Genetic Programming Approaches for Solving Elliptic Partial Differential Equations

András Sóbester, Prasanth B. Nair, and Andy J. Keane

***Abstract*—In this paper, we propose a technique based on genetic programming (GP) for meshfree solution of elliptic partial differential equations. We employ the least-squares collocation principle to define an appropriate objective function, which is optimized using GP. Two approaches are presented for the repair of the symbolic expression for the field variables evolved by the GP algorithm to ensure that the governing equations as well as the boundary conditions are satisfied. In the case of problems defined on geometrically simple domains, we augment the solution evolved by GP with additional terms, such that the boundary conditions are satisfied by construction. To satisfy the boundary conditions for geometrically irregular domains, we combine the GP model with a radial basis function network. We improve the computational efficiency and accuracy of both techniques with gradient boosting, a technique originally developed by the machine learning community. Numerical studies are presented for operator problems on regular and irregular boundaries to illustrate the performance of the proposed algorithms.**

***Index Terms*—Boosting, genetic programming (GP), meshfree collocation, partial differential equations (PDEs), radial basis functions.**

I. Introduction

MATHEMATICAL modeling of many phenomena and systems encountered in engineering and the physical sciences gives rise to partial differential equations (PDEs). There are a number of well-established techniques in the literature for solving PDEs, including the finite difference, finite-element, finite volume, and boundary element methods. These techniques employ a mesh or a grid of points in space to approximate the field variables of interest. More recently, much research has focused on the development of meshfree algorithms to solve operator problems. Work in this direction was primarily driven by the observation made by Kansa in 1990 [1], [2] that the classical collocation approach for solving PDEs solves a generalized interpolation problem. This suggested the possibility of adapting function approximation techniques as a way of developing novel meshfree numerical schemes for solving PDEs.

Manuscript received August 25, 2003; revised May 10, 2006. This work was supported in part by the University Technology Partnership for Design—A partnership between BAE Systems, Rolls-Royce, and the Universities of Southampton, Cambridge and Sheffield, U.K.

The authors are with the Computational Engineering and Design Research Group, School of Engineering Sciences, University of Southampton, Hampshire SO17 1BJ, United Kingdom (e-mail: a.sobester@soton.ac.uk; P.B.Nair@soton.ac.uk; Andy.Keane@soton.ac.uk).

Digital Object Identifier 10.1109/TEVC.2007.908467

Over the last decade, a number of meshfree schemes based on moving least-squares approximation [3], [4], radial basis functions [5]–[8], and feedforward neural networks [9]–[14] have been proposed in the literature. In these, the field variable $u(\mathbf{x})$ is approximated as

$$u(\mathbf{x}) \approx \text{span}\,\{\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_m(\mathbf{x})\} = \sum_{i=1}^{m} \beta_i \phi_i(\mathbf{x}) \quad (1)$$

where $\phi_i(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ is a trial function and β_i, $i = 1, 2, \ldots, m$ are undetermined coefficients. The accuracy of the approximation depends critically on the choice of the trial function and the scheme employed to compute the undetermined coefficients. In most meshfree schemes in the literature the structure of the trial functions as well as their number are kept fixed. The undetermined coefficients can then be computed by applying a collocation technique. Alternatively, they can be estimated by using the Galerkin or Petrov–Galerkin scheme, which involves tackling the weak form of the governing equations. In order to improve the accuracy of the approximations further, nonlinear optimization techniques are often employed to tune the trial functions; see, for example, Galperin and Kansa [15].

The elliptic PDE solver presented here is based on the genetic programming (GP) paradigm. Our objective is to generate approximate symbolic expressions for the field variable $u(\mathbf{x})$ in a more general and automatic fashion compared to existing meshfree techniques. In other words, we wish to approximate $u(\mathbf{x})$ without invoking any prior assumptions about the structure of the trial functions. Additionally, since the results are obtained in a compact, symbolic form, they can then be used in any subsequent analytical calculations.

Since its conception, the GP paradigm has been applied to solve problems via 'error-driven evolution', which covers a wide range of applications such as function approximation, classification, symbolic differentiation, integration, solving differential or integral equations, control, etc. ([16], [17]). These techniques, often referred to collectively as symbolic regression methods, generally involve finding a function in symbolic form that fits a set of observational data. A number of papers reporting the application of GP to regression can be found in the literature; see, for example, [18]–[20]. More recently, Nikolaev and Iba ([21], [22]) proposed an inductive GP approach which, in conjunction with suitable model selection criteria, can be employed to construct parsimonious polynomial networks that generalize well. In recent years, a number of researchers have also applied GP to infer differential equations which govern some phenomena of interest given a set of observational data; see, for example, the work of Sakamoto and Iba [23].

Koza's book [17] contains some studies on the application of GP to solve ordinary differential equations (ODEs). These are based on the principle of evolving a symbolic expression for the solution by minimizing an error measure that indicates the extent to which a candidate solution satisfies the governing ODEs and the initial conditions. Koza viewed the solution of differential equations as "*the search in a space of compositions of functions and terminals for a particular composition that satisfies the equation and its initial conditions*." Such a reformulation of the problem, he argues, makes it an immediate candidate for solution by GP.

More recently, Howard and Roberts [24] presented a detailed investigation in which the application of GP to solve model ODEs representing convection-diffusion phenomena was studied. The candidate solution to the ODE was represented using a polynomial augmented with additional terms to ensure that the boundary conditions are satisfied by construction. Although a polynomial representation is restrictive, it allows the least squares error measure indicating the fitness of a candidate solution to be evaluated analytically. However, the numerical studies were not very promising, particularly at high Pèclet numbers when the field variable exhibits sharp variations due to the formation of shock waves. The failure to evolve good solutions here could be a consequence of the notorious snaking problem associated with higher order polynomials.

To the best of our knowledge, the applicability of the GP paradigm to solve PDEs has not yet been fully investigated in the literature, perhaps due to the not particularly promising initial results of studies on ODEs. In the context of finding approximate solutions to PDEs, GP, in principle, offers a number of potential advantages: (1) since no mesh is required, setup is straightforward; (2) the approach is intrinsically parallel, which allows for the possibility of achieving nearly linear speedups on parallel and distributed systems; (3) the final solution is a (compact) symbolic expression and, hence, the memory requirements are significantly smaller than conventional techniques such as the finite-element and finite-difference method and, perhaps most importantly; and (4) the expression obtained can often be used in subsequent analytical calculations.

Our technique employs a combination of GP and least-squares collocation to solve elliptic PDEs, in a formulation similar to the neural network-based collocation algorithm of Lagaris *et al.* [11]. The basic idea here is to use the least-squares collocation principle to set up fitness criteria to check how well the expressions evolved by GP satisfy the governing equations and the boundary conditions. The fitness values are computed over a set of collocation points inside the domain and on its boundary.

This basic formulation, however, leads to a very large search space. To reduce it, we ensure that only solutions satisfying the boundary conditions are obtained, by repairing the expressions evolved by GP. For PDEs defined on geometrically simple boundaries, we augment the GP model with additional terms to ensure that the boundary conditions are satisfied by construction. In cases where the boundaries are geometrically complex, we hybridize the GP model with a radial basis function (RBF) network.

To further improve computational efficiency and accuracy, we propose an approach wherein GP is applied in an iterative or stagewise fashion to construct a sequence of simple approximations—an approach termed *gradient boosting* in the machine learning literature. These approximations are then aggregated to form the final expression for the field variables (see [10] for a description of this formulation in the context of increasing the accuracy of neural network solutions of linear PDEs).

