

# **CONTROL OF ROBOTIC SYSTEMS**

## **PROJECT 1**

### **REPORT ON ADAPTIVE CONTROL**

**PAPER:** ADAPTIVE OPTIMAL CONTROL FOR CONTINUOUS-TIME LINEAR TIME SYSTEMS BASED ON POLICY ITERATION

SUBMITTED BY,

MITHUN BHARADWAJ

## TABLE OF CONTENTS

<b>PART I - INTRODUCTION</b> .....	4
<b>1.1 TITLE OF THE PAPER BEING STUDIED:</b> .....	4
<b>1.2 ARTICLE INFORMATION:</b> .....	4
<b>1.3 EXPECTED RESULT:</b> .....	4
<b>1.4 REASONS FOR CHOOSING THIS PAPER:</b> .....	4
<b>1.5 TECHNICAL TERMS AND BRIEF DEFINITIONS</b> .....	5
 <b>PART II- CONTINUOUS TIME ADAPTIVE CRITIC SOLUTION FOR THE INFINITE HORIZON OPTIMAL CONTROL PROBLEM</b> .....	8
<b>2.1 POLICY ITERATION ALGORITHM</b> .....	9
<b>2.2 PROOF OF CONVERGENCE</b> .....	10
<b>2.3 SUMMARY OF PART (2)</b> .....	15
 <b>PART III – IMPLEMENTATION</b> .....	16
<b>3.1 ONLINE IMPLEMENTATION OF THE ADAPTIVE ALGORITHM BASED ON POLICY ITERATION</b> .....	16
<b>3.1.1 CALCULATING THE UNKNOWN PARAMETERS <math>pi</math></b> .....	17
<b>3.1.2 CONVERGENCE RATE:</b> .....	20
<b>3.2 SIMULATION: ONLINE LOAD FREQUENCY CONTROL DESIGN FOR A POWER SYSTEM</b> .....	21
<b>3.2.1 SIMULATION 1:</b> .....	22
<b>3.2.2 SIMULATION 2</b> .....	26
<b>3.3 MATLAB CODE</b> .....	29
<b>3.3.1 PROBLEM 1:</b> .....	29
<b>3.3.2 PROBLEM 2</b> .....	34

<b>PART IV – CONCLUSION .....</b>	<b>39</b>
<b>PART V - BIBLIOGRAPHY .....</b>	<b>40</b>
<b>PART VI – PAPERS.....</b>	<b>40</b>

## PART I – INTRODUCTION

### 1.1 TITLE OF THE PAPER BEING STUDIED:

ADAPTIVE OPTIMAL CONTROL FOR CONTINUOUS-TIME LINEAR TIME SYSTEMS  
BASED ON POLICY ITERATION

### 1.2 ARTICLE INFORMATION:

This paper authored by D. Vrabie, O. Pastravanu, M Abu-Khalaf and F.L. Lewis was accepted and published in the Automatica chapter of the Elsevier journal on 6<sup>th</sup> August, 2008. The organizations involved in this venture include

1. The University of Texas, Arlington
2. Technical University “Gh Asachi”- Automatic Control Department
3. The Mathworks Inc.

The essence of the paper can be simply put as an algorithm to solve the algebraic Riccati equation with an incomplete prior knowledge of the system dynamics. To quote from the paper abstract, **“In this paper we propose a new scheme based on adaptive critics for finding online the state feedback, infinite horizon, optimal control solution of linear continuous-time systems using only partial knowledge regarding the system dynamics”** (D. Vrabie, 2008). We will define each of these terms in a later section.

### 1.3 EXPECTED RESULT:

An algorithm based on a policy iteration technique that alternates between policy evaluation and policy update steps until an update of the control policy will no longer improve the system performance i.e an adaptive control algorithm which converges to the optimal control solution without knowing the complete internal dynamics of the system.

### 1.4 REASONS FOR CHOOSING THIS PAPER:

1. The content of the paper is in line with the course lectures and content with enough additional material to make it a good candidate for a thorough study.

2. The publication is quite recent and that favors the criteria of updating our knowledge regarding the current research areas in the field of Controls
3. Elsevier is a reputed journal with accessible information to quality publications and accessibility is an important aspect for any research.

## 1.5 TECHNICAL TERMS AND BRIEF DEFINITIONS

1. **Continuous time systems-** A continuous time system is the one that operates on a continuous time signal input and produces a continuous time signal output. In these systems, the variable *time* is viewed as continuous instead of taking discrete values.
2. **Linear Time Invariant (LTI) systems-** There are two conditions to be understood for an LTI system:

a. **Linearity:** A system  $H$  is said to be linear if it satisfies two important conditions:

i. **Additivity:**

$$H(x + y) = H(x) + H(y) \quad (1)$$

ii. **Homogeneity:** For every signal  $x$  and a scalar  $\alpha$ , we have  $H(\alpha x) = \alpha H(x)$

In a more compact form, we can say that a system is linear if for every signal  $x, y$  and scalars  $a, b$  we have that:

$$H(ax + by) = aH(x) + bH(y) \quad (2)$$

b. **Time invariance:** A time-invariant system is one whose behavior (its response to inputs) does not change with time. For continuous-time systems, we can define a system called a **delay operator**  $D_T$  so that if  $x$  is a function of time, then  $D_T(x)$  is a new function of time where:

$$(D_T(x))(t) = x(t - T) \quad (3)$$

Then for a time invariant system  $H$ , we have

$$H \cdot D_T = D_T \cdot H \quad (4)$$

A system that satisfies the conditions (a) and (b) is said to be a linear time invariant system. Linearity is a particularly important property of systems as it allows us to leverage the powerful tools of linear algebra, such as bases, eigenvectors, and eigenvalues, in their study. Time invariance is desirable because it eases computation while mirroring our

intuition that, all else equal, physical systems should react the same to identical inputs at different times.

3. **Infinite horizon-** When we talk about optimal control, it is important to specify “optimality” with respect to certain conditions and horizon is one of them. If the system has to meet the performance measures for a finite time  $T$ , then the problem is finite horizon and if we are concerned about the optimality of the system during the entire time span i.e till  $t = \infty$  then it is an infinite horizon problem.

4. **State space representation-** State space representation is the mathematical model of a physical system as a set of input, output and state variables by first order differential or difference equations. The most general form of a state space equation can be written as

$$\dot{X}(t) = A(t)X(t) + B(t)U(t) \quad (5)$$

$$Y(t) = C(t)X(t) + D(t)U(t) \quad (6)$$

where  $X(t)$  is an  $n \times 1$  vector that consists of state variables

$A(t)$  is an  $n \times n$  matrix part of the system dynamics associated with the state variables

$B(t)$  is an  $n \times m$  matrix that is part of the system dynamics associated with the inputs  $U(t)$

$Y(t)$  is the output of the system.

Here we can see that  $A(t)$  and  $B(t)$  together define the dynamics of the system and in our problem, we have stressed on the point of finding a solution for a system whose complete dynamics is not known i.e with the incomplete knowledge of these two matrices.

5. **State feedback-** This is a method employed in feedback control system theory to place the poles/eigenvalues of the system as desired (in the left half plane). By considering input as proportional to the state vector we get,

$$U(t) = KX(t) \quad (7)$$

and substituting this back into our state space representation of the system, we can rewrite equations (5) and (6) as follows,

$$\dot{X}(t) = (A + BK)X(t) \quad (8)$$

$$Y(t) = (C + DK)X(t) \quad (9)$$

6. **Optimal Control-** It is a branch of control theory which is an extension of calculus of variations and mathematical optimization that deals with the problem of finding a control

law for a given system such that a certain optimality criterion is achieved for example minimizing a cost function with respect to certain parameters.

## 7. Lyapunov Stability for LTI systems in State Space form

A function  $V: \mathbb{R}^n \rightarrow \mathbb{R}_+$  is a Lyapunov function if it satisfies the following

- $V(\vec{X}) > 0 \forall \vec{X} \neq \mathbf{0}$  and  $V(\mathbf{0}) = 0$
- In the absence of external input  $\dot{V}(\vec{X}(t)) < 0$  when  $\vec{X}(t) \neq \mathbf{0}$

**An LTI system specified in state-space form is said to be stable if and only if it has a Lyapunov function**

An LTI system specified in state space form is stable if and only if for any symmetric positive definite matrix  $Q$  there exists a symmetric positive definite matrix  $P$  such that the following Lyapunov equation holds

$$A^T P + P A = -Q$$

In which case  $V(\vec{X}) = \vec{X}^T P \vec{X}$  is a Lyapunov function.

## 8. LQR and Algebraic Ricatti Equation (ARE)

Minimizing the loss function

$$V(x(t_o), t_o) = \int_{t_o}^{\infty} (x^T(\tau) Q x(\tau) + u^T(\tau) R u(\tau)) d\tau$$

The optimal solution is given by the following controller

$$K = R^{-1} B^T P$$

Where  $P$  is in the symmetric positive solution of the following stationary Ricatti Equation

$$A^T P + P A - P B R^{-1} B^T P = -Q$$

The optimal cost is

$$V(x(t_o), t_o) = x^T(t_o) P x(t_o)$$

This part is repeated in section II in the context of developing the algorithm.

