Chapter 9
High-Dimension Model Representation via Sparse Grid Techniques

9.1 Introduction

The question of high-dimension model representation (HDMR) is of increasing importance in computational mathematics and science. We introduced this subject in Chaps. 7 and 8, where we invoked the Karhunen-Loève expansion to reduce the number of random parameters that are required to define the stochastic model. We continue the discussion of HDMR in this chapter by turning our attention to the question of determining a suitable surrogate model for computing the response of the forward problem via **VIC-3D®**. This surrogate takes the form of an interpolation table, which is then transformed into the conventional table used in NLSE for solving inverse problems. The surrogate model that we seek falls under the rubric *sparse grids*, and has been the subject of intensive research in a number of areas in recent years [15, 22, 43, 44, 46, 57, 58, 73, 121, 123, 146]. We will apply it to solving problems of model-based inversion as was developed in [111]. Sparse grids can also be used to effectively calculate high-dimensional integrals of the form (7.2).

9.2 Mathematical Structure of the Problem

The problems in this set are based on Fig. 6.3, and required 81 **VIC-3D®** runs to establish the interpolating grid. Thus, we say that the grid has 81 nodes in four-dimensional space, speaking abstractly. Each variable (the slab depth in Fig. 6.3) defines a dimension of the grid. The fifth problem introduced another variable, the width, with three possible values, making the overall grid a hypercube of 243 nodes in five-dimensional space. As we add dimensions (variables), we will soon encounter the 'curse of dimensionality,' because each node requires a **VIC-3D®** run to produce the corresponding blending function. The question arises as to

H. A. Sabbagh et al., *Advanced Electromagnetic Models for Materials Characterization and Nondestructive Evaluation*, Scientific Computation,
https://doi.org/10.1007/978-3-030-67956-9_9

whether we can reduce the number of runs by reducing the number of nodes in the interpolating grid. The answer lies in the notion of 'sparse grids.' This is an important area of study in numerical methods, and we refer the reader to [43] for a brief tutorial. We will follow the presentation and nomenclature of [43], using the complex-flaw model of Fig. 6.3 as an example. Further interesting applications of the sparse grid approach can be found in [44] and [146].

Referring to Fig. 6.3, we can define the mathematical structure of the problem by the abstract formula (0, 10, 20) ⊗ (0, 10, 20) ⊗ (0, 10, 20) ⊗ (0, 10, 20), where ⊗ represents the Cartesian or 'direct' or 'tensor' product. Thus, we have a problem that is defined on a four-dimensional hypercube with 16 corner nodes, given by (0, 20) ⊗ (0, 20) ⊗ (0, 20) ⊗ (0, 20) and 65 interior nodes given by the 'direct-difference' between the total nodes given above and the corner nodes. These involve the intermediate 10mil levels in Fig. 6.3. The question then becomes, are all of the interior nodes required for an accurate representation of the function, and if not, how do we choose which ones to keep? The answer to this is given by the sparse grid algorithm.

The sparse grid algorithm of [43] relies on a refinement of the interval of interest through successive halving of the previous interval, and then using the 'hierarchical' basis system of Fig. 9.1 as the interpolants. This system comprises, of course, our famous one-dimensional, first-order spline tent functions:

$$\phi_{l,j} = \begin{cases} 1 - |x/h_l - j_l|, & x \in [(j_l - 1)h_l,\ (j_l + 1)h_l] \cap |0, 1|; \\ 0, & \text{otherwise}, \end{cases} , \tag{9.1}$$

where h_l is the length of an interval in the lth level, and $j_l = 0, \ldots, 2^l$ determines the position of a node. It is assumed in [43] that the grid is defined on the unit cube, which is the reason for the appearance of the interval, [0, 1]. For our problem, we are only interested in levels 0 and 1 of the hierarchy, because we only use at most two intervals for each dimension (variable) of the problem, as shown in Fig. 6.3.

Multidimensional functions are obtained by taking products of the one-dimensional splines:

$$\phi_{\mathbf{l},\mathbf{j}} = \Pi_{t=1}^{d} \phi_{l_t, j_t}(x_t) \ , \tag{9.2}$$

where $\mathbf{l} = (l_1, \ldots, l_d)$ denotes the number of intervals at the lth level in each dimension. Each entry is an integer. Similarly, $\mathbf{j} = (j_1, \ldots, j_d)$, with $j_t = 0, \ldots, 2^{l_t}$, denotes the nodal ordering at the lth level in each dimension. Associated with the multidimensional functions is the index set

$$\mathbf{B_l} = \begin{cases} j_t = 1, \ldots, 2^{l_t} - 1,\ j_t \text{ odd}, & t = 1, \ldots, d, \text{if } l_t > 0, \\ j_t = 0, 1, & t = 1, \ldots, d, \text{if } l_t = 0 \ . \end{cases} \tag{9.3}$$

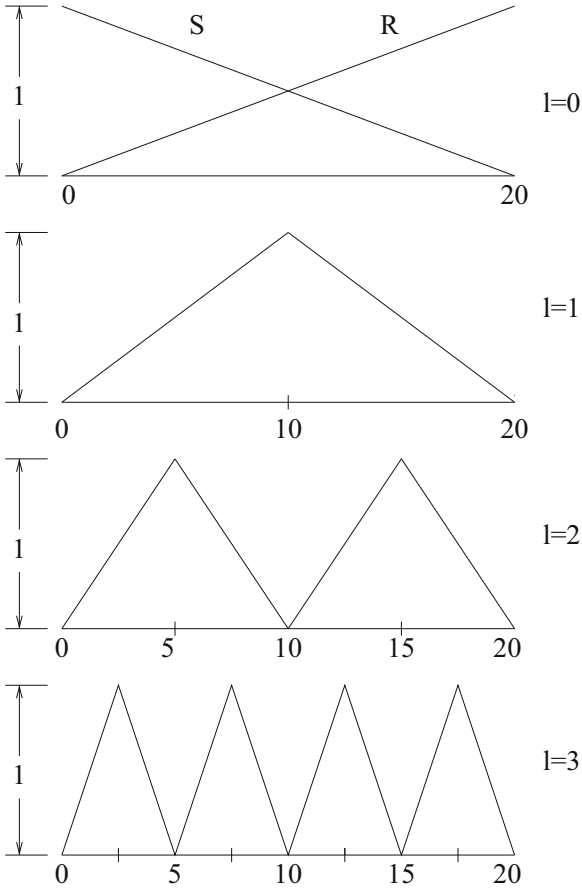

Fig. 9.1 Illustrating the one-dimensional hierarchical basis system for the function space, V_3. Each level has 2^l intervals and $2^l + 1$ nodes. Note that in our current problem, we are working in V_1, so that we are only interested in $l = 0,\ 1$. These are the usual tent functions with which we are well familiar from **VIC-3D®**. 'R' and 'S' denote 'ramp' and 'slide', respectively. The numbers along the abscissa refer to the values of the test depths of Fig. 6.3. Multidimensional functions are obtained by taking products of these functions

We define two norms for discrete multi-index vectors: $|\mathbf{l}|_\infty = \max_{1\le t\le d} l_t$, and $|\mathbf{l}|_1 = \sum_{t=1}^{d} l_t$. Using these norms, together with (9.3), we can define a full-grid function as an expansion in terms of the basis system (9.2),

$$f(\mathbf{x}) = \sum_{|\mathbf{l}|_\infty \le n} \sum_{\mathbf{j}\in\mathbf{B}_\mathbf{l}} \alpha_{\mathbf{l},\mathbf{j}} \phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}) \ , \tag{9.4}$$

and a sparse-grid function similarly,

$$f(\mathbf{x}) = \sum_{|\mathbf{l}|_1 \le n} \sum_{\mathbf{j}\in\mathbf{B}_\mathbf{l}} \alpha_{\mathbf{l},\mathbf{j}} \phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}) \ . \tag{9.5}$$

Before we discuss either expansion in detail, let's take a look at the distinction between them, which is tied up with the definition of the norms above. In the complex-flaw case that we are considering, we have $d = 4$ and $n = 1$. Hence, the

Table 9.1 Four-dimensional vectors satisfying $|\mathbf{l}|_\infty \leq 1$. The last column gives the number of nodes associated with each vector

0	0	0	0	16
1	0	0	0	8
0	1	0	0	8
0	0	1	0	8
0	0	0	1	8
1	1	0	0	4
1	0	1	0	4
1	0	0	1	4
0	1	0	1	4
0	0	1	1	4
0	1	1	0	4
1	1	0	1	2
1	0	1	1	2
0	1	1	1	2
1	1	1	0	2
1	1	1	1	1
Total				81

Table 9.2 Four-dimensional vectors satisfying $|\mathbf{l}|_1 \leq 1$. The last column gives the number of nodes associated with each vector

0	0	0	0	16
1	0	0	0	8
0	1	0	0	8
0	0	1	0	8
0	0	0	1	8
Total				48

condition $|\mathbf{l}|_\infty \leq 1$ is satisfied 16 ways, as shown in Table 9.1. The resulting total number of nodes agrees with what we knew before for the Cartesian product of three nodes in each of four variables, $3^4 = 81$. Contrast this with the vector condition for sparse grids shown in Table 9.2. It is clear that the sparse grid algorithm gives a significant reduction in the number of nodes in the interpolation grid. The difference is even more striking in the case of the five-dimensional complex flaw of Test Problem No. 5. In that case the full grid had 243 nodes, whereas the sparse grid has only $2^5 + 5 \times 16 = 112$, which is less than half the full-grid complement.

We'll interpret Table 9.2 geometrically, using Fig. 6.3 to motivate the development. The entries in Tables 9.1 and 9.2 are the exponents, l, that yield the number of intervals, 2^l, in each slab of Fig. 6.3. Hence, the first entry in Table 9.2 corresponds to the situation in which the midpoint, 10 mil, is missing in each of the slabs (yielding a single interval in each slab) and we are working with the Cartesian product $(0, 20) \otimes (0, 20) \otimes (0, 20) \otimes (0, 20)$, which are the 16 corner points of a four-dimensional hypercube of length 20 on a side.

The next entry in Table 9.2 indicates that we have introduced the 10-mil midpoint in the first slab, yielding a slab with two intervals. Geometrically, this corresponds to the intersection of the hyperplane, $x_1 = 10$, with the four-dimensional hyper-

cube, resulting in eight nodes at the boundaries of the hypercube. Similarly, the third through fifth entries correspond to the intersection of hyperplanes with the hypercube, resulting in eight nodes for each entry, yielding a total of 48 nodes. Each node carries a blending function that **VIC-3D®** must compute using the appropriate parameters of Fig. 6.3. For example, the blending function corresponding to the second entry in the table would have the first slab of Fig. 6.3 fixed at 10 mils depth, and the other three cycling through 0 and 20 mils, each, giving a **VIC-3D®** problem with 8 range values.

We turn our attention, now, to the hierarchical structure of the algorithm, which lies at the heart of (9.5). Using the format of Table 9.2, we expand (9.5) as follows:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{|\mathbf{l}|_1 \le 1} \sum_{\mathbf{j} \in \mathbf{B_l}} \alpha_{\mathbf{l},\mathbf{j}} \phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}) \\ &= \sum_{\mathbf{j} \in B_{0000}} \alpha_{0000,\mathbf{j}} \phi_{0000,\mathbf{j}}(\mathbf{x}) + \sum_{\mathbf{j} \in B_{1000}} \alpha_{1000,\mathbf{j}} \phi_{1000,\mathbf{j}}(\mathbf{x}) + \sum_{\mathbf{j} \in B_{0100}} \alpha_{0100,\mathbf{j}} \phi_{0100,\mathbf{j}}(\mathbf{x}) \\ &+ \sum_{\mathbf{j} \in B_{0010}} \alpha_{0010,\mathbf{j}} \phi_{0010,\mathbf{j}}(\mathbf{x}) + \sum_{\mathbf{j} \in B_{0001}} \alpha_{0001,\mathbf{j}} \phi_{0001,\mathbf{j}}(\mathbf{x}) \, . \end{aligned} \tag{9.6}$$

Because the expansion functions, $\{\phi_{\mathbf{l},\mathbf{j}}(\mathbf{x})\}$, are nonoverlapping for a given level, l, and have a unit amplitude, the expansion coefficients, $\{\alpha_{\mathbf{l},\mathbf{j}}\}$, are simply equal to the blending function associated with the node of the appropriate function at level l.