Finally, to complete the context in which we invite the reader to view this work, it is worth noting that there is some precedent in the literature for using evolutionary algorithms (EA) in the PDE solution process. Recently, He *et al.* [32] employed a selecto-recombinative EA to evolve the optimal relaxation factor for a successive over-relaxation (SOR) algorithm, which is used to solve PDEs. The approach proposed here differs from this in that the EA (a GP algorithm in this case) evolves *the solutions themselves*, as opposed to a hyperparameter that governs a numerical solution process.

The remainder of this paper is organized as follows. Section II outlines the basic idea of how GP can be employed to solve elliptic PDEs via least-squares collocation. Section III outlines the two approaches for repairing the solution evolved by GP to ensure that the boundary conditions are implicitly satisfied. In Section IV, we describe the gradient boosting approach for improving the computational efficiency of the GP approach. Section V presents numerical results for three model PDEs defined on geometrically regular and irregular domains to illustrate how the proposed algorithms work in practice. Section VI contains our conclusions.

II. Combining Genetic Programming (GP) With Least-Squares Collocation Techniques

We consider well-posed elliptic PDEs of the form

$$\mathcal{L}u(\mathbf{x}) = f(\mathbf{x}) \ \text{ in } \ \Omega \subset \mathbb{R}^d \tag{2}$$

subject to the boundary conditions

$$\mathcal{B}u(\mathbf{x}) = g(\mathbf{x}) \ \text{ on } \ \partial\Omega \tag{3}$$

where $\mathcal{L}$ and $\mathcal{B}$ are differential operators in the space $\mathbf{x} \in \mathbb{R}^d$, and $u(\mathbf{x})$ denotes the field variable. $\Omega \in \mathbb{R}^d$ is a bounded domain and $\partial\Omega$ denotes its boundary. For the case of Dirichlet boundary conditions, $\mathcal{B}$ is the unity operator. When Neumann boundary conditions are imposed, $\mathcal{B} = n.\nabla$, where n is the outward unit vector normal to the boundary $\partial\Omega$.

The field variable $u(\mathbf{x})$ satisfying (2) and (3) can be computed by solving the following constrained variational problem.

Problem 1:

$$\textbf{Minimize}: \quad \int_{\Omega} \left(\mathcal{L}\widehat{u}(\mathbf{x}) - f(\mathbf{x})\right)^2 \mathrm{d}\mathbf{x}$$

$$\textbf{Subject to}: \quad \int_{\partial\Omega} \left(\mathcal{B}\widehat{u}(\mathbf{x}) - g(\mathbf{x})\right)^2 \mathrm{d}\mathbf{x} = 0.$$

The objective and constraint functionals in (P1) are L_2 norms of the domain residual $\mathcal{L}\widehat{u}-f$, and the boundary residual $\mathcal{B}\widehat{u}-g$, respectively. In general, the integrals for the objective and constraint functionals can be evaluated analytically only when the

approximation $\widehat{u}(\mathbf{x})$ and the operators $\mathcal{L}$ and $\mathcal{B}$ have a special structure.[1] In practice, when $\widehat{u}$ is a general nonlinear expression, we have to resort to numerical approximations of the error integrals.

We propose to use GP to compute an approximate symbolic expression for $u(\mathbf{x})$. From the GP implementation point-of-view, the main question is how to assess the fitness of a particular symbolic candidate solution $\widehat{u}(\mathbf{x})$. In other words, we need to attach some figure of merit to $\widehat{u}(\mathbf{x})$ to describe the extent to which it satisfies the governing equations (2) and the boundary conditions (3). This objective function will control the selection process within the evolutionary algorithm.

In order to enable the application of GP to solve the constrained variational problem (P1), let us first define a set of nodes situated within the domain, as well as on the boundary, i.e.,

$$\mathcal{C} = \{(\mathbf{x}_i)|_{i=1,\ldots,n_d} \subset \Omega, (\mathbf{x}_i)|_{i=n_d+1,\ldots,n_d+n_b} \subset \partial\Omega\} \quad (4)$$

where n_d and n_b denote the number of collocation points on the domain and the boundary, respectively. We shall use the symbol n to refer to the total number of collocation points, i.e., $n = n_d + n_b$. The process of representing the domain and the boundary by a cloud of points is graphically illustrated for a domain bounded by a circle in Fig. 5. Several suggestions can be found in the literature on how to select collocation points. In our numerical studies, we have opted for full factorial experimental designs, adapted, where required, for irregular domains. Latin hypercube designs [34] or space-filling maximin distance Latin hypercube designs [35] could also be appealing choices, as they have better projective properties (i.e., they ensure better coverage of the ranges of both variables) than a full factorial design comprising the same number of points.

Using the set of collocation points $\mathcal{C}$, the integrals in problem (P1) can be cast as summations to arrive at the following least-squares collocation problem.

Problem 2:

$$\textbf{Minimize}:$$
$$E_D\left[\widehat{u}(\mathbf{x})\right] = \sum_{i=1}^{n_d} \left[\mathcal{L}\widehat{u}(\mathbf{x}_i) - f(\mathbf{x}_i)\right]^2 \quad (5)$$
$$\textbf{Subject to}:$$
$$E_B\left[\widehat{u}(\mathbf{x})\right] = \sum_{i=n_d+1}^{n} \left[\mathcal{B}\widehat{u}(\mathbf{x}_i) - g(\mathbf{x}_i)\right]^2 = 0. \quad (6)$$

Problem (P2) is amenable to solution using GP by defining a fitness function of the form $E_D[\widehat{u}(\mathbf{x})]+\lambda E_B[\widehat{u}(\mathbf{x})]$, where λ is a penalty parameter. The partial derivatives that arise when computing $\mathcal{L}\widehat{u}(\mathbf{x}_i)$ and $\mathcal{B}\widehat{u}(\mathbf{x}_i)$ may be evaluated in several ways. For example, one could obtain symbolic expressions for $\mathcal{L}\widehat{u}(\mathbf{x})$ and $\mathcal{B}\widehat{u}(\mathbf{x})$ by automatic differentiation ,and then substitute each $\mathbf{x}_i$ as required. In the present work, we opted to calculate the derivatives numerically via finite differencing for the sake of computational simplicity. To ensure accuracy, we use second-order finite-difference approximations.

[1]For example, Howard and Roberts [24] assumed $\widehat{u}$ to be a polynomial and the operators to be linear to allow for the domain and boundary residual error to be computed analytically.

Initial numerical studies suggested that a straightforward application of GP to solve problem (P2) (using the penalty term λ) is fraught with difficulties due to the requirement of searching over a complex and constrained search space. This makes finding a solution that satisfies the boundary conditions inefficient at best and almost intractable at worst. This problem motivated us to employ the formulations developed in [10]–[12] to ensure that the boundary conditions are implicitly satisfied. In a GP context, such approaches for implicitly satisfying the boundary conditions can be interpreted as chromosome repair strategies, which are similar in spirit to those commonly employed in the evolutionary optimization literature for tackling constrained numerical optimization problems; see, for example, Michalewicz [33].

