Stanford University

ENERGY 260
Uncertainty Quantification in Data-Centric Simulation

Project Report

Effectiveness of Latin Hypercube Sampling

Anthony Boukarim
Mohammad (Jabs) Aljubran

03/20/2019

Introduction:

Numerical simulations represent a common technique to model real world phenomena, e.g. fluid flow, thermodynamics, etc. These simulations can be computationally expensive, requiring several hours or days to complete a single evaluation based on a specific set of inputs. Difficulty arises when the system input parameters are uncertain—commonly treated as random variables with a probability distribution function (PDF). Hence, the numerical experiment is repeated multiple times to find the PDF of the output variable, requiring successive selections of input variables. Given the high computational expense of numerical simulations, this successive sampling process must be conducted with caution to minimize the time required to arrive to the output solution.

McKay, Beckman, and Conover (1979) proposed Latin Hypercube Sampling (LHS) as an alternative and more effective technique to random and stratified sampling techniques in fluid flow numerical simulations, i.e. computational fluid dynamics (CFD). Besides filling all portions of the input space, LHS highlights the importance of individual inputs by including all portions of the PDF of each input variable. Literature offers multiple LHS implementations: simple LHS, LHS with reduction spurious correlation, standard importance LHS, transformed importance LHS, axis importance LHS, and axis importance correlation LHS (Olsson et al. 2003). This project will explore the effectiveness of simple LHS in several applications: Monte Carlo Simulation (MCS), function proxy construction, machine learning (ML) algorithm training.

Monte Carlo Simulation:

We will start by investigating the rate of convergence of Monte Carlo using both random and Latin hypercube sampling. The simulator used is MATLAB with optimized functions for both sampling techniques. The first case represents a sampling exercise from a Gaussian distribution with mean $\mu = 1$ and variance $\sigma^2 = 0.1$. It is expected from literature that the convergence rate of a Quasi-Monte Carlo (such as Latin Hypercube Sampling) to be in the order of $O(\frac{1}{\sqrt{N}})$ whereas in the order of $O(\frac{1}{N})$ for standard Monte Carlo (Asmussen and Glenn 2007).

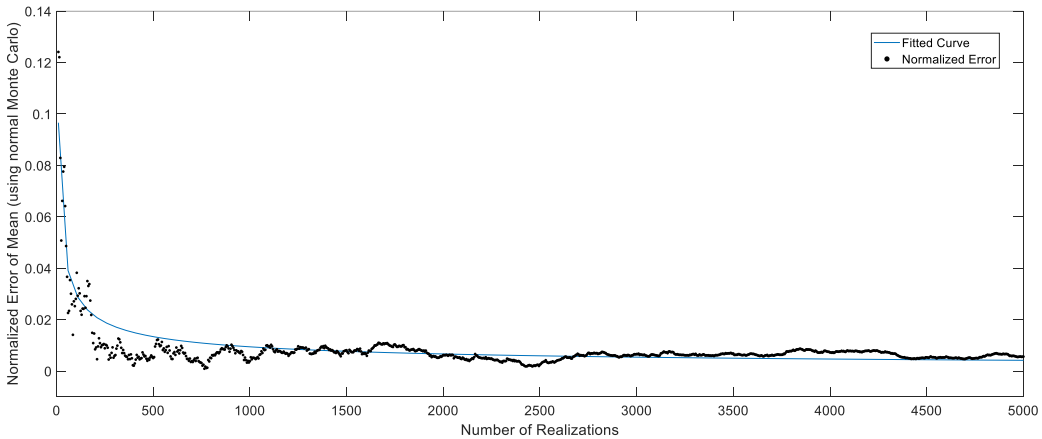

Fig. 1— Convergence of Monte Carlo for a Normal Distribution Using Simple Random Sampling: The figure above represents the normalized error of the mean using a standard Monte Carlo approach on a Gaussian distribution ($\mu = 1, \sigma^2 = 0.1$). The best fit using least squares error is given by a power law (ax^b) with coefficients: $a = 0.309$ & $b = -0.5049$. The adjusted R^2 is

81.2%. This confirms that Monte Carlo converges in the order of $\sim\frac{1}{\sqrt{N}}$. These results are reported after running the simulations 10 times and averaging the outcomes.

