SEQUENTIAL FUNCTION APPROXIMATION FOR THE SOLUTION OF DIFFERENTIAL EQUATIONS

Andrew J. Meade Jr.*, Michael Kokkolaras, and Boris A. Zeldin
Department of Mechanical Engineering
and Materials Science
William Marsh Rice University
Houston, Texas, 77251-1892, USA
Phone: (713) 527-8101 ext. 3590
FAX: (713) 285-5423
E-mail: meade@rice.edu

Revised.

Communications in Numerical Methods in Engineering.

∗ Recipient of correspondence.
This work was supported under NASA grant number CRA2-35504 and ONR grant N00014-95-1-0741.

SEQUENTIAL FUNCTION APPROXIMATION FOR THE SOLUTION OF DIFFERENTIAL EQUATIONS

Andrew J. Meade, Jr., Michael Kokkolaras, and Boris A. Zeldin
Department of Mechanical Engineering
and Materials Science,
William Marsh Rice University
Houston, TX 77251-1892, USA

Key Words: Sequential function approximation, interpolation functions, optimization, parallel direct search.

Abstract

A computational method for the solution of differential equations is proposed. With this method an accurate approximation is built by incremental additions of optimal local basis functions. The parallel direct search software package (PDS), that supports parallel objective function evaluations, is used to efficiently solve the associated optimization problem. The advantage of the method is that although it resembles adaptive methods in computational mechanics, an a-priori grid is not necessary. Moreover, the traditional matrix construction and evaluations are avoided. Computational cost is reduced while efficiency is enhanced by the low-dimensional parallel-executed optimization and parallel function evaluations. In addition, the method should be applicable to a broad class of interpolation functions. Results and global convergence rates obtained for one- and two-dimensional boundary value problems are satisfactorily compared to those obtained by the conventional Galerkin finite element method.

SEQUENTIAL FUNCTION APPROXIMATION FOR THE SOLUTION OF DIFFERENTIAL EQUATIONS

Andrew J. Meade, Jr.*, Michael Kokkolaras†, and Boris A. Zeldin‡
Department of Mechanical Engineering
and Materials Science,
William Marsh Rice University
Houston, TX 77251-1892, USA

1 Introduction

It can be argued that function approximation is the foundation of all numerical approximation methods. Corresponding problems can involve linear or nonlinear, differential or integral operators and can vary from data-fitting problems to problems governed by systems of partial differential equations. In engineering applications, the problem of function approximation is addressed by assembling interpolation functions which adequately reproduce the dependent variable of interest. For example, the popular finite element method is based on the use of simple local low-order polynomial splines which yield accurate and computationally efficient global function approximation.

This paper addresses the problem of function approximation in the context of the solution of differential equations. It is motivated by several algorithms developed for function approximation problems in the area of artificial intelligence (in particular neural networks) and their close resemblance to adaptive methods for the solution of computational me-

*Associate Professor. Recipient of correspondence, e-mail: meade@rice.edu
†Research Assistant
‡Research Associate

chanics problems. These algorithms will be reviewed in the following two subsections. In the remainder of the paper, the proposed method will be described and demonstrated by means of specific applications.

The main objective of this paper is to formulate the mathematical foundation for a novel computational procedure rather than present major applications. Consequently, linear one- and two-dimensional boundary value problems, representative of more complex problems (e.g., steady viscous fluid flow through a duct), were solved to demonstrate the potential of the novel algorithm. Computational mechanics problems of greater complexity will be the focus of future work.