Figure 9.2 illustrates the situation in one dimension at levels 0 and 1. In this example, we have $\alpha_{0,0} = BF(0)$, $\alpha_{0,20} = BF(20)$, $\alpha_{1,10} =$ SURPLUS, where SURPLUS $= BF(10) - 1/2(BF(20) + BF(0))$. Hence, the expansion shown in Fig. 9.2 is given by

$$f(x) = BF(0)\phi_{0,0}(x) + BF(20)\phi_{0,20}(x) + \text{SURPLUS}\phi_{1,10}(x) \, , \tag{9.7}$$

where $\phi_{0,0}(x)$ is the slide function, $S(x)$, and $\phi_{0,20}(x)$ is the ramp function, $R(x)$, in Fig. 9.2.

It is clear that the name 'SURPLUS' (called *hierarchical surplus* in [43]) denotes the excess in function value that the higher-order levels are supposed to accommodate. It's evaluation at level l requires only one additional blending function to be computed, while using two previously computed at level $l-1$. This is an advantage of the hierarchical structure of the algorithm. If SURPLUS=0, then the interpolator would treat this function as being linear, instead of piecewise linear, in this dimension. Thus, the expansion, (9.6), is reminiscent of a Taylor series, in which the terms corresponding to higher levels of the hierarchy correspond to higher-order polynomial terms in the Taylor series.

The full four-dimensional expansion of (9.6) is given next. The first term is:

$$BF(0,0,0,0)S(x_1)S(x_2)S(x_3)S(x_4) + BF(20,0,0,0)R(x_1)S(x_2)S(x_3)S(x_4) +$$

$$BF(0,20,0,0)S(x_1)R(x_2)S(x_3)S(x_4) + BF(20,20,0,0)R(x_1)R(x_2)S(x_3)S(x_4) +$$

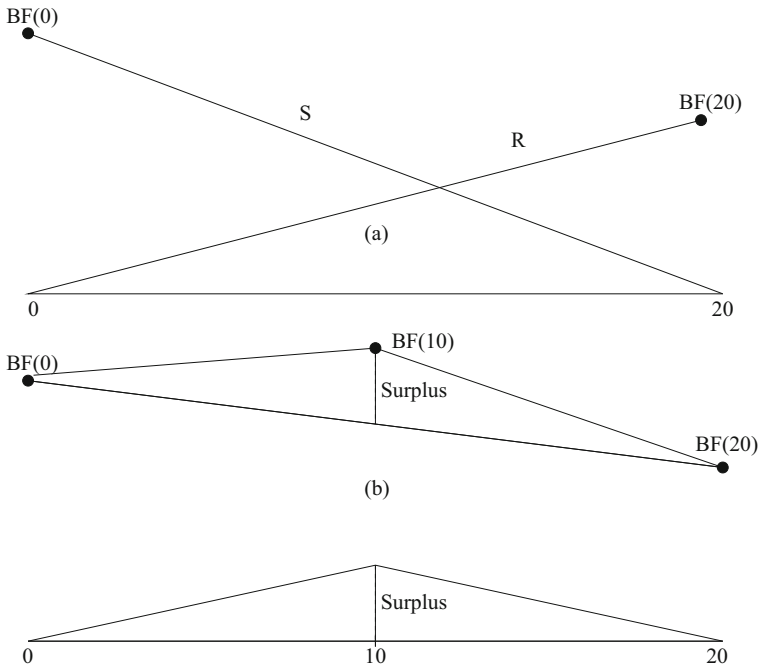

Fig. 9.2 Illustrating a one-dimensional example of the hierarchical expansion at levels 0 (**a**) and 1 (**b**). The blending functions, $BF()$, are labeled at the nodes of the one-dimensional simplex. The 'Surplus' value defines the magnitude of the level-1 spline

$$
\begin{aligned}
&BF(0,0,20,0)S(x_1)S(x_2)R(x_3)S(x_4) + BF(20,0,20,0)R(x_1)S(x_2)R(x_3)S(x_4) + \\
&BF(0,20,20,0)S(x_1)R(x_2)R(x_3)S(x_4) + BF(20,20,20,0)R(x_1)R(x_2)R(x_3)S(x_4) + \\
&BF(0,0,0,20)S(x_1)S(x_2)S(x_3)R(x_4) + BF(20,0,0,20)R(x_1)S(x_2)S(x_3)R(x_4) + \\
&BF(0,20,0,20)S(x_1)R(x_2)S(x_3)R(x_4) + BF(20,20,0,20)R(x_1)R(x_2)S(x_3)R(x_4) + \\
&BF(0,0,20,20)S(x_1)S(x_2)R(x_3)R(x_4) + BF(20,0,20,20)R(x_1)S(x_2)R(x_3)R(x_4) + \\
&BF(0,20,20,20)S(x_1)R(x_2)R(x_3)R(x_4) + BF(20,20,20,20)R(x_1)R(x_2)R(x_3)R(x_4)\ ;
\end{aligned}
\tag{9.8}
$$

the second:

$$
\begin{aligned}
&\mathrm{SURPLUS}(10,0,0,0)\phi_{1,10}(x_1)S(x_2)S(x_3)S(x_4) \\
&\quad +\mathrm{SURPLUS}(10,20,0,0)\phi_{1,10}(x_1)R(x_2)S(x_3)S(x_4) + \\
&\mathrm{SURPLUS}(10,0,20,0)\phi_{1,10}(x_1)S(x_2)R(x_3)S(x_4) \\
&\quad +\mathrm{SURPLUS}(10,20,20,0)\phi_{1,10}(x_1)R(x_2)R(x_3)S(x_4) +
\end{aligned}
$$

$$\begin{aligned}
&\text{SURPLUS}(10,0,0,20)\phi_{1,10}(x_1)S(x_2)S(x_3)R(x_4)\\
&\quad+\text{SURPLUS}(10,20,0,20)\phi_{1,10}(x_1)R(x_2)S(x_3)R(x_4)+\\
&\text{SURPLUS}(10,0,20,20)\phi_{1,10}(x_1)S(x_2)R(x_3)R(x_4)\\
&\quad+\text{SURPLUS}(10,20,20,20)\phi_{1,10}(x_1)R(x_2)R(x_3)R(x_4)\ ;
\end{aligned} \tag{9.9}$$

the third:

$$\begin{aligned}
&\text{SURPLUS}(0,10,0,0)S(x_1)\phi_{1,10}(x_2)S(x_3)S(x_4)\\
&\quad+\text{SURPLUS}(20,10,0,0)R(x_1)\phi_{1,10}(x_2)S(x_3)S(x_4)+\\
&\text{SURPLUS}(0,10,20,0)S(x_1)\phi_{1,10}(x_2)R(x_3)S(x_4)\\
&\quad+\text{SURPLUS}(20,10,20,0)R(x_1)\phi_{1,10}(x_2)R(x_3)S(x_4)+\\
&\text{SURPLUS}(0,10,0,20)S(x_1)\phi_{1,10}(x_2)S(x_3)R(x_4)\\
&\quad+\text{SURPLUS}(20,10,0,20)R(x_1)\phi_{1,10}(x_2)S(x_3)R(x_4)+\\
&\text{SURPLUS}(0,10,20,20)S(x_1)\phi_{1,10}(x_2)R(x_3)R(x_4)\\
&\quad+\text{SURPLUS}(20,10,20,20)R(x_1)\phi_{1,10}(x_2)R(x_3)R(x_4)\ ;
\end{aligned} \tag{9.10}$$

the fourth:

$$\begin{aligned}
&\text{SURPLUS}(0,0,10,0)S(x_1)S(x_2)\phi_{1,10}(x_3)S(x_4)\\
&\quad+\text{SURPLUS}(20,0,10,0)R(x_1)S(x_2)\phi_{1,10}(x_3)S(x_4)+\\
&\text{SURPLUS}(0,20,10,0)S(x_1)R(x_2)\phi_{1,10}(x_3)S(x_4)\\
&\quad+\text{SURPLUS}(20,20,10,0)R(x_1)R(x_2)\phi_{1,10}(x_3)S(x_4)+\\
&\text{SURPLUS}(0,0,10,20)S(x_1)S(x_2)\phi_{1,10}(x_3)R(x_4)\\
&\quad+\text{SURPLUS}(20,0,10,20)R(x_1)S(x_2)\phi_{1,10}(x_3)R(x_4)+\\
&\text{SURPLUS}(0,20,10,20)S(x_1)R(x_2)\phi_{1,10}(x_3)R(x_4)\\
&\quad+\text{SURPLUS}(20,20,10,20)R(x_1)R(x_2)\phi_{1,10}(x_3)R(x_4)\ ;
\end{aligned} \tag{9.11}$$

and the fifth:

$$\begin{aligned}
&\text{SURPLUS}(0,0,0,10)S(x_1)S(x_2)S(x_3)\phi_{1,10}(x_4)\\
&\quad+\text{SURPLUS}(20,0,0,10)R(x_1)S(x_2)S(x_3)\phi_{1,10}(x_4)+
\end{aligned}$$

$$\begin{aligned}
&\text{SURPLUS}(0, 20, 0, 10)S(x_1)R(x_2)S(x_3)\phi_{1,10}(x_4)\\
&\quad +\text{SURPLUS}(20, 20, 0, 10)R(x_1)R(x_2)S(x_3)\phi_{1,10}(x_4) +\\
&\text{SURPLUS}(0, 0, 20, 10)S(x_1)S(x_2)R(x_3)\phi_{1,10}(x_4)\\
&\quad +\text{SURPLUS}(20, 0, 20, 10)R(x_1)S(x_2)R(x_3)\phi_{1,10}(x_4) +\\
&\text{SURPLUS}(0, 20, 20, 10)S(x_1)R(x_2)R(x_3)\phi_{1,10}(x_4)\\
&\quad +\text{SURPLUS}(20, 20, 20, 10)R(x_1)R(x_2)R(x_3)\phi_{1,10}(x_4)\ .
\end{aligned} \tag{9.12}$$

The arguments of the BF and SURPLUS functions are the depth parameters of the corresponding slabs in Fig. 6.3. Thus, $BF(0, 20, 20, 20)$ is the blending function computed by **VIC-3D®** when slab 1 has a depth of 0, and the other three slabs are at a full depth of 20. In (9.9)–(9.12), we have

$$\begin{aligned}
\text{SURPLUS}(10, a, b, c) &= BF(10, a, b, c) - 1/2(BF(20, a, b, c) + BF(0, a, b, c))\\
\text{SURPLUS}(a, 10, b, c) &= BF(a, 10, b, c) - 1/2(BF(a, 20, b, c) + BF(a, 0, b, c))\\
\text{SURPLUS}(a, b, 10, c) &= BF(a, b, 10, c) - 1/2(BF(a, b, 20, c) + BF(a, b, 0, c))\\
\text{SURPLUS}(a, b, c, 10) &= BF(a, b, c, 10) - 1/2(BF(a, b, c, 20) + BF(a, b, c, 0))
\end{aligned} \tag{9.13}$$

9.3 Clenshaw-Curtis Grids

The Clenshaw-Curtis family of grids [57, 58] are, for the most part, superior to others of the genre that we have just described, in the sense that the number of points in C-C grids increases more slowly with the level of refinement, while retaining the same asymptotic error decay rate. Our presentation follows [58].

The points, x_j^i, of the C-C grid in one dimension are defined as

$$\begin{aligned}
m_i &= \begin{cases} 1, & \text{if } i = 1,\\ 2^{i-1} + 1, & \text{if } i > 1, \end{cases}\\
x_j^i &= \begin{cases} (j-1)/(m_i - 1) & \text{for } j = 1, \ldots, m_i \text{ if } m_i > 1,\\ 0.5 & \text{for } j = 1 \text{ if } m_i = 1. \end{cases}
\end{aligned} \tag{9.14}$$

The index, i, is used to indicate a level of refinement of the grid in the appropriate dimension. It is clear from (9.14) that the set of points, X^i, generated at the ith level, is a subset of X^{i+1}: $X^i \subset X^{i+1}$. Furthermore, this implies that the multidimensional sparse grid generated by the tensor product of the one-dimensional grids

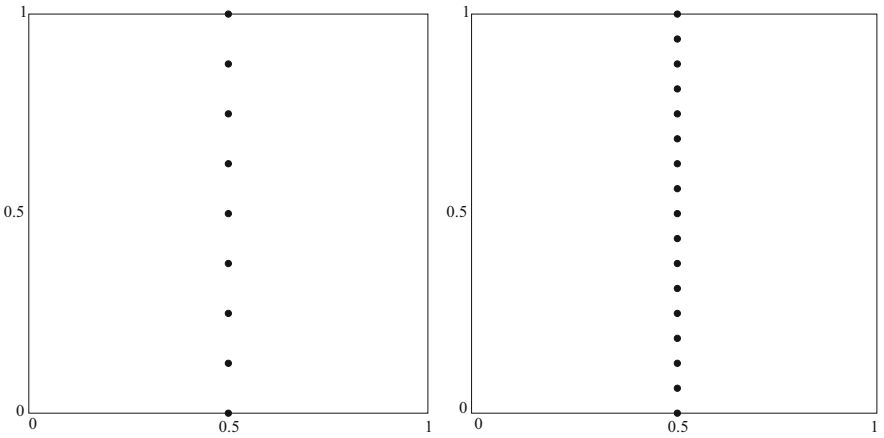

Fig. 9.3 Example of two point sets of the C-C grid. Left:$X^1 \otimes X^4$. Right:$X^1 \otimes X^5$. Note that $X^1 \otimes X^4$ is a subset of $X^1 \otimes X^5$

$$H_{q,d} = \cup_{q-d+1 \le |\mathbf{l}|_1 \le q} (X^{i_1} \otimes \cdots \otimes X^{i_d}) \,, \tag{9.15}$$

also satisfies the inclusion property: $H_{q,d} \subset H_{q+1,d}$. The index, q, is associated with the level of the dth dimension. The importance of this last relationship is that it implies that those blending functions that have already been computed on the set $H_{q,d}$ can be used in $H_{q+1,d}$, with only the surplus blending-functions being needed to be computed in $H_{q+1,d}$. When these surpluses are smaller than a threshold, we can stop refining the grid. Figure 9.3 illustrates two point sets of the C-C grid, $X^1 \otimes X^4$ and $X^1 \otimes X^5$. The former is a subset of the latter, which is at the next level of refinement.

