Parallelisation of Sparse Grids for Large Scale Data Analysis

Jochen Garcke[1], Markus Hegland[2] and Ole Nielsen[2]

[1] Institut für Angewandte Mathematik,
Rheinische Friedrich-Wilhelms-Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
garcke@iam.uni-bonn.de**
[2] Mathematical Sciences Institute, Australian National University
Canberra, ACT 0200, Australia
{Markus.Hegland, Ole.Nielsen}@anu.edu.au

Abstract. Sparse Grids (SG), due to Zenger, are the basis for efficient high dimensional approximation and have recently been applied successfully to predictive modelling. They are spanned by a collection of simpler function spaces represented by regular grids. The combination technique prescribes how approximations on simple grids can be combined to approximate the high dimensional functions. It can be improved by iterative refinement.
Fitting sparse grids admits the exploitation of parallelism at various stages. The fit can be done entirely by fitting partial models on regular grids. This allows parallelism over the partial grids. In addition, each of the partial grid fits can be parallelised as well, both in the assembly phase where parallelism is done over the data and in the solution stage using traditional parallel solvers for the resulting PDEs. A simple timing model confirms that the most effective methods are obtained when both types of parallelism are used.

AMS subject classification. 62H30, 65D10, 68Q22
Keywords: predictive modelling, sparse grids, parallelism, numerical linear algebra

1 Introduction

Data mining algorithms have to address two major computational challenges. First, they have to be able to handle large and growing datasets and secondly, they need to be able to process complex data. Datasets used in data mining studies have been doubling in size every year and many are now in the terabyte range. The second challenge is sometimes referred to as the *curse of dimensionality* as the algorithmic complexity grows exponentially in the number of features

** Part of the work was supported by the German Bundesministerium für Bildung und Forschung (BMB+F) within the project 03GRM6BN.

or dimension of the data. Data mining aims to find patterns or structure in the data. Parallel processing is a major tool in addressing the large computational requirements of data mining algorithms.

One important type of pattern discovered in data mining algorithms is represented by functions among selected dataset features. In data mining, the discovery of such functions is referred to as *predictive modelling* and includes both classification and regression. Here we will consider regression but the same algorithms are used in classification as well. Different types of models are obtained from different functional classes. The classical methods of least squares uses linear and nonlinear functions with relatively few parameters. Modern nonparametric methods are characterised by large numbers of parameters and can flexibly approximate general function sets. They include artificial neural nets, Bayesian nets, classification and regression trees (CART) [1], Multivariate Adaptive Regression Splines (MARS) [2], Support Vector Machines, ANOVA splines and additive models.

All approaches are able to characterise a large class of behaviours and involve training or fitting the model to the dataset. For example, one may wish to predict the vegetation cover of a particular region based on cartographic measurements such as elevation, slope, distance to water, etc. Other examples are prediction of the likelihood of a car insurance customer making a claim, a business customer to purchase a product or a resident to commit taxation fraud.

For a given response variable y and predictor variables $x_1, \ldots, x_d$ a predictive model is described by a function

$$y = f(x_1, \ldots, x_d).$$

We will only consider the case where the function f is an element of a linear space and we'll discuss methods to compute the its representation from data in then following.

2 Sparse Grids for Predictive Modelling

Recently a technique called sparse grids [3], based on a hierarchical basis approach, has generated considerable interest as a vehicle for reducing dimensionality problems where approximations of high dimensional functions are sought. Generalised sparse grids functions $f(\boldsymbol{x})$ are approximations which can be described by additive models of the form

$$f(\boldsymbol{x}) = \sum_{\alpha} f_\alpha(\boldsymbol{x}) \tag{1}$$

where the partial functions f_α are simpler than f in some sense. Typically, the partial functions only depend on a subset of the variables or the dependence has a coarser scale as discussed below.

Sparse grids for the solution of partial differential equations, numerical integration, and approximation problems have been studied for more than a decade

by Griebel, Zenger et al. They also developed an algorithm known as the combination technique [4] prescribing how the collection of standard grids can be combined to approximate the high dimensional function. More recently, Garcke, Griebel and Thess [5, 6] demonstrated the feasibility of sparse grids in data mining by using the combination technique in predictive modelling.