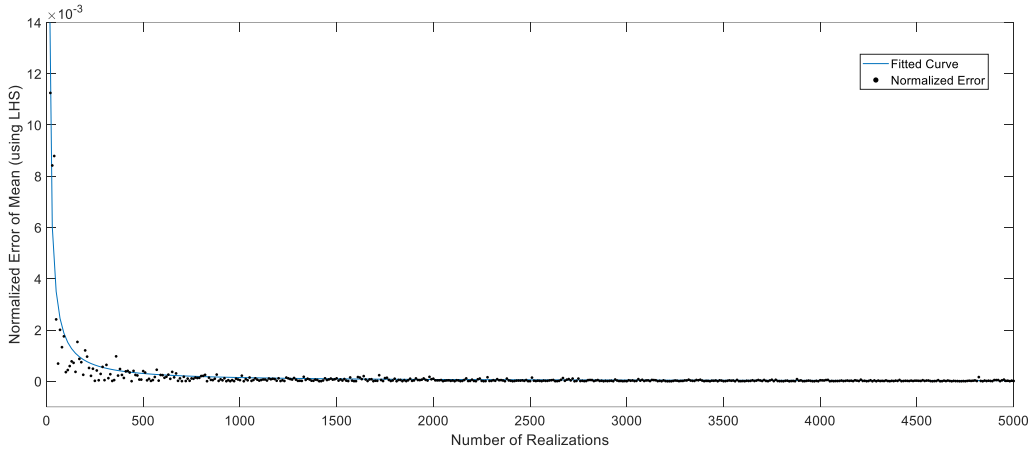

Fig. 2— Convergence of LHS for a Normal Distribution: The figure above represents the normalized error of the mean using LHS on a Gaussian distribution ($\mu = 1, \sigma^2 = 0.1$). The best fit using least squares error is given by a power law (ax^b) with coefficients: $a = 0.2231$ & $b = -1.062$. The adjusted R^2 is 92.4%. This confirms that the LHS improves the convergence to an order of $\sim\frac{1}{N}$. These results are reported after running the simulations 10 times and averaging the outcomes.

After running several simulations and averaging the outcomes, our results for this exercise can be summarized as follows: The convergence rate of standard Monte Carlo was following a power law of equation $\frac{0.309}{x^{-0.5049}}$, seen in **Fig. 1**, with an adjusted R^2 of 81.2%. Subsequently, sampling from the same distribution using LHS displayed a convergence rate following a power law of equation $\frac{0.2231}{x^{-1.062}}$, seen in **Fig. 2**, with an adjusted R^2 of 92.4%. These results are in line with the predicted convergence rates from literature review.

The second case consisted of repeating the same sampling exercise but from an Exponential distribution with mean $\mu = 0.69$. There is no clear literature review describing convergence rates depending on the chosen distributions but the same convergence range is expected.

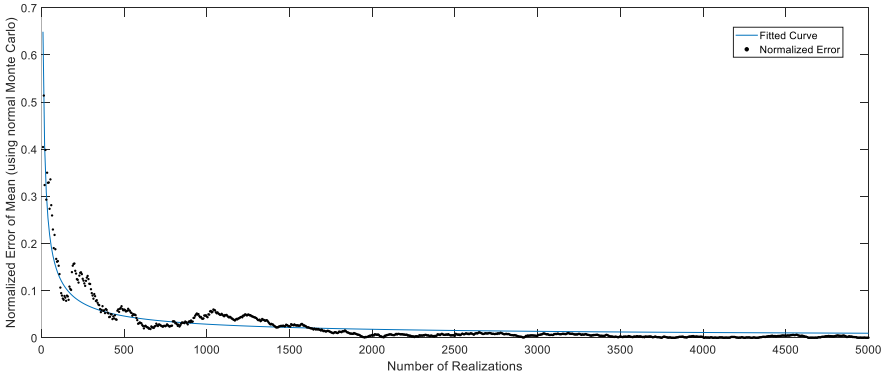

Fig. 3— Convergence of Monte Carlo for an Exponential Distribution Using Simple Random Sampling: The figure above represents the normalized error of the mean using a standard Monte Carlo approach on an Exponential distribution ($\mu = 0.69$). The best fit using least squares error is given by a power law (ax^b) with coefficients: $a = 0.3073$ & $b = -0.6752$. The adjusted R^2 is 85.57%. This shows a trend of convergence in the order of $\sim\frac{1}{\sqrt{N}}$. These results are reported after running the simulations 10 times and averaging the outcomes.