An example of (9.15) for the case $q = 6,\ d = 4$ is given here:

$$\begin{aligned} H_{6,2} &= \cup_{5 \le i_1 + i_2 \le 6} X^{i_1} \otimes X^{i_2} \\ &= X^1 \otimes X^4 \cup X^1 \otimes X^5 \cup X^2 \otimes X^3 \cup X^2 \otimes X^4 \cup X^3 \otimes X^2 \cup X^3 \otimes X^3 \\ &\cup X^4 \otimes X^1 \cup X^4 \otimes X^2 \cup X^5 \otimes X^0 \cup X^5 \otimes X^1 \,, \end{aligned} \tag{9.16}$$

with $X^0 = \emptyset$. The C-C grid corresponding to (9.16) is shown in Fig. 9.4.

With $X^i_\Delta = X^i \backslash X^{i-1}$, namely those points in X^i that are not in X^{i-1} (the 'excess' points), (9.15) can be expanded as

$$H_{q,d} = \cup_{|\mathbf{i}|_1 \le q} (X^{i_1}_\Delta \otimes \cdots \otimes X^{i_d}_\Delta) = H_{q-1,d} \cup \Delta H_{q,d} \,, \tag{9.17}$$

where

$$\Delta H_{q,d} = \cup_{|\mathbf{i}|_1 = q} (X^{i_1}_\Delta \otimes \cdots \otimes X^{i_d}_\Delta) \,, \tag{9.18}$$

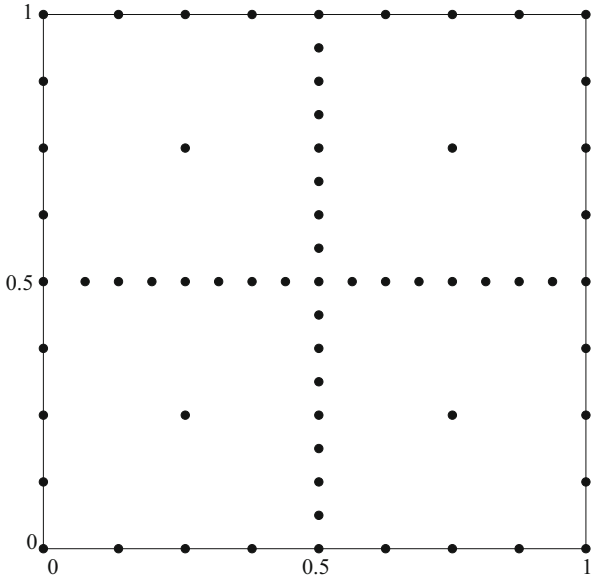

Fig. 9.4 Sparse grid $H_{6,2}^{CC}$ at level 4. There are a total of 65 points

Table 9.3 Number of grid points at level $n = q - d$ for Clenshaw-Curtis grids

n	$d = 2$	$d = 4$	$d = 8$	$d = 16$
0	1	1	1	1
1	5	9	17	33
2	13	41	145	545

and $H_{d-1,d} = \emptyset$. This is more convenient for expansion of the interpolant using successive refinements of the grid with increasing parameter, q.

Table 9.3 shows the number of grid points at level $n = q - d$ for Clenshaw-Curtis grids of up to 16 dimensions. We see that the increase in grid points with dimension is rather slow, which is a significant advantage of C-C grids. If we argue, as we did earlier, that the complex flaw of Fig. 6.3 requires only three grid points per dimension, then it follows from Table 9.3 that we may be able to get by with a total of 33 points for 16 dimensions at level 1! This is due to the fact that the midpoint is used at the 0th level, and the two end points at the first level, and is an incredible savings. If we are conservative, and decide that we need to go to level 2, we can go up to eight dimensions and require only 145 blending functions. If we used a full grid for an eight-dimensional problem with three points per dimension, we would need a total of 6561 points! The curse of dimensionality strikes again.

Klimke, [57, 58], allows one to compute the coordinates of a C-C grid, which then allows the user to determine *a priori* what blending functions to compute for an interpolation table. Table 9.4 lists the coordinates of hierarchical C-C grid points for $d = 4,\ n = 0 : 2$. The table is arranged to show the grid points that are added

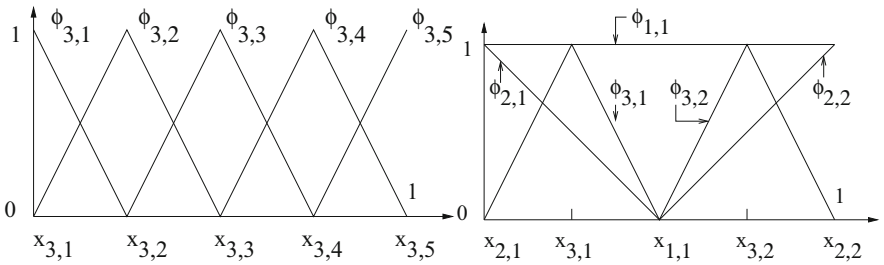

Fig. 9.5 Illustrating nodal basis functions, $\phi_{3,j}$, $x_{3,j} \in X^3$ (left) and hierarchical basis functions, $\phi_{i,j}$, with their support nodes, $x_{i,j} \in X^i_\Delta$, $i = 1, \ldots, 3$ (right) for the Clenshaw-Curtis grid (see [58])

to the interpolant at level n. The total number of points is 41, and agrees with the corresponding entry in Table 9.3.

Now that we have discussed the nature of C-C grids, it is time to address the expansion of the interpolant in function space. Figure 9.5 illustrates the typical piecewise nodal basis functions used in conventional interpolation schemes, as well as the hierarchical basis functions used with Clenshaw-Curtis grids. Note that in the C-C basis, the lowest level function in the hierarchy is simply a constant, whereas the basis system in the next level are a slide (left) and ramp (right) which do not span the entire line, but only one-half. Finally, the basis functions at level 2 are the usual disjoint tent functions that we have seen before in Fig. 9.1.

Figure 9.6 illustrates how the two sets of basis functions are used in interpolations. Our interest is in the hierarchical system shown at the bottom of the figure. The $w_{i,j}$ denote the hierarchical surpluses that are the expansion coefficients for the hierarchical interpolation formula:

$$U(x) = w_{1,1}\phi_{1,1}(x) + w_{2,1}\phi_{2,1}(x) + w_{2,2}\phi_{2,2}(x) + w_{3,1}\phi_{3,1}(x) + w_{3,2}\phi_{3,2}(x)\,. \tag{9.19}$$

They are given by: $w_{1,1} = f(0.5)$, $w_{2,1} = f(0) - f(0.5)$, $w_{2,2} = f(1) - f(0.5)$, $w_{3,1} = f(0.25) - 1/2(f(0) + f(0.5))$, and $w_{3,2} = f(0.75) - 1/2(f(1) + f(0.5))$, where $f(x)$ is shown as the dashed curve in the figure. (The surpluses are the expansion coefficients because the basis functions all have unit amplitude, and do not overlap with each other at a given level.) Note the efficiency in the hierarchical algorithm, in that function values computed at one level are reused in the next higher level.

In reality, the nodal values in Fig. 9.6 are blending functions, comprising an entire 1- or 2-D scan impedance response. Here's how we use the coordinate information in Table 9.4 to determine the computation of the associated blending function in Fig. 6.3. For $n = 0$, we set the boundary of each slab in Fig. 6.3 to be 10 mils (recall that we are scaling the physical dimensions to fit into a unit hypercube in 4-space), and use **VIC-3D®** to compute the response. This, then, is the blending function associated with the midpoint of the hypercube.

Table 9.4 Coordinates of hierarchical C-C grid points for $d = 4,\ n = 0 : 2$

$n = 0$	0.5	0.5	0.5	0.5
$n = 1$	0.0	0.5	0.5	0.5
	1.0	0.5	0.5	0.5
	0.5	0.0	0.5	0.5
	0.5	1.0	0.5	0.5
	0.5	0.5	0.0	0.5
	0.5	0.5	1.0	0.5
	0.5	0.5	0.5	0.0
	0.5	0.5	0.5	1.0
$n = 2$	0.25	0.5	0.5	0.5
	0.75	0.5	0.5	0.5
	0.0	0.0	0.5	0.5
	1.0	0.0	0.5	0.5
	0.0	1.0	0.5	0.5
	1.0	1.0	0.5	0.5
	0.5	0.25	0.5	0.5
	0.5	0.75	0.5	0.5
	0.0	0.5	0.0	0.5
	1.0	0.5	0.0	0.5
	0.0	0.5	1.0	0.5
	1.0	0.5	1.0	0.5
	0.5	0.0	0.0	0.5
	0.5	1.0	0.0	0.5
	0.5	0.0	1.0	0.5
	0.5	1.0	1.0	0.5
	0.5	0.5	0.25	0.5
	0.5	0.5	0.75	0.5
	0.0	0.5	0.5	0.0
	1.0	0.5	0.5	0.0
	0.0	0.5	0.5	1.0
	1.0	0.5	0.5	1.0
	0.5	0.0	0.5	0.0
	0.5	1.0	0.5	0.0
	0.5	0.0	0.5	1.0
	0.5	1.0	0.5	1.0
	0.5	0.5	0.0	0.0
	0.5	0.5	1.0	0.0
	0.5	0.5	0.0	1.0
	0.5	0.5	1.0	1.0
	0.5	0.5	0.5	0.25
	0.5	0.5	0.5	0.75

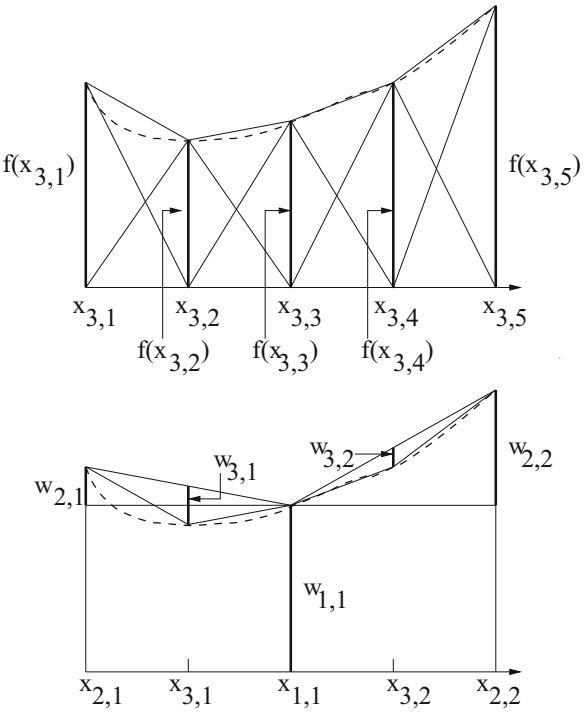

Fig. 9.6 Illustrating nodal (top) versus hierarchical (bottom) interpolation in one dimension (see [58]). The $w_{i,j}$ denote the hierarchical surpluses that are the expansion coefficients for the hierarchical interpolation formula: $U(x) = w_{1,1}\phi_{1,1}(x) + w_{2,1}\phi_{2,1}(x) + w_{2,2}\phi_{2,2}(x) + w_{3,1}\phi_{3,1}(x) + w_{3,2}\phi_{3,2}(x)$. They are given by: $w_{1,1} = f(0.5)$, $w_{2,1} = f(0) - f(0.5)$, $w_{2,2} = f(1) - f(0.5)$, $w_{3,1} = f(0.25) - 1/2(f(0) + f(0.5))$, and $w_{3,2} = f(0.75) - 1/2(f(1) + f(0.5))$, where $f(x)$ is shown as the dashed curve in the figure

The problem configuration for computing the blending function associated with the first entry at $n = 1$ of Table 9.4 would have the depth of the first slab of Fig. 6.3 equal to zero, and the other three fixed at 10 mils. For the first entry at $n = 2$, the depth of the first slab would be 5 mils, and the other three depths would be 10 mils, and so on.

