Research, Society and Development, v. 9, n. 2, e188922128, 2020
(CC BY 4.0) | ISSN 2525-3409 | DOI: http://dx.doi.org/10.33448/rsd-v9i2.2128

Sintonia Heurística e Análise de Convergência de Algoritmo de Aprendizagem por Reforço para Projeto de Controle Ótimo Baseado em Dados

Tuning Heuristics and Convergence Analysis of Reinforcement Learning Algorithm for Online Data-Based Optimal Control Design

Heurística de Ajuste y Análisis de Convergencia Del Algoritmo de Aprendizaje por Refuerzo para el Proyecto de Control Óptimo Basado en Datos Online

Recebido: 18/11/2019 | Revisado: 19/11/2019 | Aceito: 03/12/2019 | Publicado: 12/12/2019

Fábio Nogueira da Silva
ORCID: https://orcid.org/0000-0001-7215-6520
Universidade Federal do Maranhão, Brasil
E-mail: fndasilva@hotmail.com
João Viana da Fonseca Neto
ORCID: http://orcid.org/0000-0003-4606-7510
Universidade Federal do Maranhão, Brasil
E-mail: vianafonseca@gmail.com

Resumo

Uma heurística para sintonia e análise de convergência do algoritmo de aprendizado por reforço para controle com realimentação de saída com apenas dados de entrada / saída, gerados por um modelo, são apresentados. Para promover a análise de convergência, é necessário realizar o ajuste dos parâmetros nos algoritmos utilizados para a geração de dados, e iterativamente resolver o problema de controle. É proposta uma heurística para ajustar os parâmetros do gerador de dados criando superfícies para auxiliar no processo de análise de convergência e robustez da metodologia de controle ótimo on-line. O algoritmo testado é o regulador quadrático linear discreto (DLQR) com realimentação de saída, baseado em algoritmos de aprendizado por reforço através do aprendizado por diferença temporal no esquema de iteração de política para determinar a política ideal usando apenas dados de entrada / saída. No algoritmo de iteração de política, o RLS (Mínimos Quadrados Recursivos) é usado para estimar parâmetros on-line associados ao DLQR com realimentação de saída. Após a aplicação das heurísticas propostas para o ajuste, a influência dos parâmetros pôde ser vista claramente, e a análise de convergência e facilitada.

Palavras-chave: Controle Ótimo; Aprendizagem por Reforço; Programação Dinâmica Aproximada; Realimentação de Saída; Sintonia.

Abstract

A heuristic for tuning and convergence analysis of the reinforcement learning algorithm for control with output feedback with only input / output data generated by a model is presented. To promote convergence analysis, it is necessary to perform the parameter adjustment in the algorithms used for data generation, and iteratively solve the control problem. A heuristic is proposed to adjust the data generator parameters creating surfaces to assist in the convergence and robustness analysis process of the optimal online control methodology. The algorithm tested is the discrete linear quadratic regulator (DLQR) with output feedback, based on reinforcement learning algorithms through temporal difference learning in the policy iteration scheme to determine the optimal policy using input / output data only. In the policy iteration algorithm, recursive least squares (RLS) is used to estimate online parameters associated with output feedback DLQR. After applying the proposed tuning heuristics, the influence of the parameters could be clearly seen, and the convergence analysis facilitated.

Keywords: Optimal Control; Reinforcement Learning; Approximate Dynamic Programming; Output Feedback; Tuning.

Resumen

Se presenta una heurística para el análisis de sintonía y convergencia del algoritmo de aprendizaje de refuerzo para el control con retroalimentación de salida con solo datos de entrada / salida generados por un modelo. Para promover el análisis de convergencia, es necesario realizar el ajuste de parámetros en los algoritmos utilizados para la generación de datos y resolver de forma iterativa el problema de control. Se propone una heurística para ajustar los parámetros del generador de datos creando superficies para ayudar en el proceso de análisis de convergencia y robustez de la metodología óptima de control online. El algoritmo probado es el regulador cuadrático lineal discreto (DLQR) con retroalimentación de salida, basado en algoritmos de aprendizaje de refuerzo a través del aprendizaje de diferencia temporal en el esquema de iteración de políticas para determinar la política óptima utilizando solo datos de entrada / salida. En el algoritmo de iteración de políticas, se utilizan mínimos cuadrados recursivos (RLS) para estimar los parámetros online asociados con la retroalimentación de salida DLQR. Después de aplicar las heurísticas de ajuste propuestas, se pudo ver claramente la influencia de los parámetros y se facilitó el análisis de convergencia.

Palabras clave: Control Óptimo; Aprendizaje por Refuerzo; Programación Dinámica Aproximada; Realimentación de Salida; Sintonización.

1. INTRODUCTION

The system states gather information from system dynamic, either for control or monitoring purposes in various kinds of systems such as: industrial, aerospace, energy, health, economics. The measurement of these states is done mostly by the use of sensors, which depending on the precision need in the study can be very expensive.

Sometimes is not possible to measure states with sensors, either by the unavailability of the sensor, because the it is either expensive, unsuitable for the application or purely mathematical. To solve these problems is done by the inclusion of a state observer, which is a mathematical formulation that from past observations of the system can estimate the states ahead in time. Applications with parameters estimators such as recurrent neural networks like Hopfield, were applied for real time estimation in (Alonso, Mendonça, & Rocha, 2009).

Sampled data from experiments are largely described by Markov Decision Processes Partially Observable (POMDP). The control methods applied to such systems are approached in (Sondik, 1971) and (Fleming, 1968).

Methodologies to speed up the convergence process of value interation algorithms in POMDP are discussed in (N. L. Zhang & Zhang, 2001). Optimal control problems with finite and infinite horizons are presented in (Smallwood & Sondik, 1973) and (Sondik, 1978), respectively. Works related to theoretical basis, as survey on the state of the art, theory, models, algorithms and tutorials for POMDP can be found in (Monahan, 1982; Lovejoy, 1991; W. Zhang, 2001; Littman, 2009; White III, 1991), respectively.

Alternatives to predict the behavior is to build a model from data, using statistical models, heuristics, modeling by the laws which governs the studied phenomena or using mathematical model associated with artificial intelligent algorithms to learn systems dynamics, which are discussed in (Chen, Billings, & Grant, 1990; Chu, Shoureshi, & Tenorio, 1990), (Alexander S. Poznyak, 2001) and (Nechyba & Xu, 1994). Each one of these methods have their own set of parameters to be tuning in order to achieve some desired performance.

Application of system identification techniques into process control is largely used by researches, as well as state observers for system state estimation that could not be directly measured. The inclusion of state observers is to minimize the effort related to the control action efficiently. Foundations of observers in nonlinear systems can be found in (Guildas, 2007) and (Anguelova, 2004).

The process of building models from data has advantages and disadvantages. A positive point is related to the possibility of building models for complex systems, with many variables. When the laws are not well understood, or when there is little information about the system dynamics available, building the model would be impossible or too costly. Disadvantage in using approximate models from data in some methods is related to memory requirements and power computing. Sometimes large amounts of memory are needed and great computing power as well, making the computation process time consuming or too expensive to be implemented.

Artificial intelligent algorithms based on reinforcement learning techniques with integral action are discussed in (Modares, Peen, Zhu, Lewis, & Yue, 2014) for on-line controller design, considering knowledge of partial information only. Other approaches for control application with partial information and learning are discussed in (Safonov & Tsao, 1997; Battistelli, Mari, Selvi, & Tesi, 2014). These references evaluate the possibility of identifying control actions that are able to achieve the performance specifications before being inserted in the feedback process. This technique is called "Unfalsified Control". This type of methodology is presented as an adaptive controller for selection of the most appropriate control actions with real time applications.

Recent projects for data-driven on-line controllers, data-driven for navigation system, pilot air-crafting and adaptive dweel-time switching using the methodology "Unfalsified Control" are discussed in (Saeki, Kondo, Wada, & Satoh, 2014; Yongqiang, Jiabin, Xiaochun, & Nan, 2015; Liming, Shan, & Dan, 2015; Sajjanshetty & Safonov, 2015), respectively.

Both control based on models and their advantages, as well as data-driven control are presented and discussed in (Hou, 2013), with theory and applications for adaptive controllers. The nomenclatures referring to methodologies based on data are large and sometimes confusing, such as driven or fed by data (Data-Driven), model-free, without model or not based on model methodologies (Model-Free). That book (Hou, 2013) proposes a classification of methods not based on models in two ways: one regarding on-line, off-line and hybrid methodologies, and another with respect to knowledge of the controller structure.

Model-free controllers based on subspace methods have been used to perform system identification, based on projections. One application of model-free Linear Quadratic Gaussian (LQG) subspace predictive control to TCP congestion control is discussed in (Chiera & White, 2008). Application techniques of optimal control and regulation, such as LQG, which makes the inclusion of a state observer are discussed in (Favoreel, De Moor, Gevers, & Van Overschee, 1999) and (Favoreel, De Moor, Van Overschee, & Gevers, 1999). Design of

robust controllers of type, that uses the subspace technique to compute the observer gain without the knowledge of system dynamic, only using input/output data, this methods are discussed in details in $\mathcal{H}_2$ and (Hinnen, Verhaegen, & Doelman, 2008) and $\mathcal{H}_\infty$ in (Woodley, How, & Kosut, 2001).

Independent of the model-free, data driven, heuristic, algorithm or methodology used, there are parameters and initial conditions to be set that influences the performance, the training, the convergence speed in order to solve a problem from a chosen method.

The study of methods for tuning controller parameters are of great importance. There are many applications in many fields such as, control, system identification and estimation problems. When one wants to perform algorithms for control, system identification or parameter estimation, it is necessary to "set up" the variables related to the algorithm, i.e. start up the initial setup, then apply the methodology and afterwards check the performance achieved for that particular set up. For each set up initialization, a different algorithm performance is achieved. That is the importance of tuning the parameters, because it impacts on the global methodology performance.

In this paper we propose a heuristic for tuning and convergence analysis of a model-free optimal control problem based on the discrete linear quadratic regulator with output feedback using reinforcement learning algorithm in the presence of noise.

This work is organized into sections containing in Section 2 the Preliminaries with the output feedback problem formulation and implementations aspects. In Section 3 the Tuning Problem Formulation is presented. In Section 4 the proposed methodology for the convergence analysis, tuning heuristics and metrics for data and surface generation. The computer Simulations and Analysis of the influence of the parameters in the data generator and algorithm performance is presented in Section 5. The Final Considerations are presented in Section 6, followed by Section 7 with the Acknowledges and Section 9 the References presented in the paper.

2. PRELIMINARIES

In this section we will show how to describe the states as function of the past input and output data of the dynamic system and the output feedback algorithm.

Consider the following discrete linear time invariant system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{1}$$

where $A \in R^{n\times n}$, $x_k \in R^n$ the state vector, $B \in R^{n\times m}$, $u_k \in R^m$ control action vector, $C \in R^{p\times n}$ and $y_k \in R^p$ the output vector. Assume (A,B) is controllable and (A,C) observable.