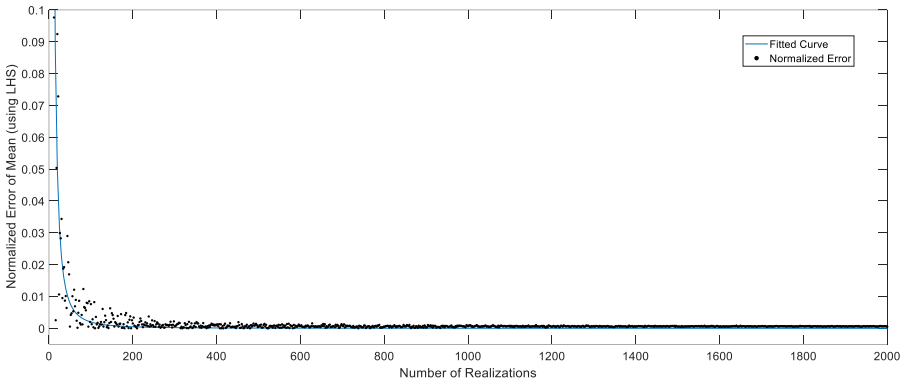

Fig. 4— Convergence of LHS for an Exponential Distribution: The figure above represents the normalized error of the mean using LHS on an Exponential distribution ($\mu = 0.69$). The best fit using least squares error is given by a power law with one term (ax^b) with coefficients: $a = 19.77$ & $b = -1.999$. The adjusted R^2 is 85.68%. This predicts that the LHS on an exponential distribution improves the convergence to an order of $\sim\frac{1}{N^2}$. These results are reported after running the simulations 10 times and averaging the outcomes.

As seen in **Fig. 3**, the best fit curve using least squares when using simple random sampling has a power law equation of $\frac{0.3073}{x^{-0.6752}}$ with an adjusted R^2 of 85.57%. Conversely, using LHS resulted in a much faster convergence rate in the order of $O(\frac{1}{N^2})$. This is demonstrated in **Fig. 4** by a power equation of $\frac{19.77}{x^{-1.999}}$ with an adjusted R^2 of 85.68%.

This exercise has proven (using simulation) that LHS can drastically improve the convergence rate of the mean, and thus the overall computational cost of a stochastic problem.

Function Proxy Construction:

To complement our analysis on LHS, we performed a Gaussian-Kernel Support Vector Machine (SVM) curve-fitting exercise to build a proxy for complicated functions (mostly non-linear). We experimented on three common functions: $f(x, y) = \sin(x) + \cos(y)$, Rosenbrock and Himmelblau functions. The mean squared error (MSE) was used to compare the proxy with the actual function for both random sampling and LHS techniques.

Looking at the MSE analysis in **Figs. 5 and 6**, we observe that LHS outperforms random sampling for both the simple trigonometric and Rosenbrock function. However, applying LHS to the Himmelblau function, seen in **Fig. 7**, did not provide an evident advantage over random sampling. We conclude that combining LHS with the Gaussian-Kernel SVM fitting can be beneficial for proxy building, yet this cannot be generalized given that it depends on the non-linearity of the function at hand.

