A VARIABLE PROJECTION METHOD FOR COMPUTATIONAL PDES WITH ARTIFICIAL NEURAL NETWORKS

Suchuan Dong
Center for Computational & Applied Mathematics
Department of Mathematics
Purdue University
West Lafayette, IN 47907
sdong@purdue.edu

ABSTRACT

We present a method for solving linear and nonlinear partial differential equations (PDE) based on the variable projection framework and artificial neural networks. For linear PDEs, enforcing the boundary/initial value problem on the collocation points gives rise to a separable nonlinear least squares problem about the network coefficients. We reformulate this problem by the variable projection approach to eliminate the linear output-layer coefficients, leading to a reduced problem about the hidden-layer coefficients only. The reduced problem is computed first by the nonlinear least squares method to determine the hidden-layer coefficients, and then the output-layer coefficients are determined by the linear least squares method. For nonlinear PDEs, enforcing the boundary/initial value problem on the collocation points gives rise to a nonlinear least squares problem that is not separable. To enable the variable projection approach for nonlinear PDEs, we first linearize the problem with a Newton iteration. The linearized system is solved by the variable projection framework together with artificial neural networks. The current method exhibits a spectral-like accuracy, with its errors decreasing exponentially with respect to the number of collocation points or the output-layer coefficients. We compare this method with PINN, and show that the current method is much more accurate and computationally efficient.

Keywords Variable projection · Neural Network · Nolinear least squares · Spectral accuracy

1 Introduction

This work concerns the approximation of partial differential equations (PDE) with artificial neural networks (ANN), and we exploit the the variable projection (VarPro) approach for solving linear and nonlinear PDEs. Neural network-based PDE methods, especially those based on deep neural networks (DNN) [13], have flourished in the past few years; see the review [14] and the references therein.

Variable projection (VarPro) is a classical approach for solving separable nonlinear least squares (SNLLS) problems [11, 12]. These problems are separable in the sense that the unknown parameters can be separated into two sets: the linear parameters and the nonlinear parameters. The basic idea of VarPro is to treat the linear parameters as dependent on the nonlinear parameters, and then eliminate the linear parameters from the problem by using the linear least squares method. This gives rise to a reduced, but generally more complicated, nonlinear least squares problem that involves only the nonlinear parameters [12]. One can then solve the reduced problem for the nonlinear parameters by a nonlinear least squares method, typically involving a Gauss-Newton type algorithm coupled with trust region or backtracking line search strategies [5, 2, 6, 9, 20]. Upon attaining the nonlinear parameters, one then computes the linear parameters by the linear least squares method. Although the reduced problem is in general more complicated, the benefits of variable projection are typically very significant. These include the reduced dimension of the parameter space, better conditioning, and faster convergence with the reduced problem [25, 26, 12]. In some sense the idea of

variable projection to least squares problems can be analogized to the Schur complement in linear algebra or the static condensation in computational mechanics.

The VarPro algorithm was originally developed in [11], and has been improved and generalized by a number of researchers and applied to many areas in the past few decades [15, 25, 12, 4, 22, 17, 21, 1, 3, 27, 10]. The VarPro algorithm or its variants for training neural networks have been the subject of several studies in the literature [29, 31, 30, 26, 23, 16, 19, 18, 8]. In [26] the authors have proved that the reduced nonlinear functional of the variable projection approach, while seemingly more complicated, leads to a better-conditioned problem and always converges faster than the original problem; see also [25]. The VarPro method together with the Levenberg-Marquardt algorithm is employed for the training of two-layered neural networks in [23, 16] and compared with other related approaches. In [19, 18] the authors extend the variable projection approach to deal with non-quadratic objective functions, and also present a stochastic optimization method based on variable projection for training deep neural networks with attractive properties.

In this paper we focus on the variable projection approach together with ANN for solving partial differential equations. The general strategy is as follows. We employ a feed-forward neural network to represent the field solution to the PDE, requiring that the output layer be linear and with zero bias. We enforce the PDEs and boundary/initial conditions on a set of collocation points in the domain and on the boundary. This gives rise to a set of discrete equations about the weight/bias coefficients in the neural network. In turn, this set of equations leads to a nonlinear least squares problem about the network coefficients.