1.1 Function Approximation in Artificial Neural Network Applications

Any approximation algorithm that uses a combination of basis functions that can be mapped into a graph-directed representation can be called an artificial neural network. In this regard, function approximation in the framework of neural computations has been primarily based on the results of Cybenko [1] and Hornik et al. [2], who showed that a continuous d-dimensional function can be arbitrarily well approximated by a linear combination of one-dimensional functions ϕ_i.

$$u \approx u_n^a(\bar{\xi}) = c_0 + \sum_{i=1}^{n} c_i \phi_i(\eta(\bar{\xi}, \bar{p}_i)) \tag{1}$$

where $\eta \in \mathbf{R}$, $\bar{\xi} \in \mathbf{R}^d$, $\bar{p}_i \in \mathbf{R}^m$, and $c_i \in \mathbf{R}$ represent the function argument, independent variables, function parameters, and linear coefficients, respectively.

The appropriate linear and nonlinear network parameters in equation (1) are traditionally selected by solving a non-linear optimization problem with the objective function given by the mean square error over some domain Ω

$$\epsilon^2 \equiv \int_\Omega (u - u_n^a)^2 d\bar{\xi} \quad . \tag{2}$$

In the neural network literature, the numerical minimization of equation (2) by the steepest descent method is known as the backpropagation algorithm [3]. More sophisticated optimization methods, including the conjugate gradient and the Levenberg-Marquardt methods, have been also used for neural network training. However, it has been found that even these advanced optimization methods are prone to poor convergence [4]. Clearly, the training algorithms must address a multi-dimensional optimization problem with non-linear dependence on the network parameters $\bar{p}_i$.

As an alternative, Jones [5], [6] and Barron [7] proposed the following iterative algorithm for sequential approximation:

$$u_n^a(\bar{\xi}) = \alpha_n u_{n-1}^a(\bar{\xi}) + c_n \phi(\eta(\bar{\xi}, \bar{p}_n)) \tag{3}$$

where $\bar{p}_n, c_n$, and α_n are optimally selected at each iteration of the algorithm. As a result, the high-dimensional optimization problem associated with neural network training is reduced to a series of simpler low-dimensional problems. A general principle of statistics was utilized to show that the upper bound of the error ϵ is of the order $C/\sqrt{n}$, where C is a positive constant. Further, Orr [8] introduced a forward selection method of sequential network training; this is essentially a method of incremental function approximation. At each iteration of the algorithm an additional basis function, which produces the largest reduction of error in the previous iteration, is chosen from a given set of functions and added to the approximation. However, the forward selection training method can be inefficient in that it may require significant computational resources when the set of trial functions is large. A similar principle is utilized in Platt's resource allocating networks (RAN) [9]. Whenever an unusual pattern is presented to the network, in on- or off-line network training, a new computational "unit" is allocated. Note that these computational units respond to local regions of the input space.

The concept of sequential approximation is one of the major features of the method proposed in this paper for the solution of computational mechanics problems.

1.2 Optimization in Computational Mechanics

Most numerical methods in computational mechanics rely on the discretization of a finite dimensional domain into a grid of nodal points. The requisite computational cost and the efficiency are strongly affected by the type of the discretization. For example, slow convergence and numerical instability, resulting from poor condition numbers of the associated matrices, have been reported in the literature for uniform grids. Therefore, optimization in computational mechanics has focused primarily in the selection of the optimal grid. Oliveira [10] showed that optimal node distribution provides minimal total potential energy of the system. In fact, it was proven that such distributed nodes lie along isoenergetic contours.

Felippa [11], [12] followed an optimal node distribution approach. The direct and computationally expensive grid optimization problem is solved for a given configuration of finite elements and the nodes are then relocated. This process is repeated iteratively until convergence to an optimal grid is achieved. However, this approach is limited to linear self-adjoint differential operators where energy variational principles are readily available. Results were reported for extremely coarse grids, due to the computational complexity of the associated multi-dimensional optimization problem. Diaz et al. [13] discussed an adaptive method to improve the accuracy of the finite element analysis; grid optimization was based on the minimization of an estimated upper bound on the potential energy of the finite element solution. It is noted that recently developed adaptive methods of computational mechanics substitute the principle of the system potential energy minimum with the weaker criterion of the homogeneous distribution of the approximation error [14] - [16].