Additive models of the form of equation (1) generalise linear models and thus form a core technique in nonparametric regression. They include the Multivariate Adaptive Regression Splines (MARS) [2], and the Additive Models by Hastie and Tibshirani [7, 8].

Challenges include the selection of function spaces, variable selection etc. Here we will discuss algorithms for the determination of the function f and hence the partial functions f_α given observed function values when the function spaces are given. For observed data points $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n$ and function values $y^1, \ldots, y^n$ we define the function f from some finite dimensional function space V to be the solution of a penalised least squares problem of the form

$$J(f) = \frac{1}{n} \sum_{i=1}^{n} \left(f(\boldsymbol{x}^{(i)}) - y^i \right)^2 + \beta \left\| Lf \right\|^2 \tag{2}$$

for some (differential) operator L. The solution of this problem can be viewed as a projection of a generalised thin plate spline function, see [9]. If the partial functions f_α are known to be orthogonal with respect to the corresponding norm (here the standard 2-norm), then they can be computed independently as minima of J. A slightly more general case is considered in the case of the combination technique where the projections into the spaces of the partial functions are known to commute. In this case, if g^α are the projections into the partial spaces, then the partial functions are known to be multiples of these projections with known (integer) coefficients, the combination coefficients c_α and thus

$$f = \sum_\alpha c_\alpha g_\alpha.$$

However, these approximations can break down when the projections do not commute. As a generalisation of this approach, an approximation has been suggested in [9] where the partial functions are again multiples of the projections g_α but the coefficients are this time determined by minimising the functional J. Thus one gets

$$f = \sum_\alpha \gamma_\alpha g_\alpha.$$

This also generalises the approximations obtained from the additive Schwarz method which is

$$f = \gamma \sum_\alpha g_\alpha$$

and experimental evidence shows that the performance is in many cases close to that of the multiplicative Schwarz method, which in statistics is known under

the term of backfitting [7]. The approaches above can be further improved by iterative refinement.

The interesting aspect of these problems which we will discuss here is the opportunities for parallel processing and the tradeoff between parallel processing and the performance of the solvers. We use a two level iterative solver and parallelism is exploited at both levels.

2.1 Multiresolution Analysis and Sparse Grids

The sparse grid idea stems from a hierarchical subspace splitting [3]. Consider the space $V_{\mathbf{l}}$, with $\mathbf{l} = (l_1, ..., l_d) \in \mathbf{N}_0^d$, of piecewise d-linear functions which is spanned by the usual d-linear 'hat' functions

$$\varphi_{\mathbf{l},\mathbf{j}}(\mathbf{x}) \;:=\; \prod_{t=1}^{d} \varphi_{l_t,j_t}(x_t), \qquad j_t = 0, ..., 2^{l_t}.$$

Here, the 1 D functions $\varphi_{l,j}(x)$ are

$$\varphi_{l,j}(x) = \begin{cases} 1 - |2^l \cdot x - j|, & x \in [\frac{j-1}{2^l}, \frac{j+1}{2^l}]; \\ 0, & \text{otherwise.} \end{cases}$$

The number of basis functions needed to resolve any $f \in V_l := V_{l,...,l}$ is now larger than 2^{ld}. With a resolution of just 17 points in each dimension ($l = 4$), say, a ten dimensional problem would require computation and storage of about 2×10^{12} coefficients which is more than one can expect on computers available today – the curse of dimensionality.

Now we define the difference spaces $W_{\mathbf{l}}$, with $\mathbf{e}_t$ denoting the t-th unit vector,

$$W_{\mathbf{l}} \;:=\; V_{\mathbf{l}} - \sum_{t=1}^{d} V_{\mathbf{l}-\mathbf{e}_t}.$$

These hierarchical difference spaces lead to the definition of a multilevel subspace splitting, i.e., the definition of the space V_l as a direct sum of subspaces,

$$V_l \;:=\; \sum_{l_1=0}^{l} ... \sum_{l_d=0}^{l} W_{(l_1,..l_d)} \;=\; \bigoplus_{|\mathbf{l}|_\infty \le l} W_{\mathbf{l}}, \tag{3}$$