Given an instant k, the dynamics in a time horizon, described as $[k-N, k]$, the equation of state and system output, can be described by a simplified model and be expressed by defining some variables.

$$x_k = A^N x_{k-N} + U_N \overline{u}_{k-1,k-N} \tag{2}$$

$$\overline{y}_{[k-1,k-N]} = V^N x_{k-N} + T_N \overline{u}_{[k-1,k-N]} \tag{3}$$

where U_N is the controllability matrix and V_N the observability matrix, with $V_N \in R^{pN\times n}$

$$U_N = \begin{bmatrix} B & AB & \cdots & A^{N-1} \end{bmatrix} \tag{4}$$

$$V_N = \begin{bmatrix} CA^{N-1} \\ \vdots \\ CA \\ C \end{bmatrix} \tag{5}$$

and T_N , the Toeplitz matrix with the Markov parameters.

$$T_N = \begin{bmatrix} 0 & CB & CAB & \cdots & CA^{N-2}B \\ 0 & 0 & CB & \cdots & CA^{N-3}B \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & CB \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

The vectors $\overline{u}_{[k-1,k-N]}$ and $\overline{y}_{[k-1,k-N]}$, which are the pasts inputs and outputs on the time horizon $[k-1,k-N]$, represent the available measured data available .

$$\overline{u}_{[k-1,k-N]} = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \in R^{pN}; \quad \overline{y}_{[k-1,k-N]} = \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix} \in R^{mN} \tag{7}$$

Since (A,C) is observable there is an observability index K, such that $rank\ (V_N) < n$ if $N < K$ and $rank\ (V_N) = n$ if $N \geq K$. So V_N has full rank n, and there is a matrix $M \in R^{n\times pN}$ such that

$$A^N = MV_N \tag{8}$$

The observability property means that there are enough observations y_k within the time horizon so that one can completely rebuild the state x_k. This procedure was used for (Aangenent, Kostic, de Jager, van de Molengraft, & Steinbuch, 2005), to determine the control by identifying the Markov parameters. Since V_N has full rank, its left inverse is given by

$$V_N^+ = (V_N^T V_N)^{-1} V_N^T \tag{9}$$

such that

$$M = A^N V_N^+ + Z(I - V_N V_N^+) \equiv M_0 + M_1 \tag{10}$$

for any matrix Z, where M_0 is the minimum norm operator and $P(R^{\perp}(V_N)) = I - V_N V_N^+$, being the projection in the image, perpendicular to V_N (Lewis & Vamvoudakis, 2011).

Theorem 1: Let system 1 be observable. Then the system states are given solely in terms of the measured data

$$x_k = M_0 \overline{y}_{[k-1,k-N]} + (U_N - M_0 T_N)\overline{u}_{[k-1,k-N]} \tag{11}$$

$$\equiv M_y \overline{y}_{[k-1,k-N]} + M_u \overline{u}_{[k-1,k-N]} \tag{12}$$

or

$$x_k = [M_u \; M_y] \begin{bmatrix} \overline{u}_{[k-1,k-N]} \\ \overline{y}_{[k-1,k-N]} \end{bmatrix} \tag{13}$$

where $M_u = U_N - M_0 T_N$ and $M_y = M_0$, with , $M_0 = A^N V_N^+, V_N^+ = (V_N^T V_N)^{-1} V_N^T$, the left inverse of the observability matrix Eq.(5) and $N > K$, and K the observability index, (Lewis & Vamvoudakis, 2011).

Next will be presented the formulation of the output feedback problem for the *DLQR*.

A. Problem Formulation-Output Feedback for the DLQR

In this section, the theoretical basis of the approximate dynamic programming and optimal control theory are presented. The formulations of the control policy, Bellman equation and dynamic system states, in terms of only measured input/output data from the system, assemble a framework for the solution of the OPFB problem. The RL approach is presented in the context of the on-line DLQR design, considering policy iteration schemes to determine on-line the optimal control policies, (Lewis & Vamvoudakis, 2011).

The output feedback methodology can perform, through reinforcement learning (RL) methods, determine the new control policy that minimizes a quadratic performance index associated with the past weighted inputs and outputs with the addition of a white noise. When we refer to inputs, we are referring to control actions.

B. DLQR Control Policy and Bellman's Equation

In this section, the control policy and Bellman Equation will be formulated presenting the association of the value function of the states and the input/output values.

Once a stabilizing control policy $u_k = \mu(x_k)$ is chosen, the performance index or value function is defined as

$$V^{\mu}(x_k) = \sum_{i=k}^{\infty} (y_i^T Q y_i + u_i^T R u_i) \equiv \sum_{i=k}^{\infty} r_i \tag{14}$$

with $Q = Q^T \geq 0$ and $R = R^T > 0$, the weighting matrices in relation to the outputs and control actions, respectively and the pair $(A, C\sqrt{Q})$ being observable (Lewis & Syrmos, 1995).

It can be shown that for any policy μ (not necessarily optimal), the value function Eq.(14) is quadratic in state (Athans & Falb, 2013), such that

$$V^{\mu}(x_k) = x_k^T P x_k \tag{15}$$

for any matrix P, $(n \times n)$. This function is represented by the Bellman Equation

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \tag{16}$$

Up to now the issues discussed here were about the theoretical development related to the dynamic programming foundations with regards to system states.

Next, the change of the dependence of the states and the formulation in terms of measured data only and present the policy iteration algorithm (PI) used in this work will be presented.

C. Value Function Formulation in Terms of Measured Data

In order to write the state equation Eq.(15) in terms of past inputs and past outputs, consider $\bar{z}$ the vector containing the past inputs/outputs on the horizon [k − 1,k − N]:

$$\bar{z}_{[k-1,k-N]} = \begin{bmatrix} \bar{u}_{[k-1,k-N]} \\ \bar{y}_{[k-1,k-N]} \end{bmatrix} \tag{17}$$

Let $V^{\mu}(x_k)$ the value function associated to the states and using Eq.(13), then

$$V^{\mu}(x_k) = x_k^T P x_k = \bar{z}_{[k-1,k-N]}^T \begin{bmatrix} M_u^T \\ M_y^T \end{bmatrix} P \begin{bmatrix} M_u & M_y \end{bmatrix} \bar{z}_{[k-1,k-N]} \tag{18}$$

$$V^{\mu}(x_k) = \bar{z}_{[k-1,k-N]}^T \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix} \bar{z}_{[k-1,k-N]} \tag{19}$$

$$= \bar{z}_{[k-1,k-N]}^T \bar{P} \bar{z}_{[k-1,k-N]} \tag{20}$$

Note that , $\bar{u}_{[k-1,k-N]} \in R^{mN}$, $\bar{y}_{[k-1,k-N]} \in R^{pN}$, $\bar{z}_{[k-1,k-N]} \in R^{(m+p)N}$ and $\bar{P} \in R^{(m+p)N \times (m+p)N}$.

The value function at the k instant as a quadratic form of the past inputs and past outputs terms and the kernel of matrix $\bar{P}$ are expressed in Eq.(20). The Matrix $\bar{P}$ still depends on matrices A, B and C by M_0, M_u and M_y.

Next section presents the structure of the temporal difference error in terms of the measured past data.

D. Temporal Difference Error Based on Measured Data

This section presents the reinforcement learning method based on temporal differences to determine the value function on-line.

Let the Bellman Temporal Difference Error equation for the DLQR with respect to the states be

$$e_k = -x_k^T P x_k + y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \tag{21}$$

The structure of Eq.(21) can be written only as function of observed output sequences y_k and past control action sequences u_k (Lewis & Vamvoudakis, 2011).

To learn the internal structure of the matrix $\bar{P}$, without depending on the matrices A, B and C, we can write Bellman temporal difference error, Eq.(21) for the DLQR as function of the measured data as follows

$$e_k = -\bar{z}_{[k-1,k-N]}^T \bar{P} \bar{z}_{[k-1,k-N]} + y_k^T Q y_k + u_k^T R u + \bar{z}_{[k,k-N+1]}^T \bar{P} \bar{z}_{[k,k-N+1]} \tag{22}$$

Using the temporal difference error, the policy evaluation step, based on Bellman's equation, Eq.(22), may be performed using only the measured data, without the information of the states and, thus, learn without knowledge of the matrices A, B and C as follows in the next section.

E. Writing Policy Update in Terms of Measured Data

The Q-learning method is a model-free reinforcement learning technique that can be used to find an optimal action policy for any given (finite) Markov decision process (MDP) that can carry out the Value Iteration and Policy Iteration, without the knowledge of the system dynamics.

From the construction already presented, we can define an improvement policy in terms of measured data.

$$\mu(x_k) = \min_{u_k} (y_k^T Q y_k + u_k^T R u + x_{k+1}^T P x_{k+1}) \tag{23}$$

$$\mu(x_k) = \min_{u_k}(y_k^T Q y_k + u_k^T R u + \bar{z}_{[k,k-N+1]}^T \bar{P} \bar{z}_{[k,k-N+1]}) \tag{24}$$

Partitioning $\bar{z}_{[k,k-N+1]}^T \bar{P} \bar{z}_{[k,k-N+1]}$,

$$\bar{z}_{[k,k-N+1]}^T \bar{P} \bar{z}_{[k,k-N+1]} = \begin{bmatrix} u_k \\ \bar{u}_{[k-1,k-N+1]} \\ \bar{y}_{[k,k-N+1]} \end{bmatrix}^T \begin{bmatrix} p_0 & p_u & p_y \\ p_u & P_{22} & P_{23} \\ p_y & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} u_k \\ \bar{u}_{[k-1,k-N+1]} \\ \bar{y}_{[k,k-N+1]} \end{bmatrix} \tag{25}$$

where $p_o \in R^{m \times m}$, $p_u \in R^{m \times (m(N-1))}$ and $p_y \in R^{m \times pN}$.

Now the optimization problem in Eq.(24), can be determined by differentiating with respect to u_k and equaling to 0, so we have,

$$0 = R u_k + p_0 u_k + p_u \bar{u}_{[k-1,k-N+1]} + p_y \bar{y}_{[k,k-N+1]} \tag{26}$$

or

$$u_k = -(R + p_0)^{-1} + (p_u \bar{u}_{[k-1,k-N+1]} + p_y \bar{y}_{[k,k-N+1]}) \tag{27}$$

This controller is an autoregressive moving average dynamic model that generates the current control actions using only past data sequences. As in the *Q*-learning method, called by (Werbos, 1992) as "learning dependent action", the minimization problem may be developed with respect to the matrix kernel of $\bar{P}$, thus, dispensing the knowledge of the system dynamics. The algorithm of policy iteration using output feedback with policy evaluation and policy improvement is described next.