The method proposed in this paper for solving computational mechanics problems closely resembles the previously discussed adaptive numerical methods. However, it is not confined by grid requirements and it can be used in conjunction with a broad class of basis functions.

2 The Proposed Method

The basic principles of the proposed computational method for solving differential equations are presented in this section. The development of this method was motivated by the similarities between iterative optimization procedures reviewed in Sections 1.1 and 1.2. The main features of the proposed method are a) the iterative nature of the developed algorithm for sequential, or incremental, approximations and b) the solution of an optimization problem at each stage of the algorithm.

The developed algorithm can be summarized as follows. Given a problem, the dependent variable is approximated by a basis expansion; an incremental set of interpolation functions is sequentially built to improve the expansion-based approximation. At each stage of the algorithm the parameters of the new interpolation function are selected by solving a nonlinear low-dimensional problem, while the associated coefficient is determined by evaluating the orthogonality requirement. For example, assuming that the algorithm has reached step i, the approximate function is given by

$$u_i^a(\bar{\xi}) = c_0 + \sum_{j=1}^{i} c_j \Phi_j(\bar{\xi}, \bar{p}_j) = u_{i-1}^a(\bar{\xi}) + c_i \Phi_i(\bar{\xi}, \bar{p}_i) \tag{4}$$

where the one-dimensional function ϕ of equation (1) has been replaced with a multi-dimensional Lagrangian basis Φ. The coefficients c_j and the interpolation functions $\Phi_j(\bar{\xi}, \bar{p}_j)$, $j = 1, \ldots, i-1$, are held fixed while the coefficient c_i and the interpolation function $\Phi_i(\bar{\xi}, \bar{p}_i)$ are optimally computed according to an appropriate criterion (see Section 3.1 for further discussion). Note that there are no restrictions on the distribution of the interpolation functions over the domain of interest; any degree of overlapping is possible. As a result, nodal points and the associated grid are not known a-priori. The sequential algorithm is halted when the relative difference between the previous and current approximations falls below a user-selected tolerance level.

2.1 The Parallel Direct Search (PDS) optimization algorithm

The computational efficiency of the method depends on the solution cost of the nonlinear optimization problem. The dimensionality of the nonlinear problem is kept low by the iterative nature of the algorithm. In addition to the reduction of the optimization problem size, parallel direct search methods [17] are used to solve the optimization problem efficiently. The selected parallel direct search optimization software package (PDS) is a collection of Fortran routines, operating in double precision, used for solving both constrained and unconstrained nonlinear optimization problems. PDS does not require derivative information and offers the advantage of supporting parallel objective function evaluations by means of the Message Passing Interface (MPI). PDS is scalable in that it fully exploits additional processors by performing new searches and by refining existing ones. The only requirement to guarantee convergence is that the objective function must be continuous. PDS works best when the ratio of the number of function evaluations to the number of the optimization variables is large. Clearly, the number of optimization variables should be kept as low as possible. This requirement is satisfied by the proposed method for sequential function approximation.

3 Mathematical Formulation

In this section, the proposed method for optimal sequential function approximation is described and implemented for the solution of differential equations. Algorithm and implementation issues are addressed by means of one- and two-dimensional numerical examples. Results are presented, compared, and discussed.

3.1 Solution of Differential Equations

Consider a general boundary value problem with homogeneous boundary conditions

$$L(u(\bar{\xi})) = f(\bar{\xi}) \ \text{ in } \Omega, \quad \text{with} \quad B(u(\bar{\xi})) = 0 \ \text{ on } \partial\Omega \tag{5}$$

where $L(\cdot)$ is a linear, self-adjoint differential operator, $B(\cdot)$ is a corresponding boundary operator and $u(\bar{\xi})$ is the exact solution to the differential equation. Alternatively, the solution to equation (5) can be determined by minimizing the following functional