If the boundary/initial value problem is linear, then the aforementioned nonlinear least squares problem is separable, and the problem can be solved by VarPro. On the other hand, if the boundary/initial value problem is nonlinear, the aforementioned nonlinear least squares problem is not separable. As a result, the variable projection cannot be directly used for nonlinear PDEs. In this case we present a combined Newton-variable projection method together with ANNs for solving the problem. We first linearize the problem for the Newton iteration. The linearized system is linear with respect to the updated approximation field, and it is solved by the variable projection approach together with ANNs. Therefore, to solve nonlinear PDEs, our method involves an overall Newton iteration. Within each iteration, we use VarPro together with ANNs to solve the linearized system to attain the updated field approximation. Upon convergence of the Newton iteration, the neural-network coefficients contain the representation of the solution field to the original nonlinear problem.

We present numerical examples with both linear and nonlinear PDEs to test the performance of VarPro. For smooth field solutions, the VarPro errors decrease exponentially as the number of collocation points or the number of output-layer coefficients increases, which is reminiscent of the spectral convergence of traditional high-order methods. We also compare the performance of the VarPro method with that of PINN [24].

2 Variable Projection with Artificial Neural Networks for Computational PDEs

2.1 Linear PDEs

Consider the following generic linear boundary-value problem on domain Ω,

$$Lu = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{1a}$$

$$Bu = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \tag{1b}$$

Here $u(\mathbf{x})$ is the field solution to be solved for, L denotes a linear differential operator, B denotes a linear operator on the boundary $\partial\Omega$ representing the boundary conditions, and $f(\mathbf{x})$ and $g(\mathbf{x})$ are prescribed non-homogeneous terms in the domain or on the boundary.

We approximate $u(\mathbf{x})$ by a feed-forward neural network, and require that the output layer be linear and contain no bias. Let $\Phi_j(\boldsymbol{\theta}, \mathbf{x})$ $(1 \leqslant j \leqslant M)$ denote the output fields of the last hidden layer, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{N_h})^T$ denotes the vector of weight/bias coefficients in all the hidden layers of the network. Then the NN logic leads to the following relation,

$$u(\mathbf{x}) = \sum_{j=1}^{M} \beta_j \Phi_j(\boldsymbol{\theta}, \mathbf{x}) = \boldsymbol{\Phi}(\boldsymbol{\theta}, \mathbf{x})\boldsymbol{\beta} \tag{2}$$

where $\boldsymbol{\Phi}(\boldsymbol{\theta}, \mathbf{x}) = [\Phi_1(\boldsymbol{\theta}, \mathbf{x}), \ldots, \Phi_M(\boldsymbol{\theta}, \mathbf{x})]$ denotes the set of output fields of the last hidden layer, and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^T$ is the vector of output-layer weights. Note that $(\boldsymbol{\theta}, \boldsymbol{\beta})$ are the trainable parameters of the neural network.

We choose a set of N $(N \geqslant 1)$ collocation points on Ω. Among them N_b $(1 \leqslant N_b \leqslant N-1)$ collocation points reside on the boundary $\partial\Omega$, and the rest of the points are from the interior of Ω. We use $\mathbb{X}$ to denote the set of all the collocation points and $\mathbb{X}_b$ to denote the set of collocation points on $\partial\Omega$.

Table 1: Poisson equation: comparison of the maximum/rms errors and network training time by VarPro and PINN/Adam. NN architecture [2,100,1].

collocation	VarPro			PINN		
points	max-error	rms-error	train-time(secs)	max-error	rms-error	train-time(secs)
5×5	$6.470E+1$	$2.422E+1$	$1.85E+0$	$7.65E-1$	$2.13E-1$	$3.14E+1$
10×10	$8.388E-3$	$3.941E-3$	$5.13E+0$	$1.11E-2$	$2.03E-3$	$1.12E+2$
15×15	$6.018E-7$	$8.241E-8$	$2.17E+1$	$1.83E-2$	$2.57E-3$	$1.43E+2$
20×20	$3.693E-7$	$4.216E-8$	$2.88E+1$	$1.80E-2$	$2.57E-3$	$1.81E+2$

Inserting the relation (2) into (1), and enforcing the equation (1a) on all the collocation points from $\mathbb{X}$ and the equation (1b) on all the boundary collocation points from $\mathbb{X}_b$, we arrive at the following system,

$$\sum_{j=1}^{M} \left[L\Phi_j(\boldsymbol{\theta}, \mathbf{x}_p)\right] \beta_j = f(\mathbf{x}_p), \quad 1 \leqslant p \leqslant N, \text{ where } \mathbf{x}_p \in \mathbb{X}, \tag{3a}$$

$$\sum_{j=1}^{M} \left[B\Phi_j(\boldsymbol{\theta}, \mathbf{x}_q)\right] \beta_j = g(\mathbf{x}_q), \quad 1 \leqslant q \leqslant N_b, \text{ where } \mathbf{x}_q \in \mathbb{X}_b. \tag{3b}$$

This is a system of $(N + N_b)$ algebraic equations about the trainable parameters $(\boldsymbol{\theta}, \boldsymbol{\beta})$, with $(N_h + M)$ unknowns.