- Algorithm - Policy Iteration Using Output Feedback (OPFB)

Select a stabilizer control policy $u^0{}_k = \mu^0$ for $j = 0,1,...,$ play until convergence:

1.**Policy Evaluation:** Compute $\bar{P}^{j+1}$, such that

$$0 = -\bar{z}_{[k-1,k-N]}^T \bar{P}^{j+1} \bar{z}_{[k-1,k-N]} + y_k^T Q y_k + (u_k^j)^T R u_k^j + \bar{z}_{[k,k-N+1]}^T \bar{P}^{j+1} \bar{z}_{[k,k-N+1]} \tag{28}$$

2.**Policy Improvement** Partition $\bar{P}$ as in Eq.(25) and define updated policy by

$$u_k^{j+1} = \mu^{j+1}(x_k) = -(R + p_0^{j+1})^{-1} (p_u^{j+1} \bar{u}_{[k-1,k-N+1]}^{j+1} + p_y^{j+1} \bar{y}_{[k,k-N+1]}) \tag{29}$$

The implementation aspects of the policy iteration algorithm, noise and discount factor influence will be discussed in the next section.

F. Implementation, Noise and Discount Factor

This section will present the influence of the discount factor aspects with the purpose of mitigating the undesirable effects of noise, which can converge to biased solutions, (Al-Tamimi, Lewis, & Abu-Khalaf, 2008).

The presented Policy Iteration algorithm can be solved on-line by standard methods as Least Squares or Recursive Least Squares. In this work we used the RLS as in (Bradtke, Ydstie, & Barto, 1994), the RLS was used in the *Q*-learning algorithm to solve Eq.(28) using the following form:

$$vec(\overline{P}^{j+1})[\overline{z}_{[k-1,k-N} \otimes \overline{z}_{[k-1,k-N]} - \overline{z}_{[k,k-N+1]} \otimes \overline{z}_{[k,k-N+1]} = y_k^T Q y_k + (u_k^j)^T R u_k^T \qquad (30)$$

where $\otimes$ is the Kronecker product and *vec*, the column operator (Brewer, 1978). The terms with quadratic indexes Kronecker product are added.

To solve Eq.(30), it is necessary that the $[\overline{z}_{[k-1,k-N]} \otimes \overline{z}_{[k-1,k-N]} - \overline{z}_{[k,k-N+1]} \otimes \overline{z}_{[k,k-N+1]}]$ be linearly independent in the interval, which is a property known as persistent of excitation (PE).

By default, we inject a white noise to the control signal to generate persistent excitation, i.e., $\hat{u}_k = u_k + d_k$, with u_k a control action computed by the PI algorithm, and d_k being the white noise.

It is known that the inclusion of noise can lead to a result with bias in the output and, thus, an error in the identification of system dynamics. In (Lewis & Vamvoudakis, 2011), the effect of bias noise inclusion for Bellman equation was discussed, and so the effect of the discount factor to decrease the undesired effect of noise.

With the inclusion of the discount factor $\gamma < 1$, we have the modified version of the PI algorithm

$$vec(\overline{P}^{j+1})[\overline{z}_{[k-1,k-N]} \otimes \overline{z}_{[k-1,k-N]} - \gamma(\overline{z}_{[k,k-N+1]} \otimes \overline{z}_{[k,k-N+1]}) \qquad (31)$$

3. TUNING PROBLEM FORMULATION

This section presents the tuning problem formulation to set up the simulations in order to select the parameters that best solve the OPFB algorithm presented in previous section.

The problem formulation that minimizes the OPFB error can be an optimization problem. This problem will be presented. Suppose an error surface from the application of a random setup of the parameters, in this case, the states initial conditions, the random noise variance and the discount factor used by the PI algorithm. For different values of theses

parameters there will have different error associated with each setup simulation, affecting the training processes and the overall algorithm convergence performance.

Let e^* be the smallest error from an Error Surface generated by the states (x_1^*, x_2^*) and $V_\epsilon^{e_i}$, the neighborhood of error e_i generated by the states (x_i, x_j), with radius, such that $\left|(x_i, x_j) - (x_1^*, x_2^*)\right| < \epsilon$.

After the selection of states belonging to the neighborhood $V_\epsilon^{e_i}$ and performing simulations for each selected element, then we compute error and observe its behavior in order to select the states that generated the smallest absolute error of all, reducing the step size for simulations in order to minimize the total absolute error. Thus, we have the following optimization problem:

$$\min_{x_1, x_2, \gamma} E(x_1, x_2, \gamma) \tag{32}$$

After determining the state (x_1,x_2) that generate a smaller error, the discount factor adjustment is done, as was observed by means of simulations that for each set of states the discount factor undergoes some changes, with respect to the discount factor set in the initial simulations that generated the surfaces.

After selecting the new discount factor a check is performed around the border of the best solution, varying the states again and thus ending the adjustment process, both the states and the discount factor, since the step size decreases for each new adjustment.

4. PROPOSED METHODOLOGY

In this section, the strategies to perform the convergence analysis, through the formulation of the metrics used to evaluate the algorithms convergence behavior are presented. The heuristic for to the selection of the initial parameters used in the data generating model and for a more reliable data will be presented.

There are several difficulties in the preparation of data-based models. As data acquisition depends on actual experiments or simulations with the use of algorithms and independent of the choice, both approaches have their own characteristics and limitations. When a priori mathematical model of a system is available, the data can be generated using the model in simulations and thereby test the system response to parametric variations of the plant, and the influence of input or output noise signal and thus test the model for more realistic situations.

In subsection 4-A the metrics used to evaluate the performance of the algorithms is presented and in subsection 4-B, the heuristics for tuning the algorithm and data generator is presented.

A. Convergence Analysis

The methodology of the algorithm Output Feedback (OPFB) (Lewis & Vamvoudakis, 2011) is based on the construction of a matrix $\bar{P}$ Eq.(25), which will provide the parameter coefficients that will be used to define the next control action, such parameters are: p_0, p_u, and $p_y = \left[p_{y_1} \; p_{y_2} \right]$.

The estimation of the matrix coefficients $\bar{P}$, Eq.(25), can be made by some parameters estimation algorithm such as LMS, RLS, Batch LS, Kalman filter, Neural Networks, Fuzzy Logic.

In order to evaluate the quality of the solution, the following criteria were used:

$$E_n^i = \sum_{j=1}^{h} \left| a_j - x_j \right| \tag{33}$$

where, E_n^i, is the error of simulation *i*, after *n* iterations, $h = [(m + p)N][(m + p)(N + 1)]/2$, concerning the minimum number of independent terms referring to the matrix kernel $\bar{P}$, Eq.(25), where *m* is the number inputs, *p* the number of outputs and *N* is the time horizon, a_j is the estimation value of parameter *j* after *n* iterations, x_j, is the desired value of parameter *j* after *n* iterations.

Another metric used to observe the behavior of the algorithm is the estimation error variance for each of the estimated parameters during the iterative process, where the error variance for each simulation, is set as

$$S_i^2(E_n^i) = \sum_{j=1}^{h} \sum_{k=1}^{n} \frac{| e_{j,k} - \bar{e}_j |^2}{n}, k = 1, 2, ..., n. \tag{34}$$

where, *k* is the number of iterations, $e_{j,k}$, is the error of parameter *j* in iteration *k*, $\bar{e}_j$, is the mean estimation error of parameter *j* after *n* iterations, $S_i^2(e_n^i)$, is the sum of the error variances of estimated parameters in simulation *i* for *n* algorithm iterations.

With these two metrics to evaluate the algorithm convergence performance and the quality of the solution and some heuristics with the purpose of guiding the solution search process in the presence of noise, then it could be possible to decrease the algorithm complexity in the simulation process. Another aspect that will be available with this metrics

and heuristics is the possibility of testing different kinds of noise amplitudes, using other types of number generator by using other distribution density functions.

It is noticed in real plants the lack of information of the function associated with noise distribution, which implies difficulties to be overcome by the control methods, since the uncertainty is present in many situations.

Ideally, the simulation process should control the plant, independent of the disturbance characteristics associated with it. By this scope, it is quite usual to conduct the study of *Monte Carlo* simulations to evaluate the behavior of algorithms different sequences of random noise or small perturbations in the parameters of the system.

B. Heuristics for Surfaces Generation and Parameters Selection

In order to perform the algorithm tuning regarding the initial conditions of the parameters associated with a particular algorithm, surfaces are constructed with the initial conditions of the parameters aiming at generating the data related to past control actions and past outputs.

The following steps were used to observe the behavior of the algorithm related to the initial parameters variations, searching for better solutions.

First we generate an absolute error surface Eq.(33). Then identify the region where the best solution is found for that configuration and select the parameters values.

From the region which resulted the best solution, generate another surface near the best setting solution, and repeat this operation until the algorithm finds a configuration that generates the lowest total error, from Eq.(33) and possibly the lower variance, from Eq.(34).

If the average between the different configurations is very high, then we can use the coefficient of variance, which is the ratio between the mean and the standard deviation for each configuration.

The steps to be developed over the parameter selecting process are as follows:

- Set up the configuration of the initial parameters with varying limits with high amplitude, in accordance with the experience of the expert with respect to the parameters of the initial conditions;
- Perform various simulations and select the seed for the random number that generated the best solution, i.e., the lower error;
- Generate a figure with the mean square error to observe the influence of noise along the simulations;

- Build a surface for the absolute error for parameters variations to determine the region where the error is lower, and then select the parameters that came to that result;
- Once the initial parameters are selected, we construct another surface of the error, now around the best solution initially found, with range for the parameters variation;
- Finally, we evaluated whether the error is satisfactory or not, otherwise check if the number of iterations is enough for convergence of the algorithm, if not, increase the number of iterations.

After the analysis of the surfaces it will be possible to identify which parameters are best for starting conditions which brings the lower absolute error with the minimum variance.

5. SIMULATION AND CONVERGENCE ANALYSIS

In this section we discuss aspects related to the *OPFB* implementation algorithms, with Policy Iteration (PI).

The presented work we will cover only the Policy Iteration algorithm (PI). For the PI algorithm, a stabilizing initial policy for the controller is necessary. For this we developed a simulator that could generate randomly a symmetric and positive definite matrix P, and then calculate the gain of the closed loop system, and thus evaluate whether the eigenvalues for the closed loop system would be inside the unit circle.

A. Initial Setup and Computational Simulations

The system used in the simulations for input and output data is described by the following dynamical system,

$$x_{k+1} = \begin{bmatrix} 1.1 & -0.3 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k \tag{35}$$

$$y_k = \begin{bmatrix} 1 & -0.8 \end{bmatrix} x_k \tag{36}$$

The number of parameters $h = [(m + p)N][(m + p)(N + 1)]/2$ to be estimated for this system were 10.

Initially 1000 simulations were performed, using the PI algorithm for the OPFB and evaluated the lowest total absolute error Eq.(33) for the 10 coefficients estimated by the *RLS*, with forgetting factor $\lambda = 1$, white noise with variance of 1.2 and mean 0 and initial states $x_1 = x_2 = 200$.

The number of iterations used for control action estimations was 1,000, using temporal difference error algorithm and one iteration for the *RLS*. As the application was developed to

be on-line, we needed to store data from the first three iteration, then we perform the algorithm after the fourth iteration.