$$E \equiv \frac{1}{2} l(u_i^a(\bar{\xi}), u_i^a(\bar{\xi})) - \langle u_i^a(\bar{\xi}), f(\bar{\xi}) \rangle \tag{6}$$

where $u_i^a(\bar{\xi})$ is the approximate solution given by equation (4) and $l(\cdot,\cdot)$ denotes the bilinear symmetric differential form associated with the operator L. Note that, for this type of problem,

$$l(\cdot,\cdot) = \langle L(\cdot), \cdot \rangle \tag{7}$$

Moreover, since $\langle u_i^a(\bar{\xi}), f(\bar{\xi}) \rangle = l(u_i^a(\bar{\xi}), u(\bar{\xi}))$, equation (6) can be rewritten as

$$E = \frac{1}{2} l(u(\bar{\xi}) - u_i^a(\bar{\xi}), u(\bar{\xi}) - u_i^a(\bar{\xi})) - \frac{1}{2} l(u(\bar{\xi}), u(\bar{\xi})) \tag{8}$$

Also, since $u(\bar{\xi})$ is the nontrivial exact solution and L is positive definite,

$$\frac{1}{2} l(u(\bar{\xi}), u(\bar{\xi})) = \text{constant} = K > 0 \ .$$

Define the error at the i-th step as e_i where $e_i = u(\bar{\xi}) - u_i^a(\bar{\xi})$. Then, $e_i = e_{i-1} - c_i \Phi_i(\bar{\xi})$, and equation (8) becomes

$$E = \frac{1}{2} l(e_{i-1} - c_i \Phi_i(\bar{\xi}), e_{i-1} - c_i \Phi_i(\bar{\xi})) - K \quad =$$

$$\frac{1}{2} l(e_{i-1}, e_{i-1}) + \frac{1}{2} c_i^2 l(\Phi_i(\bar{\xi}), \Phi_i(\bar{\xi})) - c_i l(e_{i-1}, \Phi_i(\bar{\xi})) - K \tag{9}$$

The coefficient c_i can be determined by equating the partial derivative of E with respect to c_i to zero:

$$\frac{\partial E}{\partial c_i} = 0 = c_i l(\Phi_i(\bar{\xi}), \Phi_i(\bar{\xi})) - l(e_{i-1}, \Phi_i(\bar{\xi})) \tag{10}$$

That is, the coefficient c_i can then be calculated from

$$c_i = \frac{l(e_{i-1}, \Phi_i(\bar{\xi}))}{l(\Phi_i(\bar{\xi}), \Phi_i(\bar{\xi}))} \tag{11}$$

and equation (9) becomes

$$E = \frac{1}{2} l(e_{i-1}, e_{i-1}) - K - \frac{1}{2} \frac{(l(e_{i-1}, \Phi_i(\bar{\xi})))^2}{l(\Phi_i(\bar{\xi}), \Phi_i(\bar{\xi}))} \tag{12}$$

To improve the accuracy of the approximation, the magnitude of the last term must be maximized by appropriately selecting the parameters of the interpolation function $\Phi_i(\bar{\xi})$. PDS is utilized to solve this nonlinear problem, while the associated coefficient is readily determined as described above when the governing equation is linear.

Clearly, the proposed method resembles some of the conventional adaptive numerical techniques from computational mechanics in that the new interpolation function is selected to provide the largest projection on the error of the preceding iteration. Specifically, the new interpolation function is positioned at the location of the largest estimated error.

3.2 Numerical Examples

The Poisson equation,

$$-\nabla^2 u(\bar{\xi}) = f(\bar{\xi}), \tag{13}$$

which can be used as a low-fidelity representation of some problems in computational mechanics, has been successfully solved by means of the proposed method for one and two dimensions with homogeneous Dirichlet boundary conditions. For the one-dimensional problem the domain of interest was selected as $\Omega = [0, 1]$ with $\bar{\xi} = x$ and a forcing function of $f(x) = x^2$. In the two-dimensional case, $\Omega = [-1, 1] \times [-1, 1]$, $\bar{\xi} = (x, y)^{\mathrm{T}}$ and $f(x, y) = 2\pi$.