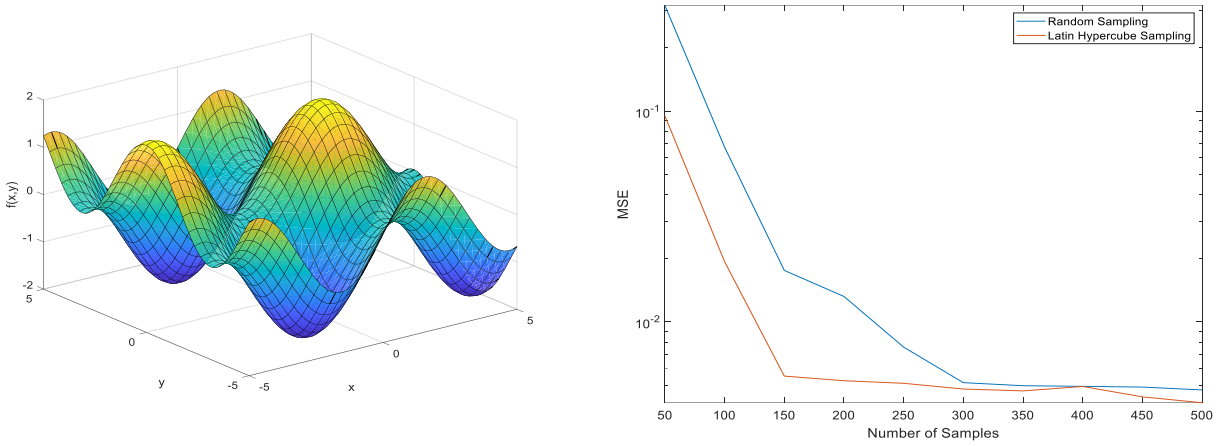

Fig. 5— a) 3D Plot of $f(x,y) = sin(x) + cos(y)$. b) Mean Squared Error of Random and Latin Hypercube Sampling Techniques: For the given trigonometric function, we conclude that LHS outperforms random sampling when using SVM.

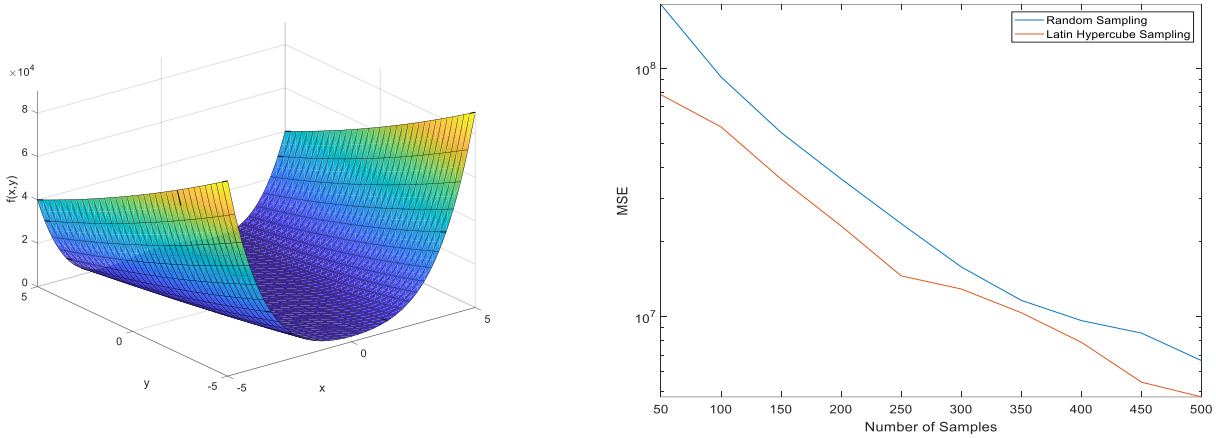

Fig. 6— a) Rosenbrock 3D Plot. b) Mean Squared Error of Random and Latin Hypercube Sampling Techniques: Fitting with SVM after sampling using LHS on the Rosenbrock function yields a better fit than random sampling.