We seek a least squares solution for $(\boldsymbol{\theta}, \boldsymbol{\beta})$ to the system (3). This system is linear with respect to $\boldsymbol{\beta}$, and nonlinear with respect to $\boldsymbol{\theta}$. This leads to a separable nonlinear least squares problem. To make the formulation more compact, we re-write the system (3) into a matrix form,

$$\mathbf{H}(\boldsymbol{\theta})\boldsymbol{\beta} = \mathbf{S}, \tag{4}$$

where $\mathbf{H}(\boldsymbol{\theta}) \in \mathbb{M}^{(N+N_b)\times M}$ and $\mathbf{S} \in \mathbb{R}^{(N+N_b)}$. For any given $\boldsymbol{\theta}$, the least squares solution for the linear parameters $\boldsymbol{\beta}$ to this system is given by $\boldsymbol{\beta} = [\mathbf{H}(\boldsymbol{\theta})]^{+}\,\mathbf{S}$, where the superscript in $\mathbf{H}^{+}$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{H}$. Define the residual of the system (4) by

$$\mathbf{r}(\boldsymbol{\theta}) = \mathbf{H}(\boldsymbol{\theta})\boldsymbol{\beta} - \mathbf{S} = \mathbf{H}(\boldsymbol{\theta})\mathbf{H}^{+}(\boldsymbol{\theta})\mathbf{S} - \mathbf{S}. \tag{5}$$

We compute the optimal nonlinear parameters $\boldsymbol{\theta}_{opt}$ by minimizing the Euclidean norm of $\mathbf{r}$,

$$\boldsymbol{\theta}_{opt} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\mathbf{r}(\boldsymbol{\theta})\|^2 = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\mathbf{H}(\boldsymbol{\theta})\mathbf{H}^{+}(\boldsymbol{\theta})\mathbf{S} - \mathbf{S}\|^2. \tag{6}$$

After $\boldsymbol{\theta}_{opt}$ is obtained, we can compute the optimal linear parameters $\boldsymbol{\beta}_{opt}$ by solving equation (4) using the linear least squares method. The problem represented by (6) is a nonlinear least squares problem about $\boldsymbol{\theta}$ only, where the linear parameter $\boldsymbol{\beta}$ has been eliminated. We solve this problem by a Gauss-Newton algorithm combined with a trust region strategy [6, 9, 7, 28].

2.2 Nonlinear PDEs

Consider the following nonlinear boundary value problem,

$$Lu + F(u) = f(\mathbf{x}), \tag{7a}$$

$$Bu + G(u) = g(\mathbf{x}), \quad \text{on } \partial\Omega, \tag{7b}$$

where $F(u)$ and $G(u)$ are nonlinear operators on the solution field $u(\mathbf{x})$ and also possibly on its derivatives, and L, B, f and g have the same meanings as in the equations (1a)–(1b).

To exploit VarPro, we first linearize the system (7a)–(7b) with the Newton's method. Let u^k denote the approximation of the solution at the k-th Newton iteration. We linearize this system as follows,

$$Lu^{k+1} + F(u^k) + F'(u^k)\left(u^{k+1} - u^k\right) = f(\mathbf{x}), \tag{8a}$$

$$Bu^{k+1} + G(u^k) + G'(u^k)\left(u^{k+1} - u^k\right) = g(\mathbf{x}), \quad \text{on } \partial\Omega, \tag{8b}$$

where $F'(u)$ and $G'(u)$ denote the derivatives with respect to u. We further re-write the linearized system into,

$$Lu^{k+1} + F'(u^k)u^{k+1} = f(\mathbf{x}) - F(u^k) + F'(u^k)u^k, \tag{9a}$$

$$Bu^{k+1} + G'(u^k)u^{k+1} = g(\mathbf{x}) - G(u^k) + G'(u^k)u^k, \quad \text{on } \partial\Omega. \tag{9b}$$

Given u^k, this system represents a linear boundary value problem about the updated approximation field u^{k+1}. Therefore, the VarPro/ANN algorithm from Section 2.1 can be used to solve this linearized system (9a)–(9b) for u^{k+1}. Upon convergence of the Newton iteration, the solution to the original nonlinear system (7a)–(7b) will be obtained and represented by the neural-network coefficients.