Piecewise linear interpolation functions (B_1 splines) were used. The interpolation function is defined by the equation

$$\Phi(x) = \begin{cases} \frac{x-(x_M-\Delta x_l)}{\Delta x_l} & \text{if} \quad x_M - \Delta x_l \le x \le x_M \\ \frac{(x_M+\Delta x_r)-x}{\Delta x_r} & \text{if} \quad x_M \le x \le x_M + \Delta x_r \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where the parameters x_M, Δx_l, and Δx_r denote the location of the center of the function, its width to the left and right, respectively. Note that the simple product of two one-dimensional functions has been used in the two-dimensional case,

$$\Phi(x, y) = \Phi(x)\Phi(y). \tag{15}$$

The optimization design variables are given by $\bar{p}_i = (x_M, \Delta x_l, \Delta x_r)_i^{\mathrm{T}}$ for the one-dimensional case and by $\bar{p}_i = (x_M, \Delta x_l, \Delta x_r, y_M, \Delta y_l, \Delta y_r)_i^{\mathrm{T}}$ for the two-dimensional case. The initial values for the optimization variables required by the PDS algorithm are chosen so that each new interpolation function is initially centered in, and covers, the problem domain. The interpolation functions were constrained to vanish at the domain boundaries in order to satisfy the homogeneous Dirichlet boundary conditions exactly. The algorithm was initialized with an empty set of basis functions, however, a non-empty set of basis functions can be used to initialize the algorithm. All calculations were done in double precision to accommodate PDS.

Two PDS-input parameters control the number of required PDS iterations and function and constraint evaluations; the number of search directions d and the user-selected tolerance tol, which forms the convergence criterion for PDS. The PDS package guarantees convergence for $d \geq 2m$ where m is the number of design variables ($m = 6$). However, d should be chosen to be larger than $2m$ in order to secure a detailed search. The number of search directions should be relatively small if the objective function is convex and large if the objective function is suspected to be nearly concave. Typically, if a small d is chosen, a small tolerance should also be chosen, since the search interval is relatively course. Similarly, if a large d is selected for the same number of design variables, a larger tolerance should be chosen to ensure computational efficiency.

The solution obtained for the one-dimensional problem, with $d = 100$ and $tol = 10^{-4}$, is compared to the exact in Fig. 1(a). Figure 1(b) compares the global convergence rate of the proposed method to that from the Galerkin technique using linear Lagrangian shape functions on a uniform grid. The RMS errors were calculated from 101 uniformly distributed trial points.

The evolution of the incrementally built solution and the squared absolute error distribution for the two-dimensional problem are shown in Fig. 2(a) and 2(b), respectively for $d = 1000$ and $tol = 10^{-4}$. The RMS error global convergence rate, compared to the rate from the Galerkin technique using uniformly distributed bilinear Lagrangian shape functions, is presented in Fig. 3. The RMS errors were calculated based on 961 uniformly distributed trial points.

The "kinks" in the global convergence curves of Fig. 1(b) and Fig. 3 come from a combination of two sources. Firstly, PDS optimizes by direct search. As such it can only evaluate a finite combination of parameters which results in a finite interval of uncertainty. Secondly, the algorithm is not designed to explicitly minimize the RMS error; the algorithm effectively minimizes equation (12). However, equation (12) acts as an upper bound on the RMS, so though successive RMS values can increase they cannot exceed the value generated by equation (12). This can result in a sawtooth or step-like oscillation in the convergence curves.

Tables 1 and 2 summarize the performance characteristics of PDS, for the two-dimensional numerical example, with both a relatively small and large number of directions. It is clear from Table 1 that a relatively small number of directions ($d = 100$) will require more PDS iterations, due to the smaller tolerance (10^{-6}), but fewer function and constraint evaluations. Table 2 shows that a large number of directions ($d = 1000$) will require fewer PDS iterations, due to the larger tolerance (10^{-4}), but more function and constraint evaluations.