## PART II- CONTINUOUS TIME ADAPTIVE CRITIC SOLUTION FOR THE INFINITE HORIZON OPTIMAL CONTROL PROBLEM

This corresponds to section 2 of the paper involving the development of the algorithm and the proof of convergence.

Consider the linear time-invariant system described by

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (10)$$

where  $x(t) \in \mathbb{R}^n$  and  $u(t) \in \mathbb{R}^m$  and  $(A, B)$  is stabilizable, subject to the optimization control problem

$$u^*(t) = \arg \min_{u(t)} V(t_0, x(t_0), u(t)) \quad \forall t_0 \leq t \leq \infty \quad (11)$$

where the infinite horizon cost function to be minimized is expressed as

$$V(x(t_0), t_0) = \int_{t_0}^{\infty} (x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau))d\tau \quad (12)$$

with  $Q \geq 0$ ,  $R > 0$  and the pair  $(Q^{1/2}, A)$  is detectable.

The solution to this optimal control problem, determined by Bellman's optimality principle is given by

$$u(t) = -Kx(t) \quad (13)$$

$$K = R^{-1}B^TP \quad (14)$$

**(Note that these are the same formulas taught in the lectures with a negative sign reversed on K)**

where the  $P$  matrix is the positive definite solution of the Algebraic Riccati Equation (ARE)

$$A^TP + PA - PBR^{-1}B^TP = -Q \quad (15)$$

From the above equation, we can see that knowledge of the complete system dynamics (i.e both system matrix  $A$  and control input matrix  $B$ ) is required to solve for  $P$  and this underpins the motivation to develop algorithms that converge to the optimal control solution without the complete knowledge of the system.

The algorithm proposed as a way of progress to the solution for the above problem will solve online for the optimal control gain, the solution of the LQR problem, without using the knowledge regarding the system internal dynamics (i.e system matrix  $A$ ). The algorithm is based on an



Actor/Critic and consists in a two-step iteration namely the Critic update and the Actor update. The update of the Critic structure results in calculating the infinite horizon cost associated with the use of a given stabilizing controller. The Actor parameters (i.e. the controller feedback gain) are then updated in the sense of reducing the cost compared to the present control policy.

## 2.1 POLICY ITERATION ALGORITHM

Let  $K$  be a stabilizing gain for (10) under the assumption that  $(A, B)$  is stabilizable such that  $\dot{x}(t) = (A - BK)x(t)$  since we have taken  $u(t) = -Kx(t)$ . This comes directly from the state feedback control explanation.

Substituting the value of  $u$  in the equation for infinite horizon quadratic cost becomes,

$$V(x(t)) = \int_t^{\infty} \left( x^T(\tau) Q x(\tau) + (-Kx(\tau))^T R (-Kx(\tau)) \right) d\tau \quad (16)$$

$$= \int_t^{\infty} \left( x^T(\tau) Q x(\tau) + x(\tau)^T K^T R K x(\tau) \right) d\tau$$

$$V(x(t)) = \int_t^{\infty} \left( x^T(\tau) (Q + K^T R K) x(\tau) \right) d\tau = x^T(t) P x(t) \quad (17)$$

where  $P$  is the real symmetric positive definite solution of the Lyapunov matrix equation

$$(A - BK)^T P + P(A - BK) = -(Q + K^T R K) \quad (18)$$

and  $V(x(t))$  serves as a Lyapunov function for (10) with the controller gain  $K$ .

The cost function can be written as

$$V(x(t)) = \int_t^{t+T} \left( x^T(\tau) (Q + K^T R K) x(\tau) \right) d\tau + \int_{t+T}^{\infty} \left( x^T(\tau) (Q + K^T R K) x(\tau) \right) d\tau \quad (19)$$

The second term of the equation can be written as  $V(x(t+T))$  deriving from the way we defined our cost function. Thus we get,

$$V(x(t)) = \int_t^{t+T} \left( x^T(\tau) (Q + K^T R K) x(\tau) \right) d\tau + V(x(t+T)) \quad (20)$$

Using the convention  $x(t) = x_t$  (As it is followed in the paper), From (17), we can write the cost function as  $V(x_t) = x_t^T P x_t$ , and considering the initial stabilizing control gain  $K_1$ , we can get the following policy iteration scheme,

$$x_t^T P_i x_t = \int_t^{t+T} (x_\tau^T (Q + K_i^T R K_i) x_\tau) d\tau + x_{t+T}^T P_i x_{t+T} \quad (21)$$

$$K_{i+1} = R^{-1} B^T P_i \quad (22)$$

Note that equations (21) and (22) form a new policy iteration algorithm that does not involve the plant matrix  $A$ .

## 2.2 PROOF OF CONVERGENCE

**Lemma 1:** Assuming that  $A_i = (A - B K_i)$ , solving for  $P_i$  in equation (21) is equivalent to finding the solution of the underlying Lyapunov equation

$$A_i^T P_i + P_i A_i = -(Q + K_i^T R K_i) \quad (23)$$

**Proof:** Since we know there is a double implication on the Lyapunov criteria (i.e. an LTI system specified in state space is stable if and only if there exists a symmetric positive definite solution to the Lyapunov equation). In our case, we have assumed that  $A_i$  is stable and  $Q + K_i^T R K_i > 0$ , then there has to exist a unique solution to the Lyapunov equation  $P_i > 0$ . We also know that

$$V_i(x_t) = x_t^T P_i x_t \text{ is a Lyapunov function for the system } \dot{x} = A_i x.$$

Now, differentiating the Lyapunov function with respect to time, we get

$$\frac{d(x_t^T P_i x_t)}{dt} = \frac{d(x_t^T)}{dt} P_i x_t + x_t^T P_i \frac{d(x_t)}{dt}$$

Substituting for  $\dot{x}$  from the equation of the system, we can rewrite the above equation as

$$\frac{d(x_t^T P_i x_t)}{dt} = x_t^T (A_i^T P_i + P_i A_i) x_t = -x_t^T (Q + K_i^T R K_i) x_t \quad (24)$$

then  $\forall T > 0$ , the unique solution of the Lyapunov equation satisfies

$$\begin{aligned} \int_t^{t+T} (x_\tau^T (Q + K_i^T R K_i) x_\tau) d\tau &= - \int_t^{t+T} \frac{d(x_\tau^T P_i x_\tau)}{d\tau} d\tau \\ &= x_t^T P_i x_t - x_{t+T}^T P_i x_{t+T} \end{aligned} \quad (25)$$

This is the same as equation (21) and provided that  $\mathbf{A}_i$  is asymptotically stable the solution of (21) is the unique solution of (23)

As proved above, we get the same solution from solving (21) or (23) but notice that we can solve (21) without any prior knowledge of the system matrix  $\mathbf{A}$ .

Therefore we have formed the new policy iteration algorithm with the equations (21) and (22) but with the caveat that  $\mathbf{A}_i$  has to be stable since we need the solution to the Lyapunov equation to exist as shown in the proof.

**Lemma 2:** Assuming that the control policy  $\mathbf{K}_i$  is stabilizing with  $\mathbf{V}_i(\mathbf{x}_t) = \mathbf{x}_t^T \mathbf{P}_i \mathbf{x}_t$  (the cost associated with it), if (22) is used to update the control policy, then the new control policy will be stabilizing.

**Proof:** Taking the positive definite cost function  $\mathbf{V}_i(\mathbf{x}_t)$  as a Lyapunov function candidate for the state trajectories while using the control policy  $\mathbf{K}_{i+1}$ . From the Lyapunov stability for LTI systems, we know that one of the properties of  $\mathbf{V}_i(\mathbf{x}_t)$  is that it's derivative  $\dot{\mathbf{V}}_i(\mathbf{x}_t) < 0$ . Hence, if we can prove that the derivative of the Lyapunov candidate for the update control policy is less than 0, we can say that the new control policy will be stabilizing.