III. Incorporation of Boundary Conditions

For geometrically simple domains, it is possible to modify the expression for the field variables evolved by GP such that the boundary conditions are satisfied by construction. To illustrate this, consider a two-dimensional PDE of the form given in (2) and (3). Let us define the coordinate system by $\mathbf{x} = (x_1, x_2)$ and represent the boundary by the parametric equation $x_2 = r(x_1)$. Further, let us define the Dirichlet boundary operator $\mathcal{B} = 1$. Then, the expression evolved by GP ($u_{GP}(\mathbf{x})$) can be repaired as follows to ensure that the boundary conditions are satisfied by construction:

$$\widehat{u}(\mathbf{x}) = [x_2 - r(x_1)]\, u_{GP}(\mathbf{x}) + g(\mathbf{x}). \quad (7)$$

It can be seen clearly from the preceding expression that since the first term becomes zero on the boundary, the boundary conditions will be exactly satisfied. A similar procedure can be carried out for the case when the domain is a rectangular box. The interested reader may consult [11] for more examples on how to derive expressions which ensure that the boundary conditions are satisfied by construction.

However, for geometrically complex boundaries, this is not always straightforward to do. In order to tackle such problems, we hybridize the GP algorithm with a RBF network. This idea is similar in spirit to that employed by Lagaris *et al.* [12] for solving PDEs on complex geometries using neural networks. Here, the current trial solution (evolved by GP) is repaired by adding a linear combination of RBFs centered around the boundary points in the set $\mathcal{C}$, thus obtaining a new candidate solution of the form

$$\widehat{u}(\mathbf{x}) = u_{\mathrm{GP}}(\mathbf{x}) + u_{\mathrm{RBF}}(\mathbf{x}) \quad (8)$$

where

$$u_{\mathrm{RBF}}(\mathbf{x}) = \sum_{i=1}^{n_b} \alpha_i \phi\left(\|\mathbf{x} - \mathbf{x}_{i+n_d}\|\right) \quad (9)$$

where $\phi(\|\mathbf{x}-\mathbf{c}\|) : \mathbb{R}^d \to \mathbb{R}$ is a RBF with center $\mathbf{c} \in \mathbb{R}^d$ and α_i, $i = 1, 2, \ldots, n_b$ are undetermined coefficients in the RBF approximation.

To ensure that the boundary conditions are satisfied by $\widehat{u}(\mathbf{x})$, we need to enforce $\mathcal{B}\widehat{u}(\mathbf{x}) = g(\mathbf{x})$ at the n_b boundary nodes in the set $\mathcal{C}$, that is

$$\mathcal{B}\left[u_{\mathrm{GP}}(\mathbf{x}_i) + u_{\mathrm{RBF}}(\mathbf{x}_i)\right] = g(\mathbf{x}_i),\ \forall\, i = n_d+1, \ldots, n. \quad (10)$$

Substituting (9) in (10) and assuming the boundary operator $\mathcal{B}$ to be linear, we obtain a system of linear equations for the undetermined coefficients, $\alpha_i, i = 1, 2, \ldots, n_b$ as given below

$$\mathbf{A}\boldsymbol{\alpha} = \mathbf{b} \quad (11)$$

where

$$\mathbf{A} = \begin{bmatrix} \mathcal{B}\phi\left(\|\mathbf{x}_{n_d+1} - \mathbf{x}_{n_d+1}\|\right) & \ldots & \ldots & \mathcal{B}\phi\left(\|\mathbf{x}_{n_d+1} - \mathbf{x}_n\|\right) \\ \mathcal{B}\phi\left(\|\mathbf{x}_{n_d+2} - \mathbf{x}_{n_d+1}\|\right) & \ldots & \ldots & \mathcal{B}\phi\left(\|\mathbf{x}_{n_d+2} - \mathbf{x}_n\|\right) \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \mathcal{B}\phi\left(\|\mathbf{x}_n - \mathbf{x}_{n_d+1}\|\right) & \ldots & \ldots & \mathcal{B}\phi\left(\|\mathbf{x}_n - \mathbf{x}_n\|\right) \end{bmatrix} \in \mathbb{R}^{n_b \times n_b} \quad (12)$$

$$\mathbf{b} = \begin{bmatrix} g\left(\mathbf{x}_{n_d+1}\right) - \mathcal{B}u_{\mathrm{GP}}\left(\mathbf{x}_{n_d+1}\right) \\ g\left(\mathbf{x}_{n_d+2}\right) - \mathcal{B}u_{\mathrm{GP}}\left(\mathbf{x}_{n_d+2}\right) \\ \cdots \\ \cdots \\ g(\mathbf{x}_n) - \mathcal{B}u_{\mathrm{GP}}(\mathbf{x}_n) \end{bmatrix} \in \mathbb{R}^{n_b} \quad (13)$$

and

$$\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_{n_d}\}^T \in \mathbb{R}^{n_b}. \quad (14)$$

After solving the preceding system of equations for α_i, $i = 1, 2, \ldots, n_b$, we can substitute these values into (9). Subsequently, using (8), we obtain a new symbolic expression for $\widehat{u}(\mathbf{x})$. This modified trial solution is now guaranteed to satisfy the boundary conditions and can be inserted into the expression for $E_D[\widehat{u}(\mathbf{x})]$ (see Problem P2), which now becomes the objective function to be minimized during evolutionary search.

To summarize: the goal that drives the evolutionary process in the GP algorithm is *the evolution of solutions that, when modified in a certain way to satisfy the boundary conditions, satisfy the governing equations as accurately as possible*. In other words, the chromosomes evolved by the GP algorithm do not represent the actual candidate solutions—they form the basis upon which solutions that satisfy the boundary conditions can be built.

In the following section, we discuss a strategy for improving the efficiency and the accuracy of this process. Before that, however, a note is due on the convergence of the baseline algorithm described above. The convergence of evolutionary algorithms is notoriously difficult to show and, in all but a few very simple cases, the task of estimating GP solution errors is intractable due to the large variety of possible building blocks and the complex probabilistic mechanics of the operators. Therefore, we rely here on an *a posteriori* safety net instead, which is guaranteed by two assumptions we made at the outset, namely, that the PDE under consideration is linear and well posed.

To illustrate the assumption of well-posedness from a mathematical viewpoint, let us first rewrite the governing PDE as a map of the form $\mathcal{L}u = f$, $L : U \rightarrow F$, where U and F are normed linear spaces which u and f belong to, respectively. The map L takes the solution $u \in U$ to its data $f \in F$. The assumption of well-posedness implies that the solution is *continuously dependent* on the data, i.e., the inverse $\mathcal{L}^{-1}$ is a bounded linear map and an inequality of the following form holds:

$$\|u\|_U \leq C\|\mathcal{L}u\|_F \ \text{ for all } u \in U \quad (15)$$

where C is a positive constant. Under this assumption, the following theorem can be used to justify the application of an algorithm that seeks solutions with small residuals (see, for example, Schaback and Wendland [36]).

Theorem: Let $\mathcal{L}u = f$ be a well-posed linear operator equation in the sense of (15). Then, the following inequality holds:

$$\|u - \widehat{u}\|_U \leq C\,\|\mathcal{L}(u - \widehat{u})\|_F = C\|f - \widehat{f}\|_F \quad (16)$$

where $\widehat{u} \in U$ is an approximate solution and $\widehat{f} = \mathcal{L}\widehat{u}$.

As a consequence of the above result, any numerical technique that produces approximate solutions with small residuals will automatically guarantee small errors in the solution. Note that the constant C is independent of the solution u and, hence, even if this constant is not known, (16) can be used as a safe *a posteriori* error estimate to compare two approximate solutions. In other words, the use of the residual error norm as a fitness function for evolutionary search can be theoretically justified for well-posed linear PDEs.