Note that both Tables 1 and 2 show that the number of PDS iterations and function and constraint evaluations fluctuate but remain "bounded" and are independent from the number of sequential algorithm steps, for both small and large values of d. Moreover, these evaluations can always be computed faster with increasing number of available parallel processors. The only cost linked to the increase in algorithm steps stems from the increasing number of numerical integrations as more interpolation functions are added to the series expansion; this cost is mitigated by performing numerical integrations in parallel.

4 Conclusions

A computational method based on sequential approximation concepts, combined with interpolation function optimization, has been proposed for the solution of differential equations. The developments of this paper are motivated by observed similarities between numerical procedures developed in the areas of neural networks and computational mechanics.

It has been shown that the proposed method can be successfully applied to one- and two-dimensional linear differential equations and yield accurate results. The advantage of the method is that although it resembles adaptive methods in computational mechanics, an a-priori grid is not necessary. Moreover, the traditional matrix evaluations are avoided. Computational cost is reduced while efficiency is enhanced by the low-dimensional parallel-executed optimization and parallel function evaluations. In addition, the method should be applicable to a broad class of interpolation functions.

Finally, note that from equation (12)

$$l(e_{i-1}, \Phi_i(\bar{\xi})) = l((u(\bar{\xi}) - u^a_{i-1}(\bar{\xi})), \Phi_i(\bar{\xi})) \quad =$$

$$l(u(\bar{\xi}), \Phi_i(\bar{\xi})) - l(u^a_{i-1}(\bar{\xi}), \Phi_i(\bar{\xi})) = \langle L(u(\bar{\xi})), \Phi_i(\bar{\xi})\rangle - \langle L(u^a_{i-1}(\bar{\xi})), \Phi_i(\bar{\xi})\rangle \quad =$$

$$\langle f(\bar{\xi}), \Phi_i(\bar{\xi})\rangle - \langle L(u^a_{i-1}(\bar{\xi})), \Phi_i(\bar{\xi})\rangle = \langle R_{i-1}, \Phi_i(\bar{\xi})\rangle \tag{16}$$

where the equation residual R_{i-1} is equal to $-\Big[L(u^a_{i-1}(\bar{\xi})) - f(\bar{\xi})\Big]$. From a geometric perspective, maximizing the third term in equation (12) is identical to constructing a basis vector that is parallel to the vector R_{i-1}. This use of the equation residual shows that it may be possible to avoid energy variational principles in practice. Specifically, equation (16) provides a solid foundation for the extension of the method to problems involving various types of differential operators.

Acknowledgments

The authors would like to thank David Serafini for his help with the parallel execution of PDS in the MPI environment and the Center for Research for Parallel Computation (CRPC) at Rice University for the use of its computer facilities. This work was supported under NASA grant number CRA2-35504 and ONR grant N00014-95-1-0741.

REFERENCES

[1] G. Cybenko, "Approximations by Superpositions of a Sigmoidal Function", *Math. Contr. Signals, Syst.*, **2**, 303-314, (1989).

[2] K. Hornik, M. Stinchcombe, and H. White, "Multi-Layer Feedforward Networks are Universal Approximators", *Neural Networks*, **2**, 359-366, (1989).

[3] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, 318-362, edited by D. Rumelhart, J. L. McClelland, and the PDP Research Group, MIT Press, Cambridge, MA, 1986.

[4] S. Saarinen, R. Bramley, and G. Cybenko, "Ill-Conditioning in Neural Network Training Problems", *SIAM J. Sci. Comput.*, **14**, No. 3, 693-714, (1993).

[5] L. K. Jones, "Constructive Approximations for Neural Networks by Sigmoidal Functions", *Proceedings of the IEEE*, **78**, No. 10, 1586-1589, (1990).