The mean square error for 1000 simulations is presented in Figure 1. Most of the results except for one simulation, near iteration 800, the *MSE* was below 20. After 1000 simulations the lowest total absolute error Eq.(33) reached was 0.4963.

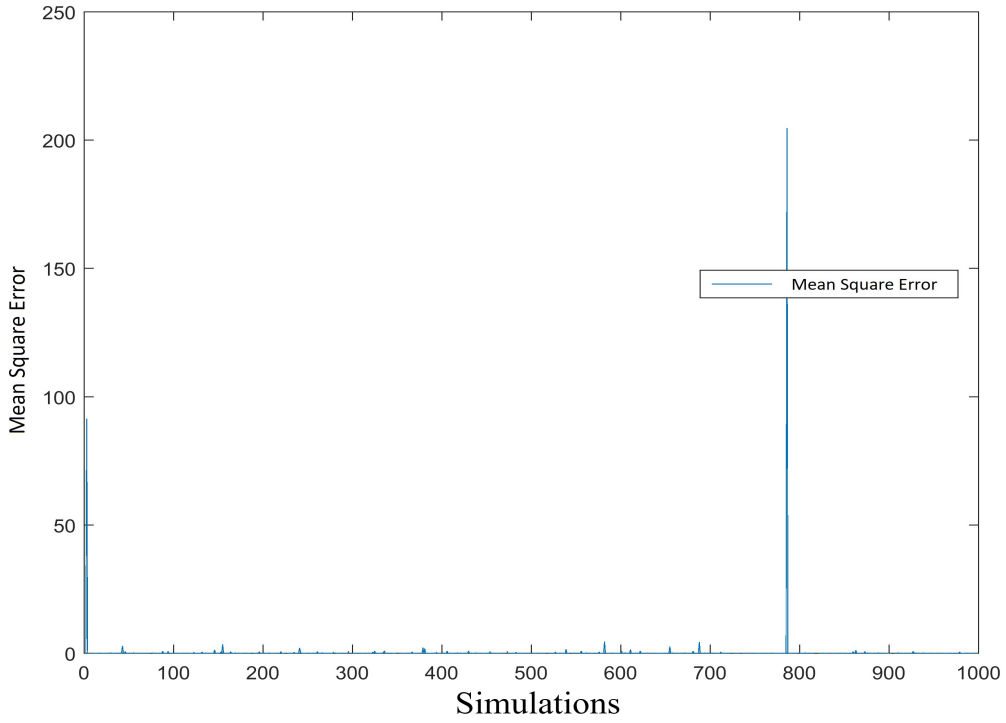

Figure. 1: Mean Squared Error for 1000 Simulations.

Source: MATLAB® Simulation by Authors

From Figure 1 can be view that near iteration 800 a peak could be done by a particular random noise and also by some roundoff through the iteration process. The objective here is to select an initial noise signal to be fixed for further use to investigate the parameters sensibility in the algorithm. So, the noise signal which produced the lowest total absolute error will the be the chosen one.

The true (P_T) and estimated (P_E) matrices solutions from the best result from the 1000 simulations presented in Figure 1 which produced the smallest total absolute error Eq.(33) are presented below.

$$\bar{P}_T = \begin{bmatrix} 1.0150 & -0.8440 & 1.1455 & -0.3165 \\ -0.8440 & 0.7918 & -1.0341 & 0.2969 \\ 1.1455 & -1.0341 & 1.3667 & -0.3878 \\ -0.3165 & 0.2969 & -0.3878 & 0.1113 \end{bmatrix} \quad (37)$$

$$\bar{P}_E = \begin{bmatrix} 1.0836 & -0.8324 & 1.1263 & -0.3067 \\ -0.8324 & 0.8195 & -0.9302 & 0.2391 \\ 1.1263 & -0.9302 & 1.4073 & -0.3764 \\ -0.3067 & 0.2391 & -0.3764 & 0.1450 \end{bmatrix} \quad (38)$$

The total absolute error was used and presented in Figure 2, because the MSE could not show some information about variability.

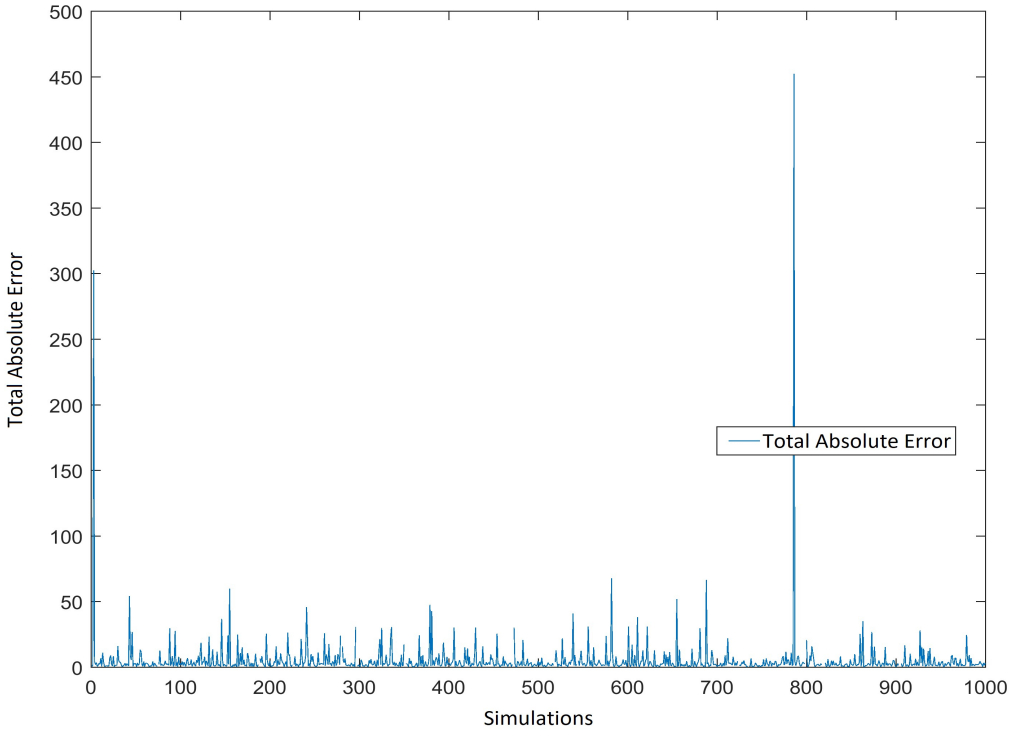

Figure 2: Total Absolute Error for 1000 Simulations.

Source: MATLAB® Simulation by Authors

From Figure 2 can see that there was a very high value near simulation 800, but most of the simulations returned error less than 50. The total absolute error Eq.(33) is the sum of the absolute values of the errors of the estimated coefficients of the matrix $\bar{P}$.

The true values of matrices presented by Eq.(37) and estimated parameters using the RLS in Eq.(38), shows some errors, but this is the first step through the process of tuning.

The coefficients evolution through the reinforcement learning and parameter estimation of the estimated matrix is presented in Figure 3.

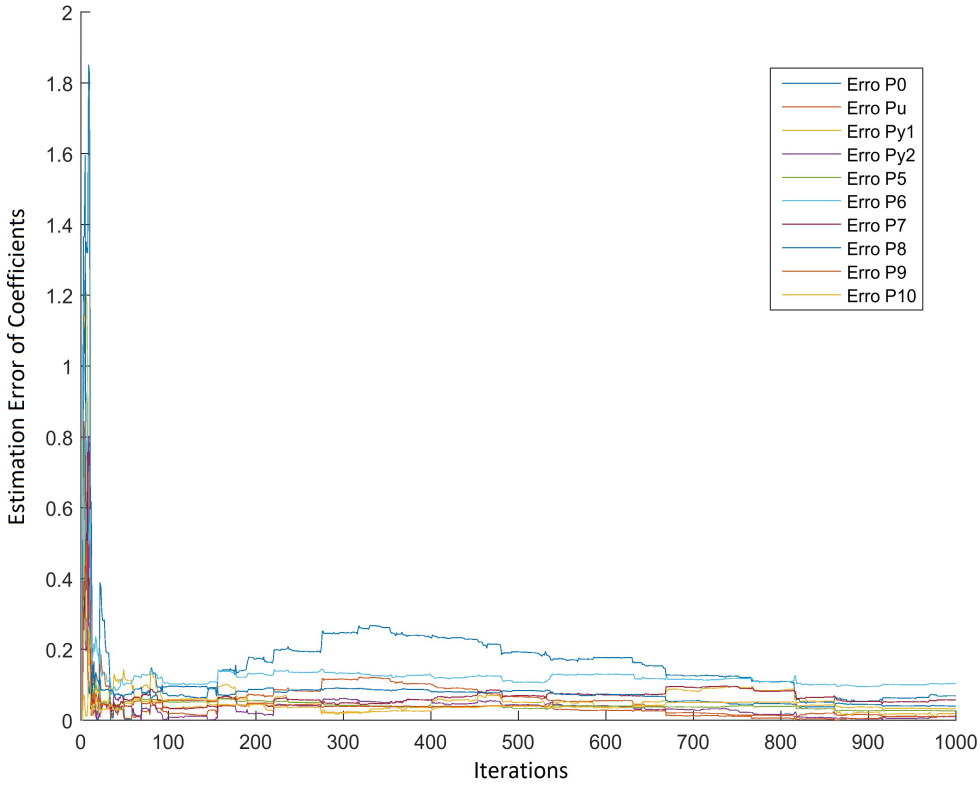

Figure 3: Estimation Error of Coefficients by RLS.

Source: MATLAB® Simulation by Authors

The convergence of the estimated coefficients was reached by iteration number 900. Simulations using more iterations were performed, but as no further improvement was achieved, 1000 iterations were chosen in order to decrease the computational cost.

From Figure 3 can be seen that the coefficient p_8, with blue color is the coefficient with high vales even after 800 iterations. It will be that these coefficients will get better as we apply the proposed methodology for tuning the algorithm.

B. Discount Factor γ and x_k Parameter Variations

The performance of the OPFB algorithm is related with to discount factor and the data model used to generate the output data. In this section we will discuss the discount factor and parameter x_k variations which are used in the data model to generate the output data. Surfaces

are built to show the variations of these two parameters to search for the best solution and to better understand their influence toward the total absolute error.

Surface of the total absolute error with respect to variations of the discount factor is presented in Figure 4. The discount factor is used to decrease the noise, and the multiplicative factor is used to weight the initial condition of the states for generating the output data.