9.4 The TASMANIAN Sparse Grids Module

The Toolkit for Adaptive Stochastic Modeling and Non-Intrusive ApproximatioN is a robust library for high-dimensional integration and interpolation, as well as parameter calibration. The code consists of several modules that can be used individually or conjointly. The project is sponsored by Oak Ridge National Laboratory Directed Research and Development as well as the Department of Energy Office for Advanced Scientific Computing Research (see tasmanian.ornl.org/about.html).

Sparse Grids is a family of algorithms for constructing multidimensional quadrature and interpolation rules from tensor products of one-dimensional rules. The TASMANIAN Sparse Grid code implements a number of different quadrature rules and basis functions (see [123] for details). The rules are grouped into three categories:

- **Global Grids:** suitable for globally smooth functions. Quadrature is based on a number of rules, including Clenshaw-Curtis, and interpolation is based on global Lagrange polynomials. Nodal point selection follows the same rules as quadrature. These grids are most suitable for our use, and will be discussed in more detail below.
- **Local Polynomial Grids:** suitable for non-smooth functions with locally sharp behavior. Interpolation is based on hierarchical piecewise polynomials with local support and user-specified order. These grids are suitable for local refinement.
- **Wavelet Grids:** are similar to local polynomials, except that it is assumed that the order is either 1 or 3. When coupled with local refinement, these grids often provide the same accuracy with fewer abscissas.

Lagrange Polynomial Interpolation Given nodal values, $f(x_i)$, the LP interpolator is given by

$$f(x) = \sum_{i=0}^{N} l_i(x) f(x_i) \ , \tag{9.20}$$

where

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \ldots, N \ . \tag{9.21}$$

The interpolating polynomials, $\{l_i(x)\}$, satisfy $l_i(x_j) = \delta_{ij}$. An example for $N = 3$ is given here:

$$\begin{aligned}
l_0(x) &= \frac{x - x_1}{x_0 - x_1} \frac{x - x_2}{x_0 - x_2} \frac{x - x_3}{x_0 - x_3} \\
l_1(x) &= \frac{x - x_0}{x_1 - x_0} \frac{x - x_2}{x_1 - x_2} \frac{x - x_3}{x_1 - x_3} \\
l_2(x) &= \frac{x - x_0}{x_2 - x_0} \frac{x - x_1}{x_2 - x_1} \frac{x - x_3}{x_2 - x_3} \\
l_3(x) &= \frac{x - x_0}{x_3 - x_0} \frac{x - x_1}{x_3 - x_1} \frac{x - x_2}{x_3 - x_2} \ .
\end{aligned} \tag{9.22}$$

The nodal points, or knots, are located at the extrema (maxima or minima) of Chebyshev polynomials. If $m_i > 1$ is the number of knots at the ith level of approximation in a given dimension, then the knots, over the interval $[-1, +1]$, are given by

$$x_j^i = -\cos \frac{\pi(j-1)}{m_i - 1}, \quad j = 1, \ldots, m_i \ , \tag{9.23}$$

with $x_1^i = 0$ if $m_i = 1$. In order for the knots to be nested at the next level of approximation, we choose $m_1 = 1$ and $m_i = 2^{i-1} + 1$ for $i > 1$ [15]. The Lagrange interpolator is an example of Eq. (1.3) in the TASMANIAN User Manual.

9.5 First TASMANIAN Results

In order to test the accuracy of interpolating high-dimensional models at various levels,[1] we start with a simple example in which we compare the interpolated result at various levels with original data that are computed directly with **VIC-3D®** for the complex flaw with 'coordinates' (20,0,0,0) shown in Fig. 9.7. These results are shown in Figs. 9.8 and 9.9. Additional test results are shown in Figs. 9.10, 9.11, and 9.12. It is clear from these tests that TASMANIAN works well as long as the level is chosen correctly. It seems likely that one could determine a suitable level for problems of a given dimension by using theoretical rates of convergence, but in these tests we used an empirical approach to determine such levels.

9.6 Results for 4D-Level 8

We have done a number of numerical experiments to test TASMANIAN, and learn more about the relationship between the number of dimensions in a grid and the level of the grid. For example, we found that there was reasonable convergence from 8D-Level3(593 points) to 8D-Level4(1953 points), but to use an eight-dimensional grid

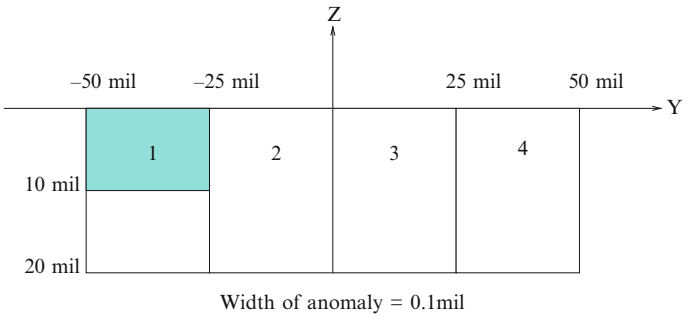

Fig. 9.7 Showing a complex flaw extending over one-half of the first block of Fig. 6.3 and vanishing elsewhere

[1] 'Level', in the context of Lagrange interpolation with the Chebyshev rule, implies the highest order polynomial that can be interpolated exactly.

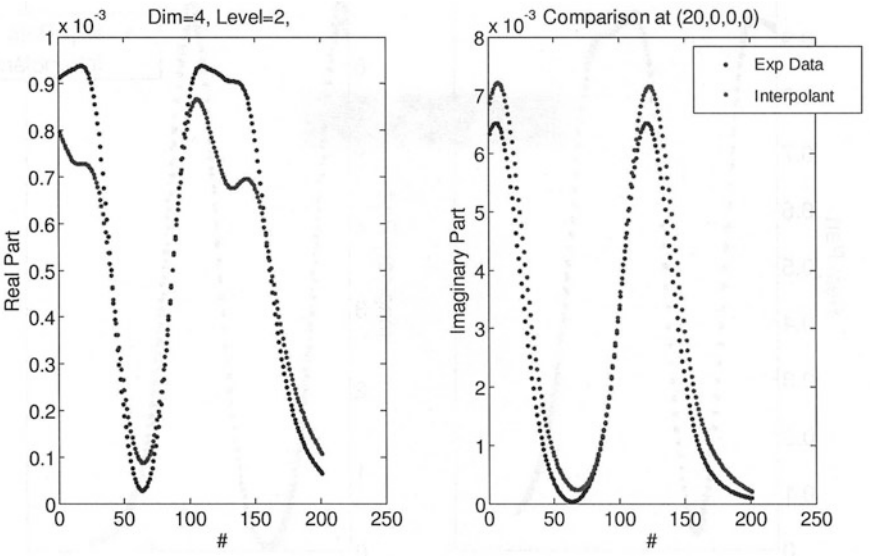

Fig. 9.8 Comparison of interpolated result with original data for the complex flaw with 'coordinates' (20,0,0,0). (See Fig. 9.7.) The four-dimensional TASMANIAN sparse grid was generated at level 2, and required 33 Chebyshev points

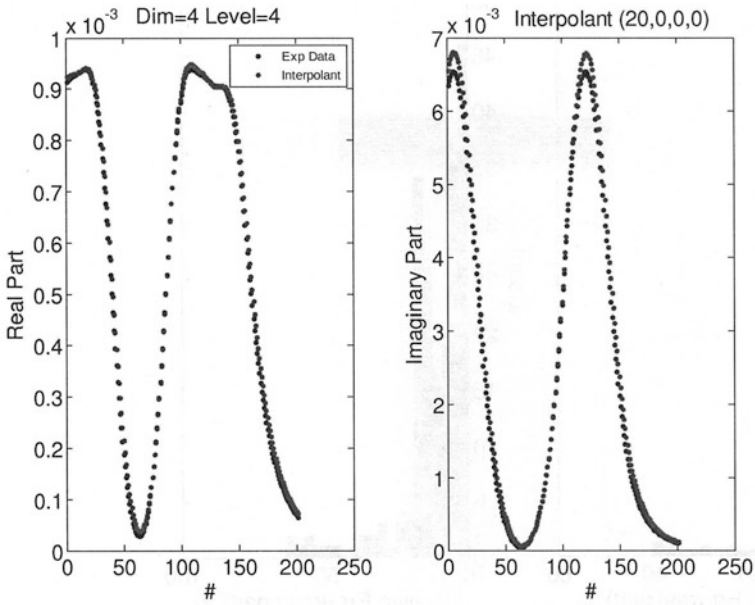

Fig. 9.9 Comparison of interpolated result with original data for the complex flaw with 'coordinates' (20,0,0,0). (See Fig. 9.7.) The four-dimensional TASMANIAN sparse grid was generated at level 4, with 145 points. Note the significant improvement over Fig. 9.8

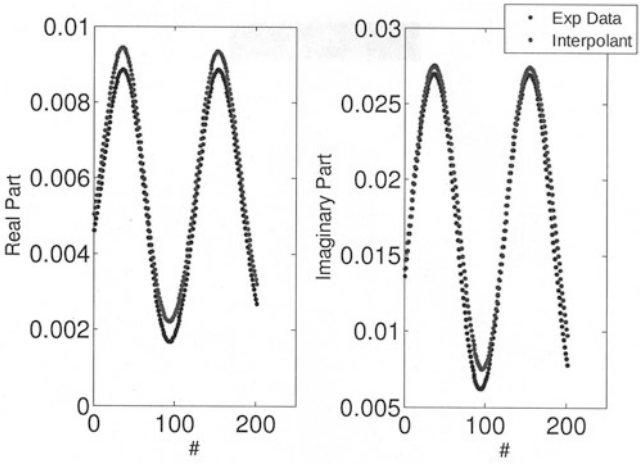

Fig. 9.10 Comparison of interpolated result with original data for the complex flaw with 'coordinates' (20,40,30,10). (See Fig. 6.4.) The four-dimensional TASMANIAN sparse grid was generated at level 2 (33 points)

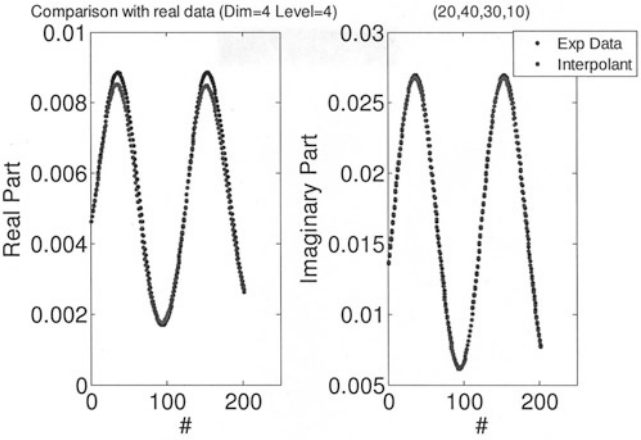

Fig. 9.11 Comparison of interpolated result with original data for the complex flaw with 'coordinates' (20,40,30,10). (See Fig. 6.4.) The four-dimensional TASMANIAN sparse grid was generated at level 4 (145 points). Note the improvement over Fig. 9.10. The relative errors are less than 7% in the real part (resistance), less than 3% in the imaginary part (reactance). The major part of the relative error occurs when the real and imaginary parts, especially the imaginary part, are both small. The relative error is calculated as the ratio of the difference of the 'experimental data' and the interpolated data to the experimental data. The absolute error is the absolute value of the difference between the experimental and interpolated data

