Neural, Parallel & Scientific Computations 15 (2007) 125-135

NEURO APPROACH FOR SOLVING MATRIX RICCATI DIFFERENTIAL EQUATION

P. Balasubramaniam[§], J. Abdul Samath[†], N. Kumaresan[§]
and
A. Vincent Antony Kumar[§1]

[§]Department of Mathematics
Gandhigram Rural University
Gandhigram - 624 302, Tamilnadu, India
Email: pbalgri@rediffmail.com

†Department of Computer Science and Applications
Gandhigram Rural University
Gandhigram - 624 302, Tamilnadu, India
Email: abdul_samath@yahoo.com

Abstract: In this paper, neuro approach is proposed for solving Matrix Riccati Differential Equation(MRDE). Experiments show that the proposed Levenberg-Marquardt training method can be efficient for obtaining solutions in shorter computation time. The goal is to provide optimal control with reduced calculus effort by comparing the solutions of the MRDE in the well known RK-method and RK-Butcher method. A numerical example is presented to illustrate the implementation of artificial neural network (ANN) for the proposed method is given.

Key words: Matrix Riccati Differential Equation, Levenberg-Marquardt algorithm, RK-method, RK-Butcher method, Neural networks.

AMS (MOS) Subject Classifications: 49N05, 92B20, 93B52

1. Introduction

Neural networks or simply neural nets are computing systems, which can be trained to learn a complex relationship between two or many variables or data sets. Having the structures similar to their biological counterparts, neural networks are representational and computational models processing information in a parallel distributed fashion composed of interconnecting simple processing nodes [17]. Neural net techniques have been successfully applied in various fields such as function approximation, signal processing and adaptive (or) learning control for nonlinear systems. Using neural networks a variety of off-line learning control algorithms have been developed for nonlinear systems [9, 11]. A variety of numerical algorithms have been discussed for solving the algebraic Riccati equation[3]. Neural networks problem have attracted considerable attentions in recent years of many researchers for numerical aspects for algebraic Riccati equations (see [4, 6, 18, 19]).

Solving the Matrix Riccati Differential Equation(MRDE) is a central issue in optimal control theory. The needs for solving such equations often arise in analysis and synthesis such as linear-quadratic optimal control systems, robust control systems with H_2 and H_∞-control [20] performance criteria, stochastic filtering and control systems, model reduction, differential games etc., and many others. In particular, the solution of algebraic Riccati matrix equations(ARME) in continuous-time (or) discrete time is an important task for nonlinear matrix equations arising in mathematics and engineering. This equation, in one form or another, has an important role in optimal control problems, multivariable and large scale systems, scattering theory estimation, detection, transportation and radiative transfer [7]. The solution of this equation is difficult to obtain from two points of view. One is nonlinear and

[1]The author also working in the Department of Mathematics, PSNA College of Engineering and Technogly, Dindigul-624622. Tamilnadu, INDIA, Email: vincypsna@rediffmail.com

Received December 16, 2005

the other is matrix form. Most general methods to solve MRDE with a terminal boundary condition are obtained on transforming MRDE into an equivalent linear differential Hamiltonian system [8]. By using this approach the solution of MRDE is obtained by partitioning the transition matrix of the associated Hamiltonian system (see [13, 16]). Another class of method is based on transforming MRDE into a linear matrix differential equation and then solving MRDE analytically (or) computationally (see [10, 14, 15]). In [10], an analytic procedure of solving the MRDE of the linear quadratic control problem for homing missile systems is presented. The solution $K(t)$ of the MRDE is obtained by using $K(t) = p(t)/f(t)$, where $f(t)$ and $p(t)$ are solutions of certain first order ordinary linear differential equations. However the given technique is restricted to single input.

Although parallel algorithms can compute the solutions faster than sequential algorithms, there is no report on neural network solutions for MRDE. Recently Balasubramaniam et al [1, 2] focused upon to find the optimal control for singular systems using neural networks. This paper focuses upon the implementation of neurocomputing approach for solving MRDE so for the optimal solution. The structured neural network architecture is trained using Levenberg-Marquardt algorithm [5] to prove the efficiency for solutions in shorter computation time. An example is given which illustrates the advantage of the fast and accurate solutions compared with RK-method and RK-Butcher method solutions.

