Unsupervised neural networks for solving Troesch's problem

Muhammad Asif Zahoor Raja†

Department of Electrical Engineering, COMSATS Institute of Information Technology, Attock Campus, Attock, Pakistan

(Received 23 February 2013; revised manuscript received 18 May 2013; published 3 December 2013)

In this study, stochastic computational intelligence techniques are presented for the solution of Troesch's boundary value problem. The proposed stochastic solvers use the competency of a feed-forward artificial neural network for mathematical modeling of the problem in an unsupervised manner, whereas the learning of unknown parameters is made with local and global optimization methods as well as their combinations. Genetic algorithm (GA) and pattern search (PS) techniques are used as the global search methods and the interior point method (IPM) is used for an efficient local search. The combination of techniques like GA hybridized with IPM (GA-IPM) and PS hybridized with IPM (PS-IPM) are also applied to solve different forms of the equation. A comparison of the proposed results obtained from GA, PS, IPM, PS-IPM and GA-IPM has been made with the standard solutions including well known analytic techniques of the Adomian decomposition method, the variational iterational method and the homotopy perturbation method. The reliability and effectiveness of the proposed schemes, in term of accuracy and convergence, are evaluated from the results of statistical analysis based on sufficiently large independent runs.

Keywords: Troesch's problem, artificial neural network, genetic algorithm, hybrid methods

PACS: 89.20.–a, 89.20.Ff, 89.20.Kk **DOI:** 10.1088/1674-1056/23/1/018903

1. Introduction

Troesch's problem is an inherently unstable boundary value problem of the non-linear ordinary differential equation. Its broad description was provided by Weibel.[1] It originates from a system of non-linear ordinary differential equations which arise in the study of the confinement of a plasma column by a radiation pressure and also in the theory of gas porous electrodes.[2,3] This problem has been investigated extensively by many researchers: the numerical solution of this problem was provided by Troesch using the shooting method (SM);[4] by Deeba *et al.* using the Adomian decomposition method (ADM);[5] by Roberts *et al.* using a combination of techniques including multipoint, continuation and perturbation;[6] by Khuri using a combination of a Laplace transform with a modified decomposition technique;[7] by Chin *et al.* with the hybrid asymptotic finite-element method;[8] by Momani *et al.* using the variational iteration method (VIM);[9] and by Feng *et al.* using the modified homotopy perturbation technique.[10] Recently, Chang has provided a solution using the differential transferred method and the variable transformation combined with VIM and SM.[11–13] Besides this, there is a wide range of articles in which researchers addressed Troesch's problem.[14–19]

Through reviewing the literature, it can be seen that all the methods applied to the equation are based on techniques other than the stochastic solvers. Therefore, we are motivated to apply artificial intelligence techniques to determine the solution of this boundary value problem. It is well known that such solvers, based on artificial neural network (ANN) mathematical modeling hybridized with global and local search methodologies, have a universal capability to solve a variety of problems associated with linear and non-linear differential equations.[20–24] For example, these solvers provide viable solutions for nonlinear Van der Pol oscillators,[25,26] nonlinear singular systems associated with the Emden–Fowler type equations,[27] the moveable singularity problem of Painlevé equation I,[28] nonlinear Schrodinger equations,[29] fluid flow problems,[30] nonlinear Bratu type equations arising in the fuel ignition model of combustion theory[31,32] and a transformed form of Troesch's boundary value problem.[33] A good source of reference in this regard is a survey article,[34] which describes the history, recent trend and future prospects of research in this field. Recently, these methodologies are extended to solve even the linear and non-linear differential equations of fractional order.[35,36] Moreover, a special kind of fractional order systems in applied sciences and engineering provided with the help of Riccati and Bagley–Torvik equations have also been solved by the said methodology.[37–39]

In this paper, a numerical treatment on Troesch's problem is presented using stochastic solvers. It is sensible to introduce Troesch's problem before providing the details of the proposed methodology; the governing equation is given by

$$\frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2} = \mu \sinh(\mu y(t)), \quad t \in (0,\ 1), \tag{1}$$

with boundary conditions given as

$$y(0) = 0, \quad y(1) = 1, \tag{2}$$

†Corresponding author. E-mail: Muhammad.asif@ciit-attock.edu.pk, rasifzahoor@yahoo.com

© 2014 Chinese Physical Society and IOP Publishing Ltd

where μ is a constant with $\mu > 0$.

In the proposed scheme, the ANN is used to formulate the mathematical model of the equation by defining an unsupervised error. The reduction of the error is subject to the availability of appropriate weights of the ANNs. To find the most suitable weights of these networks, global and local search techniques as well as their combinations are used. An evolutionary computation technique based on genetic algorithms and a pattern search (PS) technique are used as the global optimization method, while an interior point method (IPM) is used for an efficient local search. The hybridized approaches based on the combination of GA or pattern search techniques with IPM (GA-IPM and PS-IPM) are also used to solve the problem. Moreover, the reliability and effectiveness of these designed stochastic schemes are validated through comprehensive statistical analysis. In addition to that, a comparison of results obtained by these stochastic solvers is also made with the standard deterministic numerical techniques.

2. Mathematical modeling

In this section, the details of the mathematical model for Troesch's problem, with the help of a feed-forward ANN are presented. The formulation of the objective function is also described.

It is well known that an arbitrary continuous function and its derivatives on a compact set can be arbitrarily approximated by multiple inputs, single output and single hidden layer feed-forward ANNs with a linear output layer having no bias. By using the ANNs, the solution y of a generic ordinary differential equation (ODE) can be approximated by the following relation:[40,41]

$$\hat{y}(t) = \sum_{i=1}^{k} \alpha_i f(w_i t + \beta_i), \tag{3}$$

and the n order derivative $\mathrm{d}^n y/\mathrm{d}t^n$ of y is written as

$$\frac{\mathrm{d}^n \hat{y}}{\mathrm{d}t^n} = \sum_{i=1}^{k} \alpha_i \frac{\mathrm{d}^n}{\mathrm{d}t^n} f(w_i t + \beta_i), \tag{4}$$

where α_i, w_i and β_i are bounded real-valued adaptive parameters, m is the number of neurons and f is the activation function, normally taken as a log sigmoid function, given as

$$f(x) = 1/(1 + \mathrm{e}^{-x}). \tag{5}$$

The mathematical model for Eq. (1) can be formulated by a linear combination of networks, (3) and (4) with $n = 2$, called a differential equation neural network (DENN). The architecture of the DENN for Troesch's problem is shown in Fig. 1.

The objective function is formulated by defining an unsupervised error function of DENN for the equation and its associated boundary conditions. It is written as

$$e_j = e_1 + e_2|_j, \quad j = 1, 2, 3, \ldots, \tag{6}$$

where j is the cycle index. The e_1 is the error induced due to terms involved in the equation and is given as

$$e_1 = \frac{1}{N+1} \sum_{i=0}^{N} \left(\frac{\mathrm{d}^2 \hat{y}_m}{\mathrm{d}t_m^2} - \mu \sinh(\mu \hat{y}_m) \right)^2,$$
$$N = 1/h, \quad \hat{y}_m = \hat{y}(t_m), \quad t_m = mh, \tag{7}$$

where interval $t \in (0, 1)$ is divided into N steps ($t_0 = 0$, t_1, t_2, $\ldots$, $t_N = 1$) with step size h; y and $\mathrm{d}^2 y/\mathrm{d}t^2$ are the DENNs given by Eqs. (3) and (4) by using $n = 2$, respectively. Similarly, error e_2, due to the terms used in the boundary conditions as given in Eq. (2), can be written as

$$e_2 = \frac{1}{2} \left((\hat{y}_0)^2 + (\hat{y}_N - 1)^2 \right). \tag{8}$$

Now our intention is to find out appropriate weights for the DENN, such that the value of fitness function e_j is minimum. In such a way, with the weights for which the fitness value of function e_j approaches zero, the solution y of Troesch's problem is accurately approximated by the DENN solution $\hat{y}$.

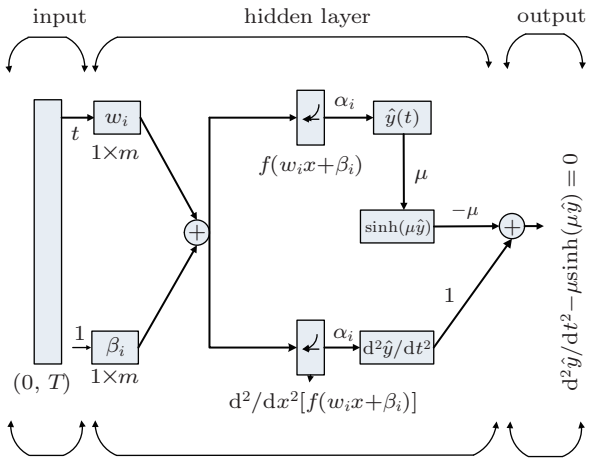


Fig. 1. DENN architecture for Troesch's problem.