Taking the derivative of  $\mathbf{V}_i(\mathbf{x}_t)$  along the trajectories generated by  $\mathbf{K}_{i+1}$ , we get

$$\dot{\mathbf{V}}_i(\mathbf{x}_t) = \mathbf{x}_t^T (\mathbf{A}_i^T \mathbf{P}_i + \mathbf{P}_i \mathbf{A}_i) \mathbf{x}_t$$

Updating the values of  $\mathbf{A}_i$  with the state feedback matrices with the updated control policy,

$$\dot{\mathbf{V}}_i(\mathbf{x}_t) = \mathbf{x}_t^T (\mathbf{P}_i (\mathbf{A} - \mathbf{B} \mathbf{K}_{i+1}) + (\mathbf{A} - \mathbf{B} \mathbf{K}_{i+1})^T \mathbf{P}_i) \mathbf{x}_t$$

Adding and subtracting  $\mathbf{B} \mathbf{K}_i$ ,

$$\dot{\mathbf{V}}_i(\mathbf{x}_t) = \mathbf{x}_t^T (\mathbf{P}_i (\mathbf{A} - \mathbf{B} \mathbf{K}_i + \mathbf{B} \mathbf{K}_i - \mathbf{B} \mathbf{K}_{i+1}) + (\mathbf{A} - \mathbf{B} \mathbf{K}_i + \mathbf{B} \mathbf{K}_i - \mathbf{B} \mathbf{K}_{i+1})^T \mathbf{P}_i) \mathbf{x}_t$$

Rearranging the terms, we get,

$$\begin{aligned} \dot{\mathbf{V}}_i(\mathbf{x}_t) &= \mathbf{x}_t^T (\mathbf{P}_i (\mathbf{A} - \mathbf{B} \mathbf{K}_i) + (\mathbf{A} - \mathbf{B} \mathbf{K}_i)^T \mathbf{P}_i) \mathbf{x}_t \\ &\quad + \mathbf{x}_t^T (\mathbf{P}_i \mathbf{B} (\mathbf{K}_i - \mathbf{K}_{i+1}) + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{B}^T \mathbf{P}_i) \mathbf{x}_t \end{aligned} \quad (26)$$

The second term using the update given by (22), can be written as

$$\mathbf{x}_t^T (\mathbf{K}_{i+1}^T \mathbf{R} (\mathbf{K}_i - \mathbf{K}_{i+1}) + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t$$

Completing the squares of the second term through the following procedure,

$$\begin{aligned} & \mathbf{x}_t^T (\mathbf{K}_{i+1}^T \mathbf{R} (\mathbf{K}_i - \mathbf{K}_{i+1}) + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t \\ &= \mathbf{x}_t^T (\mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_i - \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1} + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t \end{aligned}$$

Adding and subtracting  $\mathbf{K}_i^T \mathbf{R} \mathbf{K}_i$ ,

$$\begin{aligned} &= \mathbf{x}_t^T (-\mathbf{K}_i^T \mathbf{R} \mathbf{K}_i + \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_i - \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i \\ &\quad + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t \\ &= \mathbf{x}_t^T (-(\mathbf{K}_i^T - \mathbf{K}_{i+1}^T) \mathbf{R} \mathbf{K}_i - \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i \\ &\quad + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t \\ &= \mathbf{x}_t^T (-(\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_i - \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i \\ &\quad + (\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t \\ &= \mathbf{x}_t^T (-(\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} (\mathbf{K}_i - \mathbf{K}_{i+1}) - \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i) \mathbf{x}_t \end{aligned} \quad (27)$$

The first term of equation (26) can be rewritten using equation (23),

$$\mathbf{x}_t^T (\mathbf{P}_i (\mathbf{A} - \mathbf{B} \mathbf{K}_i) + (\mathbf{A} - \mathbf{B} \mathbf{K}_i)^T \mathbf{P}_i) \mathbf{x}_t = -\mathbf{x}_t^T (\mathbf{Q} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i) \mathbf{x}_t \quad (28)$$

Adding equations (27) and (28), we get equation (26)0

$$\begin{aligned} \dot{V}_i(\mathbf{x}_t) &= -\mathbf{x}_t^T (\mathbf{Q} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i) \mathbf{x}_t + \\ &\quad \mathbf{x}_t^T (-(\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} (\mathbf{K}_i - \mathbf{K}_{i+1}) - \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i) \mathbf{x}_t \end{aligned}$$

$$\dot{V}_i(\mathbf{x}_t) = -\mathbf{x}_t^T ((\mathbf{K}_i - \mathbf{K}_{i+1})^T \mathbf{R} (\mathbf{K}_i - \mathbf{K}_{i+1})) \mathbf{x}_t - \mathbf{x}_t^T (\mathbf{Q} + \mathbf{K}_{i+1}^T \mathbf{R} \mathbf{K}_{i+1}) \mathbf{x}_t \quad (29)$$

Therefore, with the initial assumptions when we setup the problem that  $\mathbf{Q} > \mathbf{0}$ ,  $\mathbf{R} > \mathbf{0}$ ,  $V_i(\mathbf{x}_t)$  is a Lyapunov function proving that the updated control policy  $\mathbf{u} = -\mathbf{K}_{i+1} \mathbf{x}$ , with  $\mathbf{K}_{i+1}$  given by equation (22) is stabilizing since we were able to prove that  $\dot{V}_i(\mathbf{x}_t) < \mathbf{0}$ , through equation (29)

**Remark 2:** It can be concluded from the previous proof that if the initial control policy  $K_1$  is stabilizing, then all policies obtained using the iteration (21) and (22) will also be stabilizing.

Let  $\mathbf{Ric}(\mathbf{P}_i)$  be the matrix valued function defined as

$$\mathbf{Ric}(\mathbf{P}_i) = \mathbf{A}^T \mathbf{P}_i + \mathbf{P}_i \mathbf{A} + \mathbf{Q} - \mathbf{P}_i \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}_i \quad (30)$$

And  $\mathbf{Ric}'_{\mathbf{P}_i}$  denote the Fréchet derivative of  $\mathbf{Ric}(\mathbf{P}_i)$  taken with respect to  $\mathbf{P}_i$ .

### ASIDE 1: THE ESSENCE OF FRÉCHET DERIVATIVE

The primary difference is that we move everything in the first principle formula to one side of the equation.

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$$

becomes

$$0 = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} - \frac{hf'(a)}{h}$$

so that

$$0 = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a) - f'(a)h}{h}$$

Taking the absolute values doesn't change the equation

$$0 = \lim_{h \rightarrow 0} \frac{|f(a+h) - f(a) - f'(a)h|}{|h|}$$

The utility of defining derivatives this way is that it extends to situations other than that of functions of one variable. We can recast this and this time instead of sending  $h$  to  $0$  we can equivalently send  $x$  to  $a$  ( $h = x - a$ )

Then we get

$$0 = \lim_{x \rightarrow a} \frac{|f(x) - f(a) - f'(a)(x-a)|}{|(x-a)|}$$

Now if we replace  $\mathbf{x}$  and  $\mathbf{a}$  with vectors,  $\mathbf{f}$  with a function from vectors to vectors and think of the absolute value as the length of a vector, we have a perfectly reasonable definition for a derivative, except for  $\mathbf{f}'(\mathbf{a})(\mathbf{x} - \mathbf{a})$ .

We need to replace the number  $\mathbf{f}'(\mathbf{a})$  with a linear operator (or a matrix = Jacobian) for it to make sense.

This is an especially convenient way of presenting derivatives in multivariable calculus. We see the derivative as being a linearization which well approximates our function:

$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{a}) + \mathbf{f}'(\mathbf{a})(\mathbf{x} - \mathbf{a})$  (the tangent). When  $\mathbf{f}$  is a scalar-valued function,  $\mathbf{f}'(\mathbf{a})$  is just the gradient. Also, we get that approaching this multivariate limit along coordinate axes reduces to partial derivatives. This then explains why partials (and in fact all directional derivatives) can exist at a point even when a function is not differentiable (a limit can exist along all lines but still fail to exist).

**Lemma 3:** The iteration between (21) and (22) is equivalent to Newton's method of solving the ARE

$$\mathbf{P}_i = \mathbf{P}_{i-1} - (\mathbf{Ric}'_{\mathbf{P}_{i-1}})^{-1} \mathbf{Ric}(\mathbf{P}_{i-1}) \quad (31)$$

**Proof:** Combining equations (22) and (23), we get

$$\mathbf{A}_i^T \mathbf{P}_i + \mathbf{P}_i \mathbf{A}_i = -(\mathbf{Q} - \mathbf{P}_{i-1} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}_{i-1}) \quad (32)$$

Subtracting  $\mathbf{A}_i^T \mathbf{P}_{i-1} + \mathbf{P}_{i-1} \mathbf{A}_i$  on both sides of the equation gives,

$$\mathbf{A}_i^T (\mathbf{P}_i - \mathbf{P}_{i-1}) + (\mathbf{P}_i - \mathbf{P}_{i-1}) \mathbf{A}_i = -(\mathbf{P}_{i-1} \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_{i-1} + \mathbf{Q} - \mathbf{P}_{i-1} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}_{i-1}) \quad (33)$$

Using our definitions of  $\mathbf{Ric}(\mathbf{P}_i)$  and  $\mathbf{Ric}'_{\mathbf{P}_i}$  defined in remark (2), we can see that this is equivalent to the Newton method formulation. (Chun-Hua Guo, 1998)

**NOTE:** I will not be going into the details of Newton method formulation for the sake of brevity. But to give this topic some closure, I have referenced a paper that gives the formulation necessary to understand **lemma 3**. I will also be attaching the paper at the end of the report.

**Lemma 4:** Under the assumptions of stabilizability of  $(A, B)$  and detectability of  $(Q^{1/2}, A)$ , with  $Q \geq 0, R > 0$ , in the cost index (12), the policy iteration (21) and (22), conditioned by an initial stabilizing controller, converges to the optimal control solution given by (14), where matrix  $P$  satisfies the ARE (15).