Figure 1, showing the norms of the errors for the reconstruction of a two-dimensional function on a logarithmic scale, indicates that spaces $W_{\mathbf{l}}$ with large $|\mathbf{l}|_1$ contribute very little. In fact, it can be shown [10, 3] that the size of the error committed by removing the space $W_{\mathbf{l}}$ is proportional to $2^{-r|\mathbf{l}|_1}$, where $r = 2$ in the case of piecewise linear functions. This suggests removing all spaces where the sum of resolutions is 'large'. The choice of $|\mathbf{l}|_1 \le l$ in (3) results in the sparse grid of [3] (see figure 1 for an example in three dimensions) but the grids can be chosen more generally.

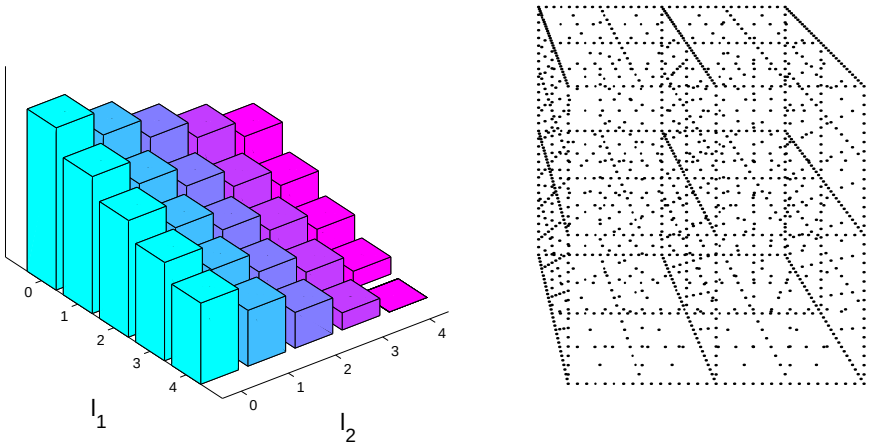

Fig. 1. Left: Norms of errors of the difference spaces $W_{\mathbf{l}}$ on a logarithmic scale. The example function is $u(x_1, x_2) = e^{-(x_1^2+x_2^2)}$ with $(x_1, x_2) \in [0, 1] \times [0, 1]$. Right: Sparse grid of refinement level $l = 5$ in three dimensions

Sparse grid spaces can also be achieved with the so-called combination technique [4] through the combination of certain spaces $V_{\mathbf{l}}$ instead of the difference spaces $W_{\mathbf{l}}$. As mentioned the grids can be chosen more generally, so we will now let the indices belong to an unspecified index set I, which leads to the generalised sparse grid space

$$S_I^d = \bigcup_{\mathbf{l} \in I} V_{\mathbf{l}}. \tag{4}$$

Each term in this sum is a tensor product of one dimensional spaces, but they are now restricted by the index set I and will generally have a much lower complexity.

See [5, 4, 9, 10] for details and further references on this subsection.

2.2 Penalised Least Squares on Sparse Grids

To compute the partial functions $f_\alpha = f_{\mathbf{l}} \in V_{\mathbf{l}}$ on each grid, the functional $J(f)$ in equation (2) has to be minimised. Substituting the representation of $f_{\mathbf{l}} = \sum_i^m \alpha_i \varphi_i$, with $\{\varphi_i\}_{i=1}^m$ a basis of $V_{\mathbf{l}}$, into (2) and differentiation with respect to α_i results in the linear system, in matrix notation,

$$(\beta C + B^T B)\alpha = B^T y. \tag{5}$$

Here C is a square $m \times m$ matrix with entries $C_{j,k} = n\,(L\varphi_j, L\varphi_k)_{L_2}$ ($(\cdot,\cdot)_{L_2}$ denotes the standard scalarproduct in L_2) and B is a rectangular $n \times m$ matrix with entries $B_{i,j} = \varphi_j(\boldsymbol{x}^{(i)}), i = 1, \ldots n, j = 1, \ldots m$. Since $n \gg m$ one stores $B^T \cdot B$ and not B, this also allows to only use one matrix structure for both C and $B^T \cdot B$. See [5] for further details.

