantini's algorithm, the least-squares algorithm, Goldstein's algorithm, and the new algorithm. All of the algorithms perform better on first-order surfaces than on second-order surfaces. This is as expected, since all of the tested algorithms rely on a first-order, rather than a second-order, model for the unwrapped phase surface.

Chapter 5

Conclusions

The phase unwrapping problem in its most general form is not well-posed. It therefore is necessary to make additional assumptions in order to produce useful solutions. There are two possible approaches to making assumptions. The first approach is to make use of additional information specific to a particular application, and therefore to create an algorithm that can only be used on that application. The second approach is to make general assumptions about the behaviour of absolute phase surfaces to create an algorithm with a wide range of possible application.

When the second approach is used, it is usually the case that one of two assumptions are made about the absolute phase. Both of these are probability models that describe the smoothness of the phase. Since these algorithms use these models for the absolute phase, it is reasonable to test the algorithms on surfaces generated from the two models. Accordingly we suggest a method for testing general phase-unwrapping algorithms by generating surfaces from the assumed prior models, wrapping the surfaces and applying the algorithms, and comparing the reconstructed phase to the original phase using an error measure.

There are two advantages of this approach to testing general algorithms. The first is that since we use simulated data, we can compare the reconstruction directly to the "ground truth", and quantify the discrepancies between the two. The second advantage is that since the absolute phase surfaces are randomly generated, they can represent the full range of problems that satisfy the assumed models.

We have used this testing method to compare five algorithms, including a new algorithm. The results show that the Frey-Koetter algorithm has the best performance on surfaces generated from the two prior models, followed closely by Costantini's algorithm. The least-squares algorithm obtains tolerable results on the surfaces. Finally, Goldstein's algorithm and our new algorithm do not perform as well on the tested surfaces as the other algorithms.

5.1 Summary of contributions

- A new quantitative method for testing general phase-unwrapping algorithms, based on models used for the absolute phase.
- A new algorithm for phase unwrapping that combines a first-order prior model with a functional likelihood, and that approximates the resulting posterior distribution using a mean-field distribution.
- A categorization of general phase-unwrapping algorithms according to the prior and likelihood models used.

5.2 Future work

- When testing algorithms using the new method, noise could be added to the generated surfaces before they are wrapped, so that the effects of noise on the algorithms could be compared.
- Other error measures could be used to quantify the discrepancy between the reconstructed phase and the true phase.
- Recently an algorithm has been found that can maximize the joint probability (4.26) of the new algorithm exactly [DL01]. This method, if found to be reasonably efficient, could replace the mean-field approximation we use.
- Better general phase-unwrapping algorithms might be created by combining the prior and likelihood models of Sections 3.3 and 3.4 in new ways. Moreover, new prior and likelihood models might be developed.

Appendix A

Details of the new algorithm

We wish to minimize

$$M(\boldsymbol{\rho}, \mathbf{k}, \mathbf{w}) = \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) - \sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k}, \boldsymbol{\rho}) \log p_{\mathbf{k},\mathbf{w}}(\mathbf{k}, \mathbf{w}). \tag{A.1}$$

with respect to $\boldsymbol{\rho}$. Expanding M using the form of $p_{\mathbf{k},\mathbf{w}}$ from (4.26) and the form of approximation $q_{\boldsymbol{\rho}}$ from (4.28), the first term of M becomes

$$\begin{aligned}
\sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k},\boldsymbol{\rho}) \log q_{\boldsymbol{\rho}}(\mathbf{k},\boldsymbol{\rho}) &= \sum_{\mathbf{k}} \prod_{i'j'} \rho_{i'j'k_{i'j'}} \log \prod_{ij} \rho_{ijk_{ij}} && \text{(A.2)}\\
&= \sum_{\mathbf{k}} \prod_{i'j'} \rho_{i'j'k_{i'j'}} \sum_{ij} \log \rho_{ijk_{ij}} && \text{(A.3)}\\
&= \sum_{ij} \sum_{\mathbf{k}} \prod_{i'j'} \rho_{i'j'k_{i'j'}} \log \rho_{ijk_{ij}} && \text{(A.4)}\\
&= \sum_{ij} \sum_{k_{11}} \sum_{k_{12}} \cdots \sum_{k_{mn}} \rho_{11k_{11}} \cdots (\rho_{ijk_{ij}} \log \rho_{ijk_{ij}}) \cdots \rho_{mnk_{mn}} && \text{(A.5)}\\
&= \sum_{ij} \sum_{k_{11}} \rho_{11k_{11}} \cdots \sum_{k_{ij}} (\rho_{ijk_{ij}} \log \rho_{ijk_{ij}}) \cdots \sum_{k_{mn}} \rho_{mnk_{mn}} && \text{(A.6)}\\
&= \sum_{ij} \sum_{k_{ij}} (\rho_{ijk_{ij}} \log \rho_{ijk_{ij}}) && \text{(A.7)}
\end{aligned}$$