IV. Boosting Genetic Programming (GP)

The notion of *boosting* originated in the machine learning community and has been successfully applied to design a new class of algorithms for classification and regression [25], [26]. The fundamental idea of boosting is to impose a distribution over the examples in an observational dataset. A sequence of models are then built in an iterative fashion by appropriately modifying this distribution such that more emphasis is laid on the difficult examples. A number of papers which demonstrate how boosting can improve the performance of existing learning algorithms can be found in the literature—see, for example, Mitchell [27]. Interestingly, the notion of boosting has also been applied to enhance the performance of GP for regression and classification problems [28], [29].

In the context of solving PDEs, the gradient descent perspective of boosting ([30], [31]) appears to be more germane. To illustrate this, consider the problem of least-squares regression.

Here, gradient boosting involves fitting a model to data and subsequently fitting another model to the residuals of the first model. This process can be carried out in an iterative fashion to construct a series of models which are then ultimately aggregated to make predictions. A key advantage of this iterative scheme is that it becomes possible to model complex input–output relationships using an ensemble of so-called *weak learners*. Furthermore, since we model data using an ensemble of simple models instead of a single highly parameterized complex model, significant computational cost savings can result in the training phase.

We will now show how the notion of gradient boosting can be used to improve the computational efficiency as well as the accuracy of the proposed approach for solving PDEs. To illustrate

TABLE I
SETTING FOR GP ALGORITHM USED IN THE EXPERIMENTS

TERMINAL SET	x, y, $\mathcal{R}$ (ephemeral random constant).
FUNCTION SET	+, -, *, % (protected division, i.e. if $b = 0$, $a\%b = 1$, otherwise $a\%b = a/b$), sin, cos, exp.
OPERATORS (probability in brackets)	Reproduction (0.1), crossover (0.8), point mutation (0.05), subtree-mutation (0.05).
MISCELLANEOUS	Population size: 1500 Number of generations: 12 (including initial population) Minimum tree length: 7 Maximum tree length: 50 Selection method: tournament (tournament size = 5).

the idea, consider a linear elliptic PDE defined on a geometrically simple domain where the GP solution can be modified to satisfy the boundary conditions by construction. We first find a compact expression for $u^1(\mathbf{x})$ using GP such that $||\mathcal{L}u^1 - f||_2$ is minimized. Subsequently, we keep u^1 fixed and find an expression for $u^2(\mathbf{x})$ using GP such that $||\mathcal{L}u^2 - r^1||_2$ is minimized, where $r^1 = f - \mathcal{L}u^1$ is the residual error. The final approximation can then be written as $\widehat{u} = u^1 + u^2$. This procedure can be carried out in an iterative fashion as shown below.

1) $u^1 \leftarrow \arg\min_u ||\mathcal{L}u - f||$ subject to $||\mathcal{B}u - g|| \leq \epsilon$ on $\partial\Omega$.
2) $r^1 \leftarrow f - \mathcal{L}u^1$.
3) $i = 1$.
4) $u^{i+1} \leftarrow u^i + \arg\min_u ||\mathcal{L}u - r^i||$ subject to $||u|| \leq \epsilon$ on $\partial\Omega$.
5) $r^{i+1} \leftarrow f - \mathcal{L}u^{i+1}$.
6) $i \leftarrow i + 1$.
7) If $||r^i|| \leq \epsilon$ stop. Else go to Step 4.

The iterations can be terminated when a suitable norm of the residual error is reduced to a small value (say ϵ), which is specified *a priori*. Since the models created at each stage i are finally aggregated, each model can be constructed by constraining the GP to search over a low dimensional space of expressions in Step 4. In other words, we seek to model the field variables using an ensemble of simple expressions. This, in turn, leads to significant computational cost savings since the GP algorithm will converge faster due the constraint imposed on the size of the search space at each iteration.

As far as the boundary conditions are concerned, as indicated in the pseudocode, in the case of u^1 these are treated as in the standard GP approach, either by construction or by using RBFs. In subsequent iterations, we need to ensure that the addition of new terms to u^1 will not alter the behavior of the aggregated function on the boundaries of the domain. For example, if the Dirichlet condition $u = g(\mathbf{x})$ is imposed on a boundary, this will have to be satisfied by the expression generated during the first boosting iteration u^1. In the subsequent iterations, we enforce $u = 0$ on the boundary (meaning, in practice, that we make sure that $||u||$ is less than a small, preset value ϵ) in order to ensure that the final aggregated solution satisfies the boundary conditions.

V. EXAMPLES

For the experiments described in this section, we have employed a standard, population-based GP algorithm, the particulars of which are summarized in Table I. We note here that these settings have not resulted from a fine-tuning process, though it has been observed that the performance of the algorithm is fairly robust to changes in most of them. An exception from this rule is the tournament size, which, if set to values greater than about 5, will lead to rapid loss of population diversity, and thus premature convergence to suboptimal solutions. At the other extreme, a value of 2 was found to lead to reduced selective pressure, and thus slow convergence.

First, we assess the performance of the approaches described here on a second-order PDE defined on a rectangular domain. The aim of this example is to demonstrate how the boundary conditions can be dealt with by constructing the trial solutions so that they always satisfy them, irrespective of the structure of the expression evolved by the GP algorithm. The following two PDEs discussed in this section are defined on a domain with a circular boundary and Cassini's Oval, respectively. We use these two examples to illustrate how the GP trial expressions can be repaired by adding a linear combination of a set of RBFs to ensure that the boundary conditions are satisfied. We also present results using the boosting approach described in the previous section. All the results reported here are averaged over 20 independent runs of the GP algorithm, started from different (randomly generated) initial populations.

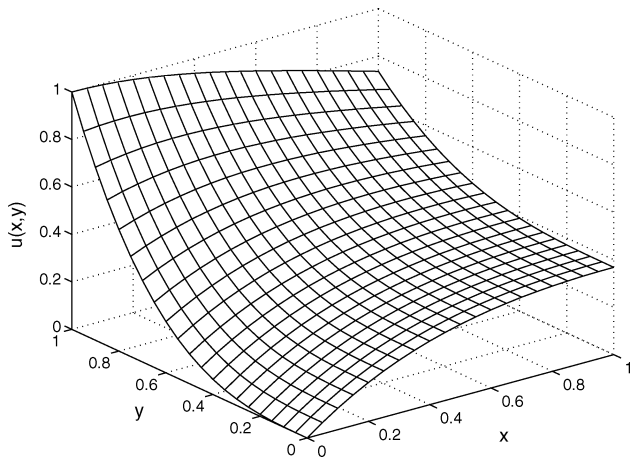

Fig. 1. Exact solution of the PDE discussed in Section V-A.

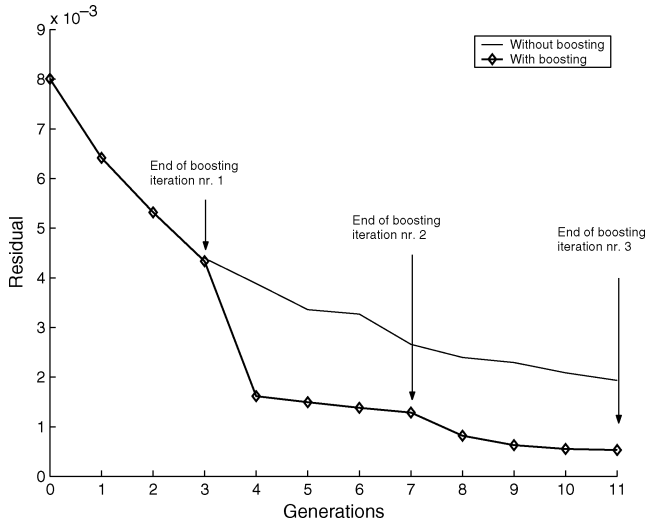

Fig. 2. Averaged GP objective function histories with and without boosting for the rectangular domain test case discussed in Section V-A.