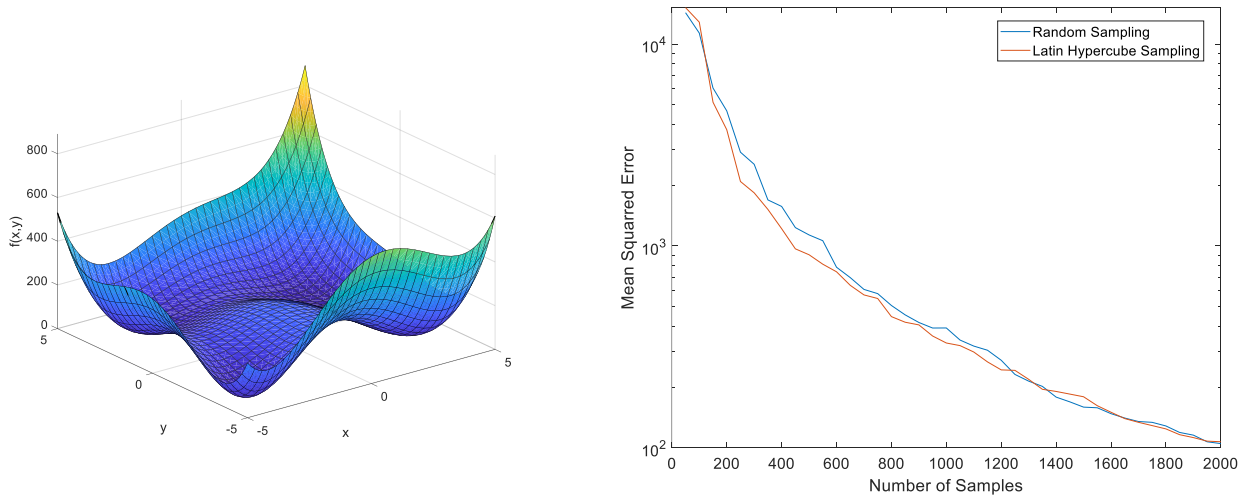

Fig. 7— a) Himmelblau Function 3D Plot. b) Mean Squared Error of Random and Latin Hypercube Sampling Techniques: The mean squared error analysis for the Himmelblau function doesn't provide a clear conclusion on whether LHS is a better sampling technique than random sampling.

Machine Learning Algorithms:

Recent technological innovation in drilling and completion allowed for achieving complex multilateral wells with significantly higher recovery rates at marginal additional costs. These advanced installations resulted in improved reservoir contact, ultimate recovery, zonal isolation and pressure drawdown control, and management of water/gas breakthrough. Equipped with state-of-the-art instrumentation and control systems, smart wells provide continuous flowrate and pressure data streams in space and time. One major component of these completions is the inflow control valve (ICV) which regulates the flow of reservoir fluids from different branches.

Numerical reservoir simulations represent the standard approach to analyzing oil and gas fields and deciding on the optimal ICV settings. However, using a solver, e.g. genetic algorithm (GA), to optimize over numerical simulators involves many computationally expensive function evaluations. Hence, surrogate-based optimization (SBO), seen in **Fig. 8**, is commonly used to efficiently find the optimal decision variables. Yet, an accurate, fast, and computationally inexpensive surrogate model is critical to SBO. Based on a few function evaluation sample points, proxy formulation techniques, i.e. Kriging, machine learning, etc., have to accurately map the objective function across the problem domain for the optimizer to locate the global optimum. This section will LHS to interleaved sampling by replicating the first step of SBO: initial sampling.

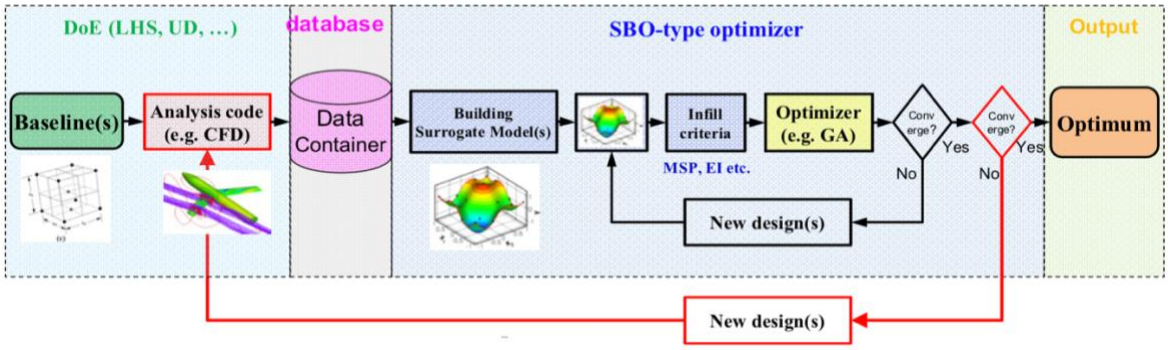

Fig. 8—SBO Approach: SBO uses a space-filling sampling design initially at which the reservoir model is evaluated to construct the initial database. A surface response is then constructed using a surrogate modeling technique which is used for optimization. Additional samples are selected adaptively to evaluate the model and add to the database for a new optimization iteration to be conducted. This process is repeated until convergence is achieved based on a set of selected criteria (Han 2016).