3. Learning methodologies

In this section, the methods used for training weights for the DENN are introduced briefly. The detailed flow of algorithms, as well as important parameter settings for their execution, is also provided.

Interior point algorithms, also referred to as barrier methods, are a class of techniques used to solve linear and nonlinear convex optimization problems. These algorithms were developed by Narendra Karmarkar in 1984 for linear programming based on Karmarkar's algorithm.[42] The IPM can solve constraint optimization problems. The basic elements of the scheme consist of a self-concordant barrier function used to encode the convex set. Contrary to the simplex method, it

reaches an optimal solution by traversing the interior of the feasible region.[43] To solve the approximate problem, the algorithm uses iteratively either the Newton step via a linear programming technique or the conjugate gradient step using a trust region.[44]

The name, pattern search, was first introduced by Hooke and Jeeves,[45] however, the convergent pattern search method was developed using the theory of positive bases by Yu.[46] A pattern search algorithm is a kind of aggressive optimization technique and belongs to the class of direct search method or derivative free techniques.[47] Hence it can be applied to optimization problems that are not continuous and where the gradient information is not already available. It can find the optimum values by using a stochastic searching technology based on a scaled and translated integer lattice. The PS techniques have been extensively used by the researchers for optimization problems like bound constrained minimization, linearly constrained minimization and globally convergent augmented Lagrangian algorithms.[48,49]

A genetic algorithm (GA) is a kind of global optimization tool based on natural selection and genetic mechanisms. One of the prominent features of GAs is that they do not get stuck in local minima. GAs use parallel procedures as well as structured strategies for randomly searching high aptitude points. Generally, GAs have three fundamental operators: selection, crossover and mutation. GAs encode the design parameters into a finite bit string to solve the required optimization problem. A GA runs iteratively using its operators randomly based upon some fitness functions. Finally, it finds and decodes the solution from the last pool of mature strings obtained by ranking strings, exchanging the portions of strings and changing some bits of the strings. GAs have been developed for the solution of many non-linear problems in engineering and applied sciences, such as the multi-user cognitive radio network resource allocation system,[50–52] an ameliorative method for two-dimensional Poisson equations,[53] an algorithm to retrieve ocean atmosphere ducts,[54] the wireless sensor network[55] and the optimization of double-screen frequency selective surface structures.[56]

Our intention is to use GA, PS and IPM techniques, as well as hybrid approaches GA-IPM and PS-IPM, for learning weights for the DENN. The MATLAB genetic algorithm and direct search toolbox are used for GA and PS with the parameter settings provided in Tables 1 and 2, respectively. The MATLAB built in function for constrained optimization FMINCON is used for IPM with the parameter setting provided in Table 2. Moreover, the generic flow diagrams of the hybrid approaches based on the GA supported with IPM (GA-IPM) and the PS aided with IPM (PS-IPM) are shown in Fig. 2.

Table 1. Parameters setting for GA algorithms.

Parameter	Setting	Parameter	Setting
Population creation	constrained dependant	Population size	40
Scaling function	rank	Chromosome size	30
Selection function	stochastic uniform	Generation	1000
Crossover fraction	0.80	Function tolerance	10^{-15}
Crossover function	scattering	StallGenLimit	500
Mutation function	constrained dependant	Bounds (lower, upper)	$(-10, 10)_{1\times 30}$
Elite count	2	Migration direction	forward
Initial penalty	10	Migration interval	20
Penalty factor	100	Nonlinear constrain tolerance	10^{-15}
Migration fraction	0.2	Fitness limit	10^{-12}

Table 2. Parameters setting for IPM and PS algorithms.

IPM		PS	
Parameter	Setting	Parameter	Setting
Start point	randomly from $(-1, 1)$	Start point	randomly from $(-1, 1)$
Derivative	solver approximate	Polling order	consecutive
Sub-problem algorithm	IDI factorization	Mesh accelerator	off
Scaling	objective and constraints	Mesh rotae/scale	on
Maximum perturbation	0.1	Poll method	GPS positive basis 2N
Finite difference types	forward differences	Mesh expansion factor	2.0
Hessian	BFGS	Mesh contraction factor	0.5
Minimum perturbation	10^{-8}	Maximum iterations	1000
X tolerance	10^{-10}	Maximum function evolutions	100000
Nonlinear constraint tolerance	10^{-15}	Nonlinear constraint tolerance	10^{-15}
Fitness limit	10^{-12}	Fitness limit	10^{-12}

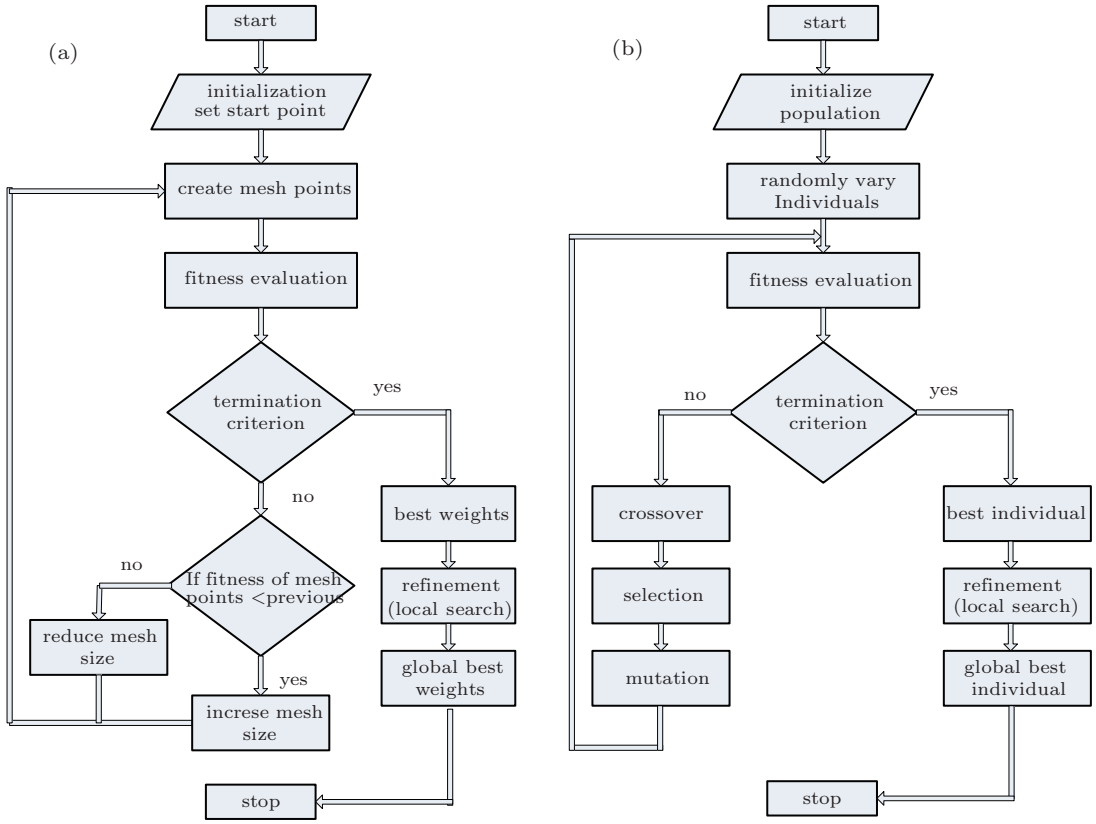


Fig. 2. Generic flow charts of the hybrid algorithms (a) PS-IPM and (b) GA-IPM.

The GA-IPM scheme consists of the following steps.

Step 1: **Initialization**: the initial population is generated randomly with bounded real values represented as chromosomes or individuals. The number of elements in each individual is equal to that of unknown weights in the DENN. Enough spread is made in the initial population for a better search space for the algorithm. Initialize parameters before executing the algorithm as given in Table 1.

Step 2: **Evaluate fitness**: calculate the fitness value for each individual of the population using Eqs. (6)–(8).

Step 3: **Termination criteria**: terminate the algorithm when either of the following criteria matches: the predefined fitness value for function $e_j \leq 10^{-12}$ is achieved; the number of generations is complete. If the termination criterion meets, then go to step 6.

Step 4: **Reproduction**: create the next generation on the basis of (i) crossover: call for scattered function; (ii) mutation: call for adaptive feasible function; and (iii) selection: call for stochastic uniform function and elitism etc.