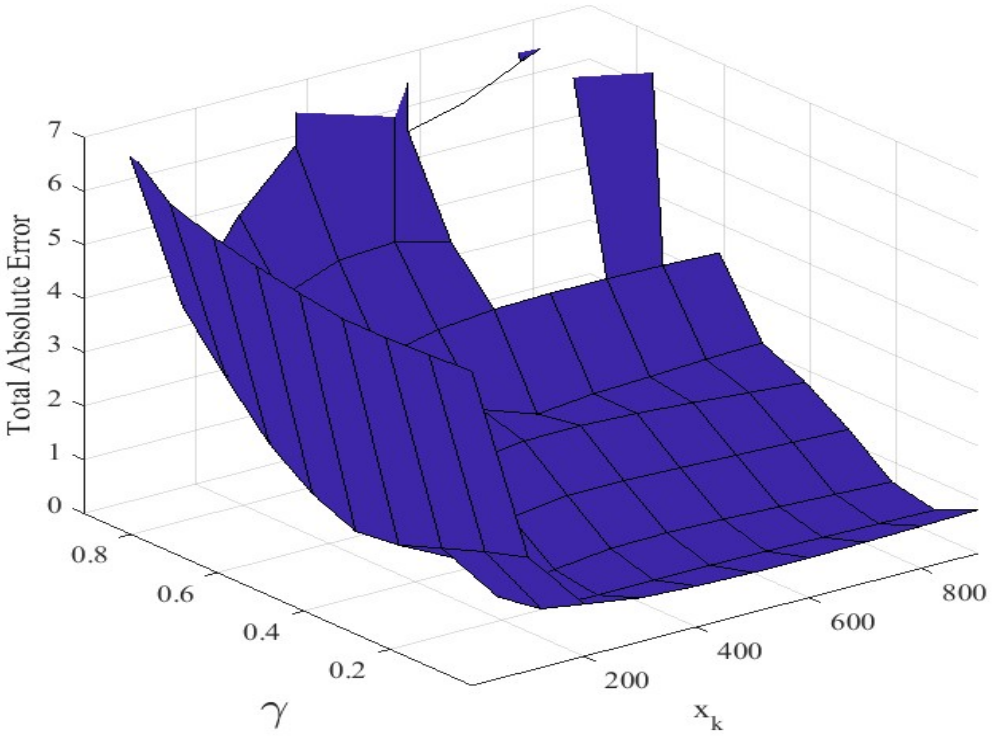

Figure 4: Error Surface for Discount Factor and x_k Parameter Variations.

Source: MATLAB® Simulation by Authors

In Figure 4 the surface represents the influence of the discount factor and parameter x_k on the total absolute error.

The best solution with the smallest absolute error in the surface was 0.3889, for the discount factor 0.21 and $x_k = 201$. The discount factor (γ) variation was established in the range of 0.01 to 1 with intervals of 0.1, and the x_k factor variation was set from 1 to 900 with intervals of 100, and a white noise of 0 mean and variance 0.4.

Now, considering a range around the best solution found on the surface of Figure 4 and varying the discount factor from 0.1 to 1 with intervals of size 0.5 and the range of parameter x_k from 1 to 300 with intervals of size 10, results in the surface shown in Figure 5.

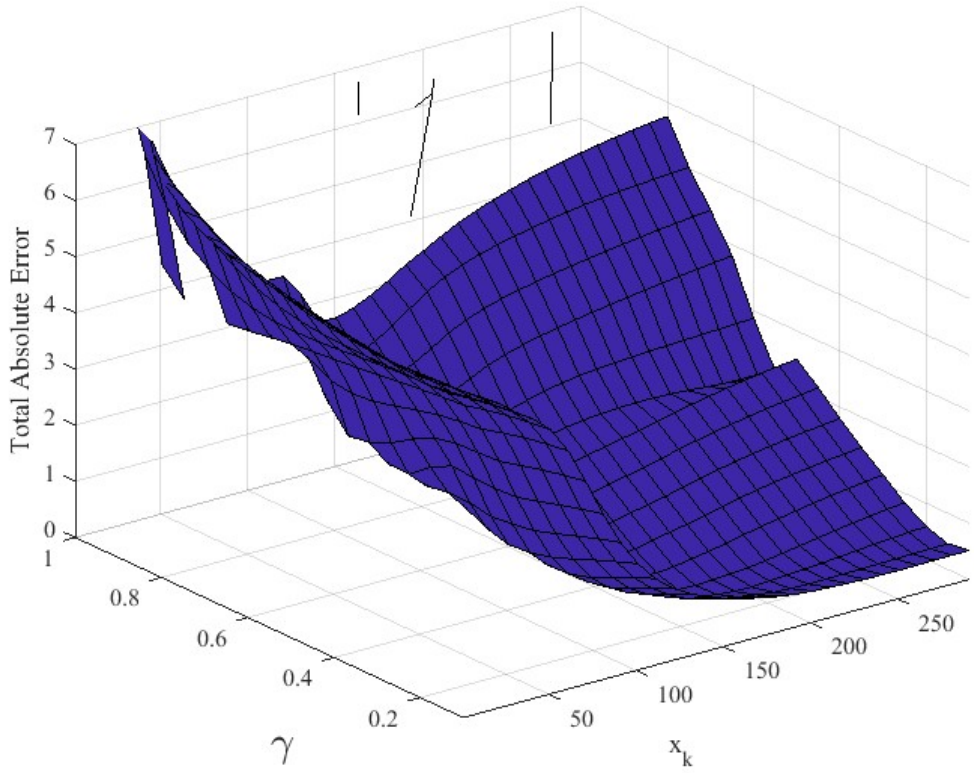

Figure 5: Error Surface for Discount Factor and x_k Variations.

Source: MATLAB® Simulation by Authors

The surface generated in Figure 5 shows the influence that the intervals size related to the shape of the error surface have. Notice that the surface of Figure 5 is smoother than that in Figure 4. The best solution found in the surface of Figure 5 resulted in error of 0.3568 with discount factor of 0.20 and factor $x_k = 191$.

C. Discount Factor γ and $Prls_k$ Variations

In this section, we will discuss the influence of the discount factor and a parameter $Prls_k$. The $Prls_k$ is a multiplicative factor of the initial covariance matrix of the RLS estimator. The influence of the discount factor and $Prls_k$ parameter variations were done in the starting set up of the *RLS* algorithm and can be seen in Figure 6.

From Figure 6, the total absolute error behavior increases as the discount factor increases too. The result which brought the lower total absolute error occur when the discount factor is lower, precisely the best result is achieved when the discount factor is 0.21. No clear pattern could be seen from $Prls_k$ increase in the chosen range.

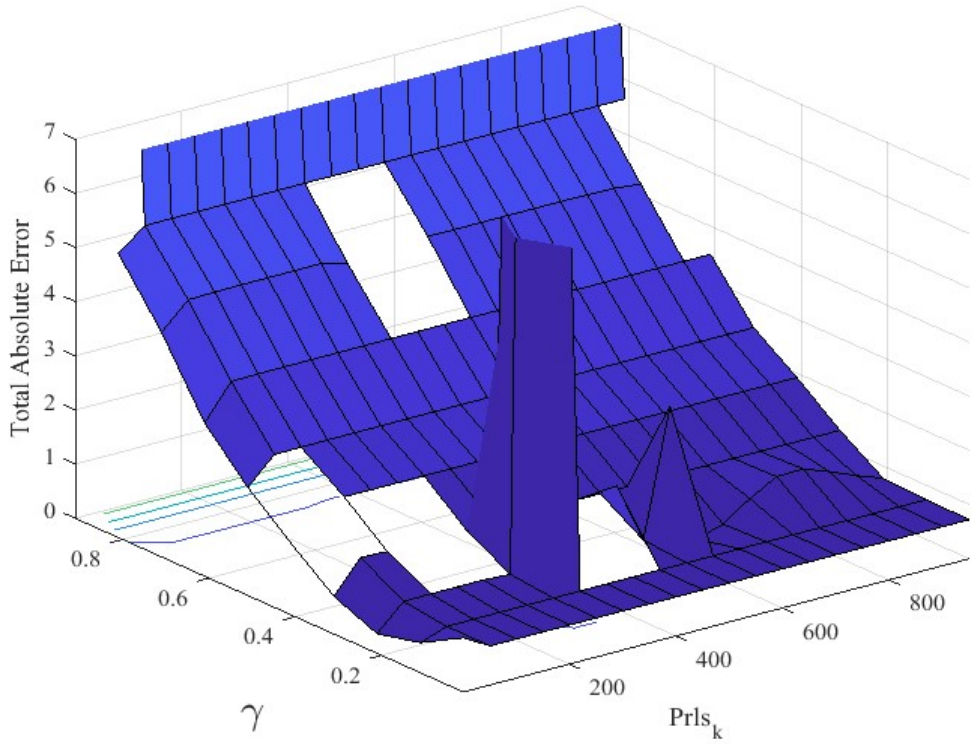

Figure 6: Error Surface for Discount Factor and $Prls_k$ Variations.

Source: MATLAB® Simulation by Authors

It can be seen from Figure 6 some discontinuity on error surface indicating that the algorithm diverges, with no solution. This could indicate that the *OPFB* algorithm has a dependence of the estimator.

The smallest error found on the surface of Figure 6 was 0.3637, with discount factor equal to 0.21 and factor $Prls_k = 1$.

Another analysis was done adding white noise with mean 0 and variance between 0.01 to 10 with increment intervals of 0.5 and discount factor with range of 0.01 to 1, with intervals of 0.01. The purpose was to analyze the algorithm performance and convergence in the presence of noise.

The surface with the results of the added noise can be seen in Figure 7. It is observed from Figure 7 that the discount factor works to reduce the unwanted effect of noise and, in this case, the single best result was achieved for the discount factor equal to 0.09. The algorithm worked well even with the increase of the noise variance, as can be seen in Figure 7.

For each point on the surface, only 5 simulations were performed, where the total absolute error of 1.1909 was computed. This error was achieved with noise variance of 3.51.

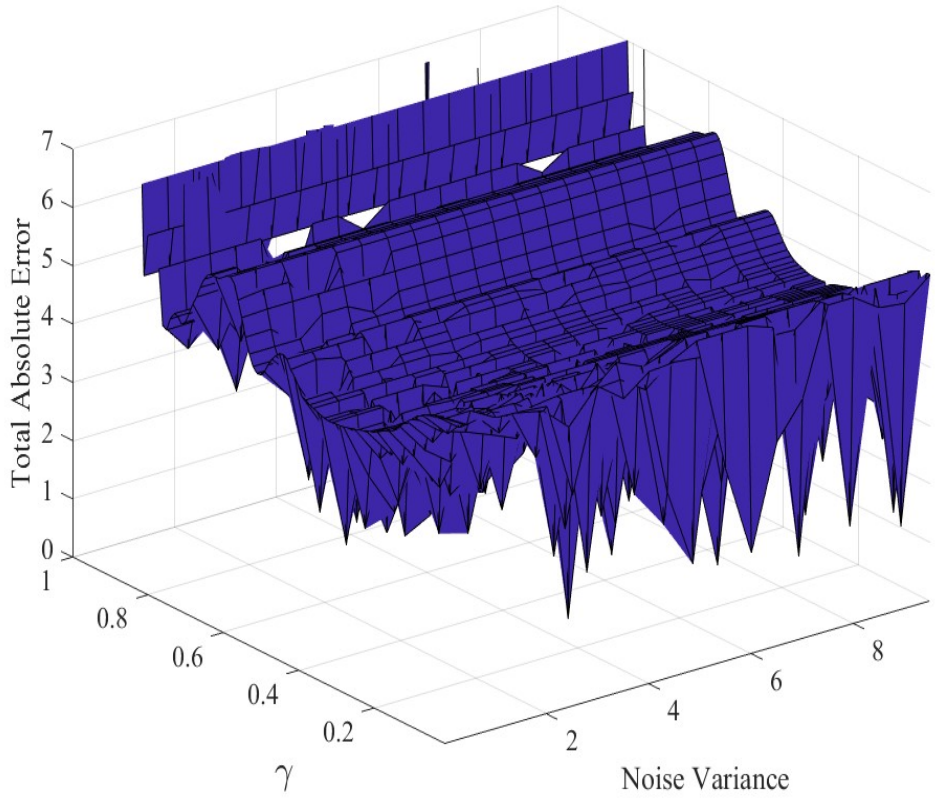

Figure 7: Error Surface for Variation in Discount Factor and Noise.