and the second term of M becomes

$$\begin{aligned}
&\sum_{\mathbf{k}} q_{\boldsymbol{\rho}}(\mathbf{k},\boldsymbol{\rho}) \log p_{\mathbf{k},\mathbf{w}}(\mathbf{k},\mathbf{w})\\
&= \sum_{\mathbf{k}} \prod_{i'j'} \rho_{i'j'k_{i'j'}} \log \prod_{ij} \left[\exp\left(\frac{1}{2\sigma^2} (w_{(i+1)j} + k_{(i+1)j} - w_{ij} - k_{ij})^2 \right) \right.\\
&\qquad\qquad \left. \cdot \exp\left(\frac{1}{2\sigma^2} (w_{i(j+1)} + k_{i(j+1)} - w_{ij} - k_{ij})^2 \right) \right] && \text{(A.8)}\\
&= \sum_{\mathbf{k}} \prod_{i'j'} \rho_{i'j'k_{i'j'}} \sum_{ij} \frac{1}{2\sigma^2} \Big((w_{(i+1)j} + k_{(i+1)j} - w_{ij} - k_{ij})^2\\
&\qquad\qquad + (w_{i(j+1)} + k_{i(j+1)} - w_{ij} - k_{ij})^2 \Big) && \text{(A.9)}\\
&= \frac{1}{2\sigma^2} \sum_{ij} \Bigg(\sum_{k_{ij}} \sum_{k_{(i+1)j}} \rho_{ijk_{ij}} \rho_{(i+1)jk_{(i+1)j}} (w_{(i+1)j} + k_{(i+1)j} - w_{ij} - k_{ij})^2\\
&\qquad\qquad + \sum_{k_{ij}} \sum_{k_{i(j+1)}} \rho_{ijk_{ij}} \rho_{i(j+1)k_{i(j+1)}} (w_{i(j+1)} + k_{i(j+1)} - w_{ij} - k_{ij})^2 \Bigg) && \text{(A.10)}
\end{aligned}$$

Thus

$$\begin{aligned}M(\boldsymbol{\rho},\mathbf{k},\mathbf{w}) = &\sum_{ij}\sum_{k_{ij}}(\rho_{ijk_{ij}}\log\rho_{ijk_{ij}})\\ &+\frac{1}{2\sigma^2}\sum_{ij}\Bigg(\sum_{k_{ij}}\sum_{k_{(i+1)j}}\rho_{ijk_{ij}}\rho_{(i+1)jk_{(i+1)j}}(w_{(i+1)j}+k_{(i+1)j}-w_{ij}-k_{ij})^2\\ &\qquad+\sum_{k_{ij}}\sum_{k_{i(j+1)}}\rho_{ijk_{ij}}\rho_{i(j+1)k_{i(j+1)}}(w_{i(j+1)}+k_{i(j+1)}-w_{ij}-k_{ij})^2\Bigg)\end{aligned} \tag{A.11}$$

We minimize (A.11) by taking its partial first derivatives

$$\begin{aligned}\frac{\partial M(\boldsymbol{\rho},\mathbf{k},\mathbf{w})}{\partial\rho_{ijk_{ij}}} =&1+\log\rho_{ijk_ij}\\ &+\frac{1}{2\sigma_p^2}\sum_{k_{(i-1)j}}\rho_{ijk_{(i-1)j}}(w_{ij}+k_{ij}-w_{(i-1)j}-k_{(i-1)j})^2\\ &+\frac{1}{2\sigma_p^2}\sum_{k_{(i+1)j}}\rho_{ijk_{(i+1)j}}(w_{(i+1)j}+k_{(i+1)j}-w_{ij}-k_{ij})^2\\ &+\frac{1}{2\sigma_p^2}\sum_{k_{i(j-1)}}\rho_{ijk_{i(j-1)}}(w_{ij}+k_{ij}-w_{i(j-1)}-k_{i(j-1)})^2\\ &+\frac{1}{2\sigma_p^2}\sum_{k_{i(j+1)}}\rho_{ijk_{i(j+1)}}(w_{i(j+1)}+k_{i(j+1)}-w_{ij}-k_{ij})^2\end{aligned} \tag{A.12}$$