Step 5: **Improvement**: repeat the procedure from step 2 to step 4 with the newly generated population for the improvement of results until the total number of cycles is complete.

Step 6: **Refinement**: the interior point method is used for further refinement of the results by taking the best individual of GAs as the starting point with the parameter settings provided in Table 2. The MATLAB optimization tool box is used for the interior point method. Store also the refined global best individual.

Step 7: **Statistical analysis**: repeat step 1 to step 6 for sufficiently large numbers to generate enough data for statistical analysis of the algorithm. Store these results in memory for detail analysis later.

Moreover, the step wise description of PS-IPM is given below.

Step 1: **Initialization**: the initial weight vector, i.e., the start point, is generated randomly with bounded real values with the number of elements equal to that of unknown weights in the DENN. Initialize the parameters before the execution of the PS algorithm as given in Table 2.

Step 2: **Mesh point creation:** the PS algorithm starts by establishing a set of points called the mesh around the given point. This current point could be the initial starting point of step 1 or could be computed from the previous step of the algorithm.

Step 3: **Evaluate fitness**: calculate the fitness value for each mesh point using Eqs. (6)–(8).

Step 4: **Termination criteria**: terminate the iterative process if any of the following criteria is fulfilled: the prede-

fined fitness value $e_j \leq 10^{-12}$ is achieved, the total number of iterations are executed. If a termination criterion is met, then go to step 6.

Step 5: **Updating poll and mesh**: based on the value of fitness, the mesh points update in a feasible direction using (i) poll: generalized pattern search (GPS) positive basis 2N method with consecutive order and (ii) mesh: expansion and contraction factors as per the setting listed in Table 2. Go to step 2.

Step 6: **Refinement**: the IPM with the setting given in Table 2 is used for further fine tuning the results by taking the best weights of PS as the initial weight vector. Store the refined final weights.

Step 7: **Statistical analysis**: repeat step 1 to step 6 for sufficiently large numbers to generate enough data for a reliable statistical analysis of the algorithm.

4. Numerical experiments

In this section, simulations have been carried out for the solution of Troesch's problem using the proposed schemes. The reported results provided by ADM,[5] VIM,[9] and HPM[10] methods are also presented along with the obtained solutions. Beside this, the numerical results for this problem are also determined using MATHEMATICA, which has an inbuilt capability to solve this problem. A comparative analysis of the results is also made.

4.1. Solution of Troesch's problem by taking $\mu = 1$

We have modeled the problem using a DENN represented by Eqs. (3) and (4) with 10 neurons, which gives 30 unknown parameters or weights to be adapted. These adaptive weights are restricted to real numbers between -10 to 10. The input of the training set is divided into $N = 100$ equal steps in $t \in (0, 1)$ with a step size $h = 0.01$. Therefore, the fitness function is formulated as

$$e_j = \frac{1}{101}\sum_{m=0}^{100}\left(\frac{\mathrm{d}^2\hat{y}_m}{\mathrm{d}t_m^2} - \sinh(\hat{y}_m)\right)^2 + \frac{1}{2}\left([\hat{y}(0)]^2 + [\hat{y}(1)-1]^2\right)\Big|_j, \quad j = 1,2,3,\ldots \quad (9)$$

Similarly, the fitness function for $N = 10$ equal steps in interval $t \in (0, 1)$ with step $h = 0.1$ is given as:

$$e_j = \frac{1}{11}\sum_{m=0}^{10}\left(\frac{\mathrm{d}^2\hat{y}_m}{\mathrm{d}t_m^2} - \sinh(\hat{y}_m)\right)^2 + \frac{1}{2}\left([\hat{y}(0)]^2 + [\hat{y}(1)-1]^2\right)\Big|_j, \quad j = 1,2,3,\ldots \quad (10)$$

Now, we have to find appropriate weights for the DENN, such that the fitness function e_j given in Eqs. (9) and (10) is minimized. We have applied the GA, PS, IPM, PS-IPM and GA-IMP approaches for training the weighs of the optimization problem. The parameters settings used by these algorithms are provided in Tables 1 and 2. The schemes run iteratively for training the weights of the DENN and are terminated when a predefined number of cycles is completed or the fitness value $e_j \leq 10^{-12}$ is achieved. One set of weights using the fitness function given in Eqs. (9) and (10) for the GA, PS, IPM, PS-IPM and GA-IPM algorithms is plotted in Fig. 3. In Fig. 3(a), the weights are provided with the values of fitness function 4.9458×10^{-4}, 4.6645×10^{-6}, 5.1853×10^{-10}, 2.3101×10^{-10} and 3.2878×10^{-11} for the GA, PS, IPM, PS-IPM and GA-IPM, respectively, while the weights with the fitness values of 1.7142×10^{-3}, 7.0521×10^{-6}, 1.2279×10^{-9}, 1.4914×10^{-10} and 6.9958×10^{-11} for the GA, PS, IPM, PS-IPM and GA-IPM, respectively, are plotted in Fig. 3(b). The weights given in Fig. 3 can be used in Eq. (3) to determine the solution of Troesch's problem for any input in interval (0, 1).

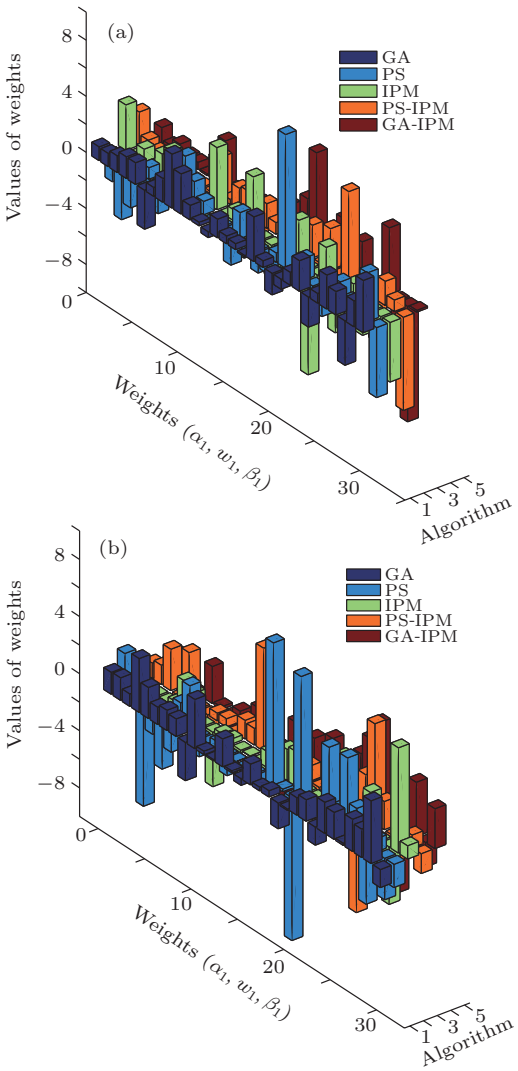


Fig. 3. (color online) Sets of weights learned for the DENN by stochastic solvers for (a) $h = 0.01$ and (b) $h = 0.1$.

Table 3. Solution of Troesch's problem obtained using different step sizes for $\mu = 1$.

h	t	y_{MATH}	y_{GA}	y_{PS}	y_{IPM}	$y_{\mathrm{PS-IPM}}$	$y_{\mathrm{GA-IPM}}$
0.01	0.0	0.0846612485	0.0012751289	–0.0000789341	–0.0000000273	0.0000000024	0.0000000010
	0.2	0.2573939014	0.1704182877	0.1701187403	0.1701713637	0.1701712989	0.1701713351
	0.4	0.4405998337	0.3461188360	0.3472284475	0.3472227040	0.3472229101	0.3472228714
	0.6	0.6421286150	0.5356385924	0.5385792776	0.5385344234	0.5385343706	0.5385343936
	0.8	0.8713625434	0.7481560486	0.7526946547	0.7526081795	0.7526081188	0.7526080996
	1.0	0.0846612485	0.9944360210	1.0002047725	0.9999999983	0.9999999993	0.9999999998
0.1	0.0	0.0846612485	-0.0029749528	0.0004652414	0.0000000799	0.0000000015	0.0000000006
	0.2	0.2573939014	0.1652829584	0.1703870024	0.1701713457	0.1701712927	0.1701713215
	0.4	0.4405998337	0.3393959310	0.3472230066	0.3472228946	0.3472229169	0.3472228951
	0.6	0.6421286150	0.5270946850	0.5382630439	0.5385342182	0.5385343730	0.5385343459
	0.8	0.8713625434	0.7382692632	0.7520711246	0.7526081887	0.7526080891	0.7526081082
	1.0	0.0846612485	0.9830390029	0.9991673886	0.9999999170	1.0000000003	0.9999999991