3 Solution of the penalised least squares system

The basis functions have typically a small compact support and thus the matrices for the partial grids are sparse. However, for any given partial space many variables do occur with a large scale. Consequently, the coefficient matrix of equation (5) can have a fairly dense structure even though it has sparse diagonal blocks. The following blocking of the matrix B originates from the partial grids:

$$B = [B_1, \ldots, B_k]\,. \tag{6}$$

The system matrix then has blocks $B_i^T B_j + \beta C_{i,j}$ where the blocks $C_{i,j}$ are the corresponding blocks of the penalty terms. Often, only diagonal blocks of the penalty terms are considered.

Solving for the partial grids amounts to solving the (sparse) equations $(B_i^T B_i + \beta C_{i,i})x_i = B_i^T y$. In general, we cannot assume that the x_i are good approximations for the components of the solution to the full problem. However, they are a good starting point. Note that in terms of the linear algebra, the large system of linear equations is usually singular, as the spaces V_j have nontrivial intersections. In fact, we will assume that the collection of partial spaces V_j is *relatively large*, and, in particular, for each pair V_i and V_j the intersection $V_i \cap V_j$ is also a space V_k. Solving the partial problems amounts to a block Jacobi preconditioning which does not directly lead to a good approximation. Instead one uses the combination formula where the partial solutions are scaled with a (combination) factor c_i. The combination formula becomes necessary as the same part of the solution may be contained in several different partial functions and thus needs to be subtracted again. This only needs to be taken into account for the block Jacobi variant; in the case of block Gauss-Seidel, the residuals are subtracted from the right-hand side after every partial fitting step. This, however, comes at a cost as the parallelism over the partial grids is lost as one needs to update the right-hand-side between fitting partial grids. As we solve our systems in parallel the block Gauss-Seidel method is less attractive.

Using the combination formula in the way described may lead to good solutions, however, these are in general an approximation to the solution of the full fitting problem. The solution can be improved with iterative refinement. This provides a block Jacobi-like variant for the solution of the full problem and thus we introduce an outer iteration to correct the approximation obtained from the combination formula. This outer iteration should also deal with errors introduced through the randomness of the data, which may be amplified by the combination method. In terms of the outer iteration, the solution of the partial systems together with the combination formula forms a preconditioner. We further accelerate the convergence of this outer iteration by using conjugate gradients. Experiments show that these outer iterations converge within a couple of steps.

The partial problems themselves are solved iteratively as well using conjugate gradients. While fitting to a regular grid is a standard procedure, it may converge slowly. Thus the question arises if one can get away with only a few iteration

steps for the inner iterations as well. We have conducted initial experiments which show that the number of inner iterations can have a substantial influence on the solution and we observed saturation effects, where the outer iterations did not seem to be able to improve beyond a certain limit for low numbers of inner iterations. This behaviour did seem to be dependent on the type of problem and further investigations will be done to understand this better.

While this interaction between inner and outer iterations is an interesting topic for further investigation we will not further pursue it here but discuss the levels of parallelism which can be exploited both over the grids (corresponding to the other iterations) and within the grids (corresponding to the inner iterations).

4 Strategies for exploiting parallelism

In general, coarse grain parallelism is preferred, in particular for the application of distributed memory computing [11] as it typically has less (communication) overhead. However, in many cases the amount of available coarse grain parallelism of the algorithm is limited. If one would like to further parallelise the computations (if sufficient parallel resources are available) one would need to look at utilising finer grain parallelism as well. Such fine grain parallelism is also well suited to some shared memory and vector computations, in which case it can be competitive with the coarse grain parallelism.

The combination technique is straightforwardly parallel on a coarse grain level [12]. A second level of parallelisation on a fine grain level for each problem in the collection of grids can be achieved through the use of threads on shared-memory multi-processor machines. Both parallelisation strategies, i.e., the direct coarse grain parallel treatment of the different grids and the fine grain approach via threads, can also be combined and used simultaneously. This leads to a parallel method which is well suited for a cluster of multi-processor machines. See [13] for first results concerning speedups and efficiency.