Source: MATLAB® Simulation by Authors

Now, narrow intervals were chosen for the discount factor and noise variance around the best solution that was achieved in the Figure 7. The noise variance range was between 0.1 to 4, with increment of 0.1. The number of simulations performed was 25 for each surface point, choosing the best solution among all the simulations. The discount factor range was from 0.01 to 0.3 with intervals of 0.01. The total absolute error surface is shown in Figure 8.

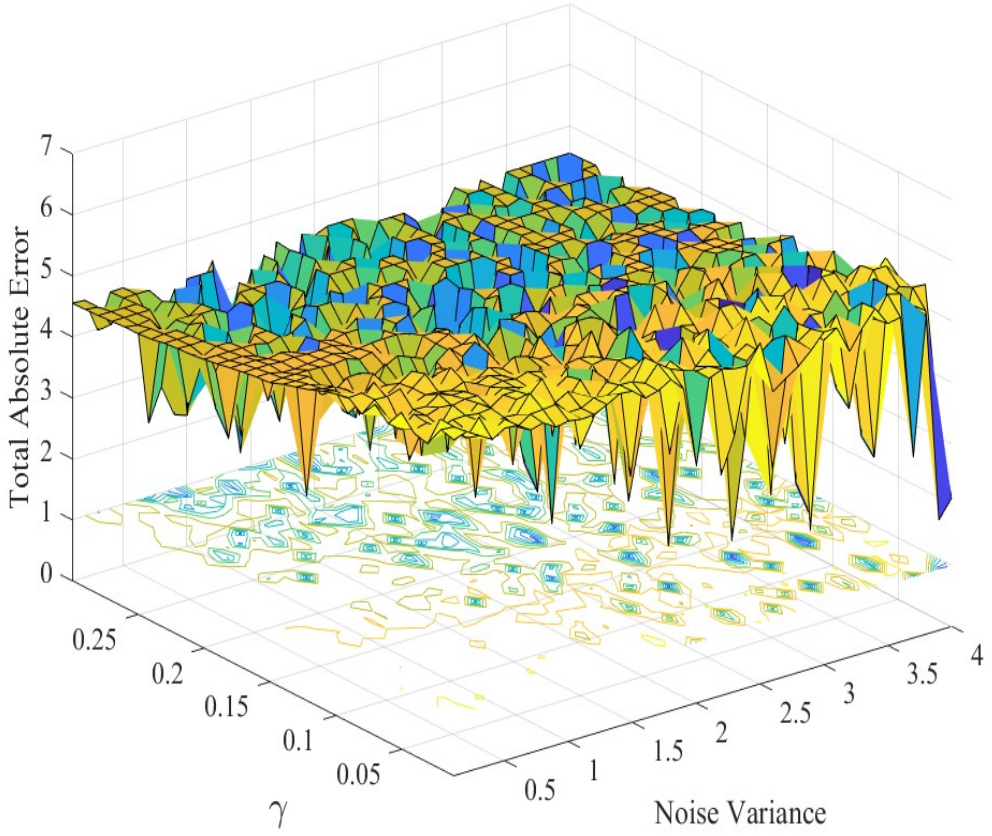

Figure 8: Error Surface for Discount Factor Range from 0.01 to 0.3 and Noise Range from 0.1 to 4.

Source: MATLAB® Simulation by Authors

As for each simulation we are varying the noise variance, we have that the total error will not follow a smooth pattern, generating several different total errors.

The purpose of the error surface generated in Figure 8 was to show the direct influence of increased noise variance related to the total absolute error. We could also calculate the *Pearson* correlation coefficient to identify whether there is a linear correlation between the noise and the total absolute error. Independently, one can see that this phenomenon does not occur. We could even compute other kind of associative measure like *Spearman* correlation coefficient, which is not necessarily linear.

Restricting the discount factor for a range between 0.01 and 0.35 and noise with variances between 0.1 and 3.5, with intervals of 0.05 it can be observed in Figure 9; just as in Figure 8, where minor errors are concentrated in the range up to 0.1 for the discount factor

and the variance of the noise around 3. The best results were obtained when the noise variance was 2.95 on this surface the smallest error achieved was 1.1019.

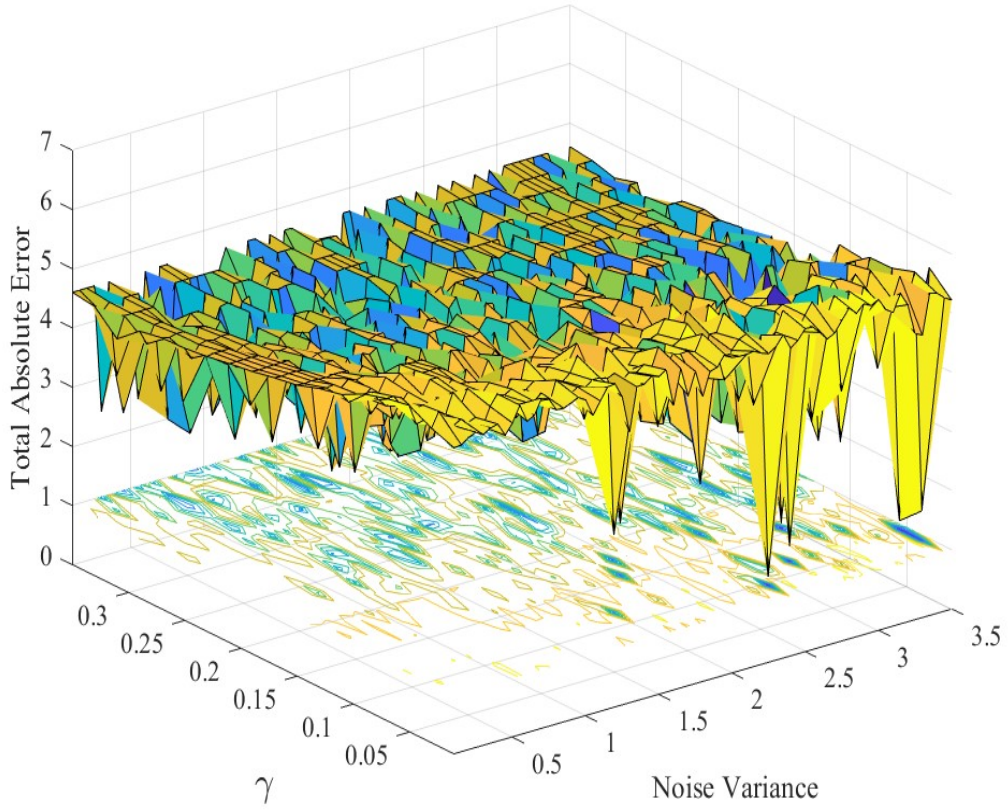

Figure 9: Error Surface for the Discount Factor Ranges from 0.01 to 0.35 and Noise Range from 0.1 to 3.5.

Source: MATLAB® Simulation by Authors

This smallest error achieved can be explain since the number of simulations was small, because the purpose was to investigate the influence of the discount factor and the noise variance in the total error and have a visual tool for this evaluation through the surface.

Once identified the influence of noise and discount factor with respect to the total absolute error, now we can also observe the behavior of initial states that will be used to generate the data and thus check if there is any significant influence from them.

For the process in question, we have two states which we will call x_1 and x_2. Initially, we will observe the total absolute error with respect to the initial state x_1 range from 0 to 5.000 with intervals of 200 and x_2 ranging from −2000 to 2000, with increment of 500.

The smallest error occurred for x_1 = 1600 and x_2 = 1000, with total absolute error of 0.3205 in Figure 10. To calculate the total absolute error for each pair of states (x_1,x_2), 25 noise free simulations, so we could identify what initial states produce the smallest error.

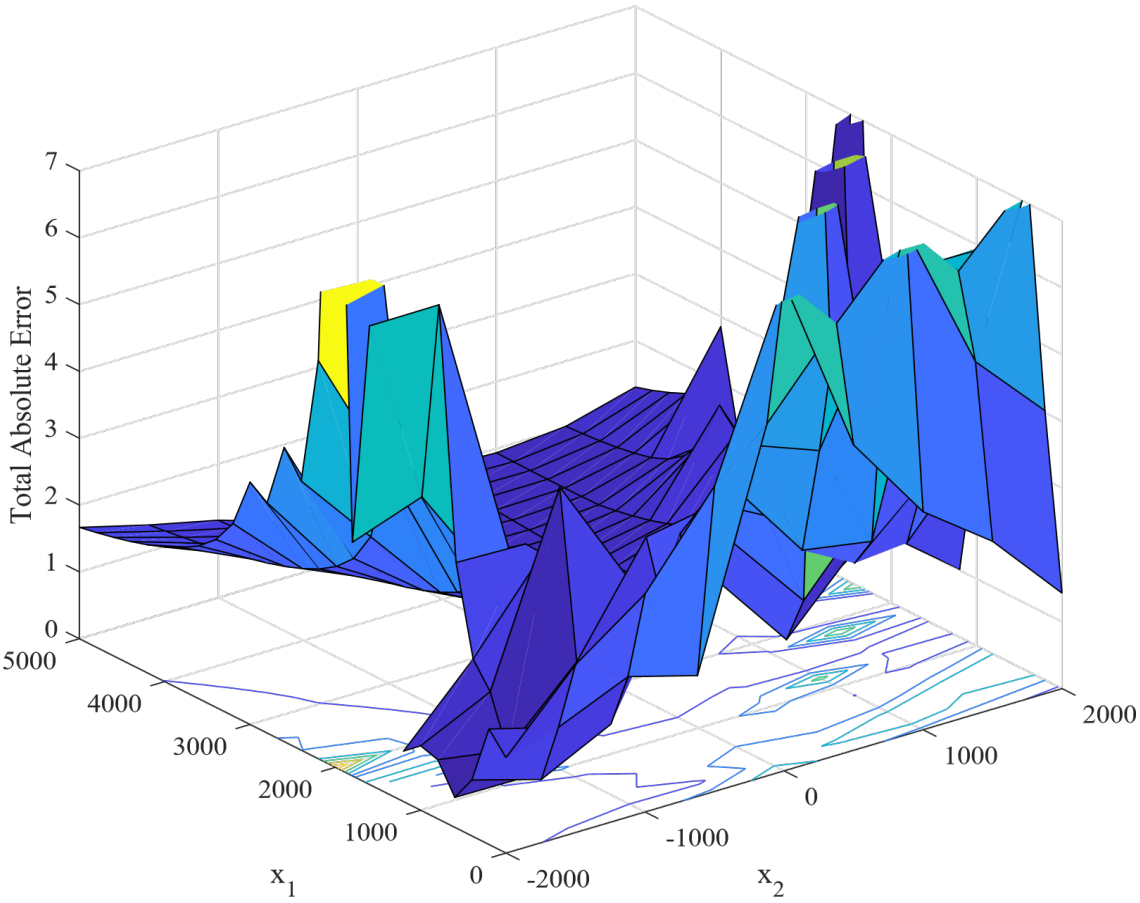

Figure 10: Initial States x_1 and x_2 Model Generator Variation.