Table 4. Reported results of Troesch's problem and the difference with the MATHEMATICA solution.

t	Approximate solution				Absolute error $\lvert y_{\mathrm{MATH}} - y\rvert$			
	y_{ADM}	y_{VIM}	y_{HPM}	$y_{\mathrm{VIM-1}}$	ADM	VIM	HPM	VIM-1
0.1	0.08492528	0.10016683	0.08541675	0.08445675	2.64×10^{-4}	1.55×10^{-2}	7.56×10^{-4}	2.04×10^{-4}
0.2	0.17067908	0.20133869	0.17166933	0.16976097	5.08×10^{-4}	3.12×10^{-2}	1.50×10^{-3}	4.10×10^{-4}
0.3	0.25810502	0.30454102	0.25960358	0.25677759	7.11×10^{-4}	4.71×10^{-2}	2.21×10^{-3}	6.16×10^{-4}
0.4	0.34807811	0.41084132	0.35008533	0.34640729	8.55×10^{-4}	6.36×10^{-2}	2.86×10^{-3}	8.15×10^{-4}
0.5	0.44152329	0.52137347	0.44401041	0.43960682	9.23×10^{-4}	8.08×10^{-2}	3.41×10^{-3}	9.92×10^{-4}
0.6	0.53943772	0.63736635	0.54231466	0.53741230	9.03×10^{-4}	9.88×10^{-2}	3.78×10^{-3}	1.12×10^{-3}
0.7	0.64291809	0.76017896	0.64598391	0.64096743	7.89×10^{-4}	1.18×10^{-1}	3.85×10^{-3}	1.16×10^{-3}
0.8	0.75319489	0.89134491	0.75606400	0.75155871	5.86×10^{-4}	1.39×10^{-1}	3.46×10^{-3}	1.05×10^{-3}
0.9	0.87167571	1.03263022	0.87367075	0.87066123	3.13×10^{-4}	1.61×10^{-1}	2.31×10^{-3}	7.01×10^{-4}
1.0	1.00000000	1.18610986	1.00000000	1.00000000	3.13×10^{-8}	1.86×10^{-1}	3.13×10^{-8}	7.12×10^{-8}

The results are determined for the GA, PS, IPM, PS-IPM and GA-IPM for inputs $t \in (0,\ 1)$with a step of 0.2 and are listed in Table 3. The numerical solution, y_{MATH}, obtained using MATHMETICA, is also presented in Table 3. Moreover, we also provide the reported results of well known techniques ADM, VIM and HPM for the approximate solution of the problem in Table 4.

Before making a comparative analysis, it is better to provide a brief overview of the reported results. The ADM has been applied to solve Troesch's Problem for $0 \le \mu \le 1$; its third term approximate solution y_{ADM} is written as[5]

$$\begin{aligned} y_{\mathrm{ADM}}(t) = & -\frac{1}{4\mu^2}[-\mu\cosh(\mu t)\sinh(\mu t) \\ & +4\mu t\sinh(\mu)\cosh(\mu t)-8\sinh(t)\sinh(\mu t) \\ & -3\mu t\sinh(\mu)\cosh(\mu)+8t\cosh^2(\mu)-8t]. \quad (11) \end{aligned}$$

The VIM has been applied for the solution of the problem for $0 \le \mu \le 1$; its third term approximate solution y_{VIM} is given as[9]

$$\begin{aligned} y_{\mathrm{VIM}}(t) = & \frac{1}{72\mu}-147\mu t+6\mu^3t^3-216\mu t\cosh(\mu t) \\ & -9\mu t\cosh(2\mu t)+333\sinh(\mu t) \\ & +36\mu^2t\sinh(\mu t)+18\sinh(2\mu t)+\sinh(3\mu t). \quad (12) \end{aligned}$$

The HPM has been also applied to solve the problem for $0 \le \mu \le 1$; its second-order approximate solution y_{HPM} is represented as[10]

$$y_{\mathrm{HPM}}(t) = t+\frac{\mu^2}{6}(t^3-t)+\frac{\mu^4}{360}(3t^5-10t^3+7t). \quad (13)$$

Recently, Chang[12,13] has also provided the solution of Troesch's problem using the transform approach with the VIM for first-order approximations (VIM-1). The results are also included in Table 4 for the same inputs.

In order to make a comparison, the absolute errors from y_{MATH} are obtained and the results are provided in Table 4 for y_{ADM}, y_{VIM}, y_{HPM} and $y_{\mathrm{VIM-1}}$, while those provided in Table 5 are for y_{GA}, y_{PS}, y_{IPM}, $y_{\mathrm{PS-IPM}}$ and $y_{\mathrm{GA-IPM}}$. It can be seen that the ranges of error in cases of y_{ADM}, y_{VIM}, y_{HPM} and $y_{\mathrm{VIM-1}}$ are 10^{-4}, 10^{-2} to 10^{-1}, 10^{-4} to 10^{-3} and 10^{-4} to 10^{-3}, respectively; while by taking h =0.01, for y_{GA}, y_{PS}, y_{IPM}, $y_{\mathrm{PS-IPM}}$ and $y_{\mathrm{GA-IPM}}$, the ranges are 10^{-4} to 10^{-3}, 10^{-6} to 10^{-4}, 10^{-8} to 10^{-6}, 10^{-8} and 10^{-9} to 10^{-8}, respectively. It is further noted that there is no significant difference in the results that can be seen for the results obtained using $h = 0.1$ and

$h = 0.01$. Therefore, we have limited our investigation only to $h = 0.1$ and $h = 0.01$ and do not decrease further the step size for more accuracy at the cost of much more computations.

The reliability and effectiveness of the stochastic solver can only be validated through a detailed statistical analysis of the results. In this regard, 50 independent runs of each solver are made. The best and the worst results are defined as minimum and maximum absolute errors from y_{MATH} numerical results. The mean and standard deviation are also determined to check the spreadness in results. The statistical parameters on the basis of 50 independent runs of each solver for inputs $t = 0.3, 0.5, 0.7$ using $h = 0.1$ and $h = 0.01$ are provided in Table 6. It can be seen that better mean results are obtained for the GA-IPM method rather than the GA, PS, IPM and PS-IPM approaches in both cases of $h = 0.1$ and $h = 0.01$.

Moreover, the values of fitness function e_j given in Eqs. (9) and (10) are determined for the 50 independent runs using the GA, PS, IPM, PS-IPM and GA-IPM algorithms. The results are plotted in Fig. 4 in descending order, fitness values against the independent runs of each algorithm. Figure 4(a) shows the results taking the fitness function e_j given in Eq. (9), while Fig. 4(b) is plotted using the fitness function given in expression (10). It can be seen that the ranges of fitness values in the case of $h = 0.01$ are 10^{-4} to 10^{-2}, 10^{-6} to 10^{-2}, 10^{-8} to 10^{-6}, 10^{-9} to 10^{-6} and 10^{-10} to 10^{-7} using the GA, PS, IPM, PS-IPM and GA-IPM algorithms, respectively. It can be further seen that for both cases of $h = 0.1$ and $h = 0.01$, the best results are obtained from the GA-IPM method.

Table 5. Error analysis of the proposed solutions obtained for Troesch's Problem.