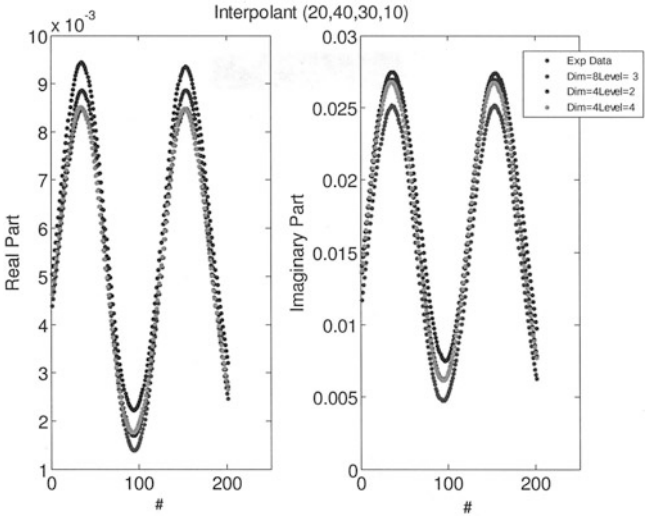

Fig. 9.12 Comparison of interpolated results with original data for the complex flaw with 'coordinates' (20,40,30,10). (See Fig. 6.4.) The coordinates of the flaw in eight dimensions are (20,20,40,40,30,30,10,10), because the length of each of the eight-dimensional slabs is one-half that of the four-dimensional slabs. The eight-dimensional TASMANIAN sparse grid was generated at level 3, with 593 Chebyshev points

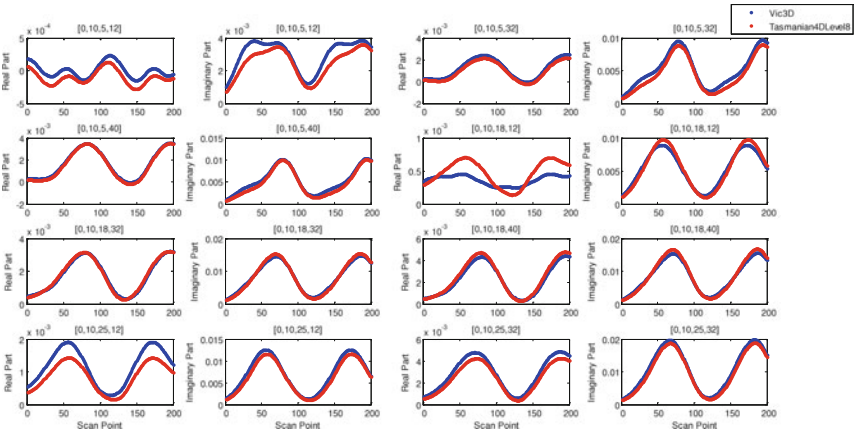

Fig. 9.13 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8, with 1857 Chebyshev points

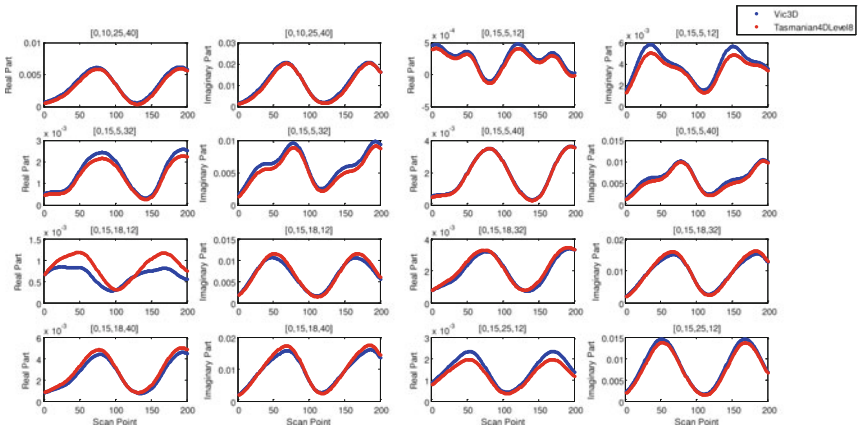

Fig. 9.14 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

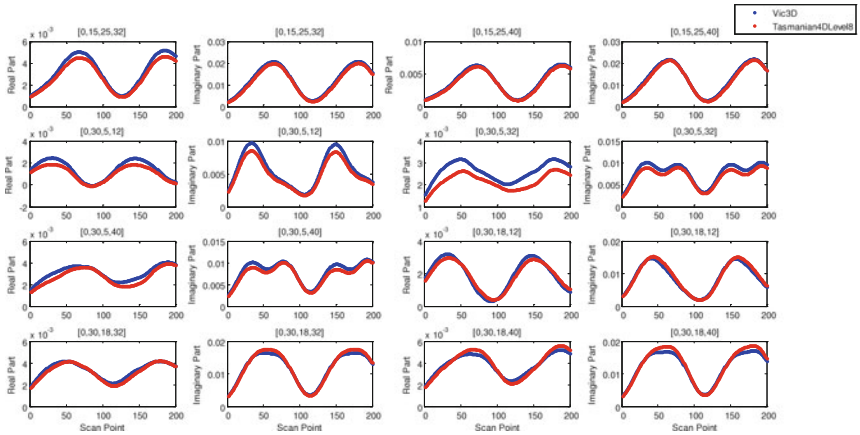

Fig. 9.15 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

accurately would require even higher levels. Since the 8D-Level4 run took 12.5 h, we decided not to go to higher dimensions, until we were absolutely forced to. In fact, it is probably wiser to use voxel-based inverse methods when the number of parameters needed to accurately model the geometry of the problem exceeds, say, 5 or 6.

We also studied the convergence of 4D-Level 6 to 4D-Level 8, and find excellent results. Hence, the remainder of our study at this point will concentrate on 4D-Level 8 situation. As a starter, we show in Figs. 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.19, 9.20, 9.21, 9.22, and 9.23 a comparison of interpolated results at 81 arbitrarily selected points with the TASMANIAN grid at level 8 with 1857 Chebyshev points.

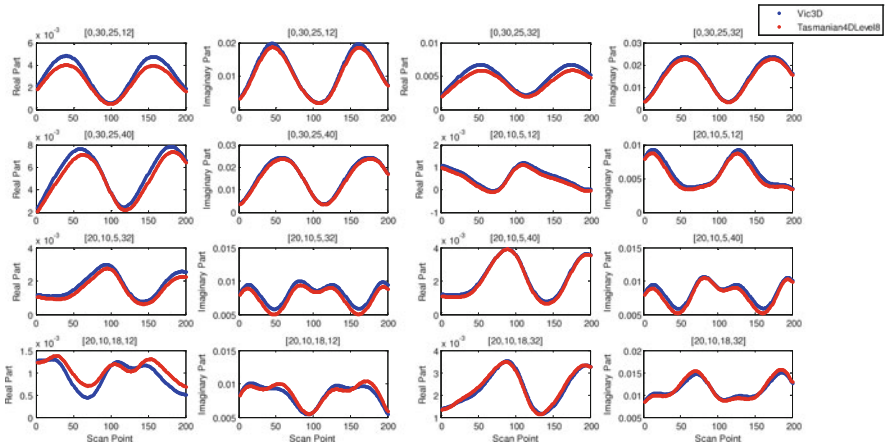

Fig. 9.16 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

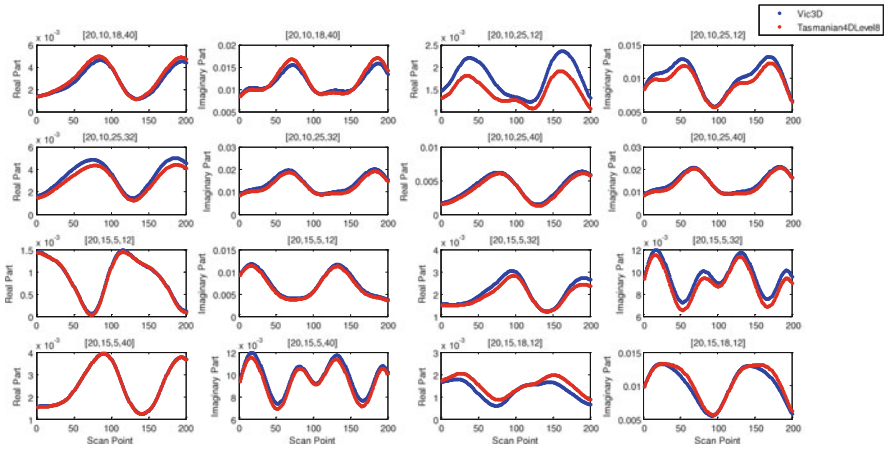

Fig. 9.17 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

Scaling may be a contributing factor to the occasionally poor fit of the real part, since the real part is much smaller than the imaginary, but it is also likely that the real and imaginary parts vary differently over this rather large range of the variables in physical space, which may contribute to the challenge of accurately interpolating each component.

Figure 9.24 shows the **VIC-3D®** model response of Fig. 6.4 when block 2 is varied in depth from 0 to 20 mils in four equal intervals, and the other blocks remain fixed at the values shown. Figure 9.25 shows a fourth-order polynomial fit to the peak values of the impedance data of Fig. 9.24, and Table 9.5 lists the coefficients

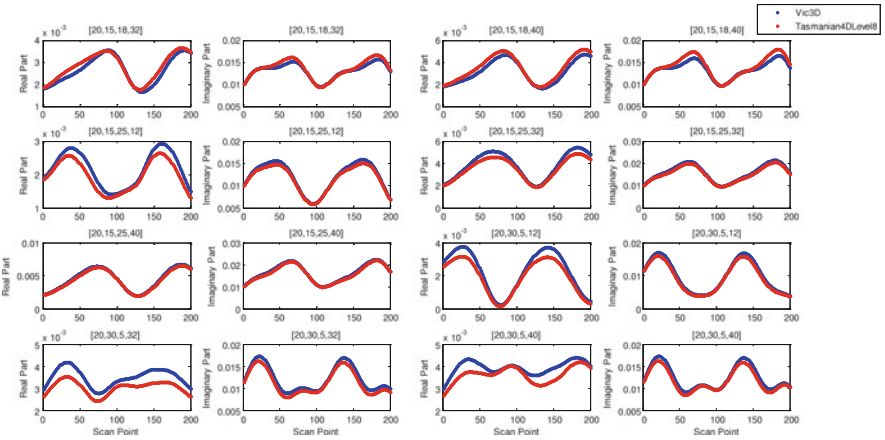

Fig. 9.18 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

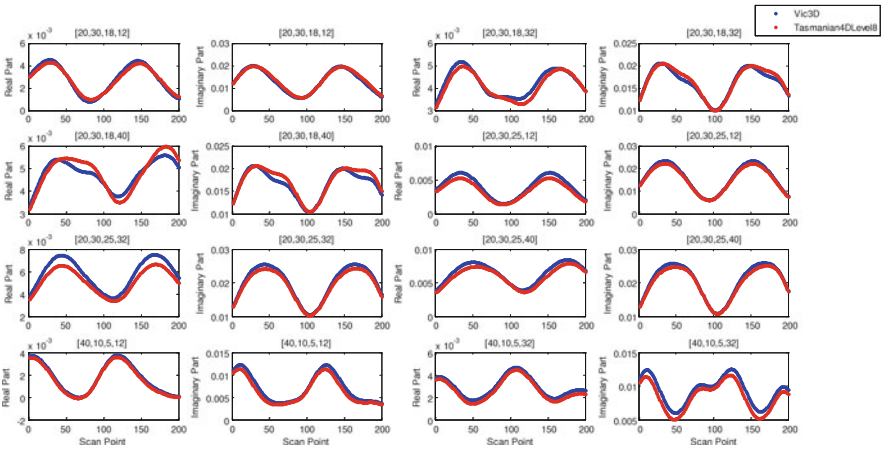

Fig. 9.19 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

of these fits. Clearly, there is a difference in the way that the real and imaginary parts vary with this particular geometric feature of the model. In both Figs. 9.24 and 9.25, the legend indicates depths of 0–40, which is due to the fact that the slabs are represented in **VIC-3D®** as canonical 'blocks,' whose reference coordinate system is at the center of the block. The blocks are defined in **VIC-3D®** by the total length of the unclipped block, so that when the clip plane passes through the origin of the block, as in this case, the 'length' parameter that **VIC-3D®** uses, and which is shown in the figures, is always twice the depth parameter shown in Fig. 6.4.