4.1 Parallelisation across grids

The linear systems 5) for the partial functions f_α of the collection of grids can be computed independently of each other, therefore their computation in each outer iteration step can be simply done completely in parallel. Each process computes the solution on a certain number of grids. If as many processors are available as there are grids in the collection of grids then each processor computes the solution for only one grid. The control process collects the results and computes the final function f on the sparse grid. Just a short setup or gather phase, respectively, is necessary. Since the cost of computation is roughly known *a-priori*, a simple but effective static load balancing strategy is available; see [14].

4.2 Parallelisation across data

To compute $B^T \cdot B$ in (5) for each data instance $\boldsymbol{x}^{(i)}$, the product of the values of all basis function, which are non-zero at $\boldsymbol{x}^{(i)}$, has to be calculated, i.e.,

$\sum_{j,k} \varphi_j(\boldsymbol{x}^{(i)}) \cdot \varphi_k(\boldsymbol{x}^{(i)})$, and the results have to be written into the matrix structure at $\left(B^T \cdot B\right)_{j,k}$. These computations only depend on the data and therefore can be done independently for all instances. Therefore the $d \times n$ array of the training set can be separated in p parts, where p is the number of processors available in the shared-memory environment. Each processor now computes the matrix entries for n/p instances. Some overhead is introduced to avoid memory conflicts when writing into the matrix structure. In a similar way the evaluation of the classifier on the data points can be threaded in the evaluation phase.

4.3 Parallelisation of solvers

After the matrix is built, threading can be used on SMP architectures in the solution phase as fine grain parallelism. Since we are using an iterative solver most of the computing time is used for the matrix-vector-multiplication. Here the vector α in (5) of size m can be split into p parts and each processor now computes the action of the matrix on a vector of size m/p.

4.4 Combination of coarse and fine grain parallelism

We have seen in [13] that coarse grain parallelism yields the highest speedups and the better efficiency. However, the number of grids may be such that the parallel resources are not fully utilised. Here we show how much additional speedup to expect when using fine grain parallelism in addition to the coarse grain parallelism.

If p processors are used to parallelise a computation with k grids one can expect a maximal speedup of $k/\lceil k/p \rceil$. This is displayed for the case of $p = 30$ and $k = 1, \ldots, 300$ in figure 2. In order to use the fine grain parallelism we first use the coarse grain approach for a first stage where $p\lfloor k/p \rfloor$ grids are processed and then, in a second stage the processors are distributed evenly among the remaining tasks. After the first step there are

$$p_x = k - p\lfloor k/p \rfloor < p$$

tasks remaining. Thus one has $p_l = \lfloor p/p_x \rfloor$ processors per remaining task. These are then used to parallelise all the remaining tasks and one thus gets a total speedup of

$$S_{p,k} = k/(\lfloor k/p \rfloor + \mathrm{sign}(p_x) * (0.1 + 0.9/p_l).$$

This is again displayed in figure 2. We assume here that the fine grain parallelism has ten percent overhead which cannot be parallelised.

This approach can be refined further through the concurrent use of fine and coarse grain parallelism, e.g., parallelising $p/2$ grids with the parallelism across grids and a 2 processor shared memory parallelism for each grid.

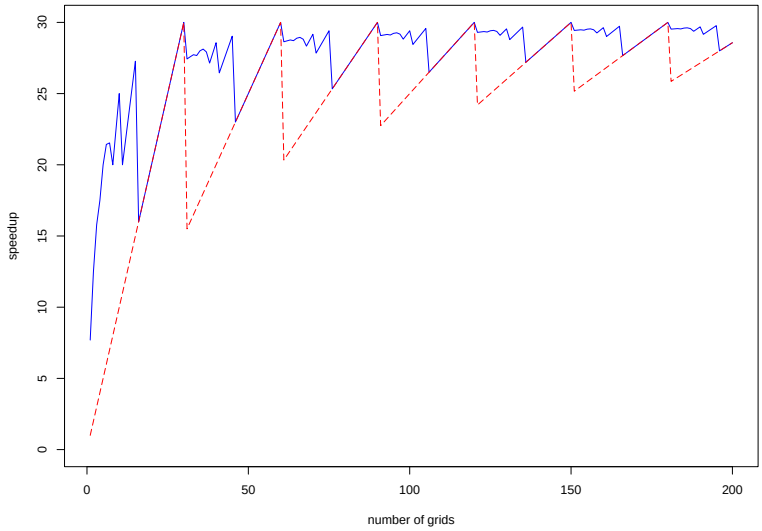

Fig. 2. Theoretical speedups for coarse grain parallelism (dashed line) and coarse grain with added fine grain.