and setting them equal to zero. To approximately solve the nonlinear system we update the value of each $\rho_{ijk_{ij}}$ individually while leaving the other variables fixed. The update equation for $\rho_{ijk_{ij}}$ is thus

$$\begin{aligned}
\rho_{ijk_{ij}} = \exp\Bigg(- \Bigg(& \frac{1}{2\sigma_p^2} \sum_{k_{(i-1)j}} \rho_{ijk_{(i-1)j}} (w_{ij} + k_{ij} - w_{(i-1)j} - k_{(i-1)j})^2 \\
& + \frac{1}{2\sigma_p^2} \sum_{k_{(i+1)j}} \rho_{ijk_{(i+1)j}} (w_{(i+1)j} + k_{(i+1)j} - w_{ij} - k_{ij})^2 \\
& + \frac{1}{2\sigma_p^2} \sum_{k_{i(j-1)}} \rho_{ijk_{i(j-1)}} (w_{ij} + k_{ij} - w_{i(j-1)} - k_{i(j-1)})^2 \\
& + \frac{1}{2\sigma_p^2} \sum_{k_{i(j+1)}} \rho_{ijk_{i(j+1)}} (w_{i(j+1)} + k_{i(j+1)} - w_{ij} - k_{ij})^2 + 1 \Bigg) \Bigg).
\end{aligned} \tag{A.13}$$

Appendix B

Gibbs sampling

Gibbs sampling is a Monte Carlo technique that is used to draw samples from a (joint) distribution that is difficult to sample from directly. It can be used when the conditional probability of each variable of the distribution given the other variables takes a form that is reasonably easy to sample from. That is, to use Gibbs sampling to sample from $p_{\mathbf{X}}(\mathbf{X})$ where $\mathbf{X} = (X_1, X_2, \ldots, X_n)$. we need to be able to easily draw samples from

$$p_{X_i|\{X_j|j\neq i\}}(X_1, X_2, \ldots, X_n) \tag{B.1}$$

for each $1 \leq i \leq n$. The process of Gibbs sampling is as follows.

1. Let $\mathbf{X}^{(0)} = (X_1^{(0)}, X_2^{(0)}, \ldots, X_n^{(0)})$. Let $t = 0$.

2. Draw an integer i without replacement from 1 to n.

3. Draw a value Y from $p_{X_i|\{X_j|j\neq i\}}(\mathbf{X}^{(t)})$.

4. Let $\mathbf{X}^{(t+1)} = (X_1^{(t)}, \ldots, X_{i-1}^{(t)}, Y, X_{i+1}^{(t)}, \ldots, X_n^{(t)})$. Let $t = t + 1$.

5. Repeat (2)-(4) until we have drawn each integer from 1 to n. Reset so that all integers from 1 to n are marked as undrawn.

6. Repeat (2)-(5) for some number of iterations.

The effect of performing the above algorithm is to simulate a single realization of a particular type of Markov chain. Markov chains in general are sequences of random variables $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots$ that share a common range S and whose distributions are related by

$$p_{\mathbf{X}^{(t+1)}|\{\mathbf{X}^{(0)},\mathbf{X}^{(1)},\ldots,\mathbf{X}^{(t)}\}} = p_{\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}} \tag{B.2}$$

that is, the distribution of $\mathbf{X}^{(t+1)}$ is conditionally independent of $\{\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}\}$ given $\mathbf{X}^{(t)}$. This means that

$$p_{\mathbf{X}^{(t+1)}}(\mathbf{X}^{(t+1)}) = \int_S p_{\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}}(\mathbf{X}^{(t+1)}, \mathbf{X}) p_{\mathbf{X}^{(t)}}(\mathbf{X}) d\mathbf{X} \tag{B.3}$$

for each t. The conditional probabilities $p_{\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}}$ are generally referred to as *transition distributions* since they represent the "transition" from $\mathbf{X}^{(t)}$ to $\mathbf{X}^{(t+1)}$. Because of this relation, the distributions of the sequence $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots$ is fully determined by specifying the transition distributions and the initial distribution $p_{\mathbf{X}^{(0)}}$.