Eclipse, a commercial reservoir simulator, is used to construct a reservoir model, seen in **Table 1**. This model, seen in **Figs. 9**, simulates a homogenous oil reservoir with two-phase flow (oil and water), mainly driven by an underlying water aquifer. A segmented, trilateral producer well is drilled at the grid center and completed across the reservoir top layer with three ICV devices (each holds 11 discrete settings from 0 to 10 indicating constriction area range of 0 to 0.001 ft^2) which are installed at each lateral tie-in segment. Each simulation run outputs wellhead flow rates, and downhole flow rates and pressure drops at each ICV over a period of 2,000 days.

Table 1—Eclipse Model Input: Rock, fluid, and well input parameters that are used to generate the physical reservoir model for the optimization problem.

Reservoir Size (ft^3)	2000 x 900 x 270	Payzone Thickness (ft)	150
Porosity	10%	Horizontal Perm (mD)	10
Vertical Perm (mD)	0.1	Compressibility (psi^{-1})	3E-6 @ 14.7 psi
OWC (ft)	7150	Reservoir Pressure (psi)	6000
Oil Density (pcf)	40	Water Density (pcf)	62
Water Aquifer	Carter-Tracy	Aquifer Thickness (ft)	120
Segment Count	16	Branch Count	3
Tubing ID (ft)	36%	Tubing Roughness (ft)	0.001
Max Oil Rate (STB/D)	5000	Target BHP (psi)	2000 psi
ICV Setting "0" Area (ft^2)	0	ICV Setting "10" Area (ft^2)	0.001

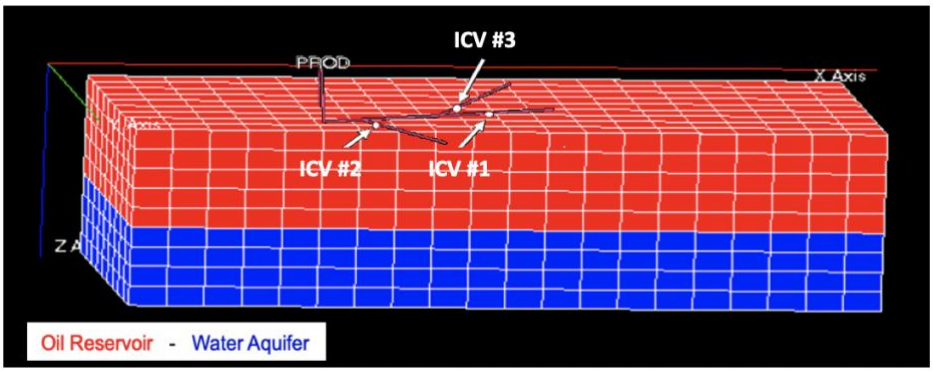

Fig. 9—Eclipse Model: Cartesian Eclipse reservoir model at the initial state where the oil reservoir (red) is 100% saturated with oil and occupies the upper layers while the water aquifer (blue) is 100% saturated with water and occupies the lower layers. A single trilateral producer situated around the center block. Each lateral is equipped with an ICV device around the tie-in point to control reservoir fluids inflow into the wellbore.

A total of 1,330 simulation runs are conducted with an eight-day timestep, generating 667,660 temporal sample points. **Table 2** shows a simplified example of the task that needs to be performed by the proxy. The oil and water cumulative flow rates of cases 1,2 and 3 are generated using Eclipse which then are used to train an ANN model. The latter acts as a surrogate to predict the cumulative production at the cases 5,6, and 7 where the ICV settings are different and originally unseen by the ANN.