A. Boundary Conditions Imposed by Construction

Let us consider the following Poisson problem (devised by Lagaris *et al.* [11]) defined on $[0,1]\times[0,1]$:

$$\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}=e^{-x}(x-2+y^3+6y) \qquad (17)$$

with the Dirichlet boundary conditions

$$\begin{aligned} u(0,y)&=y^3, \quad u(1,y)=(1+y^3)e^{-1},\\ u(x,0)&=xe^{-x}, \quad u(x,1)=e^{-x}(1+x). \end{aligned} \qquad (18)$$

The exact solution for this problem is $u_e(x,y)=e^{-x}(x+y^3)$, which is shown in Fig. 1.

To ensure that the boundary conditions are satisfied, the expression evolved by GP (u_{GP}) is modified as follows:

$$\begin{aligned} \widehat{u}(x,y)=&(1-x)y^3+x(1+y^3)e^{-1}+(1-y)x(e^{-x}-e^{-1})\\ &+y\left[(1+x)e^{-x}-(1-x+2xe^{-1})\right]+x(1-x)y(1-y)u_{\mathrm{GP}}. \end{aligned} \qquad (19)$$

When assessing the fitness of an individual, i.e., the domain residual error defined in (5), we used $n_d=49$ collocation points arranged on the domain in a seven-level full factorial experimental design. The averaged variation of this residual through the 12-generation run is shown in Fig. 2. This figure also shows the averaged fitness history for the case when the same computational budget has been split into three boosting iterations of three generations each (plus a zeroth generation for evaluating the initial population, randomly generated at the beginning of each iteration).

In order to evaluate the accuracy of the best approximate symbolic solutions $\widehat{u}_b$ resulting from the GP runs, we have computed the mean square errors of these approximations with respect to the exact solution u_e over a space-filling grid of 10 000 points inside the domain. This error value, averaged over the 20 runs, alongside the standard deviation over the 20 run samples is shown in the first row of Table II.

The second row of the table shows the same data for the boosting case: the average mean square error values for $\widehat{u}_b=u_b^1+u_b^2+u_b^3$ and their standard deviation over the 20 run sample.

TABLE II
MEANS (μ_{MSE}) AND STANDARD DEVIATIONS (σ_{MSE}) OF THE SAMPLES COMPRISING THE MEAN SQUARE ERROR VALUES OF THE SOLUTIONS EVOLVED BY THE GP AND BOOSTING APPROACHES

Problem	μ_{MSE}	σ_{MSE}
PDE on rectangular domain (simple GP approach)	7.2241×10^{-7}	5.2986×10^{-7}
PDE on rectangular domain (GP with boosting)	4.8430×10^{-7}	6.9587×10^{-7}
PDE on circular domain (simple GP approach)	5.8305×10^{-4}	1.6758×10^{-3}
PDE on circular domain (GP with boosting)	2.0471×10^{-4}	7.8746×10^{-4}
PDE on Cassini's Oval (simple GP approach)	9.3381×10^{-4}	7.7802×10^{-4}
PDE on Cassini's Oval (GP with boosting)	4.4622×10^{-4}	4.7648×10^{-4}

Fig. 3 shows the spatial distribution of solution error for a typical individual evolved by the GP.

As mentioned earlier, in addition to its performance-enhancing effect, the gradient boosting approach encourages the evolution of compact building blocks, as opposed to the complex, nested structures that often result from the standard version of the technique. This is illustrated in the appendix by examples of typical individuals generated with and without boosting for this and the subsequent test problems.

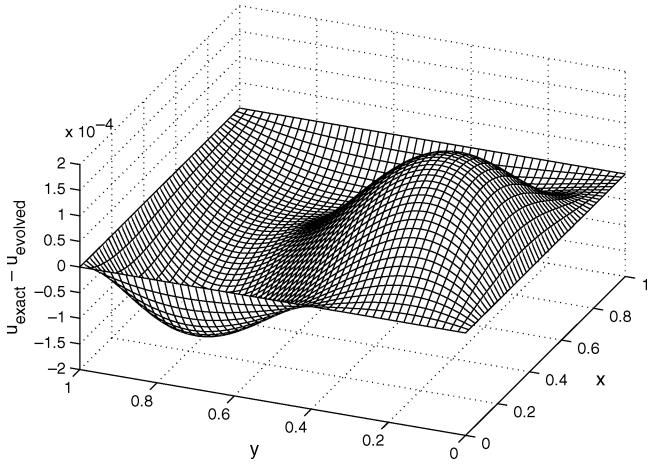

Fig. 3. Deviation from the exact solution for a typical individual generated by the GP algorithm (the problem is described in Section V-A).

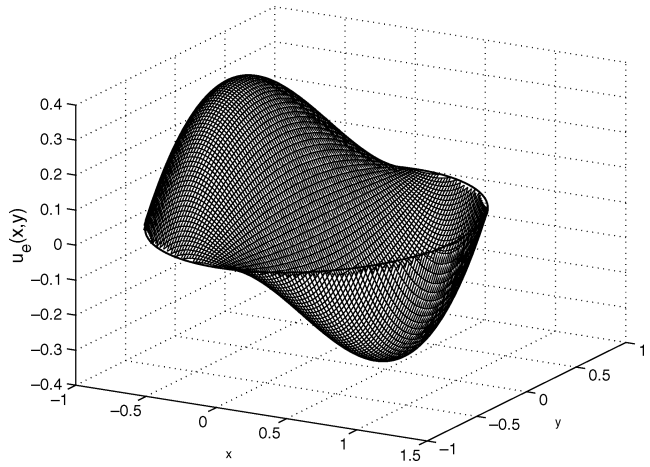

Fig. 4. Exact solution of the PDE discussed in Section V-B1.

B. Boundary Conditions Imposed by RBFs

1) Circular Domain: In order to assess the feasibility of using a set of RBFs as a means of enforcing the boundary conditions for PDEs defined on nonrectangular domains, first we consider the domain delimited by the circular boundary $x^2 + y^2 - 1 = 0$ ($x, y \in [-1, 1]$), along which the Dirichlet condition imposed on the field variable is $u = 0$. The PDE is defined as

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 4x\cos x + (5 - x^2 - y^2)\sin(x). \quad (20)$$

The exact solution, pictured in Fig. 4, is $u_e(x, y) = (x^2 + y^2 - 1)\sin(x)$.