t	Absolute error for $h = 0.01$					Absolute error for $h = 0.1$				
	GA	PS	IPM	PS-IPM	GA-IPM	GA	PS	IPM	PS-IPM	GA-IPM
0.1	7.50×10^{-4}	6.66×10^{-5}	8.53×10^{-8}	5.30×10^{-8}	2.89×10^{-8}	3.89×10^{-3}	3.31×10^{-4}	9.27×10^{-8}	6.97×10^{-8}	3.16×10^{-8}
0.2	2.50×10^{-4}	5.26×10^{-5}	1.17×10^{-8}	5.31×10^{-8}	1.69×10^{-8}	4.89×10^{-3}	2.16×10^{-4}	6.30×10^{-9}	5.93×10^{-8}	3.05×10^{-8}
0.3	3.61×10^{-4}	2.73×10^{-5}	1.05×10^{-7}	5.11×10^{-8}	2.01×10^{-8}	6.19×10^{-3}	1.14×10^{-4}	4.51×10^{-8}	4.46×10^{-8}	8.70×10^{-9}
0.4	1.10×10^{-3}	5.60×10^{-6}	1.46×10^{-7}	6.00×10^{-8}	2.13×10^{-8}	7.83×10^{-3}	1.57×10^{-7}	4.45×10^{-8}	6.68×10^{-8}	4.50×10^{-8}
0.5	1.97×10^{-3}	3.21×10^{-5}	1.01×10^{-7}	7.30×10^{-8}	2.92×10^{-8}	9.65×10^{-3}	1.35×10^{-4}	1.23×10^{-7}	7.19×10^{-8}	1.65×10^{-8}
0.6	2.90×10^{-3}	4.49×10^{-5}	2.29×10^{-8}	2.99×10^{-8}	6.90×10^{-9}	1.14×10^{-2}	2.71×10^{-4}	1.82×10^{-7}	2.75×10^{-8}	5.46×10^{-8}
0.7	3.77×10^{-3}	5.56×10^{-5}	1.40×10^{-7}	5.99×10^{-8}	2.32×10^{-8}	1.30×10^{-2}	3.95×10^{-4}	3.57×10^{-8}	6.54×10^{-8}	4.60×10^{-8}
0.8	4.45×10^{-3}	8.65×10^{-5}	7.30×10^{-8}	1.23×10^{-8}	6.90×10^{-9}	1.43×10^{-2}	5.37×10^{-4}	8.22×10^{-8}	1.74×10^{-8}	1.70×10^{-9}
0.9	4.95×10^{-3}	1.46×10^{-4}	1.18×10^{-7}	1.37×10^{-8}	2.05×10^{-8}	1.55×10^{-2}	7.00×10^{-4}	3.56×10^{-7}	3.62×10^{-8}	6.01×10^{-8}
1.0	5.56×10^{-3}	2.05×10^{-4}	3.30×10^{-8}	3.20×10^{-8}	3.15×10^{-8}	1.70×10^{-2}	8.33×10^{-4}	1.14×10^{-7}	3.10×10^{-8}	3.22×10^{-8}

Table 6. Statistical parameters of the proposed results for Troesch's problem for $\mu = 1$.

t	Method	$h = 0.01$				$h = 0.1$			
		Best	Worst	Mean	STD	Best	Worst	Mean	STD
0.3	GA	1.32×10^{-4}	2.03×10^{-2}	4.79×10^{-3}	4.97×10^{-3}	2.12×10^{-5}	1.82×10^{-2}	4.53×10^{-3}	3.90×10^{-3}
	PS	2.49×10^{-5}	1.70×10^{-2}	2.10×10^{-3}	3.65×10^{-3}	1.34×10^{-5}	3.42×10^{-2}	2.89×10^{-3}	7.04×10^{-3}
	IPM	4.00×10^{-9}	6.22×10^{-6}	1.19×10^{-6}	1.45×10^{-6}	4.51×10^{-8}	1.72×10^{-5}	2.11×10^{-6}	3.26×10^{-6}
	PS-IPM	7.31×10^{-9}	5.92×10^{-6}	7.24×10^{-7}	1.08×10^{-6}	2.59×10^{-9}	1.36×10^{-5}	1.32×10^{-6}	2.43×10^{-6}
	GA-IPM	1.64×10^{-9}	1.82×10^{-5}	1.26×10^{-6}	2.94×10^{-6}	3.69×10^{-9}	2.66×10^{-5}	1.94×10^{-6}	4.63×10^{-6}
0.5	GA	5.07×10^{-4}	4.29×10^{-2}	1.37×10^{-2}	1.04×10^{-2}	1.52×10^{-3}	5.11×10^{-2}	1.56×10^{-2}	1.06×10^{-2}
	PS	3.21×10^{-5}	3.73×10^{-2}	3.08×10^{-3}	6.09×10^{-3}	8.80×10^{-5}	4.81×10^{-2}	2.92×10^{-3}	7.43×10^{-3}
	IPM	4.14×10^{-9}	7.31×10^{-6}	1.43×10^{-6}	1.66×10^{-6}	1.09×10^{-7}	2.50×10^{-5}	2.55×10^{-6}	4.39×10^{-6}
	PS-IPM	1.63×10^{-8}	9.33×10^{-6}	1.12×10^{-6}	1.99×10^{-6}	2.75×10^{-9}	9.05×10^{-6}	1.18×10^{-6}	1.79×10^{-6}
	GA-IPM	1.11×10^{-8}	1.56×10^{-5}	1.01×10^{-6}	2.34×10^{-6}	7.88×10^{-9}	2.57×10^{-5}	1.95×10^{-6}	$4.53\mathrm{v}10^{-6}$
0.7	GA	3.17×10^{-3}	6.28×10^{-2}	2.25×10^{-2}	1.60×10^{-2}	6.62×10^{-3}	1.01×10^{-1}	2.65×10^{-2}	1.82×10^{-2}
	PS	1.44×10^{-6}	7.01×10^{-2}	4.35×10^{-3}	1.12×10^{-2}	1.88×10^{-5}	7.36×10^{-2}	3.67×10^{-3}	1.09×10^{-2}
	IPM	4.48×10^{-8}	7.22×10^{-6}	1.45×10^{-6}	1.83×10^{-6}	3.57×10^{-8}	3.46×10^{-5}	2.68×10^{-6}	5.65×10^{-6}
	PS-IPM	2.18×10^{-8}	1.50×10^{-5}	1.64×10^{-6}	3.07×10^{-6}	4.42×10^{-9}	7.05×10^{-6}	1.02×10^{-6}	1.56×10^{-6}
	GA-IPM	1.48×10^{-8}	1.03×10^{-5}	9.56×10^{-7}	1.81×10^{-6}	2.12×10^{-9}	2.16×10^{-5}	1.92×10^{-6}	4.34×10^{-6}

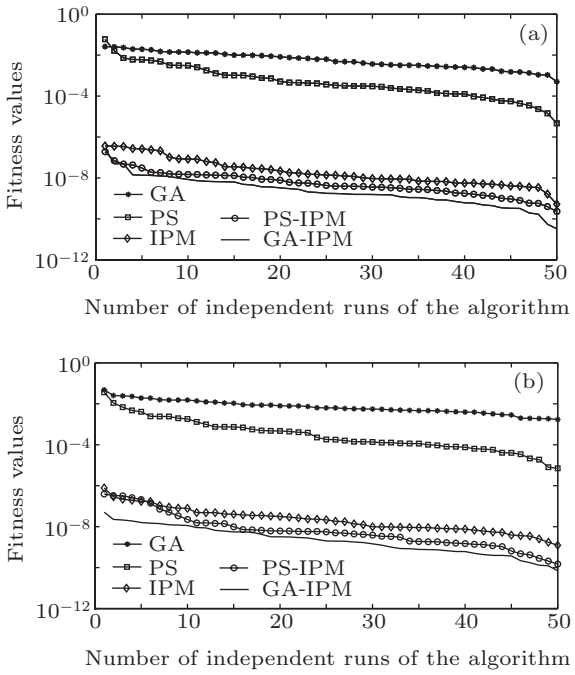


Fig. 4. Summary of the results for 50 independent runs of different stochastic solvers: (a) $h = 0.01$, (b) $h = 0.1$.

4.2. Solution of Troesch's problem by taking $\mu = 0.5$

By solving the given Troesch's problem in a similar way as in the previous case, however, the fitness functions formulated by taking $N = 100$ and $N = 10$ steps in interval (0, 1), respectively, can be written as

$$e_j = \frac{1}{101}\sum_{m=0}^{100}\left(\frac{\mathrm{d}^2\hat{y}_m}{\mathrm{d}t_m^2} - \frac{1}{2}\sinh\left(\frac{1}{2}\hat{y}_m\right)\right)^2 + \frac{1}{2}\left([\hat{y}(0)]^2 + [\hat{y}(1)-1]^2\right)\Big|_j, \quad j = 1,2,3,\ldots, \quad (14)$$

$$e_j = \frac{1}{11}\sum_{m=0}^{10}\left(\frac{\mathrm{d}^2\hat{y}_m}{\mathrm{d}t_m^2} - \frac{1}{2}\sinh\left(\frac{1}{2}\hat{y}_m\right)\right)^2 + \frac{1}{2}\left([\hat{y}(0)]^2 + [\hat{y}(1)-1]^2\right)\Big|_j, \quad j = 1,2,3,\ldots. \quad (15)$$