Table 2—Problem Example: The ANN approach is responsible for generating the oil and water cumulative production profiles for unseen ICV settings when given a few others.

Variable	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
ICV #1	10	8	1	0	3	7
ICV #2	10	10	3	6	4	5
ICV #3	10	0	2	2	5	7
Oil and Water Cumulative Production	Given Data			Predicted Output		

A simple one-layer, ten-neuron regularized ANN, seen in **Fig. 10**, was chosen for this exercise. The ANN training process followed the MATLAB implementation of the Levenberg-Marquardt algorithm. It alternates between the Gauss-Newton and gradient descent methods to approximate the Hessian matrix by using a performance/learning gain measure that is adjusted adaptively throughout the learning process to control the learning rate and ensure robust convergence (MacKay 1992; Foresee and Hagan 1997). Meanwhile, the regularization parameter λ is optimized using the Bayesian regularization backpropagation (BRB) algorithm which locates and trains on

effective weights while effectively turning off the irrelevant counterparts. Note that each training sample point represents the ICV settings (input variables) and cumulative oil and water production after 2,000 days (output variables).

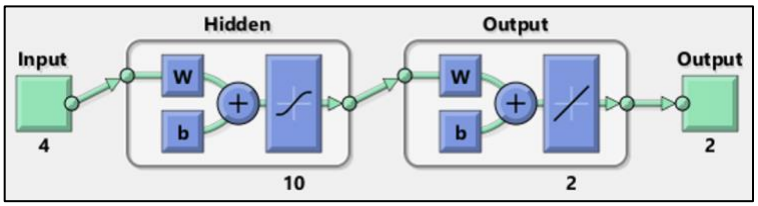

Fig. 10—ANN architecture: ANN is used to generate the statistical model. Note that there are four inputs (Time and settings of ICV #1,2 and 3) and two outputs: cumulative oil and water production at the end of the chosen 2,000-day simulation period. Note that 'W' and 'b' represent the weights and bias parameters while each neuron uses sigmoid as the transfer function.

In training ANN, choosing the training and validation data is critical to guarantee it actually represents the global sample. Otherwise, the training error could be low while the validation and/or testing errors are high, indicating loss of generalization for that specific model. Since this is a regression supervised learning ML problem, adjusted regression correlation coefficient (R^2) will be used as the evaluation metric of the trained ANN performance on the unseen testing data. The premise of this evaluation is to assess the impact of using LHS, seen in **Fig. 11**, in training ANN on the given dataset.

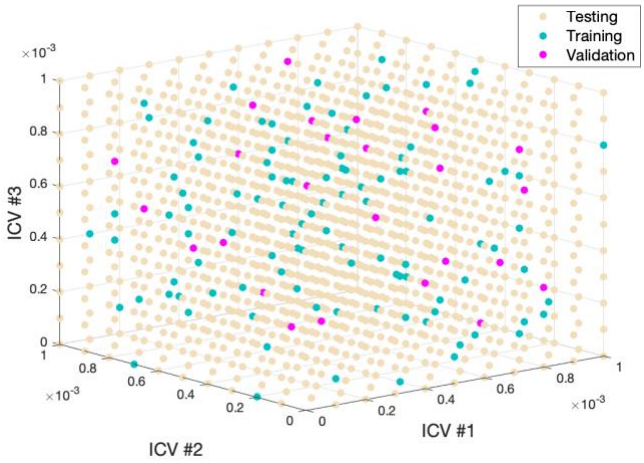

Fig. 11—LHS Approach: Example of 50 samples conducted using LHS, which shows effective space coverage and representation of the global sample which improves the results of training ANN.

Note that the ANN weights (parameters to be optimized for to find the optimal ANN fit) are initiated using the Nguyen-Widrow algorithm (1990) which ensures that the initially active region of the network neurons is evenly distributed to cover the input space. However, this algorithm involves randomness which results in different ANN optimal weight results for different training sessions. Hence, training will be repeated several times for each sampling technique to evaluate the stability and convergence to of ANN regression on testing data to $R^2 \cong 1$. ANN is trained and validated using different sample sizes using LHS and interleaved sampling, seen in **Fig. 12**. Note that the ANN R^2 performance on the testing sample generally improves as the training sample size increases. However, LHS is evidently superior to interleaved sampling at small training sample sizes. In addition, LHS reflects better stability as training is repeated compared to interleaved

sampling which suffers from oscillation. Overall, LHS is more effective as it provides a better multi-dimensional coverage.