**Proof:** It is a proven result that the iterative solution using Newton method formulation converges to the solution of the ARE. (Kirsten Morris, 2004) and the equivalence established between the proposed online policy and the newton method formulation for the solution of an ARE, ensures that the policy iteration algorithm converges to the optimal control solution.

### 2.3 SUMMARY OF PART (2)

In section 2, we have developed and proposed an algorithm that consists of two parts, namely policy iteration and policy update that converges to the optimal control solution as indicated by the ARE without knowing the complete system dynamics i.e to be specific, complete knowledge of the  $A$  matrix is not required, but  $B$  matrix has to be known for the policy update step.

Section 2 in the report follows the structure of part 2 of the paper closely as it is the crux of the idea. The derivations and explanations here are more in detail as it should be with a reference to any results that has not been explicitly proved in the report.

## PART III – IMPLEMENTATION

The developed algorithm needs knowledge of the  $\mathbf{B}$  matrix which explicitly appears in the policy update step. The information regarding the  $\mathbf{A}$  matrix is embedded in the states  $x(t)$  and  $x(t + T)$  which are observed online and hence the knowledge regarding the system matrix is not required for the computation of the optimal control solution.

### 3.1 ONLINE IMPLEMENTATION OF THE ADAPTIVE ALGORITHM BASED ON POLICY ITERATION

To find the parameters (matrix  $P_i$ ) of the cost function associated with the policy  $K_i$  in (21), the term  $\mathbf{x}^T(t)\mathbf{P}_i\mathbf{x}(t)$  can be written as

$$\mathbf{x}^T(t)\mathbf{P}_i\mathbf{x}(t) = \bar{\mathbf{p}}_i^T \bar{\mathbf{x}}(t) \quad (34)$$

Where  $\bar{\mathbf{x}}(t)$  denotes the Kronecker product quadratic polynomial basis vector with elements  $\{x_i(t)x_j(t)\}_{i=1,n;j=1,n}$  and  $\bar{\mathbf{p}} = v(P)$  with  $v(\cdot)$  a vector valued matrix function that acts on symmetric matrices and returns a column vector by stacking the elements of the diagonal and upper triangular part of the symmetric matrix into a vector.

**ASIDE 2:** For the sake of demonstration of the above statement, let us assume that  $P$  is 2x2 matrix

$$\begin{aligned} \mathbf{x}^T(t)\mathbf{P}_i\mathbf{x}(t) &= [x_1 \quad x_2] \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= [p_{11} \quad p_{12} \quad p_{22}] \begin{bmatrix} x_1^2 \\ 2x_1x_2 \\ x_2^2 \end{bmatrix} \end{aligned}$$

In this case the matrix on the left will be  $\bar{\mathbf{p}} = v(P)$  and the matrix on the right is the set of quadratic basis vectors equivalent to  $\bar{\mathbf{x}}(t)$ .



Using (34) we can rewrite equation (9) as follows

$$\bar{\mathbf{p}}_i^T (\bar{\mathbf{x}}(t) - \bar{\mathbf{x}}(t + T)) = \int_t^{t+T} (\mathbf{x}^T(\tau)(\mathbf{Q} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i) \mathbf{x}(\tau)) d\tau \quad (35)$$

In this equation  $\bar{\mathbf{p}}_i$  is the vector of unknown and  $(\bar{\mathbf{x}}(t) - \bar{\mathbf{x}}(t + T))$  acts as the regression vector (this can be calculated as we can measure the value of the states at  $t$  and  $t + T$ ). The right hand side target function,  $\mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i)$  (also known as the reinforcement on the time interval  $[t, t + T]$ ),

$$\mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i) = \int_t^{t+T} (\mathbf{x}^T(\tau)(\mathbf{Q} + \mathbf{K}_i^T \mathbf{R} \mathbf{K}_i) \mathbf{x}(\tau)) d\tau$$

Is measured based on the system states over the time interval  $[t, t + T]$ .

Considering  $\dot{\mathbf{V}}(t) = (\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t))$  as a definition of a new state  $\mathbf{V}(t)$ , augmenting the system (10), the value of  $\mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i)$  can be calculated by taking two measurements of this newly introduced system state since  $\mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i) = \mathbf{V}(t + T) - \mathbf{V}(t)$ .

The parameter vector  $\bar{\mathbf{p}}_i$  of the function  $\mathbf{V}_i(\mathbf{x}(t))$  (i.e the critic) which will then yield the matrix  $\mathbf{P}_i$ , is found by minimizing in the least square sense, the error between the target function  $\mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i)$ , and the parameterized left hand side of equation (35).

### 3.1.1 CALCULATING THE UNKNOWN PARAMETERS $\bar{\mathbf{p}}_i$

Consider the equation

$$\bar{\mathbf{p}}_i^T (\bar{\mathbf{x}}(t) - \bar{\mathbf{x}}(t + T)) = \mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i)$$

Where  $\mathbf{d}(\bar{\mathbf{x}}(t), \mathbf{K}_i)$  represents the true value of the function and the estimates of the parameters  $\bar{\mathbf{p}}_i$  is to be found such that the parameters satisfy the equation as closely as possible.

Taking,

$$\begin{aligned} X &= [\bar{x}_\Delta^1 \quad \bar{x}_\Delta^2 \quad \dots \quad \bar{x}_\Delta^N] \\ \text{where } \bar{x}_\Delta^i &= \bar{x}^i(t) - \bar{x}^i(t+T) \\ Y &= [d(\bar{x}^1, K_i) \quad d(\bar{x}^2, K_i) \quad \dots \quad d(\bar{x}^N, K_i)]^T \end{aligned}$$

Assuming the square loss function  $J(\bar{p}_i)$ , we get

$$J(\bar{p}_i) = \sum_{n=1}^N (y_n - \bar{p}_i^T \bar{x}_\Delta^n)^2 \quad (36-i)$$

We need to minimize this over  $\bar{p}_i$ . Differentiating the loss function with respect to  $\bar{p}_i$  and equating to 0 gives,

$$\nabla_{\bar{p}_i} J(\bar{p}_i) = \nabla_{\bar{p}_i} \left( \sum_{n=1}^N (y_n - \bar{p}_i^T \bar{x}_\Delta^n)^2 \right) = 0$$

$$= - \sum_{n=1}^N -2(y_n - \bar{p}_i^T \bar{x}_\Delta^n) \bar{x}_\Delta^n = 0$$

$$\sum_{n=1}^N \bar{p}_i^T \bar{x}_\Delta^n \bar{x}_\Delta^n = \sum_{n=1}^N y_n \bar{x}_\Delta^n \quad (36-ii)$$

Noting that  $\bar{p}_i^T \bar{x}_\Delta^n$  is a scalar, we can write,

$$\bar{p}_i^T \bar{x}_\Delta^n = \bar{x}_\Delta^{nT} \bar{p}_i \text{ and} \quad (36-iii)$$

$$\sum_{n=1}^N (\bar{p}_i^T \bar{x}_\Delta^n) \bar{x}_\Delta^n = \sum_{n=1}^N \bar{x}_\Delta^n (\bar{p}_i^T \bar{x}_\Delta^n) \quad (36-iv)$$

Using the above (36-iii) and (36-iv), equations, we can rewrite (36-ii) as follows,

$$\sum_{n=1}^N \bar{x}_{\Delta}^n \bar{x}_{\Delta}^{nT} \bar{p}_i = \sum_{n=1}^N y_n \bar{x}_{\Delta}^n$$

Writing the above equation in matrix form, we get,

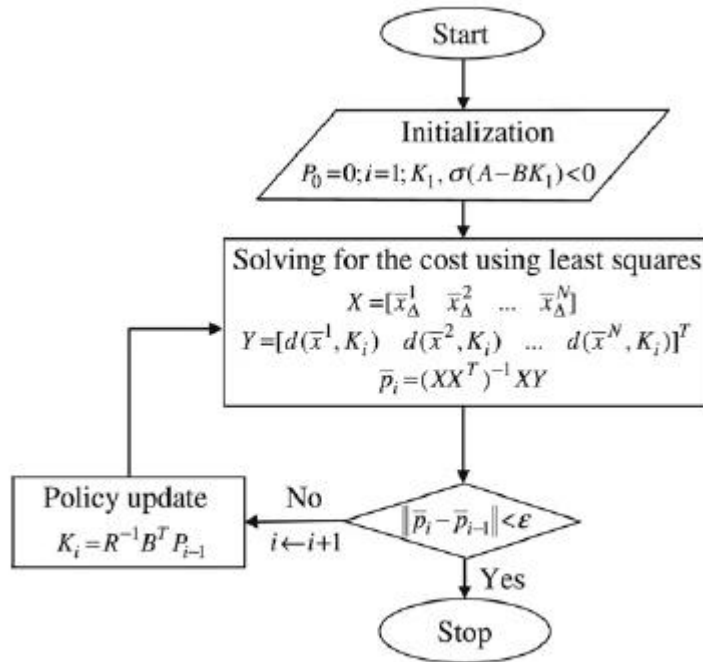
$$(XX^T)\bar{p}_i = XY$$

Therefore  $\bar{p}_i$  can be said to be,

$$\bar{p}_i = (XX^T)^{-1}XY \quad (36)$$

Equation (36) gives the value of  $\bar{p}_i$  as a minimization of the least squares criterion.