We have applied the GA, PS, IPM, PS-IPM and GA-IMP approaches for training the weights of the DENN on the same pattern as in the previous case. One set of weights using fitness function (14) for the GA, PS, IPM, PS-IPM and GA-IPM algorithms with fitness values of 1.0599×10^{-4}, 2.1807×10^{-6}, 1.9544×10^{-10}, 6.9990×10^{-12} and 2.0771×10^{-12}, respectively, are plotted in Fig. 5(a). Similarly, in Fig. 5(b), the weights are provided using fitness function (15) with fitness values of 5.1294×10^{-5}, 2.3190×10^{-6}, 7.5301×10^{-10}, 1.0193×10^{-11} and 1.1402×10^{-12} for the GA, PS, IPM, PS-IPM and GA-IPM, respectively. A solution of Troesch's problem is determined for inputs $t \in (0,1)$ with step 0.2 using Eq. (3) and weights given in Fig. 5. The results are presented in Table 7; it also includes the numerical solution y_{MATH} obtained using the inbuilt capability of MATHEMATICA. The approximate solutions for y_{ADM}, y_{VIM} and y_{HPM} are calculated using $\mu = 0.5$ in Eqs. (11)–(13), respectively. The solutions based on the ADM, VIM and HPM approaches are provided in Table 8 along with the report solution by Chang[12,13] using the first-order approximation of VIM $y_{\mathrm{VIM}-1}$. The error analyses for the results from the standard solution y_{MATH} are provided in Table 9. It can be seen from Tables 8 and 9 that the absolute errors for ADM, VIM, HPM and VIM-1 lie in the ranges of 10^{-6} to 10^{-5}, 10^{-3} to 10^{-2}, 10^{-5} to 10^{-4} and 10^{-3} to 10^{-2}, respectively, whereas the absolute errors in GA, PS, IPM, PS-IPM and GA-IPM for $h = 0.1$ lie in the ranges of 10^{-7} to 10^{-4}, 10^{-5} to 10^{-4}, 10^{-9} to 10^{-8} and 10^{-9} to 10^{-8}, respectively. It can be seen that the best results are mostly provided by the hybrid approach of GA-IPM scheme.

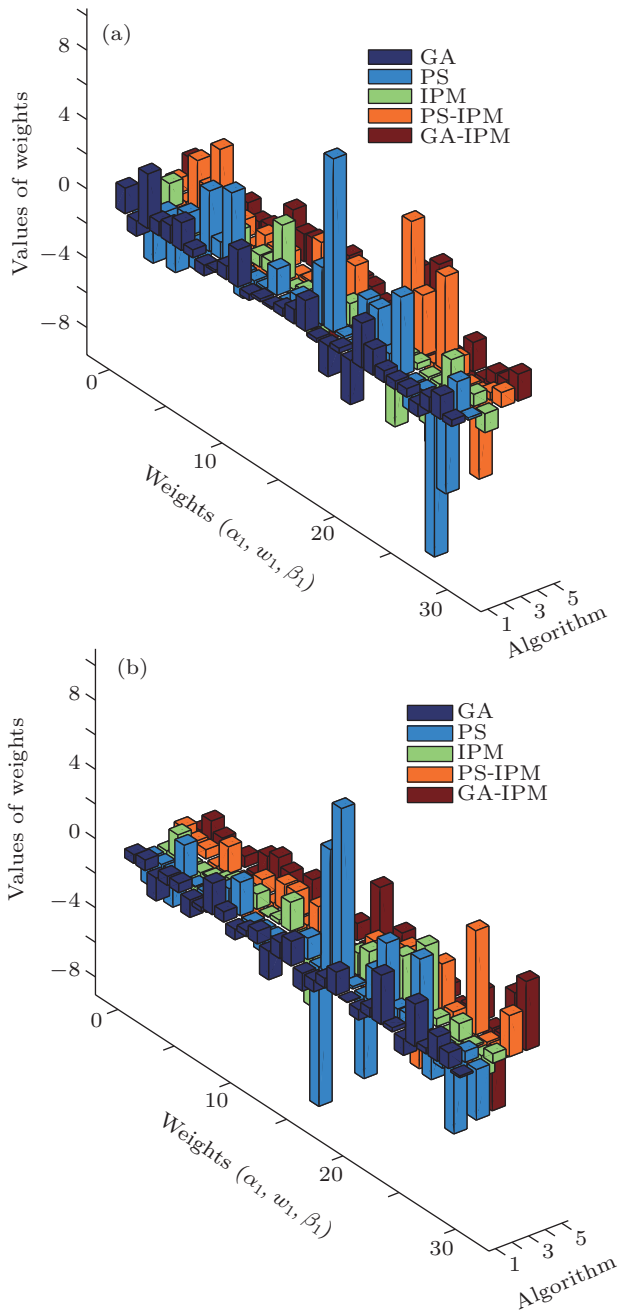


Fig. 5. (color online) Sets of weights learned for the DENN by stochastic solvers for (a) $h = 0.01$ and (b) $h = 0.1$.

Table 7. Solution of Troesch's problem obtained using different step sizes for $\mu = 0.5$.

h	t	y_{MATH}	y_{GA}	y_{PS}	y_{IPM}	y_{PS-IPM}	y_{GA-IPM}
0.01	0.0	0.000000000	0.0001365233	0.0004919299	0.0000000163	0.0000000000	0.0000000000
	0.2	0.1921287605	0.1922279739	0.1924746631	0.1921287638	0.1921287331	0.1921287511
	0.4	0.3861848604	0.3860031805	0.3863736747	0.3861847672	0.3861848627	0.3861848451
	0.6	0.5841332656	0.5835714558	0.5842155301	0.5841333516	0.5841332394	0.5841332447
	0.8	0.7880165428	0.7874112630	0.7879945174	0.7880164900	0.7880165255	0.7880165284
	1.0	1.0000000230	0.9994948197	0.9998297300	0.9999999837	1.0000000001	0.9999999999
0.1	0.0	0.0000000000	0.0016016158	0.0000003916	0.0000000192	0.0000000049	0.0000000000
	0.2	0.1921287605	0.1931828752	0.1921094137	0.1921287041	0.1921287580	0.1921287425
	0.4	0.3861848604	0.3865792590	0.3862102451	0.3861845830	0.3861848185	0.3861848478
	0.6	0.5841332656	0.5838823883	0.5841818449	0.5841335562	0.5841332695	0.5841332523
	0.8	0.7880165428	0.7873125957	0.7880358506	0.7880165579	0.7880165295	0.7880165204
	1.0	1.0000000230	0.9988599269	1.0000052640	0.9999999989	1.0000000049	0.9999999999

Table 8. Reported results of Troesch's problem for $\mu = 0.5$ with absolute errors.

t	Approximate solution				Absolute error $\lvert y_{MATH} - y \rvert$			
	y_{ADM}	y_{VIM}	y_{HPM}	y_{VIM-1}	ADM	VIM	HPM	VIM-1
0.1	0.09594775	0.10004167	0.09599479	0.09297271	3.39×10^{-6}	4.10×10^{-3}	5.04×10^{-5}	2.97×10^{-3}
0.2	0.19213525	0.20033366	0.19222933	0.18636669	6.49×10^{-6}	8.20×10^{-3}	1.01×10^{-4}	5.76×10^{-3}
0.3	0.28880342	0.30112753	0.28894397	0.28060408	9.01×10^{-6}	1.23×10^{-2}	1.50×10^{-4}	8.19×10^{-3}
0.4	0.38619555	0.40267739	0.38638033	0.37610883	1.07×10^{-5}	1.65×10^{-2}	1.95×10^{-4}	1.00×10^{-2}
0.5	0.48455854	0.50524117	0.48478190	0.47330770	1.14×10^{-5}	2.07×10^{-2}	2.35×10^{-4}	1.12×10^{-2}
0.6	0.58414420	0.60908205	0.58439466	0.57263131	1.09×10^{-5}	2.49×10^{-2}	2.61×10^{-4}	1.15×10^{-2}
0.7	0.68521057	0.71446985	0.68546774	0.67451551	9.40×10^{-6}	2.93×10^{-2}	2.67×10^{-4}	1.06×10^{-2}
0.8	0.78802343	0.82168265	0.78825400	0.77940273	6.89×10^{-6}	3.37×10^{-2}	2.37×10^{-4}	8.61×10^{-3}
0.9	0.89285787	0.93100843	0.89301067	0.88774380	3.63×10^{-6}	3.82×10^{-2}	1.56×10^{-4}	5.11×10^{-3}
1.0	1.00000000	1.04274695	1.00000000	1.00000000	2.30×10^{-8}	4.27×10^{-2}	2.30×10^{-8}	4.98×10^{-7}

Table 9. Error analyses of the proposed solutions obtained for Troesch's problem.