[6] L. K. Jones, "A Simple Lemma on Greedy Approximation in Hilbert Space and Convergence Rates For Projection Pursuit Regression and Neural Network Training", *The Annals of Statistics* **20**, No. 1, 608-613, (1992).

[7] A. R. Barron, "Universal Approximation Bounds for Superpositions of a Sigmoidal Function", *IEEE Transactions on Information Theory*, **39**, No. 3, 930-945, (1993).

[8] M. J. Orr, "Regularization in the Selection of Radial Basis Function Centres", *Neural Computation*, **7**, No. 3, 606-623, (1995).

[9] J. Platt, "A Resource-Allocating Network for Function Interpolation", *Neural Computation*, **3**, 213-225, (1991).

[10] E. R. Oliveira, in *Proc. Third Conf. Matrix Methods in Struct. Mech.*, AFFDL-TR-71-160, 423-446, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, 1971.

[11] C. A. Felippa, "Optimization of Finite Element Grids by Direct Energy Search," *Appl. Math. Modelling*, **1**, 93-96, (1976).

[12] C. A. Felippa, "Numerical Experiments in Finite Element Grid Optimization by Direct Energy Search," *Appl. Math. Modelling*, **1**, 239-244, (1977).

[13] A. R. Diaz, N. Kikuchi, and J. E. Taylor, "A Method of Grid Optimization for Finite Element Methods", *Computer Methods in Applied Mechanics and Engineering*, **41**, 29-45, (1983).

[14] L. Demkowicz and J. T. Oden, "On a Mesh Optimization Method Based on a Minimization of Interpolation Error", *Int. J. Engng. Sci.*, **24**, No. 1, 55-68, (1986).

[15] A. R. Diaz, N. Kikuchi, and J. E. Taylor, "Optimal Design Formulations for Finite Element Grid Adaptation", Lecture Notes in Mathematics, **1086**, *Sensitivity of Functionals with Applications in Engineering Science*, edited by V. Komokov, Springer-Verlag, Berlin, 1984.

[16] R. A. DeVore, "Degree of Nonlinear Approximation", *Approximation Theory VI*, **1**, 175-201, edited by C. K. Chui, L. L. Schumaker, and J. D. Ward, Academic Press, 1989.

[17] J. E. Dennis, Jr. and V. Torczon, "Direct Search Methods on Parallel Machines", *SIAM J. Optimization*, **1**, No. 4, 448-474, (1991).

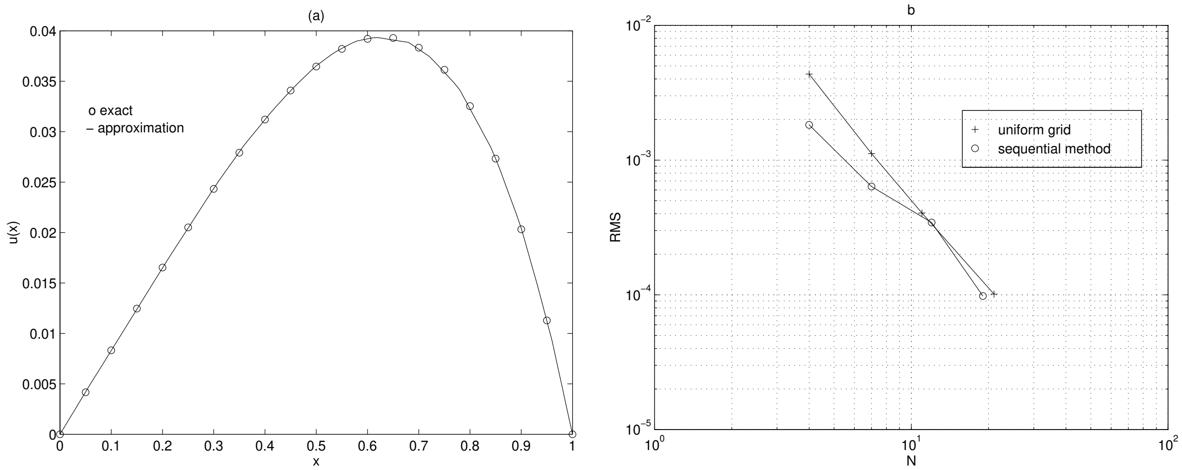

Figure 1: One-dimensional Poisson's equation: (a) solution comparison using 19 optimal interpolation functions, (b) convergence rate comparison.