The algorithm can be picturized as follows:



**Fig. 1.** Continuous-time linear policy iteration algorithm.

(D. Vrabie, 2008)

### 3.1.2 CONVERGENCE RATE:

The least-squares problem can be solved in real-time after a sufficient number of data points are collected along a single state trajectory, under the regular presence of an excitation requirement. The number of data points in the  $\mathbf{X}$  and  $\mathbf{Y}$  matrices should be more than the number of independent points in the  $\mathbf{P}$  matrix ie assuming  $\mathbf{P}$  is an  $n \times n$  matrix, we need at least  $N$  data points where  $N$  is given by  $N \geq \frac{n(n+1)}{2}$  (Since we know  $\mathbf{P}$  is a symmetric matrix).

The solution given by the square loss minimization can also be obtained using recursive estimation algorithms such as gradient descent or the recursive least square algorithm, in which case a persistence of excitation condition is required.

Newton's method to solve an ARE has quadratic convergence (Kirsten Morris, 2004) and hence by proven equivalence, the proposed algorithm will have the same speed. For the case in which solution of equation (21) is obtained iteratively, the convergence speed of the online algorithm proposed in this algorithm decreases. In this case at each step in the policy iteration algorithm (solving equations (21) and (22)), a recursive gradient algorithm which most often has exponential convergence will be used. Therefore depending on the technique chosen to solve the policy iteration, we can vary the convergence rate of the proposed online algorithm.

The proposed online policy iteration procedure requires only measurements of the states at discrete moments in time,  $t$  and  $t + T$ , as well as knowledge of the observed cost over the time interval  $[t, t + T]$ , which is  $d(\bar{x}(t), K_i)$ . Therefore there is no required knowledge about the system  $\mathbf{A}$  matrix for the evaluation of the cost or the update of the control policy. The  $\mathbf{B}$  matrix is required for the update of the control policy, using (22), and this makes the tuning algorithm only partially model-free. In this case, the control policy is updated at time  $t + T$ , after observing the state  $x(t + T)$  and it is used for controlling the system during the time interval  $[t + T, t + 2T]$ ; thus the algorithm is suitable for online implementation from the control theory point of view.

The structure of the system with the adaptive controller is presented in Fig. 2. Most important is that the system was augmented with an extra state  $\mathbf{V}(t)$  defined as

$\dot{V}(t) = (x^T(t)Qx(t) + u^T(t)Ru(t))$  in order to extract the information regarding the cost associated with the given policy. This newly introduced system dynamics is part of the adaptive critic based controller thus the control scheme is actually a dynamic controller with the state given by the cost function  $V$ . One can observe that the adaptive optimal controller has a hybrid structure with a continuous-time internal state followed by a sampler and discrete time update rule i.e the update of both the actor and the critic is performed at discrete moments in time. Nevertheless, the control action is a full-fledged continuous-time control, only that its constant gain is updated only at certain points in time.

The critic will stop updating the control policy when the difference between the performances of the system evaluated at two consecutive steps will cross below a designer specified threshold, i.e. the algorithm has converged to the optimal controller.

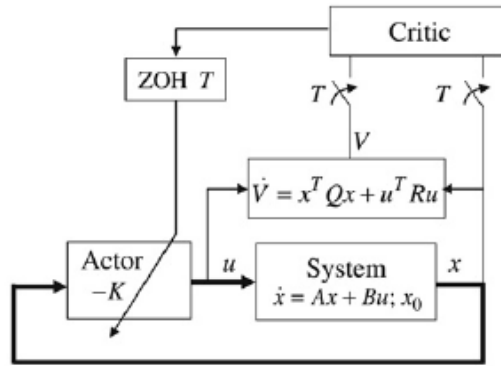


Fig. 2. Structure of the system with optimal adaptive controller.

(D. Vrabie, 2008)

### 3.2 SIMULATION: ONLINE LOAD FREQUENCY CONTROL DESIGN FOR A POWER SYSTEM

#### Problem selection:

The problem chosen is that of designing a controller for a power system. Power systems are characterized by nonlinearities, linear state feedback control is regularly employed for load-frequency control at a certain nominal operating point. This simplifies the design problem, but a new issue appears as only the range of the plant parameters can be determined. Thus it is

particularly advantageous to apply model-free methods to obtain the optimal LQR controller for a given operating point of the power system.

### Problem Specifics:

- The matrices of the plant are,

$$A_{nom} = \begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix}$$

$$B = [0 \quad 0 \quad 13.736 \quad 0]^T$$

- For this simulation it was considered that the linear model of the real plant internal dynamics is given by

$$A = \begin{bmatrix} -0.0665 & 11.5 & 0 & 0 \\ 0 & -2.5 & 2.5 & 0 \\ -9.5 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix},$$

- Taking  $T = 0.05s$  and initial condition  $x_0 = [0 \quad 0.1 \quad 0 \quad 0]$
- $Q$  and  $R$  are chosen to be identity matrices of appropriate dimensions
- In this case  $P$  will be a  $4 \times 4$  matrix, so the number of independent elements will be  $\frac{n(n+1)}{2} = 10$ . Therefore, we need at least  $N \geq 10$  points to calculate  $P$ . Here we have chosen that to be 20. We also know  $T = 0.05s$ , so we can say that the update happens every 1s.

### 3.2.1 SIMULATION 1:

We start the iterative algorithm while using the controller calculated for the nominal model of the plant  $A_{nom}$ , and the controller parameters will be adapted online to converge to the optimal controller for the real plant.

The values of the  $P$  matrix parameters at  $t = 0s$  corresponds to the solution of the Riccati equation that was solved considering the approximate model of the system  $A_{nom}$

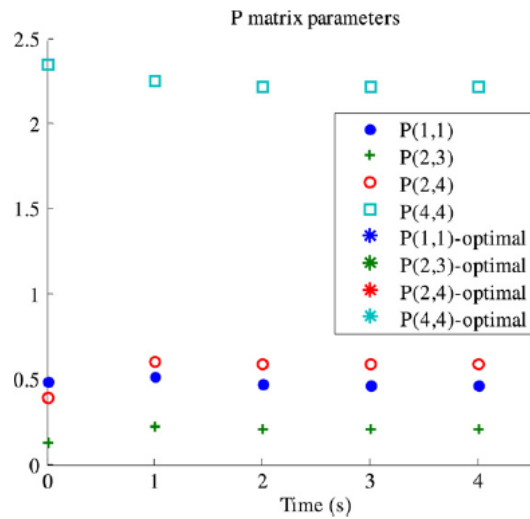
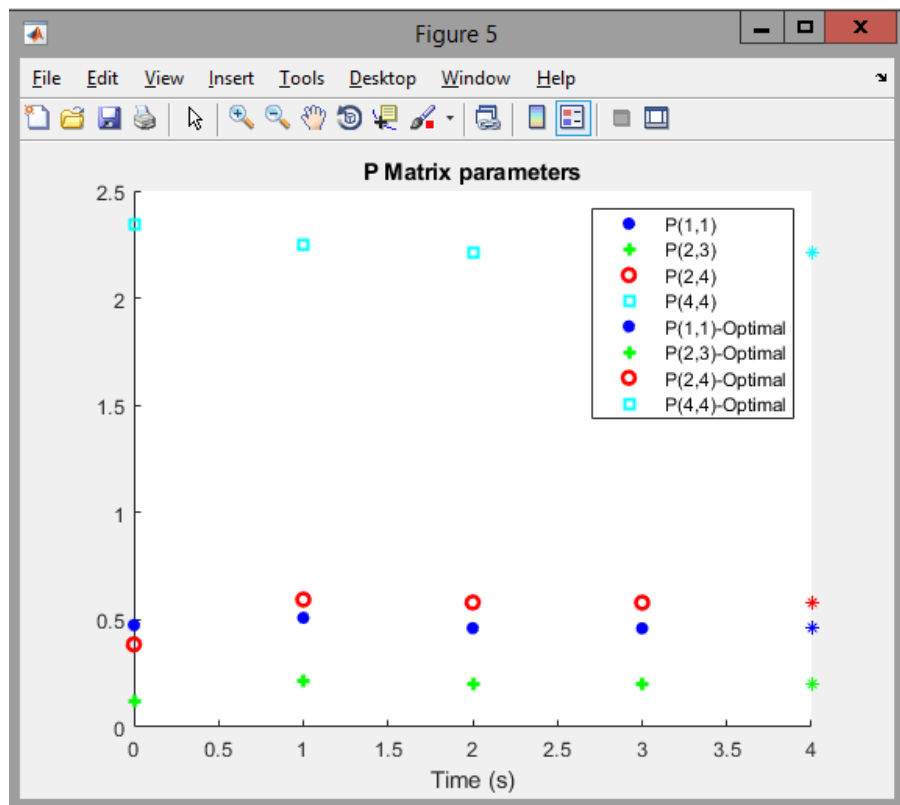


Fig. 3. Evolution of the parameters of the  $P$  matrix for the duration of the experiment.