The Markov chain to which Gibbs sampling is related has transition distributions

$$\begin{aligned} p_{\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}}(\mathbf{X}^{(t+1)}, \mathbf{X}^{(t)}) = & p_{X_i|\{X_j|j\neq i\}}(X_1^{(t)}, \ldots, X_{i-1}^{(t)}, X_i^{(t+1)}, X_{i+1}^{(t)}, \ldots, X_n^{(t)}) \\ & \cdot \prod_{j\neq i} \delta(X_j^{(t+1)}, X_j^{(t)}) \end{aligned} \tag{B.4}$$

for each t. We can show that if one of the distributions $p_{\mathbf{X}^{(t)}}$ is equal to the original joint distribution $p_{\mathbf{X}}$, then $p_{\mathbf{X}^{(t+1)}}$ will be equal to $p_{\mathbf{X}}$ as well:

$$\begin{aligned}
&p_{\mathbf{X}^{(t+1)}}(\mathbf{X}^{(t+1)}) \\
&= \int p_{\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}}(\mathbf{X}^{(t+1)}, \mathbf{X}) p_{\mathbf{X}^{(t)}}(\mathbf{X}) \, d\mathbf{X} && \text{(B.5)} \\
&= \int p_{X_i|\{X_j|j\neq i\}}(X_1, \ldots, X_{i-1}, X_i^{(t+1)}, X_{i+1}, \ldots, X_n) \prod_{j\neq i} \delta(X_j^{(t+1)}, X_j) \\
&\qquad \cdot p_{\mathbf{X}}(\mathbf{X}) \, d\mathbf{X} && \text{(B.6)} \\
&= \int p_{X_i|\{X_j|j\neq i\}}(X_1, \ldots, X_{i-1}, X_i^{(t+1)}, X_{i+1}, \ldots, X_n) \prod_{j\neq i} \delta(X_j^{(t+1)}, X_j) \\
&\qquad \cdot p_{X_i|\{X_j|j\neq i\}}(\mathbf{X}) p_{\{X_j|j\neq i\}}(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n) \, d\mathbf{X} && \text{(B.7)} \\
&= p_{X_i|\{X_j|j\neq i\}}(\mathbf{X}^{(t+1)}) p_{\{X_j|j\neq i\}}(X_1^{(t+1)}, \ldots, X_{i-1}^{(t+1)}, X_{i+1}^{(t+1)}, \ldots, X_n^{(t+1)}) \\
&\qquad \int p_{X_i|\{X_j|j\neq i\}}(X_1^{(t+1)}, \ldots, X_{i-1}^{(t+1)}, X_i, X_{i+1}^{(t+1)}, \ldots, X_n^{(t+1)}) \, dX_i && \text{(B.8)} \\
&= p_{\mathbf{X}}(\mathbf{X}^{(t+1)}) && \text{(B.9)}
\end{aligned}$$

Because $p_{\mathbf{X}}$ is unchanged by the transitions of the Gibbs sampling Markov chain, we call it an *invariant* distribution of the chain. Note that if $p_{\mathbf{X}^{(T)}} = p_{\mathbf{X}}$ then $p_{\mathbf{X}^{(t)}} = p_{\mathbf{X}}$ for all $t > T$.

As long as each of the conditional distributions (B.1) is not identically zero, we are assured that (1) there is a nonzero probability that the Markov chain will move from one state to any other state after enough iterations (the chain is then said to be *irreducible*) and (2) that the chain has a nonzero probability of remaining in the same state [Nea93]. The combination of these conditions imply that the Markov chain is *ergodic*: for t large enough, $p_{\mathbf{X}^{(t)}}(\mathbf{X}^{(t)}) = \pi(\mathbf{X}^{(t)})$ for some (unique) invariant distribution π, no matter what is chosen for the initial distribution $p_{\mathbf{X}^{(0)}}$. Since this invariant distribution must be unique, and we have already shown that $p_{\mathbf{X}}$ is invariant under the transition distributions (B.4), we can conclude that this Markov chain converges to the joint distribution.

Therefore, as long as each of the conditional probabilities (B.1) is not identically zero, for all iterations t after some iteration T each of the values $\mathbf{X}^{(t)}$ produced by the algorithm described at the top of this section is a sample from the joint distribution $p_{\mathbf{X}}$.

Bibliography

[AFKM01] Kannan Achan, Brendan J. Frey, Ralf Koetter, and David Munson. Unwrapping phases by relaxed mean field inference. In *Proceedings of the 2001 International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, 2001.

[BS94] José M. Bernardo and Adrian F. M. Smith. *Bayesian theory.* Wiley series in probability and mathematical statistics. John Wiley and Sons, Chichester, 1994.