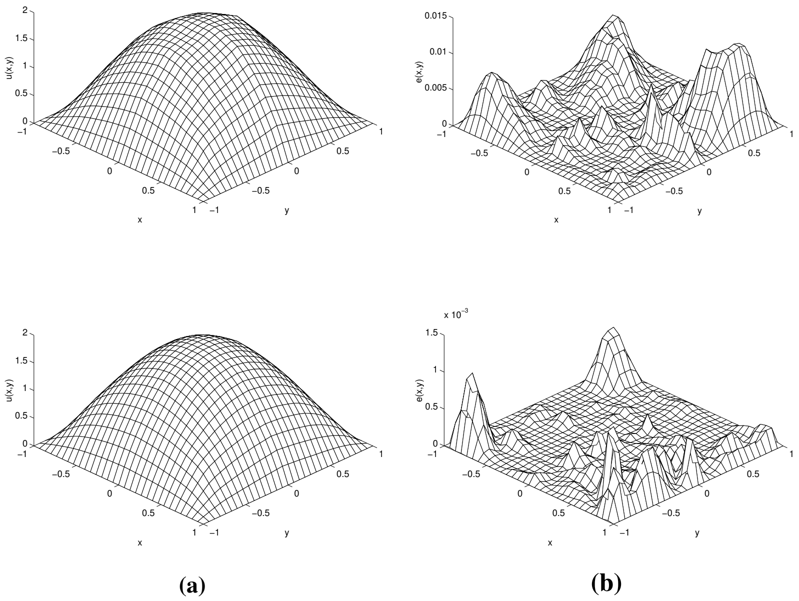

Figure 2: Two-dimensional Poisson's equation: (a) solution and (b) squared error distribution after (top to bottom) 25 and 100 optimal interpolation functions, respectively.

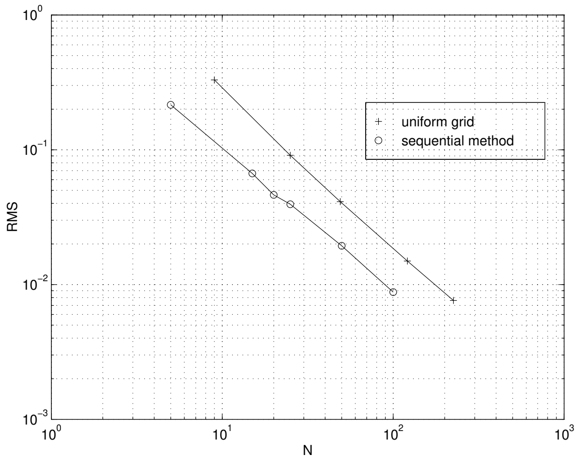

Figure 3: Two-dimensional Poisson's equation: convergence rate comparison.

Table 1: PDS performance characteristics for the two-dimensional Poisson problem using $d = 100$ and $tol = 10^{-6}$.

Algorithm step	# of PDS iterations	Total # of function evaluations	Total # of constraint evaluations
1	19	459	1812
30	39	2303	3818
60	39	1535	3588
90	39	3431	3812

Table 2: PDS performance characteristics for the two-dimensional Poisson problem using $d = 1000$ and $tol = 10^{-4}$.

Algorithm step	# of PDS iterations	Total # of function evaluations	Total # of constraint evaluations
1	7	1066	6097
30	13	3156	11632
60	15	12718	13996
90	8	909	6574