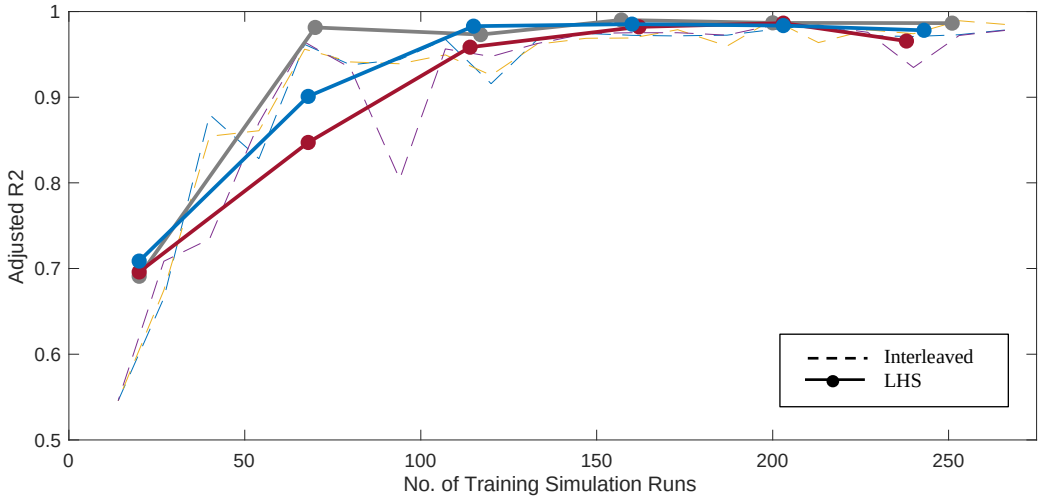

Fig. 12—LHS and Interleaved Sampling: Comparison of LHS and interleaved sampling for ANN training purposes proves that LHS is more robust and especially superior to interleaved sampling at lower sample sizes.

Conclusions:

Latin Hypercube Sampling is found to be superior to simple random sampling in Monte Carlo simulations when using both Gaussian and exponential probability distribution functions. Experimental results on given examples and exercises matched the theoretically derived literature convergence rates. LHS was also applied to enhance the process of building proxy function using support vector machines with Gaussian kernel, and showed improved results in some cases and almost neutral results when compared to random sampling. Furthermore, LHS showed an improved performance in artificial neural network training efficiency when higher regression fit results for a given sample size. This eventually facilitates the applicability of surrogate-based optimization approach. LHS

References:

[1] Abukhamsin, A.Y. (2017). Inflow profiling and production optimization in Smart wells using distributed acoustic and Temperature measurements (Unpublished doctoral dissertation). Stanford University, CA.

[2] Asmussen, S., & Glynn, P. W. (2007). *Stochastic simulation: algorithms and analysis* (Vol. 57). Springer Science & Business Media.

[3] Burden, F., & Winkler, D. (2008). Bayesian regularization of neural networks. In Artificial neural networks (pp. 23-42). Humana Press.

[4] Foresee, F. D., & Hagan, M. T. (1997, June). Gauss-Newton approximation to Bayesian learning. In Proceedings of International Conference on Neural Networks (ICNN'97) (Vol. 3, pp. 1930-1935). IEEE.

[5] MacKay, D.J. (1992). Bayesian Interpolation. Neural Computation, Vol. 4, No. 3, 1992, pp. 415 to 447).

[6] McKay, M. D., Beckman, R. J., & Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 42(1), 55-61.

[7] Nguyen, D., & Widrow, B. (1990, June). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In 1990 IJCNN International Joint Conference on Neural Networks (pp. 21-26). IEEE.

[8] Olsson, A., Sandberg, G., & Dahlblom, O. (2003). On Latin hypercube sampling for structural reliability analysis. Structural safety, 25(1), 47-68.