3 Numerical Examples

3.1 Poisson Equation

We consider the 2D Poisson equation $\nabla^2 u = f(x, y)$ with Dirichlet BCs on the unit square domain $\Omega = [0, 1] \times [0, 1]$, with an analytic solution as shown in Fig. 1(a). Fig. 1(b) depicts the point-wise error distribution of the VarPro solution obtained on an NN architecture $[2, 20, 100, 1]$ with the cosine activation function, showing a maximum error on the order 10^{-9} in the domain. Fig. 1(c) shows the the VarPro maximum/rms errors versus the collocation points, indicating an exponential convergence. Table 1 compares the maximum/rms errors and the network training time between VarPro and PINN, showing that VarPro is much more accurate and computationally more efficient.

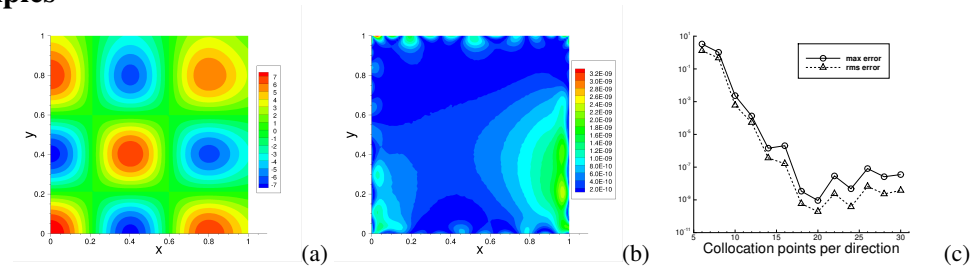

Figure 1: Poisson equation: (a) exact solution. (b) Pointwise error of the VarPro solution. (c) Maximum/rms errors of VarPro versus the number of collocation points per direction. NN architecture [2, 20, 100, 1], cos activation function.

3.2 Viscous Burgers Equation

We next consider the viscous Burgers equation on the domain $(x, t) \in [-1, 1] \times [0, 1.05]$, $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{1}{100\pi}\frac{\partial^2 u}{\partial x^2}$, with homogeneous BC and the initial condition $u(x, 0) = -\sin(\pi x)$. This problem has a solution in which a sharp gradient (akin to a jump discontinuity) develops over time. Figs. 2(a,b) show distributions of the VarPro solution and its point-wise absolute error in the domain, with a maximum error on the order 10^{-5}. Fig. 2(c) compares the solution profiles between the VarPro solution and the exact solution at $t = 1$, illustrating the sharp gradient in the domain. Block time marching and domain decomposition (with 6 sub-domains) have been used with VarPro for this problem, with a local NN architecture $[2, 250, 1]$ on each sub-domain. VarPro result is highly accurate.

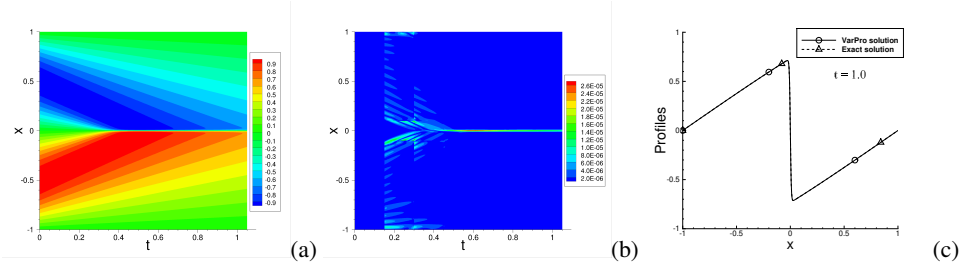

Figure 2: Burgers equation: distributions of (a) VarPro solution, and (b) its point-wise error. (c) Comparison of profiles between the VarPro solution and exact solution at $t = 1$. Domain decomposition and block time marching are used.

4 Concluding Remarks

We have presented a variable projection-based method together with artificial neural networks for solving linear and nonlinear partial differential equations. The basic idea of variable projection (VarPro) is to distinguish the linear parameters from the nonlinear parameters, and then eliminate the linear parameters to attain a reduced formulation of the problem. One can then solve the reduced problem for the nonlinear parameters first, and then compute the linear parameters by the linear least squares method afterwards. For smooth field solutions, the errors of the VarPro method decrease exponentially or nearly exponentially with increasing number of collocation points or with increasing number of output-layer coefficients. The test results show that the VarPro method is highly accurate. Even with a fairly small number of nodes in the neural network, or with a fairly small set of collocation points, the VarPro method can produce very accurate simulation results.

Acknowledgments

This work was partially supported by US National Science Foundation (DMS-2012415).

References

[1] T. Askham and J.N. Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM J. Applied Dynamical Systems*, 17:380–416, 2018.