The distribution of collocation points used for calculating the residual error is shown in Fig. 5. In order to ensure that any trial solution satisfies the condition $\widehat{u} = 0$ on the boundary, we added a set of 25 RBFs to every "raw" u_{GP} expression generated by the GP algorithm, before evaluating their fitness. These basis functions are centered around equally spaced boundary points. We

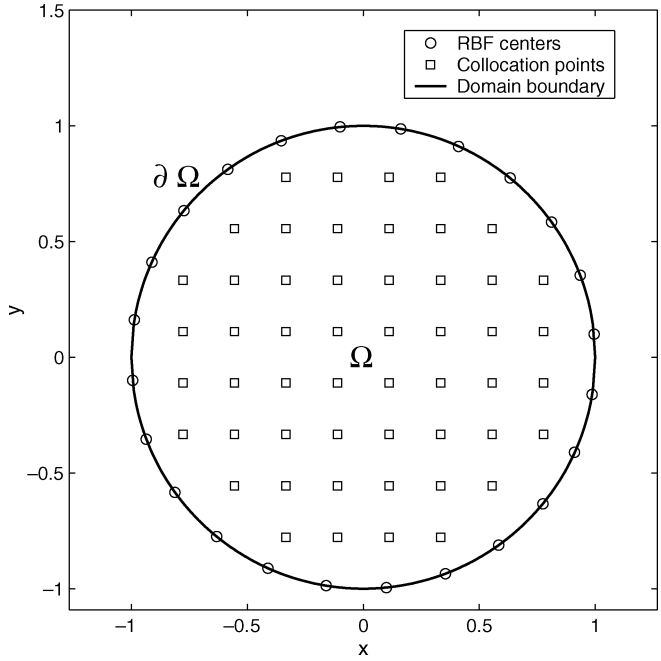

Fig. 5. The 25 RBF centers and 52 collocation points on the domain of the PDE discussed in Section V-B1.

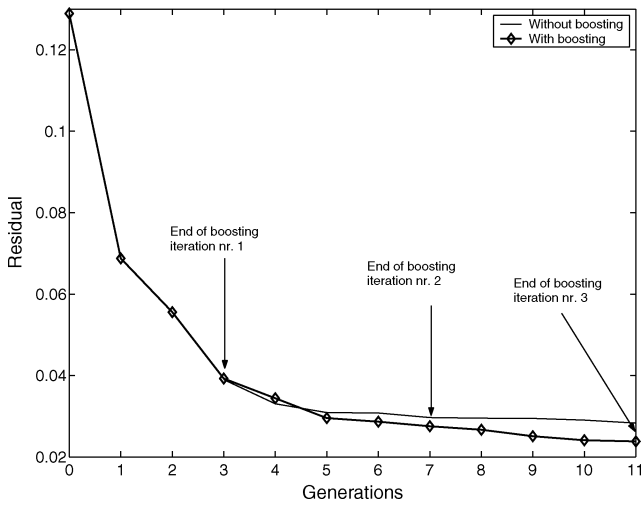

Fig. 6. Averaged GP objective function histories with and without boosting for the circular domain test case discussed in Section V-B1.

chose a Gaussian RBF of the form $\exp(-r^2/\sigma)$, where the hyperparameter σ is chosen to be equal to the distance between two adjacent collocation points on the boundary. Again, a simple full factorial design has been used to place the collocation points (52 in this case, see Fig. 5).

Fig. 6 depicts the averaged convergence trends of the residual error for the standard GP approach and the gradient boosting technique. A summary of the statistics of the solution is given in Table II. It can be seen that for this problem, the accuracy of the gradient boosting is only marginally better than the standard GP. However, the boosting approach results in more compact expressions; see the appendix. The spatial distribution of the error in a typical solution is shown in Fig. 7.

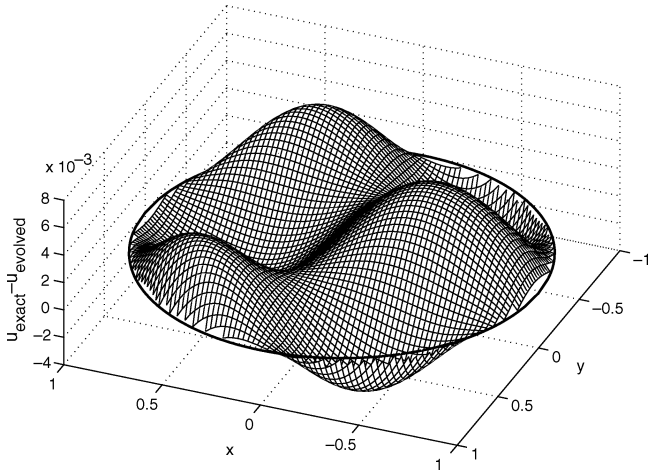

Fig. 7. Deviation from the exact solution for a typical individual generated by the GP algorithm (the problem is described in Section V-B1).

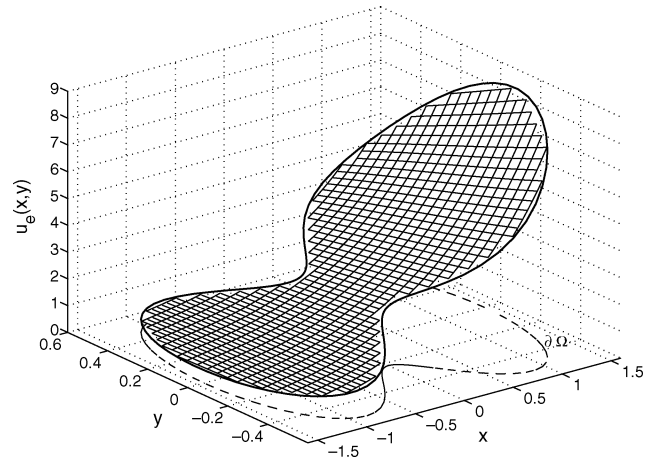

Fig. 8. Exact solution of the PDE discussed in Section V-B2.

2) Cassini's Oval: In this final example, we consider the PDE

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 2e^{x-y} \tag{21}$$

on the domain Ω delimited by Cassini's Oval, which is defined as follows:

$$\partial\Omega = \{R(\theta)(\cos\theta, \sin\theta) : 0 \le \theta \le 2\pi\}$$
$$\text{where} \quad R(\theta) = \sqrt{\cos(2\theta) + \sqrt{1.1 - \sin^2(2\theta)}}. \tag{22}$$

The Dirichlet boundary condition imposed on the boundary defined by (20) is $u = e^{x-y} + e^x \cos y$. The exact solution for this problem $u_e = e^{x-y} + e^x \cos y$ is shown in Fig. 8. The set of collocation points used to calculate the objective function is shown in Fig. 9.

The convergence characteristics and statistics of the solution obtained using the standard GP and the boosting approach for this problem are shown in Fig. 10 and Table II. The spatial distribution of error in a typical solution is shown in Fig. 11. Similar to the earlier examples, we observed that the boosting approach results in more compact expressions for the field variable as compared to the standard GP approach.

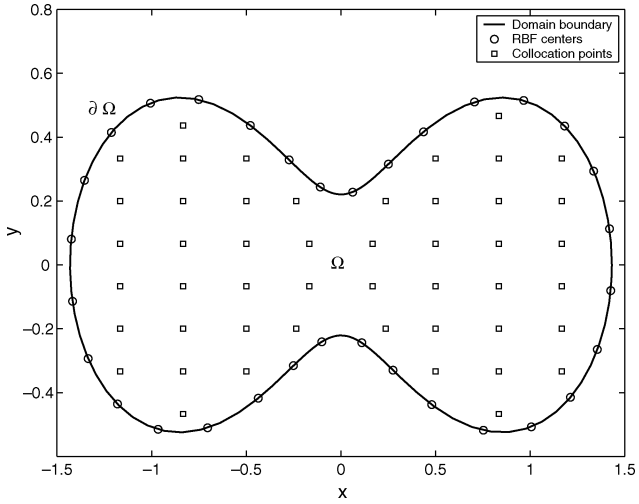

Fig. 9. The 32 RBF centers and 48 collocation points on Cassini's Oval—the domain on which the PDE discussed in Section V-B2 is defined.