### Result given in paper



### Result obtained through simulation of the algorithm

NOTE: The optimal values of  $P$  matrix elements are represented by \* at  $t=5$ . (Legend for those couldn't be corrected in time)

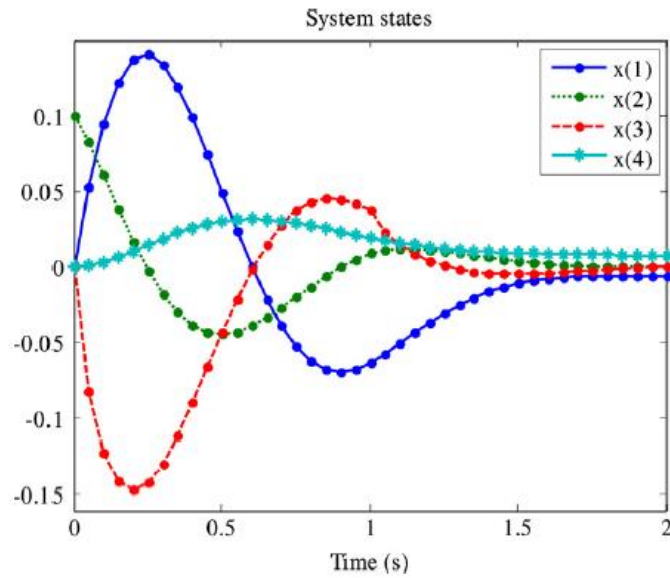
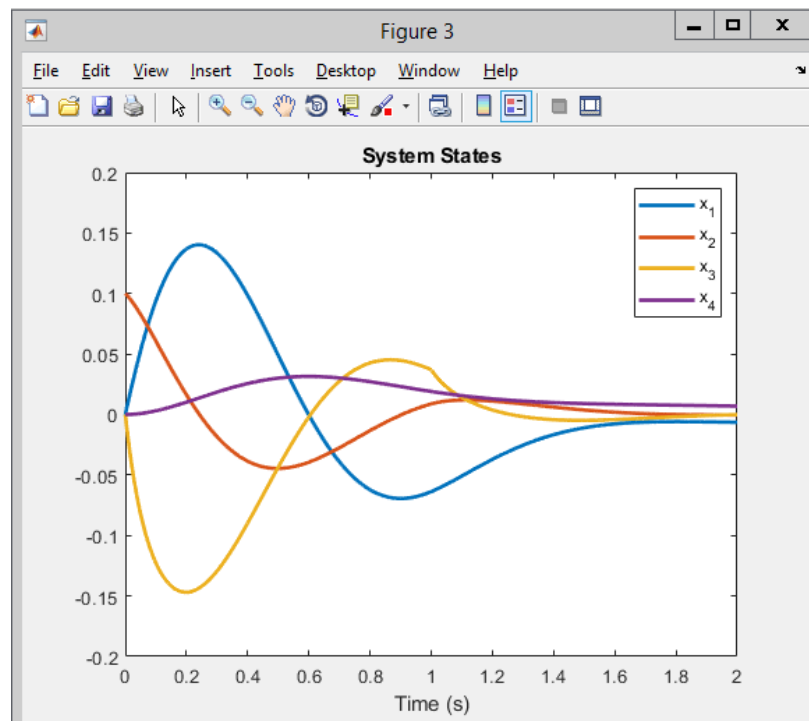


Fig. 4. System state trajectories (lines) and state information that was actually used for the Critic update (dots on the state trajectories).

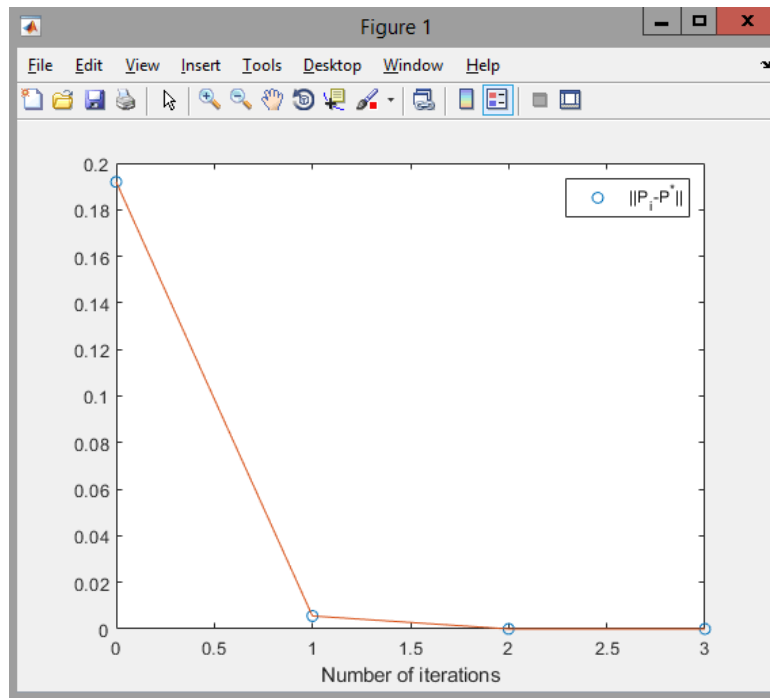
### Result for the system states in the paper



### Result for the system states through simulation



**Additional plot that verifies the convergence of the algorithm**



$$\text{norm}(P_i - P^*)$$

Furthermore the paper states that the optimal controller was obtained after 4 iterations

```
Command Window
New to MATLAB? See resources for Getting Started.

it =

    3

it =

    4

Number of iterations to obtain optimal control solution 4
>>
fx >> |
```

**Converges after 4 iterations in simulation**

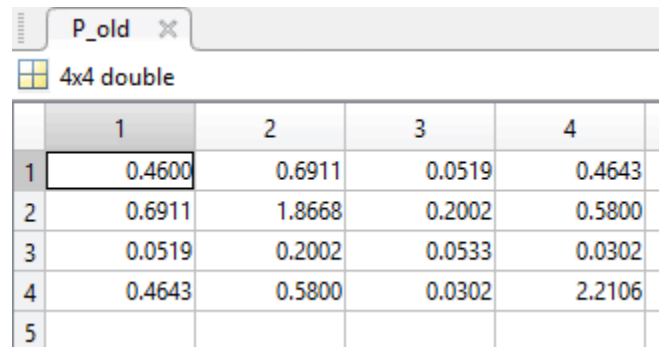
and the optimal solution obtained as given in the paper is

$$P = \begin{bmatrix} 0.4599 & 0.6910 & 0.0518 & 0.4641 \\ 0.6910 & 1.8665 & 0.2 & 0.5798 \\ 0.0518 & 0.2 & 0.0532 & 0.03 \\ 0.4641 & 0.5798 & 0.03 & 2.2105 \end{bmatrix}$$

Whereas the optimal solution as obtained by directly solving the ARE is given by,

$$P^* = \begin{bmatrix} 0.4600 & 0.6911 & 0.0519 & 0.4642 \\ 0.6911 & 1.8668 & 0.2002 & 0.5800 \\ 0.0519 & 0.2002 & 0.0532 & 0.0302 \\ 0.4642 & 0.5800 & 0.0302 & 2.2106 \end{bmatrix}$$

The result obtained through simulation of the algorithm in MATLAB is as follows



	1	2	3	4
1	0.4600	0.6911	0.0519	0.4643
2	0.6911	1.8668	0.2002	0.5800
3	0.0519	0.2002	0.0533	0.0302
4	0.4643	0.5800	0.0302	2.2106
5				

**P matrix obtained through simulation of the algorithm**

As it can be observed that the convergence is accurate to  $10^{-4}$  of the optimal solution as stated in the paper and hence the results of the simulation can be verified.

### 3.2.2 SIMULATION 2

In the case when the system to be controlled is itself stable this allows starting the iteration while using no controller (i.e. the initial controller is zero and no identification procedure needs to be performed).