[CF00] Gabriel F. Carballo and Paul W. Fieguth. Probabilistic cost functions for network flow phase unwrapping. *IEEE Transactions on Geoscience and Remote Sensing*, 38(5):2192–2201, September 2000.

[Cos96] Mario Costantini. A phase unwrapping method based on network programming. In *Proceedings of the Fringe '96 Workshop on ERS SAR Interferometry*, pages 261–272, Zurich, Switzerland, 1996.

[Cos98] Mario Costantini. A novel phase unwrapping method based on network programming. *IEEE Transactions on Geoscience and Remote Sensing*, 36(3):813–

821, May 1998.

[CZ00] Curtis W. Chen and Howard A. Zebker. Network approaches to two-dimensional phase unwrapping: intractibility and two new algorithms. *Journal of the Optical Society of America A*, 17(3):401–414, March 2000.

[CZ01] Curtis W. Chen and Howard A. Zebker. Two-dimensional phase unwrapping with use of statistical models for cost functions in nonlinear optimization. *Journal of the Optical Society of America A*, 18(2):338–351, February 2001.

[DL01] José M. B. Dias and José M. N. Leitão. A discrete/continuous minimization method in interferometric image processing. Draft accepted for EMMCVPR-2001, June 2001.

[FKMM00] Brendan J. Frey, Ralf Koetter, Jodi Moran, and David C. Munson, Jr. Variations on phase unwrapping. In *International Symposium on Information Theory and Its Applications*, Honolulu, Hawaii, November 2000.

[FKP01] Brendan J. Frey, Ralf Koetter, and Nemanja Petrovic. Very loopy belief propagation for unwrapping phase images. Submitted to Neural Information Processing Systems Conference, 2001.

[Fly97] Thomas J. Flynn. Two-dimensional phase unwrapping with minimum weighted discontinuity. *Journal of the Optical Society of America A*, 14(10):2692–2701, oct 1997.

[Fre98] Brendan J. Frey. *Graphical Models for Machine Learning and Digital Communication.* Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 1998.

[GNPS98] Luciano Guerriero, Giovanni Nico, Guido Pasquariello, and Sebastiano Stramaglia. New regularization scheme for phase unwrapping. *Applied Optics*, 37(14):3053–3058, May 1998.

[GP98] Dennis C. Ghiglia and Mark D. Pritt. *Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software.* John Wiley & Sons, Inc., 1998.

[GZW88] Richard M. Goldstein, Howard A. Zebker, and Charles L. Werner. Satellite radar interferometry: two-dimensional phase unwrapping. *Radio Science*, 23(4):713–720, July 1988.

[Ito82] Kazuyoshi Itoh. Analysis of the phase unwrapping algorithm. *Applied Optics*, 21(14):2470, July 1982.

[KFPM01] Ralf Koetter, Brendan J. Frey, Nemanja Petrovic, and David C. Munson, Jr. Unwrapping phase images by propagating probabilities across graphs. In *Proceedings of the 2001 International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT, May 2001.

[Loe00] Andreas Loebel. MCF Version 1.2 – A network simplex implementation, 2000. Available for academic use free of charge via WWW at www.zib.de.

[MR95] Jose L. Marroquin and Mariano Rivera. Quadratic regularization functionals for phase unwrapping. *Journal of the Optical Society of America A*, 12(11):2393–2400, November 1995.

[Nea93] Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, Department of Computer Science, University of Toronto, 1993.

[NPD00] Giovanni Nico, Gintautas Palubinskas, and Mihai Datcu. Bayesian approaches to phase unwrapping: theoretical study. *IEEE Transactions on Signal Processing*, 48(9):2545–2556, September 2000.

[Pfe90] Paul E. Pfeiffer. *Probability for applications.* Springer texts in statistics. Springer-Verlag, New York, 1990.

[PRS00] Giovanni Poggio, Arturo R. P. Ragozini, and Daniela Servadei. A bayesian approach for SAR interferometric phase restoration. In *Proceedings of the 2000 International Geoscience and Remote Sensing Symposium*, pages 3202–3205. IEEE, July 2000.

[SGPV99] Sebastiano Stramaglia, Luciano Guerriero, Guido Pasquariello, and Nicola Veneziani. Mean-field annealing for phase unwrapping. *Applied Optics*, 38(8):1377–1383, March 1999.

[Ste99] James Stewart. *Calculus: early vectors.* Brooks/Cole Publishing Company, Pacific Grove, preliminary edition, 1999.

[Sze90] Richard Szeliski. Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, 5(3):271–301, December 1990.