5 Conclusion

In this paper we extend results from [13], where two parallelisation strategies for the sparse grid combination technique where shown. Through the combination of both strategies the speedup results can be further improved. This leads to a parallel method which is well suited for a cluster of multi-processor machines.

The performance of the outer iteration using new combination coefficients does depend on the commutation properties of the projection operators into the partial spaces. In particular, if the projection operators commute, the outer iteration gives the correct result after one step. In addition, the performance depends on the partial spaces in a different way: If all the partial spaces are one-dimensional, then the outer iteration again gives the exact result after one step. Alternatively, if the spaces form an ordered sequence $V_1 \subset V_2 \subset \ldots V_m$ and the dimensions are $\dim(V_k) = k$ then again the new combination method gives an exact result after one step (which can be seen by using orthogonalisation). It would thus seem that in general more spaces would provide better convergence of the outer iteration. In the case of the solution of the Laplace equation with finite elements, it is known that the classical combination method acts like extrapolation and larger error terms cancel [4]. Note that these classical combination methods do also use a large number of subspaces. Thus larger numbers of subspaces would appear to provide better performance. The extrapolation properties of the general combination method for fitting problems will be further investigated in the future.

References

1. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A. (1984)
2. Friedman, J.H.: Multivariate adaptive regression splines. Ann. Statist. **19** (1991) 1–141 With discussion and a rejoinder by the author.
3. Zenger, C.: Sparse grids. In Hackbusch, W., ed.: Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990. Volume 31 of Notes on Num. Fluid Mech., Vieweg (1991) 241–251
4. Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In de Groen, P., Beauwens, R., eds.: Iterative Methods in Linear Algebra, IMACS, Elsevier, North Holland (1992) 263–281
5. Garcke, J., Griebel, M., Thess, M.: Data mining with sparse grids. Computing **67** (2001) 225–253
6. Garcke, J., Griebel, M.: Classification with sparse grids using simplicial basis functions. Intelligent Data Analysis **6** (2002) 483–502 (shortened version appeared in KDD 2001, Proc. Seventh ACM SIGKDD, F. Provost and R. Srikant (eds.), pages 87-96, ACM, 2001).
7. Hastie, T.J., Tibshirani, R.J.: Generalized additive models. Volume 43 of Monographs on Statistics and Applied Probability. Chapman and Hall Ltd., London (1990)
8. Hastie, T., Tibshirani, R.: Generalized additive models. Statist. Sci. **1** (1986) 297–318 With discussion.
9. Hegland, M.: Additive sparse grid fitting. In: Proceedings of the Fifth International Conference on Curves and Surfaces, Saint-Malo, France 2002. (2002) submitted.
10. Hegland, M., Nielsen, O.M., Shen, Z.: High dimensional smoothing based on multilevel analysis. Submitted (2000) Available at http://datamining.anu.edu.au/publications/2000/hisurf2000.ps.gz.
11. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA (1997)
12. Griebel, M.: The combination technique for the sparse grid solution of PDEs on multiprocessor machines. Parallel Processing Letters **2** (1992) 61–70
13. Garcke, J., Griebel, M.: On the parallelization of the sparse grid approach for data mining. In Margenov, S., Wasniewski, J., Yalamov, P., eds.: Large-Scale Scientific Computations, Third International Conference, Sozopol, Bulgaria. Volume 2179 of Lecture Notes in Computer Science. (2001) 22–32
14. Griebel, M., Huber, W., Störtkuhl, T., Zenger, C.: On the parallel solution of 3D PDEs on a network of workstations and on vector computers. In Bode, A., Cin, M.D., eds.: Lecture Notes in Computer Science 732, Parallel Computer Architectures: Theory, Hardware, Software, Applications, Springer Verlag (1993) 276–291