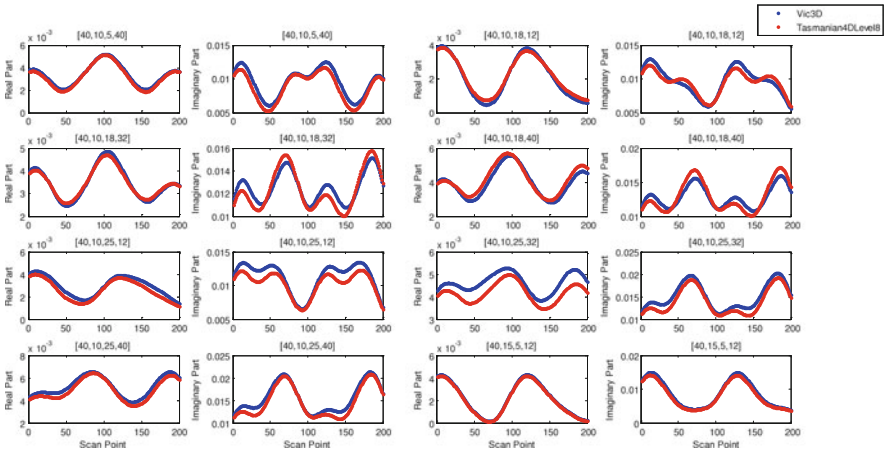

Fig. 9.20 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

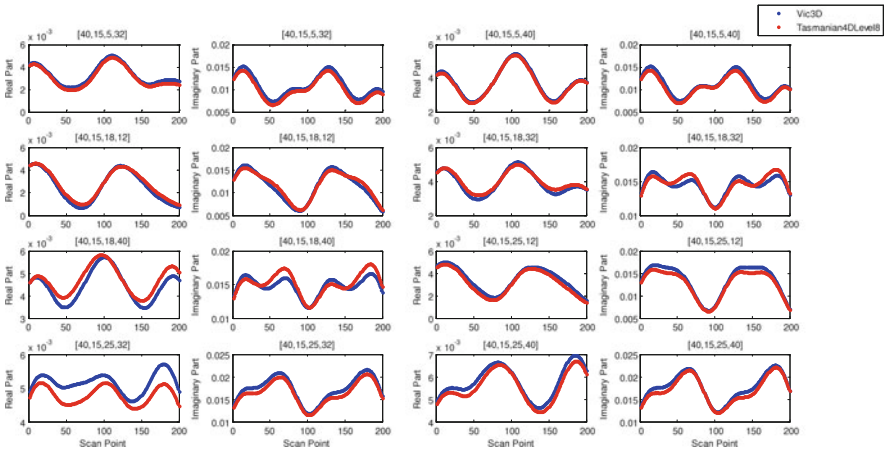

Fig. 9.21 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

9.7 The Geometry of the 4D-Level 8 Chebyshev Sparse Grid

We show in Figs. 9.26, 9.27, 9.28, 9.29, and 9.30 the distribution of points in a number of two-dimensional subspaces of the original four-dimensional grid.

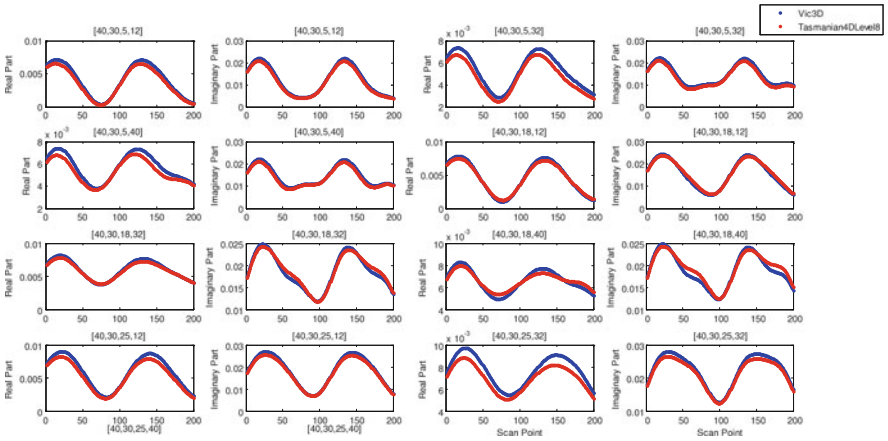

Fig. 9.22 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

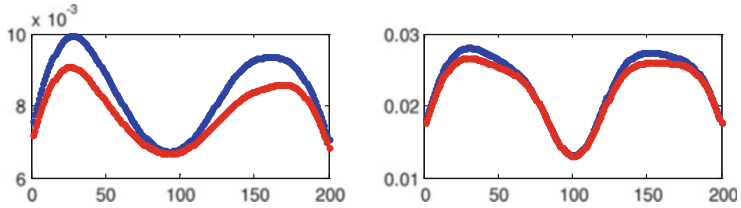

Fig. 9.23 Comparison of interpolated results at 81 arbitrarily selected points with the four-dimensional TASMANIAN grid at level 8 (continued)

Table 9.5 Coefficients of the fourth-order polynomial fits of Fig. 9.25

Order	R	X
4	−2.89708333333331e − 09	9.79583333333349e − 09
3	6.61083333333314e − 08	−1.28541666666668e − 06
2	7.25370833333338e-06	4.77454166666668e − 05
1	−6.38808333333335e − 05	−9.69083333333310e − 05
0	0.00300550000000000	0.0116510000000000

9.8 Searching the Sparse Grid for a Starting Point for Inversion

One of the applications of the sparse grid is as a surrogate for **VIC-3D®** in choosing a starting point for inversion with NLSE. To demonstrate this, we took two 'test data' sets, one with coordinates (0,10,18,32), and the other with coordinates (0,10,5,12). The first corresponds to a 'good' interpolation, as shown in the first column of the third row of Fig. 9.13, and the second to a 'not-so-good' interpolation,

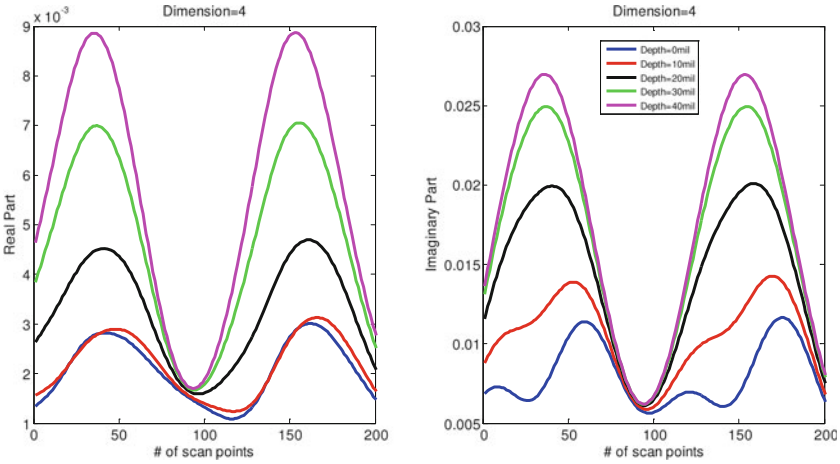

Fig. 9.24 **VIC-3D®** model response of Fig. 6.4 when block 2 is varied in depth from 0 to 20 mils in four equal intervals, and the other blocks remain fixed at the values shown

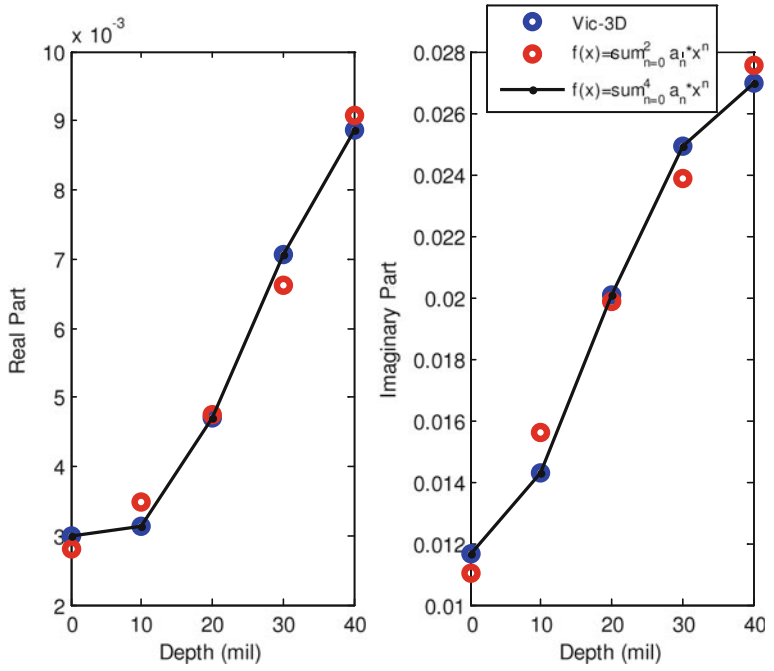

Fig. 9.25 Showing a fourth-order polynomial fit to the peak values of the impedance data of Fig. 9.24

as shown in the first column of the first row of Fig. 9.13. The algorithm for finding the starting point is to determine the 'nearest neighbor' to the test data among the

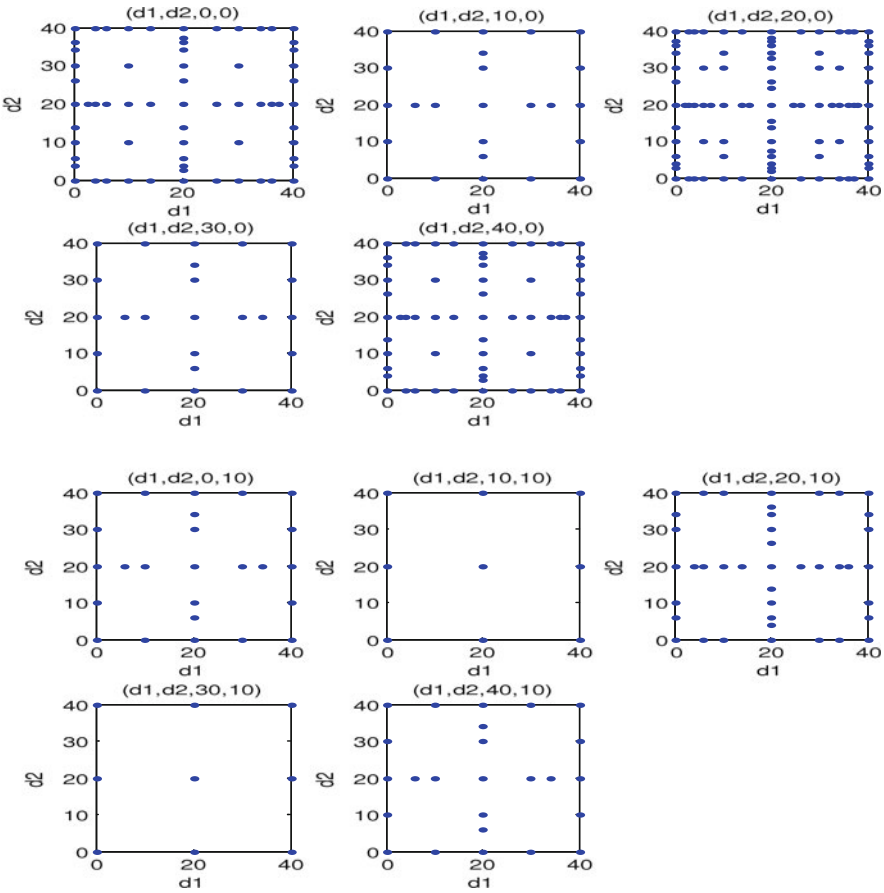

Fig. 9.26 Showing the Chebyshev distribution of points in various two-dimensional subsets of the four-dimensional grid at level 8

Table 9.6 Nearest sparse grid neighbors for two test data sets

Test set	Nearest neighbor/Φ	Second nearest neighbor/Φ
(0, 10, 18, 32)	(0, 5.8579, 20, 30)/9.42(−4)	(5.8579, 5.8579, 20, 20)/1.17(−3)
(0, 10, 5, 12)	(0, 10, 0, 10)/5.43(−4)	(0, 0, 10, 10)/1.08(−3)

1857 sparse grid points by choosing that point with the smallest norm of the residual impedance vector. The result of the experiment is shown in Table 9.6. The second nearest point is also shown for each data vector. The nearest neighbor to (0,10,18,32) lies in the subspace $(d_1, d_2, 20, 30)$ in Fig. 9.27, and the second nearest neighbor lies in the subspace $(d_1, d_2, 20, 20)$ in the same figure. As for (0,10,5,12), its nearest neighbor lies in the subspace $(d_1, d_2, 0, 10)$ shown in Fig. 9.26, and the second nearest neighbor lies in $(d_1, d_2, 10, 10)$ in the same figure.