Source: MATLAB® Simulation by Authors

Looking to the Figure 10 can be seen many edges on the surface, because the intervals between the variations were 200 for x_1 and 500 for x_2. Selecting now a smaller increment for each interval of x_1 with size 100, we can see with more details the variation of behavior and a smoother error surface, because we computed more points using smaller increments in the range of the intervals.

It can be seen in Figure 11 it was possible to achieve an error of 0.2644, for x_1 = 2700 and x_2 = 1500. In that figure we can see more clearly the behavior of the parameters x_1 and x_2. The parameter x_1 seems to produce more variable errors for smaller values, between 0-3000, and for higher values of x_2, between 02000. Now we choose to construct a surface around the smallest absolute error of Figure 11, in order to define x_1 and x_2 values, which bring the smallest absolute error.

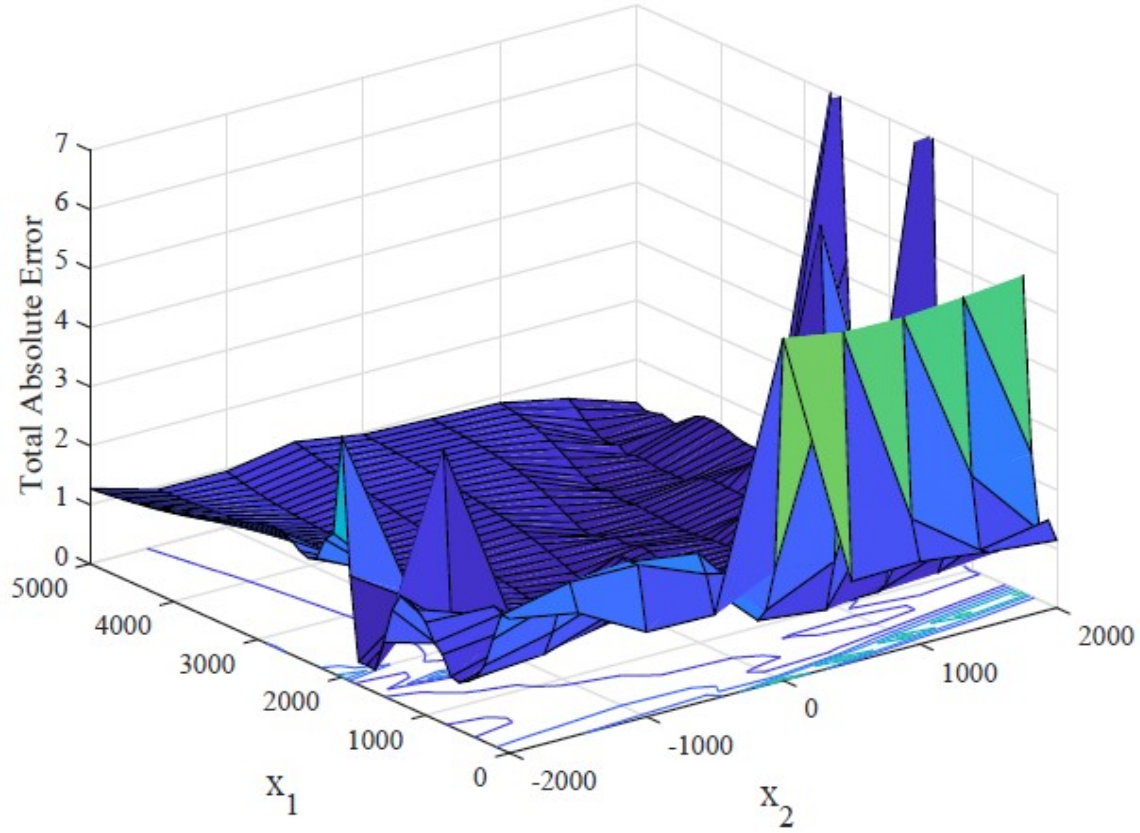

Figure 11: Initial States Set up for Variations of x_1 and x_2, with size 100 to x_1 and 500 to x_2.

Source: MATLAB® Simulation by Authors

After some other simulations with variations around the smallest error from Figure11, we can get the best value of x_1 that should be between 2750-2800 and x_2 between 1000-1020. If we restrict the range of variation of x_1 between 2750 to 2800 and x_2 between 1000 and 1020, we get the result of the absolute total error in Figure 12, with total absolute error of 0.1019 for a discount factor equals to 0.20.

Several simulations were performed to compute in the mentioned ranges, varying with intervals of 1 unit only and then compute the total absolute error in order com build the surface and then have a visual behavior of the parameters influence to the total absolute error.

From Figure 11 is clear the minimum by looking up the shape of the total absolute error behavior.

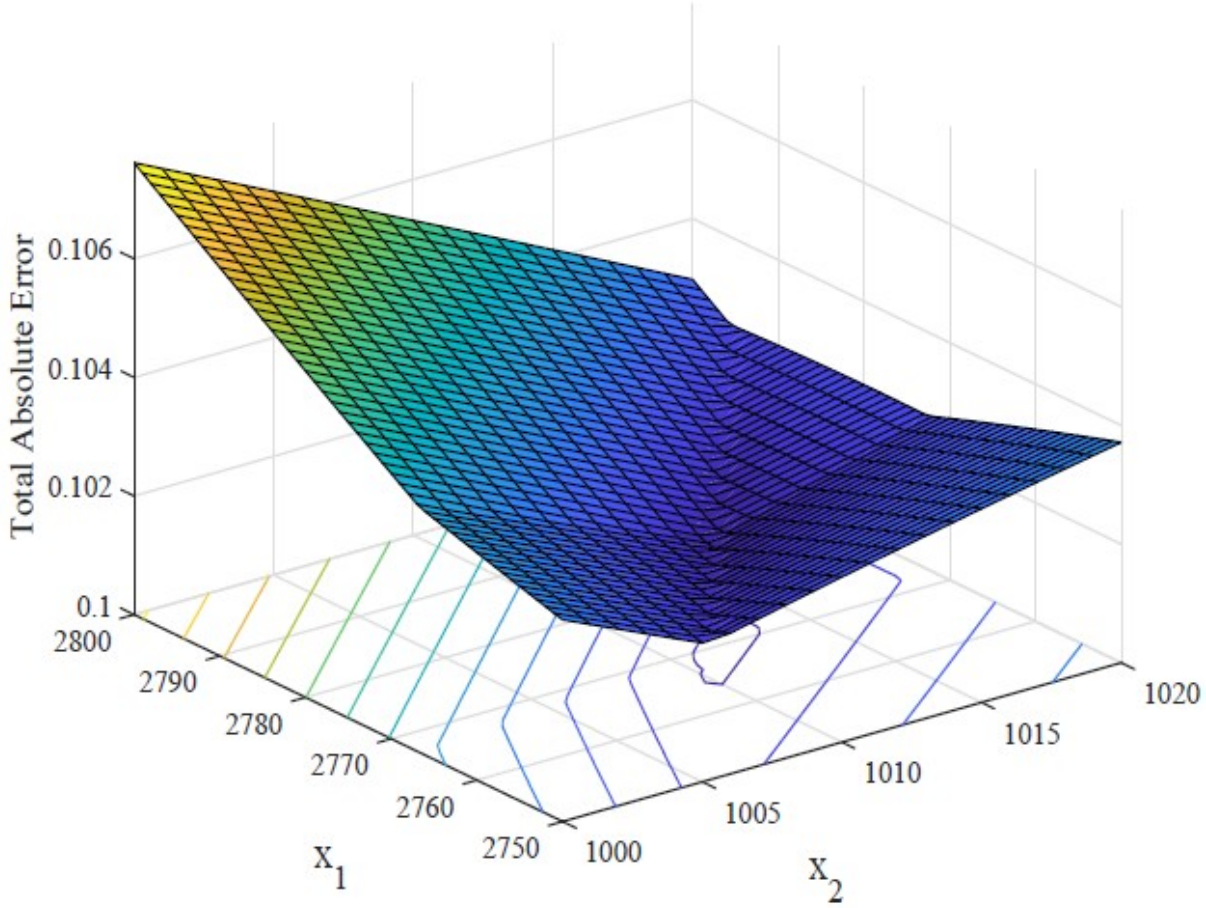

Figure 12: Initial States Set up Variations for x_1 and x_2, between 2750−2800 for x_1 and 1000−1020 for x_2.

Source: MATLAB® Simulation by Authors

From Figure 12 we can observe in the surface the minimum value of the total absolute error clearly. The values for x_1 and x_2 were $x_1 = 2771$ and $x_2 = 1012$.

We can see from the above results that the algorithm is sensitive to initial data of the states being of great importance the proper determination of them in the simulation process, in order to test and validate a methodology or algorithm. The construction of a reliable data generator is of great importance.

Finally, we decided to simulate again small variations on the discount factor in order to see if it was possible to get a smaller error, with a fine tuning of the discount factor parameter. After the changes in the discount factor and we got better results, but when we carried out these tests. We also noticed that the x_1 and x_2 changed from their values.

Performing a refinement near the best solution for x_1 and x_2 and varying the discount factor between 0.19 and 0.21 at intervals of 0.0001, we obtained an error of 0.0740 for $x_1 = 2902$, $x_2 = 977.8$ and discount factor 0.1916.

The estimated and desired matrices results achieved were:

$$\bar{P}_T = \begin{bmatrix} 1.0150 & -0.8440 & 1.1455 & -0.3165 \\ -0.8440 & 0.7918 & -1.0341 & 0.2969 \\ 1.1455 & -1.0341 & 1.3667 & -0.3878 \\ -0.3165 & 0.2969 & -0.3878 & 0.1113 \end{bmatrix} \quad (39)$$

$$\bar{P}_E = \begin{bmatrix} 1.0155 & -0.8440 & 1.1355 & -0.3200 \\ -0.8440 & 0.7883 & -1.0608 & 0.2969 \\ 1.1355 & -1.0608 & 1.3845 & -0.3973 \\ -0.3200 & 0.2969 & -0.3973 & 0.1138 \end{bmatrix} \quad (40)$$

Most of the coefficients converge before 400 iterations, only 2 coefficients, P_8 and Py_1 converged before 600 iterations. The Coefficients P_8 and Py_1 had the slowest convergence rate as can be seen from Figure 13.