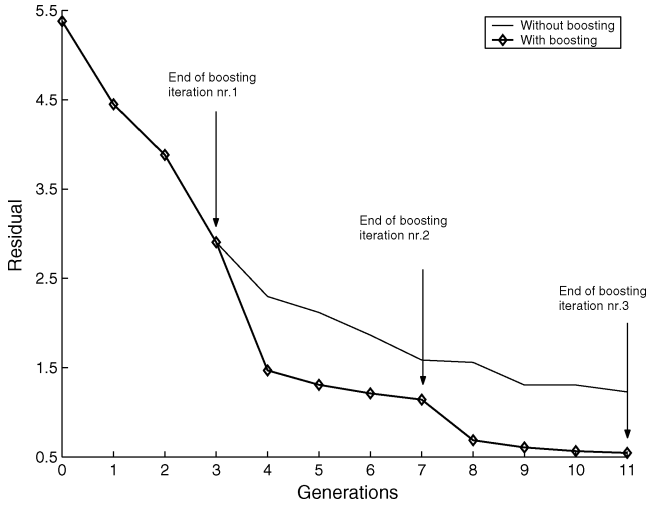

Fig. 10. Averaged GP objective function histories with and without boosting for the Cassini's Oval test case discussed in Section V-B2.

There are, of course, a number of ways in which the error of these approximations can be controlled. In decreasing order of effectiveness the principal means of reducing the error are: a) increasing the number of boosting iterations; b) increasing the number of generations within each boosting iteration; and c) increasing the number of collocation points. In our experience, the average ultimate generalization error estimate (as calculated over the test sets) can be reduced by changing these settings to around $\mu_{\mathrm{MSE}} = 1.5 \times 10^{-4}$, where the standard deviation over a sample of ten runs is around $\sigma_{\mathrm{MSE}} = 2 \times 10^{-4}$. We have also found that increasing the number of boosting iterations beyond 6–7 will not reduce the average ultimate error any further, as the weak learners of higher order will be subjected to

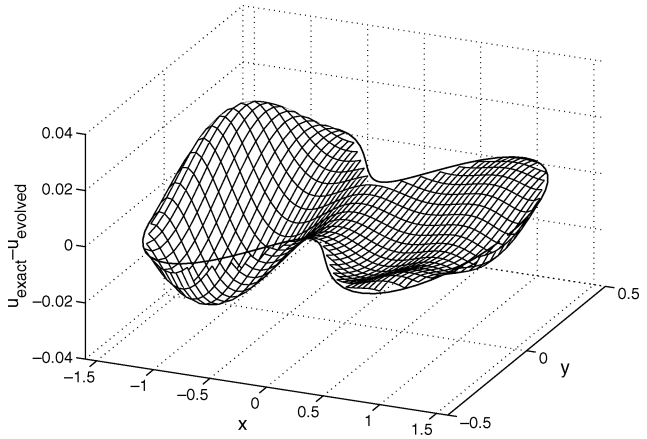

Fig. 11. Deviation from the exact solution for a typical individual generated by the GP algorithm (the problem is described in Section V-B2).

ever-reducing levels of selective pressure due to the very small-scale variations in the residuals that need to be approximated.

VI. Concluding Remarks

We have presented an algorithm based on the GP paradigm for solving elliptic PDEs). The key idea was to employ the least-squares collocation principle to define the fitness function. The boundary conditions are imposed implicitly either by repairing the GP expression using algebraic terms or a radial basis function network. The idea of gradient boosting was subsequently introduced for reducing the computational cost and improving the accuracy of the proposed approach.

Numerical studies have been presented for a number of model operator equations to illustrate the performance of the proposed algorithms. It was shown that the gradient boosting approach gives more accurate results than the standard GP approach. Further, the boosting approach results in more compact expressions for the field variables. The results obtained for the numerical examples suggest that the GP approach holds promise for quickly computing approximate symbolic expressions for the solution of PDEs.

In general, the accuracy achievable with the approach described here is inferior to that of finite difference or finite element methods. Nevertheless, the technique has a number of attractive features. Most importantly, its setup time is much shorter than that of any mesh-based technique. Further, it is easy to parallelize and a symbolic solution is obtained instead of a numerical one, which leads to significantly lower memory requirements in comparison to mesh-based methods. When compared with the neural network approach referred to earlier in the paper, the GP-based technique described here has the advantage of offering compact, relatively simple symbolic approximations, often simple enough to be incorporated into any subsequent analytical calculations. The disadvantage is that the accuracy that can be achieved is inferior to that of the neural network—moreover, the latter offers the possibility of a more precise control of the accuracy.

There are a number of potential avenues for future work in this area. For example, one might consider the introduction of bouts of Lamarckian learning to some individuals in the population, in order to improve the accuracy of the trial solutions using some local search heuristic. Additionally, the feasibility of the approach could be tested on more difficult, high-dimensional problems with boundaries of complicated shapes. Finally, further research could investigate the possibility of developing *a priori* error estimates, as well as theoretical proofs of convergence.

Appendix

Examples of solutions to the three test problems, generated with and without boosting, are shown below.

Test Problem A: Typical individual generated *without* boosting

$$\hat{u}=y\left(\cos\left(\sin\left(2y\cos\left(\frac{1}{5}\frac{y\cos\left(1/8\cos(xy)yx+3/8\right)}{x}\right)\right)\right)+3\right)^{-1}+\frac{3}{8}\cos(x).$$

Typical individual generated *with* boosting

$$\hat{u}_b = u_b^1 + u_b^2 + u_b^3$$

where

$$u_b^1 = \frac{6}{25}+\frac{2}{5}y, \quad u_b^2 = \frac{1-x}{8}, \quad u_b^3 = \frac{y}{-15+5x}.$$

Test Problem B.1: Typical individual generated *without* boosting

$$\hat{u}=\frac{55}{24}x+\frac{1}{24}-\frac{1}{24}\cos\left(3-x-\sin\left(-\cos\frac{13}{33}+x\right)\right)+u_{\mathrm{RBF}}.$$

Typical individual generated *with* boosting

$$\hat{u}_b = u_b^1 + u_b^2 + u_b^3 + u_{\mathrm{RBF}}$$

where

$$u_b^1 = \frac{9}{4}x, \quad u_b^2 = \frac{1}{7}\exp\left(x\cos(x)\right), \quad u_b^3 = \frac{\sin x - x}{y+7}.$$

Test Problem B.2: Typical individual generated *without* boosting

$$\hat{u} = \exp x - y + x + \exp(\sin 1) - 1 + \cos\left(\exp(\sin 1)+\frac{4}{7}+x\right)+u_{\mathrm{RBF}}.$$

Typical individual generated *with* boosting

$$\hat{u}_b = u_b^1 + u_b^2 + u_b^3 + u_{\mathrm{RBF}}$$

where

$$u_b^1 = 6+3x-2y, \quad u_b^2 = x(2y-1), \quad u_b^3 = 5\cos y.$$

References

[1] E. J. Kansa, "Multiquadrics—A scattered data approximation scheme with applications to computational fluid dynamics: I. Surface approximations and partial derivative estimates," *Comput. Math. Applicat.*, vol. 19, no. 6–8, pp. 127–145, 1990.