[2] A. Bjorck. *Numerical Methods in Matrix Computations*. Springer, 2015.

[3] G.-Y. Chen, M. Gan, C.L.P. Chen, and H.-X. Li. A regularized variable projection algorithm for separable nonlinear least-squares problems. *IEEE Transactions on Automatic Control*, 64:526–537, 2019.

[4] J. Chung, E. Haber, and J. Nagy. Numerical methods for coupled super-resolution. *Inverse Problems*, 22:1261–1272, 2006.

[5] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, 1996.

[6] S. Dong and Z. Li. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 387:114129, 2021. (also arXiv:2012.02895).

[7] S. Dong and Y. Wang. A method for computing inverse parametric pde problems with random-weight neural networks. *Journal of Computational Physics*, 489:112263, 2023. (also arXiv:2210.04338).

[8] S. Dong and J. Yang. Numerical approximation of partial differential equations by a variable projection method with artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, 398:115284, 2022. (also arXiv:2201.09989).

[9] S. Dong and J. Yang. On computing the hyperparameter of extreme learning machines: algorithms and applications to computational PDEs, and comparison with classical and high-order finite elements. *Journal of Computational Physics*, 463:111290, 2022. (also arXiv:2110.14121).

[10] N.B. Erichson, P. Zheng, K. Manohar, J.N. Kutz S.L. Brunton, and A.Y. Aravkin. Sparse principal component analysis via variable projection. *SIAM J. Appl. Math.*, 80:977–1002, 2020.

[11] G.H. Golub and V. Pereyra. The differentiation of pseudo-inverse and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10:413–432, 1973.

[12] G.H. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19:R1–R26, 2003.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[14] G.E. Karniadakis, G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440, 2021.

[15] L. Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975.

[16] C.-T. Kim and J.-J. Lee. Training two-layered feedforward networks with variable projection method. *IEEE Transactions on Neural Networks*, 19:371–375, 2008.

[17] K.M. Mullen and I.H.M. van Stokkum. The variable projection algorithm in time-resolved spectroscopy, microscopy and mass spectrometry applications. *Numer. Algor.*, 51:319–340, 2009.

[18] E. Newman, J. Chung, M. Chung, and L. Ruthotto. SlimTrain – a stochastic approximation method for training separable deep neural networks. *arXiv:2109.14002*, 2021.

[19] E. Newman, L. Ruthotto, J. Hart, and B. van Bloemen Waanders. Train like a (Var)Pro: Efficient training of neural networks with variable projection. *arXiv:2007.13171*, 2020.

[20] N. Ni and S. Dong. Numerical computation of partial differential equations by hidden-layer concatenated extreme learning machine. *Journal of Scientific Computing*, 95:35, 2023. (also arXiv:2204.11375).

[21] D.P. O'Leary and B.W. Rust. Variable projection for nonlinear least squares problems. *Comput. Optim. Appl.*, 54:579–593, 2013.

[22] M.R. Osborne. Separable least squares, variable projection, and the gauss-newton algorithm. *Electronic Transactions on Numerical Analysis*, 28:1–15, 2007.

[23] V. Pereyra, G. Scherer, and F. Wong. Variable projections neural network training. *Mathematics and Computers in Simulation*, 73:231–243, 2006.

[24] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[25] A. Ruhe and P.A. Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22:318–337, 1980.

[26] J. Sjoberg and M. Viberg. Separable nonlinear least squares minimization - possible improvements for neural net fitting. *Neural Networks for Signal Processing VII. Proceedings of IEEE Signal Processing Workshop*, 1997.

[27] X. Song, W. Xu, K. Hayami, and N. Zheng. Secant variable projection method for solving nonnegative separable least squares problems. *Numerical Algorithms*, 85:737–761, 2020.

[28] Y. Wang and S. Dong. An extreme learning machine-based method for computational pdes in higher dimensions. *Computer Methods in Applied Mechanics and Engineering*, 418:116578, 2024.

[29] K. Weigl and M. Berthod. Neural networks as dynamical bases in function space. *Report No 2124, INRIA, Sophis-Antipolis, France*, 1993. URL: https://hal.inria.fr/inria-00074548/document.

[30] K. Weigl and M. Berthod. Projection learning: alternative approach to the computation of the projection. *Proc. European Symp. on Artificial Neural Networks, Brussels, Belgium*, pages 19–24, 1994.

[31] K. Weigl, G. Giraudon, and M. Berthod. Application of projection learning to the detection of urban areas in SPOT satellite images. *Report No 2143, INRIA, Sophia-Antipolis, France*, 1993. URL: https://hal.inria.fr/inria-00074529.