t	Absolute error for $h = 0.01$					Absolute error for $h = 0.1$				
	GA	PS	IPM	PS-IPM	GA-IPM	GA	PS	IPM	PS-IPM	GA-IPM
0.1	1.16×10^{-4}	4.25×10^{-4}	2.74×10^{-8}	8.32×10^{-9}	1.54×10^{-8}	1.34×10^{-3}	1.57×10^{-5}	1.31×10^{-7}	2.46×10^{-8}	4.95×10^{-9}
0.2	9.92×10^{-5}	3.46×10^{-4}	3.31×10^{-9}	2.74×10^{-8}	9.37×10^{-9}	1.05×10^{-3}	1.93×10^{-5}	5.64×10^{-8}	2.45×10^{-9}	1.80×10^{-8}
0.3	9.90×10^{-7}	2.62×10^{-4}	6.73×10^{-8}	1.63×10^{-8}	5.90×10^{-9}	7.37×10^{-4}	1.73×10^{-5}	2.96×10^{-7}	3.85×10^{-8}	2.12×10^{-8}
0.4	1.82×10^{-4}	1.89×10^{-4}	9.32×10^{-8}	2.35×10^{-9}	1.53×10^{-8}	3.94×10^{-4}	2.54×10^{-5}	2.77×10^{-7}	4.19×10^{-8}	1.26×10^{-8}
0.5	3.91×10^{-4}	1.31×10^{-4}	1.97×10^{-8}	5.63×10^{-9}	2.47×10^{-8}	5.51×10^{-5}	4.54×10^{-5}	5.99×10^{-9}	1.82×10^{-8}	7.72×10^{-9}
0.6	5.62×10^{-4}	8.23×10^{-5}	8.60×10^{-8}	2.62×10^{-8}	2.09×10^{-8}	2.51×10^{-4}	4.86×10^{-5}	2.91×10^{-7}	3.90×10^{-9}	1.33×10^{-8}
0.7	6.38×10^{-4}	3.44×10^{-5}	8.62×10^{-8}	2.87×10^{-8}	1.15×10^{-8}	5.03×10^{-4}	3.65×10^{-5}	2.96×10^{-7}	4.32×10^{-9}	2.09×10^{-8}
0.8	6.05×10^{-4}	2.20×10^{-5}	5.28×10^{-8}	1.73×10^{-8}	1.44×10^{-8}	7.04×10^{-4}	1.93×10^{-5}	1.51×10^{-8}	1.33×10^{-8}	2.24×10^{-8}
0.9	5.14×10^{-4}	9.34×10^{-5}	1.40×10^{-7}	1.89×10^{-8}	2.67×10^{-8}	8.89×10^{-4}	8.15×10^{-6}	2.01×10^{-7}	2.79×10^{-8}	2.21×10^{-8}
1.0	5.05×10^{-4}	1.70×10^{-4}	3.93×10^{-8}	2.29×10^{-8}	2.31×10^{-8}	1.14×10^{-3}	5.24×10^{-6}	2.41×10^{-8}	1.81×10^{-8}	2.31×10^{-8}

Moreover, the statistical analysis of the results for the stochastic solvers is also performed. In this regard, 50 independent runs for the GA, PS, IPM, PS-IPM and GA-IPM algorithms are executed. The statistical parameters are provided for $t = 0.4$, 0.6, 0.8 using the different step sizes in Table 10. It can be inferred from the results that the mean value of absolute error is the minimum in the GA-IPM method as compare to the GA, PS, IPM and PS-IPM approaches. The values of fitness function e_j given in Eqs. (14) and (15) are obtained for 50 independent runs with the GA, PS, IPM, PS-IPM and GA-IPM algorithms. The results are presented in Fig. 5 in descending the order fitness values against the number of independent runs of each scheme. Figure 5(a) shows the results by taking the fitness function e_j as given in Eq. (14), while Fig. 5(b) is plotted using the fitness function given in expression (15). It can be seen that the ranges of fitness values in the case of $h = 0.1$ are 10^{-4} to 10^{-3}, 10^{-5} to 10^{-1}, 10^{-8} to 10^{-7}, 10^{-11} to 10^{-7} and 10^{-12} to 10^{-8} for the GA, PS, IPM, PS-IPM and GA-IPM algorithms, respectively. It can be seen that for both cases of $h = 0.01$ and $h = 0.1$, the best results are mostly obtained from the GA-IPM technique.

Table 10. Statistical parameters of the proposed results for Troesch's problem for $\mu = 0.5$.

t	Method	$h = 0.01$				$h = 0.1$			
		Best	Worst	Mean	STD	Best	Worst	Mean	STD
0.4	GA	2.01×10^{-5}	3.81×10^{-3}	1.05×10^{-3}	8.29×10^{-4}	9.51×10^{-6}	5.09×10^{-3}	1.12×10^{-3}	9.95×10^{-4}
	PS	3.84×10^{-7}	8.02×10^{-3}	5.83×10^{-4}	1.24×10^{-3}	6.63×10^{-6}	5.17×10^{-3}	8.53×10^{-4}	1.30×10^{-3}
	IPM	7.54×10^{-9}	3.05×10^{-6}	4.45×10^{-7}	4.46×10^{-7}	1.56×10^{-7}	5.38×10^{-6}	7.78×10^{-7}	8.69×10^{-7}
	PS-IPM	2.35×10^{-9}	8.22×10^{-6}	7.20×10^{-7}	1.38×10^{-6}	2.94×10^{-9}	5.68×10^{-6}	6.40×10^{-7}	1.13×10^{-6}
	GA-IPM	2.55×10^{-9}	3.66×10^{-6}	3.65×10^{-7}	5.97×10^{-7}	9.83×10^{-9}	1.47×10^{-6}	3.04×10^{-7}	2.93×10^{-7}
0.6	GA	4.86×10^{-5}	5.02×10^{-3}	1.85×10^{-3}	1.21×10^{-3}	1.76×10^{-4}	6.59×10^{-3}	2.13×10^{-3}	1.29×10^{-3}
	PS	1.76×10^{-6}	4.99×10^{-3}	6.86×10^{-4}	9.21×10^{-4}	5.43×10^{-6}	6.26×10^{-3}	9.26×10^{-4}	1.44×10^{-3}
	IPM	8.60×10^{-8}	3.27×10^{-6}	6.89×10^{-7}	7.09×10^{-7}	1.08×10^{-9}	3.77×10^{-6}	8.16×10^{-7}	7.04×10^{-7}
	PS-IPM	3.49×10^{-9}	3.82×10^{-6}	5.86×10^{-7}	9.05×10^{-7}	3.90×10^{-9}	5.53×10^{-6}	4.41×10^{-7}	8.70×10^{-7}
	GA-IPM	3.95×10^{-10}	1.95×10^{-6}	3.24×10^{-7}	3.90×10^{-7}	1.74×10^{-9}	3.27×10^{-6}	3.47×10^{-7}	5.28×10^{-7}
0.8	GA	3.18×10^{-5}	6.00×10^{-3}	2.01×10^{-3}	1.42×10^{-3}	3.19×10^{-4}	8.53×10^{-3}	2.45×10^{-3}	1.64×10^{-3}
	PS	7.28×10^{-6}	9.44×10^{-3}	1.07×10^{-3}	1.75×10^{-3}	4.18×10^{-6}	1.01×10^{-2}	1.37×10^{-3}	2.24×10^{-3}
	IPM	5.45×10^{-9}	2.24×10^{-6}	3.93×10^{-7}	5.26×10^{-7}	8.68×10^{-9}	4.77×10^{-6}	3.98×10^{-7}	8.88×10^{-7}
	PS-IPM	1.43×10^{-9}	5.96×10^{-6}	7.23×10^{-7}	1.32×10^{-6}	9.09×10^{-10}	2.72×10^{-6}	4.08×10^{-7}	5.99×10^{-7}
	GA-IPM	3.56×10^{-10}	2.41×10^{-6}	2.16×10^{-7}	4.50×10^{-7}	4.34×10^{-10}	3.79×10^{-6}	2.23×10^{-7}	5.86×10^{-7}

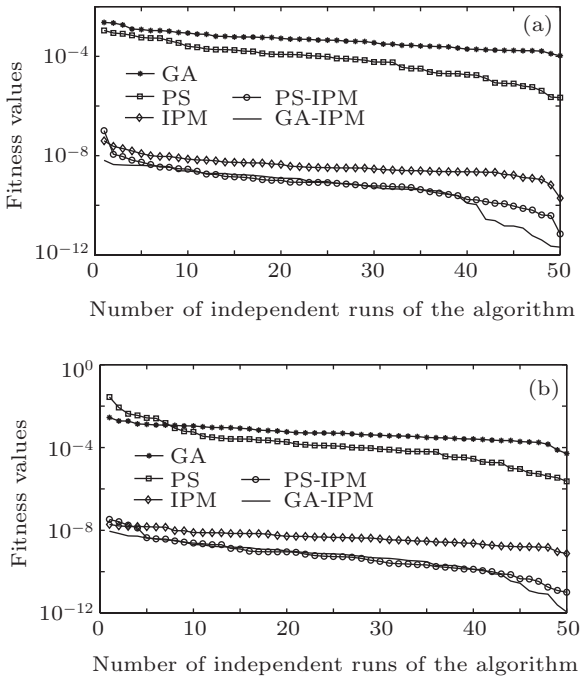


Fig. 6. Summary of the results for 50 independent runs of different stochastic solvers: (a) $h = 0.01$, (b) $h = 0.1$.