So initializing the K matrix to 0

Result provided in the paper for this is case is

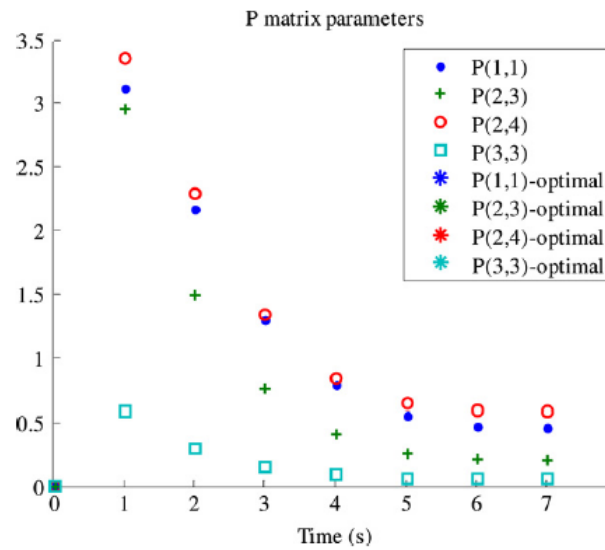
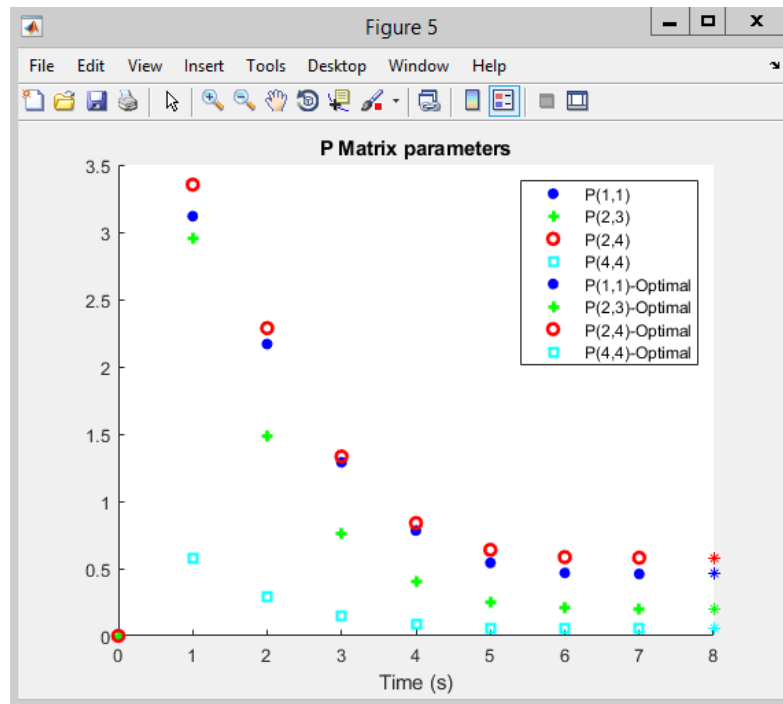


Fig. 5. Evolution of the parameters of the  $P$  matrix for the duration of the experiment when the adaptive algorithm was started without controller for the power system.

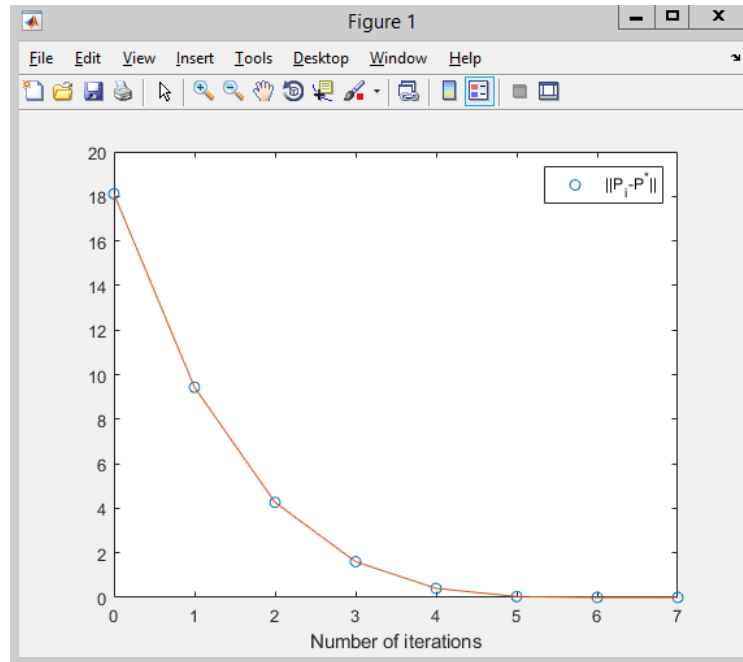
### P matrix parameters for problem 2 given in the paper

The result obtained through simulation of the algorithm in MATLAB is as follows



### P Matrix parameters obtained through simulation

Furthermore, we can verify the convergence of the solution from the following plot



$$\text{norm}(P_i - P^*)$$

The number of iterations to converge to the optimal control solution in this case is 7 and the solution as mentioned in the paper is as follows

$$P = \begin{bmatrix} 0.4601 & 0.6912 & 0.0519 & 0.4643 \\ 0.6912 & 1.8672 & 0.2003 & 0.5800 \\ 0.0519 & 0.2003 & 0.0533 & 0.0302 \\ 0.4643 & 0.5800 & 0.0302 & 2.2107 \end{bmatrix}$$

The results of the above parameters obtained through simulation in the above case are,

```

Command Window
New to MATLAB? See resources for Getting Started.

it =
    7

it =
    8

Number of iterations to obtain optimal control solution 8
fx >>

```

**The solution converges after 8 iterations in the simulation**

	1	2	3	4
1	0.4599	0.6911	0.0519	0.4642
2	0.6911	1.8665	0.2002	0.5800
3	0.0519	0.2002	0.0533	0.0302
4	0.4642	0.5800	0.0302	2.2105

**P matrix obtained through simulation for problem 2**

Hence it can be verified for both cases that the solution presented in the paper and the simulations run on MATLAB give equivalent solutions and the validity of the algorithm can be established.

### 3.3 MATLAB CODE

#### 3.3.1 PROBLEM 1:

**CODE NAME: Control\_Proj1\_Sim\_Prob1**

```
%Control of Robotic Systems course- Project 1
%Code for the paper "Adaptive optimal control for continuous-time linear
systems based on policy iteration"_Problem1

global A B K xn un
clc;
x_save=[];
t_save=[];

% System matrices used for simulation purpose
A_nom=[-0.0665 8 0 0; 0 -3.663 3.663 0; -6.86 0 -13.736 -13.736; 0.6 0 0 0];
A=[-0.0665 11.5 0 0; 0 -2.5 2.5 0; -9.5 0 -13.736 -13.736; 0.6 0 0 0];

B=[0; 0; 13.736; 0];

x0=[0;0.1;0;0]; %initial conditions

[xn,un]=size(B);%size of B. un-column #, xn row #
```

```

% Set the weighting matrices for the cost function
Q=eye(4);
R=eye(1);

% Initialize the feedback gain matrix

[P1, L, K]=care(A_nom,B,Q);
P_old=P1;

T=.05; %Duration of time for each integration

Dxx=[];XX=[];XU=[]; % Data matrices

X=[x0;kron(x0',x0')';kron(x0,zeros(un,1))];

P=eye(xn)*10; % Initialize the previous cost matrix
it=0; % Counter for iterations
p_save=[]; % Track the cost matrices in all the iterations
k_save=[]; % Track the feedback gain matrix in each iterations

[P0,K0]=care(A,B,Q); % Calculate the ideal solution for comparion purpose
k_save=[norm(K-K0)];

while norm(P-P_old)>10^-3 % Stopping criterion for learning
    for i=1:20
        % Simulation the system and at the same time collect online info.
        [t,X]=ode45(@mysys, [it+(i-1)*T,it+i*T],X(end,:));

        %Append new data to the data matrices

```

```

Dxx=[Dxx;kron(X(end,1:xn),X(end,1:xn))-kron(X(1,1:xn),X(1,1:xn))];
XX=[XX;X(end,xn+1:xn+xn^2)-X(1,xn+1:xn+xn^2)];
XU=[XU;X(end,xn+xn^2+1:end)-X(1,xn+xn^2+1:end)];

% Keep track of the system trajectories
x_save=[x_save;X];
t_save=[t_save;t];
end

if it==0
    P_old=P1;
else
    P_old=P; % Update the previous cost matrix
end

QK=Q+K'*R*K; % Update the Qk matrix
X2=XX*kron(eye(xn),K'); %
X1=[Dxx,-X2-XU]; % Left-hand side of the key equation
Y=-XX*QK(:); % Right-hand side of the key equation

pp = pinv(X1)*Y;

P = reshape(pp(1:xn*xn), [xn, xn]);
P = (P +P')/2;

figure(4)

scatter(it,P_old(1,1),'o','filled','b')
hold on
scatter(it,P_old(2,3),'+','g','Linewidth',2)
hold on
scatter(it,P_old(2,4),'o','r','Linewidth',2)
hold on
scatter(it,P_old(4,4),'s','c','Linewidth',2)
hold on

```

```

p_save=[p_save,norm(P-P0)];      % Keep track of the cost matrix

K=inv(R)*transpose(B)*P;          % Get the improved gain matrix
k_save=[k_save,norm(K-K0)];      % Keep track of the control gains

    it=it+1      % Update and display the # of iters
end
fprintf('\nNumber of iterations to obtain optimal control solution %d\n',it)

% Plot the trajectories

figure(4)

scatter(it,P0(1,1),'*','b')
hold on
scatter(it,P0(2,3),'*','g')
hold on
scatter(it,P0(2,4),'*','r')
hold on
scatter(it,P0(4,4),'*','c')
hold on
axis([0 it 0 2.5])
xlabel('Time (s)')
title('P Matrix parameters')
legend('P(1,1)', 'P(2,3)', 'P(2,4)', 'P(4,4)', 'P(1,1)-Optimal', 'P(2,3)-
Optimal', 'P(2,4)-Optimal', 'P(4,4)-Optimal')

figure(1)
plot([0:length(p_save)-1],p_save,'o',[0:length(p_save)-1],p_save)
legend('||P_i-P^*||')
xlabel('Number of iterations')