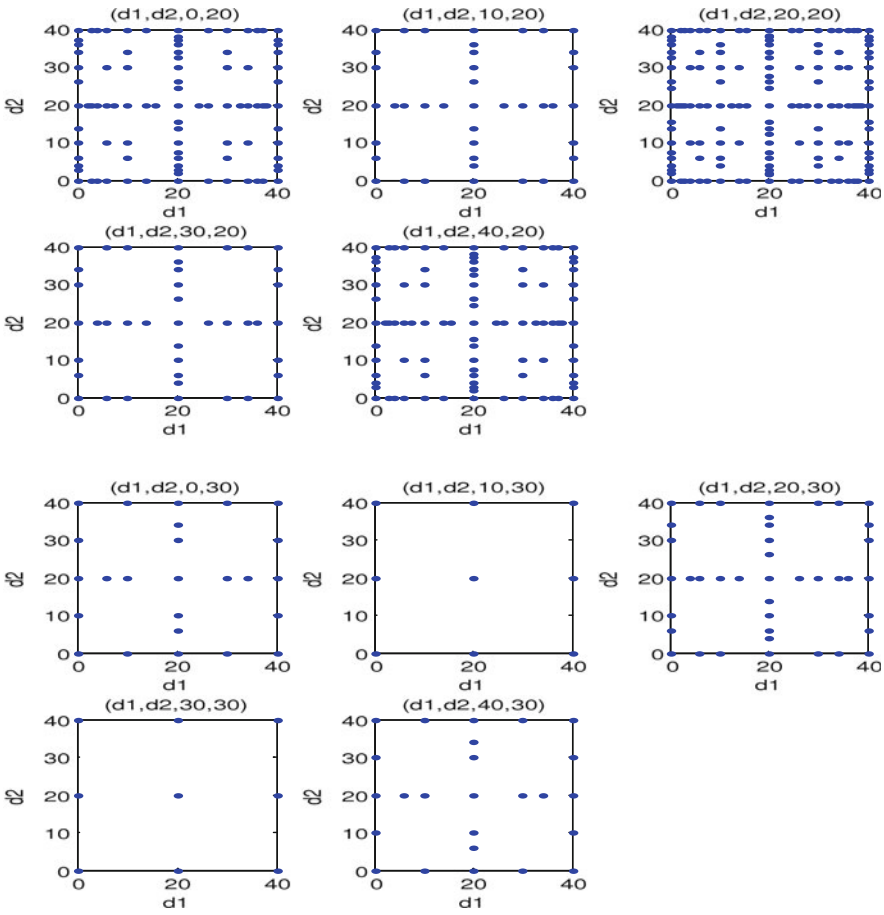

Fig. 9.27 Showing the Chebyshev distribution of points in various two-dimensional subsets of the four-dimensional grid at level 8 (continued)

In this manner we can map the 'most likely' regions of the sparse grid in which the data vector lies, and then compute a compact uniform grid for interpolation in NLSE within these regions. If the uniform grid is much smaller than the sparse grid, we would expect to get tighter estimations of confidence intervals when we perform a stochastic inversion with NLSE.

9.9 A Five-Dimensional Inverse Problem

The proof-of-the-pudding with sparse grids is their ability to simplify the solution of inverse problems that utilize internal interpolation tables as with NLSE. Our hope

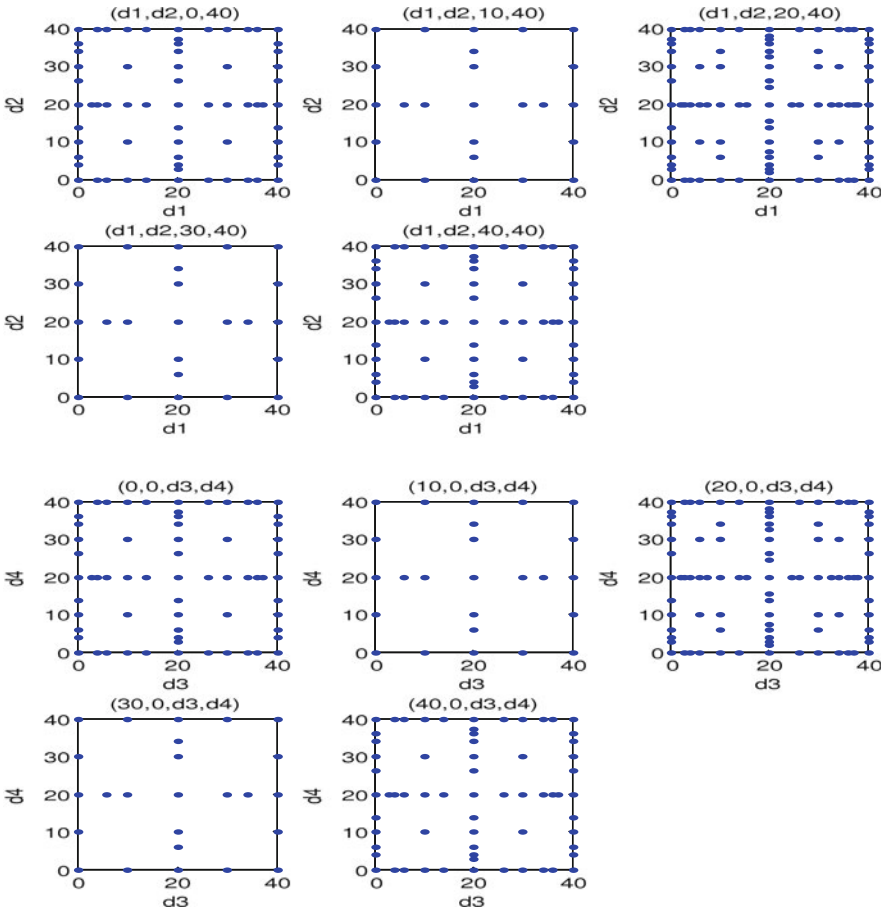

Fig. 9.28 Showing the Chebyshev distribution of points in various two-dimensional subsets of the four-dimensional grid at level 8 (continued)

is that we need to compute far fewer forward solutions with **VIC-3D®**, but then use the resulting sparse-grid solution to compute a much more refined full Cartesian grid which NLSE will then use to complete the inversion problem. This example will demonstrate the validity of this approach.

The problem consists of a split-D probe of the type shown in Fig. 9.31, and which was analyzed in [111, Section 6.6] that is scanned past a rectangular slot whose dimensions are 1 mm × 2 mm × 3 mm. The probe is vertical to the surface of the workpiece, but is rotated about its axis by 22°. The two parameters that define the orientation of the probe are the Euler angles shown in Fig. 9.32. These two angles, together with the liftoff of the probe and the length and depth of the flaw are the five parameters that are to be determined in the inverse problem. The 'true' values of the parameters are: LO = 3.5, $\Theta = 0$, $\Psi = 22$, L = 2, and D = 3.

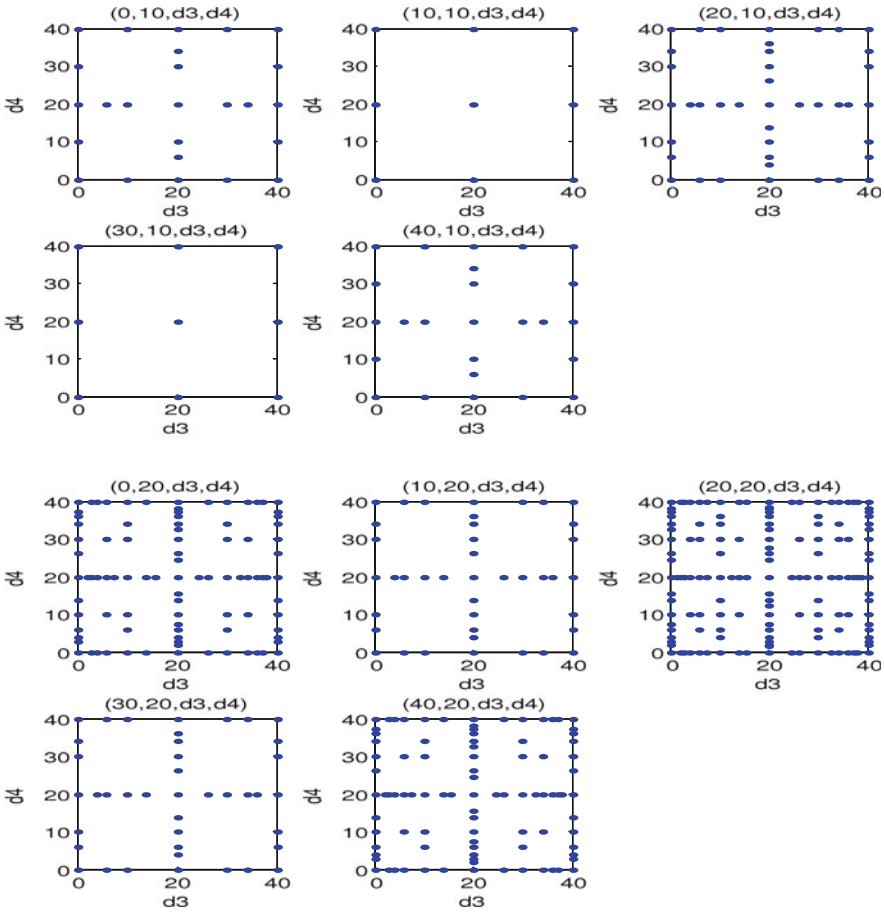

Fig. 9.29 Showing the Chebyshev distribution of points in various two-dimensional subsets of the four-dimensional grid at level 8 (continued)

The five-dimensional parameter space ranges over: LO $=$ $[0, 9.99]$, Θ $=$ $[0, 9.99]$, Ψ $=$ $[0, 90]$, L $=$ $[1.7, 2.3]$, D $=$ $[2.7, 3.3]$, which at level 4 will be covered by 311 Chebyshev points. These are the values that are presented to **VIC-3D®** to generate the sparse interpolation table. The first NLSE full Cartesian table consists of four points in each dimension, uniformly distributed over its range, which yields 1024 nodes for NLSE. The results of the first inversion test are shown in Table 9.7. The final column in the table gives the number of local minima generated by the 500 random starting points that coalesce into the global minimum. In order to improve the accuracy of the inversion of Ψ, we increase the number of NLSE nodes in this parameter to 6, and get the result shown as Test 2 in Table 9.7. Clearly, this parameter benefits from a denser nodal distribution because it covers a large range of [0,90].

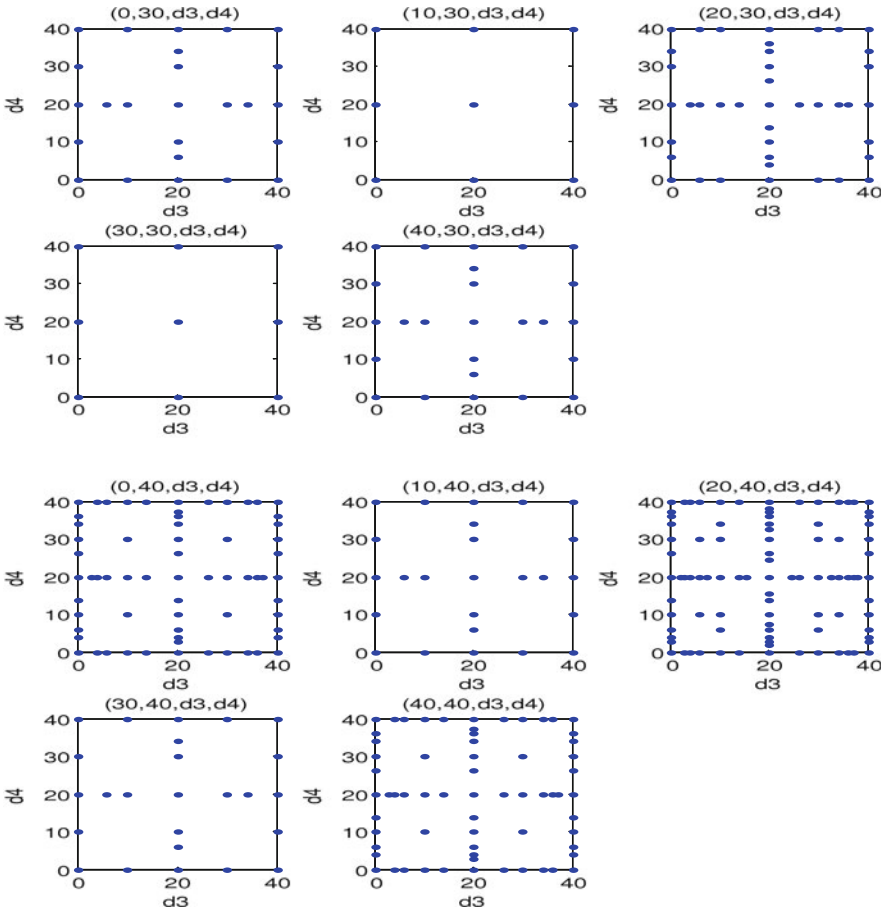

Fig. 9.30 Showing the Chebyshev distribution of points in various two-dimensional subsets of the four-dimensional grid at level 8 (continued)

The five-dimensional, level 4 inversion is quite good, but the important thing to note is the 'leverage-value' of each test in Table 9.7, namely $1024/311 = 3.3$ for the first, and 4.94 for the second. This is significant because the time to compute the 311 points completely dominates the overall inversion process, whereas the computation of function values for NLSE using the full Cartesian grid interpolation table derived from these 311 points is much faster.