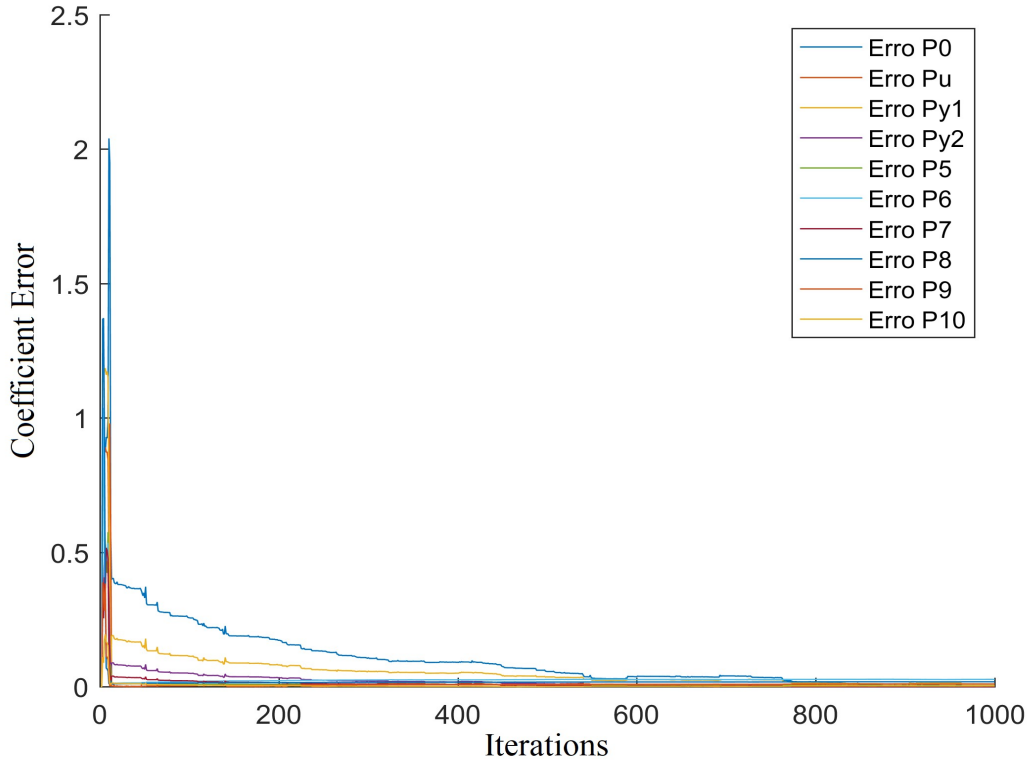

Figure 13: Error of the Estimated Coefficients Total *Error* 0.0740.

Source: MATLAB® Simulation by Authors

The error for each coefficient estimated along the iteration process is presented in Figure 13. After this last refinement on the parameters, no further improvement could be achieved, and the convergence of the parameter occurred by 800 iterations.

6. FINAL CONSIDERATIONS

The presented work showed a general methodology for the selection of the initial parameters, and a heuristic for preparing a data generator based on models for data-based methods, in order to produce more reliable data.

A data generator, statistical metrics and heuristics for selecting parameters with assistance of surfaces were proposed to support in the algorithm convergence and robustness analysis of the DLQR with output feedback.

The methodology was evaluated with respect to the influence of noise and the response of the tested algorithm which best produces a good quality data generator for data-based methodologies.

Finally, we concluded that the application of heuristics associated with the data generator and surface buildings for the selection of parameters is promising with respect to data generation in a more reliable manner, as well as the use of statistical metrics proposed to evaluate the performance, convergence and robustness for on-line DLQR based on reinforcement learning with output feedback.

7. ACKNOWLEDGENT

The authors would like to thank DEE/UFMA and CPGEE/UFMA (Department of Electrical Engineering of Federal University of Maranhão), and the Coordination for the Improvement of Higher Level Personnel (CAPES).

References

Aangenent, W., Kostic, D., de Jager, B., van de Molengraft, R., & Steinbuch, M. (2005, June). Data-based optimal control. In *Proceedings of the 2005, american control conference, 2005.* (p. 1460-1465 vol. 2). doi: 10.1109/ACC.2005.1470171

Alexander S. Poznyak, W. Y., Edgar N. Sanchez. (2001). *Differential neural networks for robust nonlinear control: Identification, state estimation and trajectory tracking* (1st ed.). World Scientific Publishing Company. Retrieved from http://gen.lib.rus.ec/book/index.php?md5=029EFF9BEA958638157E5C63C73986DA

Alonso, H., Mendonça, T., & Rocha, P. (2009). Hopfield neural networks for on-line parameter estimation. *Neural Networks*, *22*(4), 450–462.

Al-Tamimi, A., Lewis, F. L., & Abu-Khalaf, M. (2008, Aug). Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(4), 943949. doi: 10.1109/TSMCB.2008.926614

Anguelova, M. (2004). *Nonlinear observability and identifiability: 484 general theory and a case study of a kinetic model for s. cerevisiae*. Chalmers tekniska högsk. Retrieved from https://books.google.com.br/books?id=wIYLYAAACAAJ

Athans, M., & Falb, P. L. (2013). *Optimal control: an introduction to the theory and its applications*. Courier Corporation.

Battistelli, G., Mari, D., Selvi, D., & Tesi, P. (2014, Dec). Unfalsified approach to datadriven control design. In *53rd ieee conference on decision and control* (p. 6003-6008). doi: 10.1109/CDC.2014.7040329

Bradtke, S. J., Ydstie, B. E., & Barto, A. G. (1994, June). Adaptive linear quadratic control using policy iteration. In *American control conference, 1994* (Vol. 3, p. 3475-3479 vol.3). doi: 10.1109/ACC.1994.735224

Brewer, J. (1978, Sep). Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, *25*(9), 772-781. doi: 10.1109/TCS.1978.1084534

Chen, S., Billings, S., & Grant, P. (1990). Non-linear system identification using neural networks. *International journal of control*, *51*(6), 1191–1214.

Chiera, B. A., & White, L. B. (2008). Application of model-free lqg subspace predictive control to tcp congestion control. *International Journal of Adaptive Control and Signal Processing*, *22*(6), 551–568.

Chu, S. R., Shoureshi, R., & Tenorio, M. (1990). Neural networks for system identification. *Control Systems Magazine, IEEE*, *10*(3), 31–35.

Favoreel, W., De Moor, B., Gevers, M., & Van Overschee, P. (1999). Closed loop modelfree subspace-based lqg-design. In *Proc. of the 7th ieee mediterranean conference on control and automation, june* (pp. 28–30).

Favoreel, W., De Moor, B., Van Overschee, P., & Gevers, M. (1999). Model-free subspacebased lqg-design. In *American control conference, 1999. proceedings of the 1999* (Vol. 5, pp. 3372–3376).

Fleming, W. H. (1968). Optimal control of partially observable diffusions. *SIAM Journal on Control*, *6*(2), 194–214.

Guildas, B. (2007). *Nonlinear observers and applications*. Springer-Verlag Berlin Heidelberg.

Hinnen, K., Verhaegen, M., & Doelman, N. (2008, May). A data-driven $\mathcal{H}_2$-optimal control approach for adaptive optics. *IEEE Transactions on Control Systems Technology*, *16*(3), 381-395. doi: 10.1109/TCST.2007.903374.

Hou, S., Zhongsheng; Jin. (2013). *Model free adaptive control : Theory and applications*. CRC Press.

Lewis, F. L., & Syrmos, V. L. (1995). *Optimal control*. John Wiley & Sons.

Lewis, F. L., & Vamvoudakis, K. G. (2011, Feb). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *41*(1), 14-25. doi: 10.1109/TSMCB.2010.2043839.

Liming, W., Shan, F., & Dan, H. (2015, Oct). Unfalsified adaptive controller design for pilot-aircraft system(iccas 2015). In *Control, automation and systems (iccas), 2015 15th international conference on* (p. 1494-1499). doi: 10.1109/ICCAS.2015.7364589.

Littman, M. L. (2009). A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, *53*(3), 119–125.

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, *28*(1), 47–65.

Modares, H., Peen, G. O., Zhu, L., Lewis, F. L., & Yue, B. (2014, June). Datadriven optimal control with reduced output measurements. In *Intelligent control and automation (wcica), 2014 11th world congress on* (p. 1775-1780). doi: 10.1109/WCICA.2014.7052989.

Monahan, G. E. (1982). State of the art a survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, *28*(1), 1–16.

Nechyba, M. C., & Xu, Y. (1994). Neural network approach to control system identification with variable activation functions. In *Intelligent control, 1994., proceedings of the 1994 ieee international symposium on* (pp. 358–363).

Saeki, M., Kondo, K., Wada, N., & Satoh, S. (2014, Dec). Data-driven online unfalsified control by using analytic center. In *53rd ieee conference on decision and control* (p. 2026-2031). doi: 10.1109/CDC.2014.7039696.

Safonov, M. G., & Tsao, T.-C. (1997, Jun). The unfalsified control concept and learning. *IEEE Transactions on Automatic Control*, *42*(6), 843-847. doi: 10.1109/9.587340.

Sajjanshetty, K. S., & Safonov, M. G. (2015, Sept). Adaptive dwell-time switching in unfalsified control. In *2015 ieee conference on control applications (cca)* (p. 1773-1778). doi: 10.1109/CCA.2015.7320866.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, *21*(5), 1071–1088.

Sondik, E. J. (1971). *The optimal control of partially observable markov processes.* (Tech. Rep.). DTIC Document.

Sondik, E. J. (1978). The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations research*, *26*(2), 282–304.

Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches*, *15*, 493–525.

White III, C. C. (1991). A survey of solution techniques for the partially observed markov decision process. *Annals of Operations Research*, *32*(1), 215–230.

Woodley, B. R., How, J. P., & Kosut, R. L. (2001). Model free subspace based infinity control. In *American control conference, 2001. proceedings of the 2001* (Vol. 4, pp. 2712–2717).

Yongqiang, H., Jiabin, C., Xiaochun, T., & Nan, L. (2015, July). A robust data driven error damping method for inertial navigation system based on unfalsified adaptive control. In *Control conference (ccc), 2015 34th chinese* (p. 5455-5460). doi: 10.1109/ChiCC.2015.7260492

Zhang, N. L., & Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable markov decision processes. *Journal of Artificial Intelligence Research*, *14*, 29–51.

Zhang, W. (2001). *Algorithms for partially observable markov decision processes* (Unpublished doctoral dissertation). Citeseer.

Percentage contribution of each author in the manuscript

Fábio Nogueira da Silva – 80%

João Viana da Fonseca Neto – 20%