```



```

figure(2)
plot([0:length(k_save)-1],k_save,'^',[0:length(k_save)-1],k_save)
legend('||K_i-K^*||')
xlabel('Number of iterations')

% Post-learning simulation
[tt,xx]=ode23(@mysys,[t(end) 20],X(end,:));

% Keep track of the post-learning trajectories
t_final=[t_save;tt];
x_final=[x_save;xx];

figure(3)

plot(t_final,x_final(:,1:4),'Linewidth',2)
axis([0,2,-0.2,0.2]) %axis can be changed to get the complete
trajectory
legend('x_1','x_2','x_3','x_4')
xlabel('Time (s)')
title('System States')

% The following function gives the dynamics of the system. Also,

function dX=mysys(t,X)
    global A B K xn
    x=X(1:xn);

    u=-K*x;
    dx=A*x+B*u;
    dxx=kron(x',x)';
    dux=kron(x',u)';
    dX=[dx;dxx;dux];
end

```

### 3.3.2 PROBLEM 2

#### CODE NAME: Control\_Proj1\_Sim\_Prob2

```
%Control of Robotic Systems course- Project 1
%Code for the paper "Adaptive optimal control for continuous-time linear
systems based on policy iteration"_Problem2

global A B K xn un
clc;
x_save=[];
t_save=[];

% System matrices used for simulation purpose
A_nom=[-0.0665 8 0 0; 0 -3.663 3.663 0; -6.86 0 -13.736 -13.736; 0.6 0 0 0];
A=[-0.0665 11.5 0 0; 0 -2.5 2.5 0; -9.5 0 -13.736 -13.736; 0.6 0 0 0];

B=[0; 0; 13.736; 0];

x0=[0;0.1;0;0]; %initial conditions

[xn,un]=size(B);%size of B. un-column #, xn row #

% Set the weighting matrices for the cost function
Q=eye(4);
R=eye(1);

% Initialize the feedback gain matrix
K=zeros(un,xn); % Only if A is Hurwitz, K can be set as zero.
P_old=zeros(xn);

T=.05; %Duration of time for each integration

Dxx=[];XX=[];XU=[]; % Data matrices
```

```

X=[x0;kron(x0',x0')';kron(x0,zeros(un,1))];

P=eye(xn)*10; % Initialize the previous cost matrix
it=0;          % Counter for iterations
p_save=[];     % Track the cost matrices in all the iterations
k_save=[];     % Track the feedback gain matrix in each iterations

[P0,K0]=care(A,B,Q); % Calculate the ideal solution for comparion purpose
k_save=[norm(K-K0)];

while norm(P-P_old)>10^-3 % Stopping criterion for learning
    for i=1:20
        % Simulation the system and at the same time collect online info.
        [t,X]=ode45(@mysys, [it+(i-1)*T,it+i*T],X(end,:));

        %Append new data to the data matrices
        Dxx=[Dxx;kron(X(end,1:xn),X(end,1:xn))-kron(X(1,1:xn),X(1,1:xn))];
        XX=[XX;X(end,xn+1:xn+xn^2)-X(1,xn+1:xn+xn^2)];
        XU=[XU;X(end,xn+xn^2+1:end)-X(1,xn+xn^2+1:end)];

        % Keep track of the system trajectories
        x_save=[x_save;X];
        t_save=[t_save;t];
    end

    P_old=P; % Update the previous cost matrix

    QK=Q+K'*R*K; % Update the Qk matrix
    X2=XX*kron(eye(xn),K'); %
    X1=[Dxx,-X2-XU]; % Left-hand side of the key equation
    Y=-XX*QK(:); % Right-hand side of the key equation

```

```

pp = pinv(X1)*Y;

P = reshape(pp(1:xn*xn), [xn, xn]);
P = (P +P')/2;

figure(4)

scatter(it,P_old(1,1),'o','filled','b')
hold on
scatter(it,P_old(2,3),'+','g','Linewidth',2)
hold on
scatter(it,P_old(2,4),'o','r','Linewidth',2)
hold on
scatter(it,P_old(3,3),'s','c','Linewidth',2)
hold on

p_save=[p_save,norm(P-P0)];      % Keep track of the cost matrix

K=inv(R)*transpose(B)*P;          % Get the improved gain matrix
k_save=[k_save,norm(K-K0)];      % Keep track of the control gains

it=it+1      % Update and display the # of iters
end
fprintf('\nNumber of iterations to obtain optimal control solution %d\n',it)

% Plot the trajectories

figure(4)

scatter(it,P0(1,1),'*','b')
hold on
scatter(it,P0(2,3),'*','g')
hold on
scatter(it,P0(2,4),'*','r')

```

```

hold on
scatter(it,P0(3,3),'*','c')
hold on
axis([0 it 0 3.5])
xlabel('Time (s)')
title('P Matrix parameters')
legend('P(1,1)', 'P(2,3)', 'P(2,4)', 'P(4,4)', 'P(1,1)-Optimal', 'P(2,3)-Optimal', 'P(2,4)-Optimal', 'P(4,4)-Optimal')

figure(1)
plot([0:length(p_save)-1],p_save,'o',[0:length(p_save)-1],p_save)
legend('||P_i-P^*||')
xlabel('Number of iterations')

figure(2)
plot([0:length(k_save)-1],k_save,'^',[0:length(k_save)-1],k_save)
legend('||K_i-K^*||')
xlabel('Number of iterations')

% Post-learning simulation
[tt,xx]=ode23(@mysys,[t(end) 20],X(end,:));

% Keep track of the post-learning trajectories
t_final=[t_save;tt];
x_final=[x_save;xx];

figure(3)

plot(t_final,x_final(:,1:4),'Linewidth',2)
axis([0,2,-0.2,0.2]) %axis can be changed to get the complete trejectory
legend('x_1', 'x_2', 'x_3', 'x_4')

```

```

xlabel('Time (s)')
title('System States')

% The following nested function gives the dynamics of the system. Also,
function dX=mysys(t,X)
    global A B K xn
    x=X(1:xn);

    u=-K*x;
    dx=A*x+B*u;
    dxx=kron(x',x')';
    dux=kron(x',u')';
    dX=[dx;dxx;dux];
end

```

## PART IV – CONCLUSION

In this report, we have reviewed in detail a paper that proposed a new policy iteration technique to solve online the continuous time LQR problem without using knowledge about the system's internal dynamics (system matrix  $A$ ). The algorithm is an online adaptive optimal controller based on an adaptive critic scheme in which the actor performs continuous time control while the critic incrementally corrects the actor's behavior at discrete moments in time until best performance is obtained. The critic evaluates the actor performance over a period of time and formulates it in a parameterized form. Based on the critic's evaluation the actor behavior policy is updated for improved control performance.

The result can be summarized as an algorithm which effectively provides solution to the algebraic Riccati equation associated with the optimal control problem without using knowledge of the system matrix  $A$ . The convergence and the development of the algorithm is discussed in detail in section II.

The results published in the paper have been verified through a simulation by implementing the proposed algorithm in MATLAB. It is shown that both the problems result in similar solution as published in the paper and through simulation. The MATLAB code used for simulation has been provided for reference. The problem considered in this paper is taken from another publication but I couldn't get access to it as it was paid and hence I am unable to attach it in this report. Nevertheless, there was no shortage of information to simulate the result.

## PART V - BIBLIOGRAPHY

1. Chun-Hua Guo, P. L. (1998). Analysis and modification of Newton's method for Algebraic Ricatti Equation. *Mathematics of Computation*, 1089-1105.
2. D. Vrabie, O. P.-K. (2008). Adaptive optimal control for continuous-time linear systems based on policy iterations. *Elsevier*, 477-484.
3. Kirsten Morris, C. N. (2004). Iterative Solution of Algebraic Riccati Equations using a Modified Newton-Kleinman Method. *MTNS*.
4. D. Vrabie, F.L.L (2008). Adaptive Optimal Control Algorithm for Continuous-Time Non Linear Systems Based on Policy Iterations. IEEE

## PART VI – PAPERS

### 6.1 MAIN PAPER



Adaptive optimal  
control.pdf

### 6.2 REFERENCE PAPER (1- BIBLIOGRAPHY)



ANALYSIS AND  
MODIFICATON OF N

### 6.3 REFERENCE PAPER (3- BIBLIOGRAPHY)



Modified  
Newton-Kleinman M

### 6.4 REFERENCE PAPER (4-BIBLIOGRAPHY)



Adaptive Optimal  
Control Algorithm.p