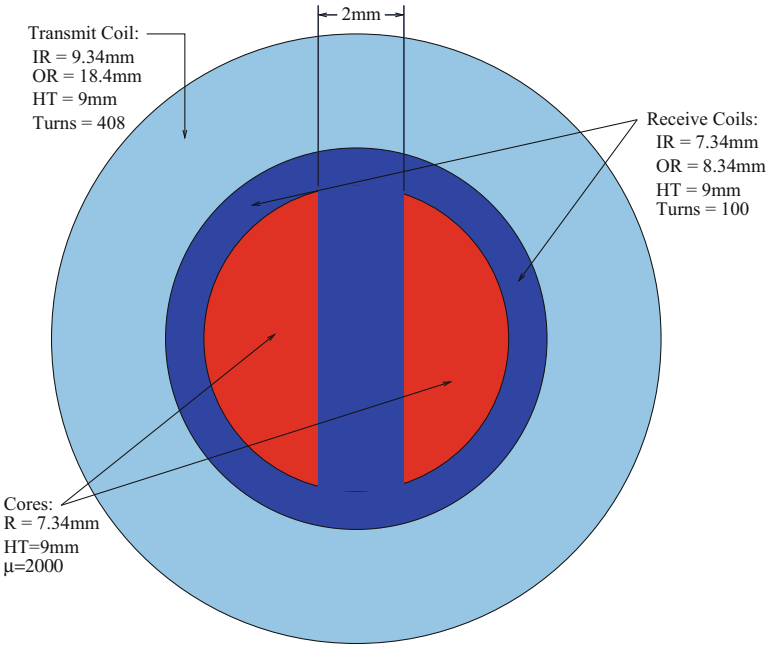

Fig. 9.31 The split-D coil configuration

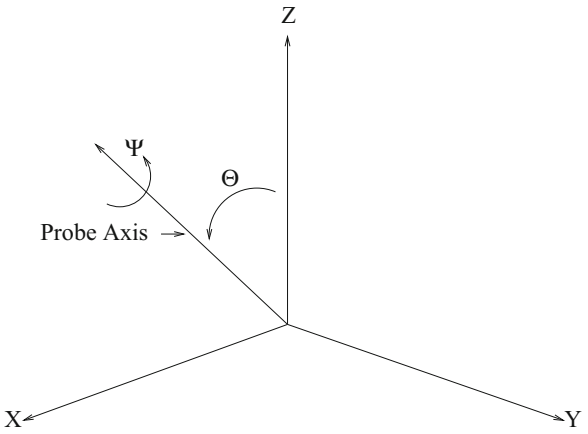

Fig. 9.32 Illustrating the two Euler angles that define the orientation of the split-D probe in the test problem. The probe axis is orthogonal to the plane of Fig. 9.31. The input model data for the inversion are $\Psi = 22°$ and $\Theta = 0°$

9.10 Noisy Data and Uncertainty Propagation

In developing the stochastic inverse model of Chap. 6, we assumed that the input data were given, and that the only stochastic feature of the problem was the random vector of unknown parameters that were to be determined. Each component of the vector was uniformly distributed over a certain range.

Table 9.7 Inversion results for the five-dimensional problem at level 4 with two different interpolation tables for NLSE

Test	Size (NLSE)	Level	No. Cal.	No. Nodes	ϕ	No. Pts.
1	$4 \times 4 \times 4 \times 4 \times 4$	4	311	1024	0.639(−5)	87
2	$4 \times 4 \times 6 \times 4 \times 4$	4	311	1536	0.578(−5)	78

Test	LO/Sensit	Θ/Sensit	Ψ/Sensit	L/Sensit	D/Sensit
1	3.5/2.24(−2)	5.46(−2)/4.17(−2)	20.4/0.47	1.99/4.25(−3)	2.98/0.28
2	3.5/2.02(−2)	0.11/3.71(−2)	21.51/0.36	1.99/3.83(−3)	2.99/0.25

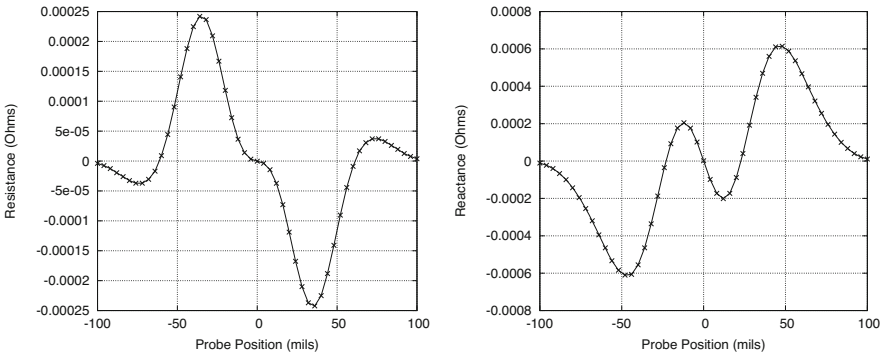

Fig. 9.33 Illustrating the noise-free input to the five-dimensional inverse problem. Left: resistance, Right: reactance

Now we want to extend the model to include the effects of Gaussian random noise that is superimposed on the noise-free input, shown in Fig. 9.33, to the five-dimensional inverse problem discussed in the preceding section. We consider two levels of noise, one with an RMS value of 1×10^{-5} and the other with an RMS value of 3×10^{-5}. Figure 9.34 illustrates a sample function of the former process superimposed on the noiseless data, and Fig. 9.35 illustrates a sample function from the second process superimposed on the noiseless data.

Our interest is in determining how uncertainty in the input data is propagated through the nonlinear least-squares filter into uncertainty in the output parameters. To accomplish this, we do a Monte Carlo analysis, in which the data of Fig. 9.33 are corrupted by ten samples from each noise source, as in Figs. 9.34 and 9.35, and then applied to NLSE using the interpolation table shown as Test 2 in Table 9.7.

The results are shown in Fig. 9.36, which depicts the relative error, defined to be the ratio of the computed value to the 'true' value, except for Θ, which uses an artificial value of 1×10^{-8} for zero. The 'Noise Level' in the figure is the ratio of the RMS value of noise to the peak value of the noiseless resistance, 0.00025, in Fig. 9.33. The ten sample points for each reconstructed parameter are shown as small dots, and the mean of the results is shown as the large red dot. The large black dot is the true value of the parameter. Note that many of the sample points are hidden behind either of the large dots.

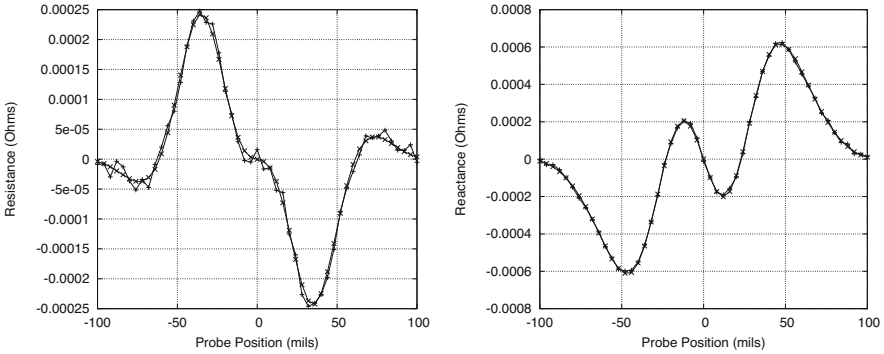

Fig. 9.34 Illustrating the noise-free input to the five-dimensional inverse problem with a sample function of noise at an RMS level of 1×10^{-5} superimposed. Left: resistance, Right: reactance

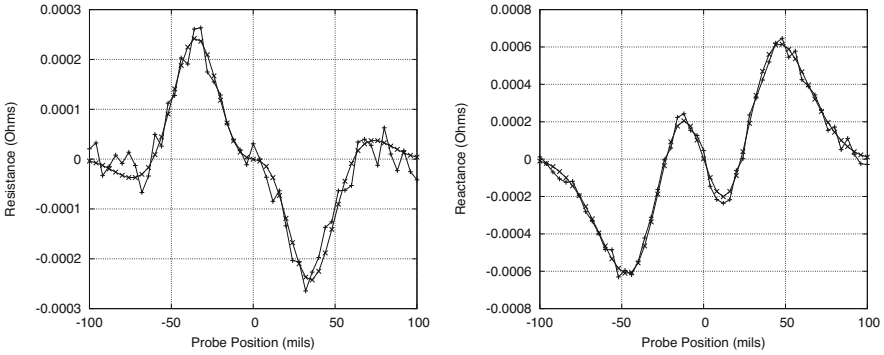

Fig. 9.35 Illustrating the noise-free input to the five-dimensional inverse problem with a sample function of noise at an RMS level of 3×10^{-5} superimposed. Left: resistance, Right: reactance

The mean values are in reasonable agreement with the true values, which suggests that the inversions are reasonable with these levels of input noise. The results for depth, D, may appear strange, in that the error in the mean value for the 1×10^{-5} noise source is greater than that for the 3×10^{-5} source, but keep in mind that the sensitivity coefficient for D in Table 9.7 is large, which, following our discussion in Chap. 6, indicates that the inversion process alone will introduce significant uncertainty in the estimated value of D. Our intuition is restored, however, when we look at the distribution of the errors over the ten samples: it is much larger for all five parameters when the noise level is 0.12. (Note that there is a small dot at ± 0.1 in D for Noise Level $= 0.12$.)

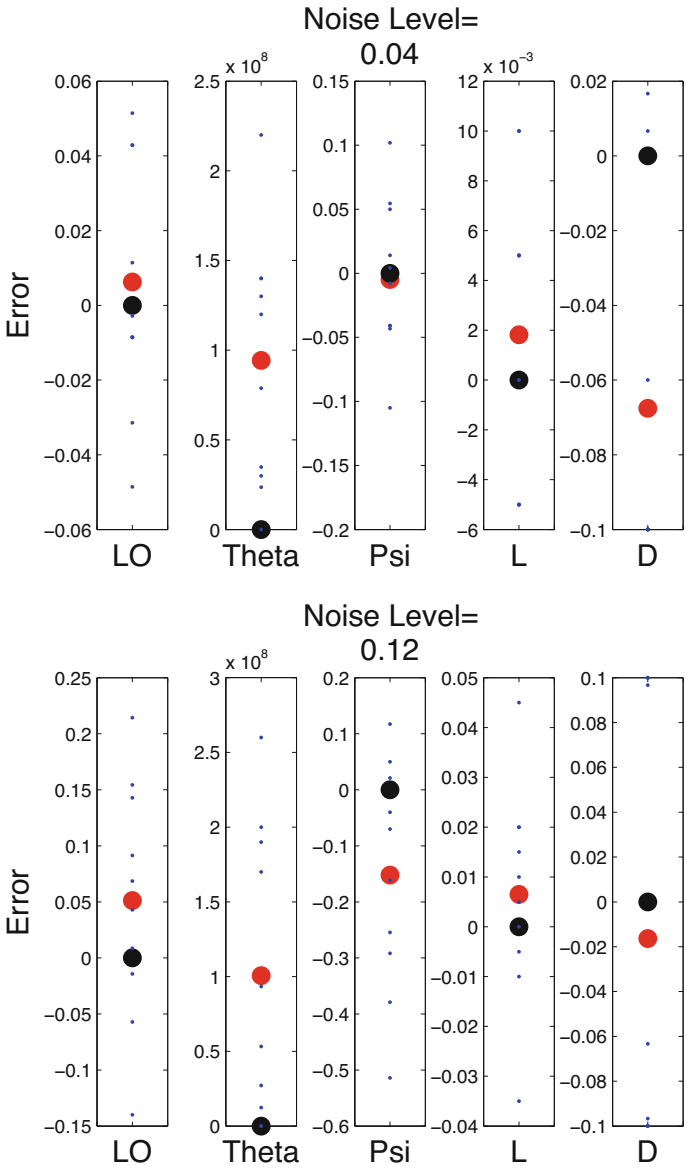

Fig. 9.36 Result of the Monte Carlo test to determine the error in inversion due to random noise at two different RMS values in the input data. Top: RMS $= 1 \times 10^5$; Bottom: RMS $= 3 \times 10^5$. The 'Noise Level' is the ratio of the RMS value of noise to the peak value of the noiseless resistance, 0.00025, in Fig. 9.33