[2] E. J. Kansa, "Multiquadrics—A scattered data approximation scheme with applications to computational fluid dynamics: II. Solutions to parabolic, hyperbolic, and elliptic partial differential equations," *Comput. Math. Applicat.*, vol. 19, no. 6–8, pp. 147–161, 1990.

[3] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, "Meshless methods: An overview and recent developments," *Comput. Methods Appl. Mech. Eng.*, vol. 139, pp. 3–47, 1996.

[4] S. Li and W. K. Liu, "Meshfree and particle methods and their applications," *Appl. Mech. Rev.*, vol. 55, pp. 1–34, 2002.

[5] G. E. Fasshauer, "Solving partial differential equations by collocation with radial basis functions," in *Surface Fitting and Multiresolution Methods*, A. LeMhaut, C. Rabut, and L. Schumaker, Eds. Nashville, TN: Vanderbilt Univ. Press, 1997.

[6] C. Franke and R. Schaback, "Solving partial differential equations by collocation using radial basis functions," *Appl. Math. Comput.*, vol. 93, pp. 73–83, 1998.

[7] G. E. Fasshauer, "Solving differential equations with radial basis functions: Multilevel methods and smoothing," *Advances in Comput. Math.*, vol. 11, no. 2–3, pp. 139–159, 1999.

[8] H. Wendland, "Meshless Galerkin methods using radial basis functions," *Math. Comput.*, vol. 68, no. 228, pp. 1521–1531, 1999.

[9] B. P. Van Milligen, V. Tribaldos, and J. A. Jimenez, "Neural differential equation and plasma equilibrium solver," *Phys. Rev. Lett.*, vol. 75, no. 20, pp. 3594–3597, 1995.

[10] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural network methods in quantum mechanics," *Comput. Phy. Commun.*, vol. 104, pp. 1–14, 1997.

[11] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[12] I. E. Lagaris, A. Likas, and D. G. Papageorgiou, "Neural networks methods for boundary value problems with irregular boundaries," *IEEE Trans. Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.

[13] M. Quito, Jr., C. Monterola, and C. Saloma, "Solving n-body problems with neural networks," *Phys. Rev. Lett.*, vol. 86, no. 21, pp. 4741–4743.

[14] C. Montelora and C. Saloma, "Solving the nonlinear schrodinger equation with an unsupervised neural network: Estimation of error in solution," *Opt. Commun.*, vol. 222, no. 1–6, pp. 331–339, 2003.

[15] E. A. Galperin and E. J. Kansa, "Application of global optimization and radial basis functions to numerical solutions of weakly singular volterra integral equations," *Comput. Math. Applicat.*, vol. 43, no. 3–5, pp. 491–499, 2002.

[16] J. R. Koza, "Genetically breeding populations of computer programs to solve problems in artificial intelligence," in *Proc. 2nd Int. Conf. Tools for AI*, Herndon, VA, 1990, pp. 819–827.

[17] J. R. Koza, *Genetic Programming—On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[18] A. Salhi, H. Glaser, and D. D. Roure, "Parallel implementation of a genetic-programming based tool for symbolic regression," *Inf. Process. Lett.*, vol. 66, pp. 299–307, 1998.

[19] M. Keijzer and V. Babovic, "Genetic programming, ensemble methods and the bias/variance tradeoff—Introductory investigations," *Lecture Notes in Computer Science*, vol. 1802, pp. 76–90, 2000.

[20] L. Sanchez, "Interval-valued GA-P algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 64–72, 2000.

[21] N. Nikolaev and H. Iba, "Regularization approach to inductive genetic programming," *IEEE Trans. Evol. Comput.*, vol. 5, no. 4, pp. 359–375, 2001.

[22] N. Nikolaev and H. Iba, "Learning polynomial feedforward neural networks by genetic programming and backpropagation," *IEEE Trans. Neural Networks*, vol. 14, no. 2, pp. 337–350, 2003.

[23] E. Sakamoto and H. Iba, "Evolutionary inference of a biological network as differential equations by genetic programming," *Genome Informatics*, vol. 12, pp. 276–277, 2001.

[24] D. Howard and S. C. Roberts, "Genetic programming solution of the convection-diffusion equation," in *Proc. Genetic Evol. Comput. Conf. (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., 2001, pp. 34–41.

[25] Y. Freund, "Boosting a weak learning algorithm by majority," *Inf. Comput.*, vol. 121, no. 2, pp. 256–285, 1995.

[26] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[27] T. M. Mitchell, *Machine Learning*. : McGraw-Hill, 1997.

[28] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proc. Genetic Evol. Comput. Conf. GECCO-99*, 1999, pp. 1053–1060.

[29] G. Paris, D. Robilliard, and C. Fonlupt, "Applying boosting techniques to genetic programming," in *Proc. 6th Int. Conf. Artif. Evol.*, Le Creusot, France, 2001, pp. 315–326.

[30] J. H. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussion)," *Annals of Statistics*, vol. 28, pp. 337–407, 2000.

[31] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[32] J. He, J. Xu, and X. Yao, "Solving equations by hybrid evolutionary computation techniques," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 2959–304, 2000.

[33] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[34] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 1, pp. 23–245, 1979.

[35] M. D. Morris and T. J. Mitchell, "Exploratory designs for computer experiments," *J. Statist. Planning and Inference*, vol. 43, pp. 381–402, 1995.

[36] R. Schaback and H. Wendland, "Kernel techniques: From machine learning to meshless methods," *Acta Numerica*, pp. 1–97, 2006.

András Sóbester holds a B.Eng. degree in mechanical engineering from the Technical University of Cluj, Romania, the B.Sc. degree in design and manufacture from the University of Central Lancashire, Lancashire, U.K., and the Ph.D. degree in aerospace engineering from the University of Southampton, Hampshire, U.K.

After receiving the Ph.D. degree, he worked in the Computational Engineering and Design Group of the University of Southampton School of Engineering Sciences, first as a Research Fellow and then as a Lecturer. Since August 2007, he holds a Royal Academy of Engineering Research Fellowship.

His research interests include aircraft design, aerodynamic shape optimization, shape parameterization, design technologies (including CAD), unmanned air vehicles, and the use of expert systems in conceptual design.

Prasanth B. Nair holds a B.Tech. and M.Tech. degree in aerospace engineering from the Indian Institute of Technology, Mumbai, in 1995 and 1997, respectively. He then joined the Computational Engineering and Design Center, University of Southampton, Hampshire, U.K., where he received the Ph.D. degree in 2000.

He is currently a Senior Lecturer in the School of Engineering Sciences at the University of Southampton. He is the coauthor of the book *Computational Approaches for Aerospace Design* (Wiley, 2005) and he has coauthored over 70 papers in refereed journals, edited books, and conference proceedings. He serves on the editorial board of the *International Journal of Reliability and Safety*. His current research focuses on computational methods for analysis of deterministic and stochastic systems governed by partial differential equations, with applications to design and control.

Andy J. Keane holds a Ph.D. degree from Brunel University, Uxbridge, U.K., after having worked in the concept design division of the U.K. Ministry of Defence Warship Design Agency.

He then moved to Oxford, where as a Fellow of Pembroke College, he lectured on structural dynamics. He has held his current appointment as a Professor at the University of Southampton since 1996, chairing the Computational Engineering and Design Group (now numbering over 60 staff and Ph.D. students). He is the coauthor of the book *Computational Approaches for Aerospace Design* (Wiley, 2005) and also coauthor of over 200 papers in refereed journals, edited books, and conference proceedings.

Dr. Keane is a Fellow of the Institution of Mechanical Engineers.