5. Conclusion

The numerical solvers based on a DENN optimized with the GA, PS, IPM, PS-IPM and GA-IPM provide a reliable solution of Troesch's problem. It is found that the absolute error from the standard MATHEMATICA solution lies in ranges of 10^{-4} to 10^{-3}, 10^{-6} to 10^{-4}, 10^{-8} to 10^{-6}, 10^{-8} and 10^{-9} to 10^{-8} for the GA, PS, IPM, PS-IPM and GA-IPM, respectively.

Among the proposed schemes, the results from the hybrid approaches like PS-IPM and GA-IPM are more accurate than those from the GA, PS and IPM techniques. In hybrid approaches, the results of GA-IPM are relatively more precise than that of the PS-IPM method.

A comparison of the results is made with the approximate solutions of well known analytic techniques ADM, VIM and HPM. It is found that among the proposed techniques, the results of GA and PS techniques are comparable to those of ADM, VIM and HPM approaches, whereas IPM, PS-IPM and GA-IPM provide better solutions of Troesch's problem.

The reliability and effectiveness of the proposed schemes are validated from a detailed statistical analysis. It is found that the mean absolute error is the lowest in the GA-IPM scheme compared to those in the GA, PS, IPM and PS-IPM algorithms.

Moreover, the proposed computational intelligence techniques give results readily on a continuous input within the entire trained interval unlike numerical methods based on built-in routines of MATHEMATICA using fully explicit Range–Kutta solvers, which give results only on a predefined grid on discrete inputs and require the repetition of the whole cumbersome procedure to find results for any other input. On the other hand, the state-of-the-art analytical solvers including ADM, VIM and HAM give accurate results only in the close vicinity of the initial guess and, as the input range expands, they start to accumulate errors. The results of the proposed techniques are less prone to these effects. Besides, simplicity of concept, ease of implementation and a broader domain of applicability are other perks of the proposed scheme.

In future, one can look for the application of swarm intelligence techniques based on ant/bee colony optimization or particle swarm optimization, differential evolution and genetic programming etc for the solution of Troesch's problem.

References

[1] Weibel E S 1958 *The Plasma in Magentic Field* (Landshoff R K M, Ed.) (Stanford: Stanford University Press)

[2] Markin V S, Chernenko A A, Chizmadehev Y A and Chirkov Y G 1966 *Aspects of the Theory of Gas Porous Electrodes* (Bagotskii V S and Vasilev Y B, Ed.) (New York: Consultants Bureau) pp. 21–33
[3] Godaspow D and Baker B S 1973 *J. Electrochem. Soc.* **120** 1005
[4] Troesch B A 1976 *J. Comput. Phys.* **21** 279
[5] Deeba E, Khuri S A and Xie S 2000 *J. Comput. Phys.* **159** 125
[6] Roberts S M and Shipman J S 1972 *J. Comput. Phys.* **10** 232
[7] Khuri S A 2003 *Int. J. Comput. Math.* **80** 493
[8] Chin R C Y and Krasny R 1983 *SIAM J. Sci. Stat. Comput.* **10** 229
[9] Momani S, Abuasad S and Odibat Z 2006 *Appl. Math. Comput.* **183** 1351
[10] Feng X, Mei L and He G 2007 *Appl. Math. Comput.* **189** 500
[11] Chang S H and Chang I L 2008 *Appl. Math. Comput.* **195** 799
[12] Chang S H 2010 *J. Comput. Appl. Math.* **234** 3043
[13] Chang S H 2010 *Appl. Math. Comput.* **216** 3303
[14] Khuri S A and Sayfy A 2011 *Math. Comput. Model.* **56** 1907
[15] Mirmoradi S H, Hosseinpour I, Ghanbarpour S and Barari A 2009 *Applied Mathematical Sciences* **3** 1579
[16] Mohamad A J 2010 *Eng. Tech. J.* **28** 369
[17] Snyman J A 1979 *J. Comput. Appl. Math.* **5** 171
[18] Babolian E and Javadi S h 2004 *Appl. Math. Comput.* **153** 253
[19] Chin R C Y 1981 *J. Comput. Appl. Math.* **7** 181
[20] Parisi D R, Mariani M C and Laborde M A 2003 *Chem. Eng. Processing* **42** 715
[21] Khan J A, Raja M A Z and Qureshi I M 2011 *Chin. Phys. Lett.* **28** 020206
[22] Yazdi H S and Shahri R P 2010 *Appl. Soft Comput.* **10** 267
[23] Yazdan S, Hayati M and Moradian R 2009 *Appl. Soft Comput.* **9** 20
[24] Beidokhti R S and Malek A 2009 *J. Franklin Institute* **346** 898
[25] Khan J A, Raja M A Z and Qureshi I M 2011 *Int. J. Phys. Sci.* **6** 7247
[26] Khan J A, Raja M A Z and Qureshi I M 2011 *Chin. Phys. Lett.* **28** 110205
[27] Khan J A, Raja M A Z and Qureshi I M 2011 *Ann. Math. Artif. Intell.* **63** 185
[28] Raja M A Z, Khan J A, Ahmad S I and Qureshi I M 2013 *Innovations in Intelligent Machines – 3, SCI 442* (Berlin Heidelberg: Springer-Verlag) pp. 103–117
[29] Monterola C and Saloma C 2001 *Opt. Express* **9** 72
[30] El-Emam N N and Al-Rabeh R H 2011 *Appl. Soft Comput.* **11** 3283
[31] Raja M A Z, Ahmad S I and Samar R 2012 *Neural Comput. Appl.*
[32] Raja M A Z, and Ahmad S I 2012 *Neural Comput. Appl.*
[33] Raja M A Z submitted to *Information Sciences Journal*
[34] Kumar M and Yadav N 2011 *Comput. Math. Appl.* **62** 3796
[35] Raja M A Z, Qureshi I M and Khan J A 2011 *Int. J. Innov. Comput I* **7** 621
[36] Raja M A Z, Khan J A and Qureshi I M 2010 *Lecture Notes in Computer Science*, Vol. **5990** (Berlin Heidelberg: Springer-Verlag) pp. 231–240
[37] Raja M A Z, Khan J A and Qureshi I M 2010 *GECCO (Companion)* **2010** 2023
[38] Raja M A Z, Khan J A and Qureshi I M 2011 *Math. Probl. Eng.* **2011** 76507501
[39] Raja M A Z, Khan J A and Qureshi I M 2010 *Ann. Math. Artif. Intell.* **60** 229
[40] Aarts L P and Veer P V D 2001 *Neural Process. Lett.* **14** 261
[41] Khan J A and Raja M A Z 2013 *Research Journal of Applied Sciences, Engineering and Technology* **6** 450
[42] Karmarkar N 1984 *Combinatorica* **4** 373
[43] Wright S 1997 *Primal-Dual Interior-Point Methods*. (Philadelphia: SIAM)
[44] Wright M H 2005 *Bull. Amer. Math. Soc.* **42** 39
[45] Hooke R and Jeeves T A 1961 *Journal of the Association for Computing Machinery* **8** 212
[46] Yu W C 1979 *Positive Basis and a Class of Direct Search Techniques*, (Zhongguo Kexue: Scientia Sinica) pp. 53–68
[47] Dolan E D, Lewis R M and Torczon V J 2003 *SIAM J. Optimiz.* **14** 567
[48] Michael L R and Torczon V 1999 *SIAM J. Optimiz.* **9** 1082
[49] Michael L R and Torczon V 2000 *SIAM J. Optimiz.* **10** 917
[50] Zu Y X and Zhou J 2012 *Chin. Phys. B* **21** 019501
[51] Zu Y X and Zhou J 2011 *Acta Phys. Sin.* **60** 079501 (in Chinese)
[52] Zu Y X, Zhou J and Zeng CC 2010 *Chin. Phys. B* **19** 119501
[53] Wu P, He Y G and Fang G F 2013 *Acta Phys. Sin.* **62** 020301 (in Chinese)
[54] He R, Huang S X, Zhou C T and Jiang Z H 2012 *Acta Phys. Sin.* **61** 049201 (in Chinese)
[55] Zhou J, Liu Y A, Wu F, Zhang H G and Zu Y X 2011 *Acta Phys. Sin.* **60** 090504 (in Chinese)
[56] Wang J B and Lu J 2011 *Acta Phys. Sin.* **60** 057304 (in Chinese)