This paper is organized as follows. In section 2 the statement of the problem is given. In section 3, solution of the MRDE is presented. In section 4, a numerical example is discussed. The final conclusion section demonstrates the efficiency of the method.

2. Statement of the Problem

Consider a completely state controllable linear system as follows:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0 \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control vector, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are known coefficient matrices associated with $x(t)$ and $u(t)$ respectively, x_0 is a given initial state vector and $m \leq n$. If all state variables are measurable, then a linear state feedback control law

$$u(t) = -R^{-1}B^T\lambda(t) \tag{2}$$

can be obtained to the system described by equation (1) where

$$\lambda(t) = K(t)x(t), \tag{3}$$

$K(t) \in \mathbb{R}^{n\times n}$ matrix such that $K(t_f) = S$.

In order to minimize both the state and control signals of the feedback control system a quadratic performance index is usually minimized :

$$min\ J = \frac{1}{2}x^T(t_f)Sx(t_f) + \frac{1}{2}\int_0^{t_f}[x^T(t)Qx(t) + u^T(t)Ru(t)]dt \tag{4}$$

where the superscript T denotes the transpose operator, $Q \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite (or semidefinite) weighting matrix for $x(t)$, $R \in \mathbb{R}^{m\times m}$ is a symmetric and positive definite weighting matrix for $u(t)$. It is well known in the control literature that to minimize J is equivalent to minimize the Hamiltonian equation

$$H(x,u,\lambda,t) = \frac{1}{2}x^T(t)Qx(t) + \frac{1}{2}u^TRu(t) + \lambda^T(t)[Ax(t) + Bu(t)]$$

along the optimal trajectory.

Using the calculus of variations, we obtain

$$\frac{\partial H}{\partial u}(x, u, \lambda, t) = 0$$

implies that

$$Ru(t) + B^T\lambda(t) = 0$$

and

$$\frac{\partial H}{\partial x} = \dot{\lambda}(t)$$

$$\Rightarrow \dot{\lambda}(t) = -Qx(t) - A^T\lambda(t) \tag{5}$$

$$\frac{\partial H}{\partial \lambda} = \dot{x}(t)$$

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

and from (2), we have

$$\dot{x}(t) = Ax(t) - BR^{-1}B^T\lambda(t). \tag{6}$$

Equations (5) and (6) can be written in a matrix form as follows:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\lambda}(t) \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix}$$

where $x(0) = x_0$. Assuming $|R| \neq 0$, from (3) we have

$$\dot{\lambda}(t) = \dot{K}(t)x(t) + K(t)\dot{x}(t). \tag{7}$$

Using the equations (5) and (6) in (7), we obtain

$$[\dot{K}(t) + K(t)A + A^TK(t) + Q - K(t)BR^{-1}B^TK(t)]x(t) = 0. \tag{8}$$

Since equation (8) holds for all non-zero x(t), then the term pre-multiplying x(t) must be zero. Therefore, we obtain the following (MRDE) for the linear system (1)

$$\dot{K}(t) + K(t)A + A^TK(t) + Q - K(t)BR^{-1}B^TK(t) = 0. \tag{9}$$

In the following section the MRDE (9) is going to be solved for $K(t)$ so for the optimal solution.

3. Solution of MRDE

In the process of solving the equation (9), the MRDE is transformed into a system of nonlinear differential equations.

Hence the equation (9) can be written as

$$\dot{k}_{ij} = f(t, k_{ij}), \quad (i, j = 1, 2, 3, ..., n) \tag{10}$$

In the following subsections the above system will be solved by RK-method, RK-Butcher method and Neural networks approaches respectively.

3.1. **Runge-Kutta Method.** RK-algorithms have always been considered as the best tool for the numerical integration of ordinary differential equations($ODEs$). Since the system (10) contains n^2 first order ODEs with n^2 variables, RK method is explained for a system of three first order ODEs with three variables.

$$\begin{aligned} k_{11}(i+1) &= k_{11}(i) + \frac{1}{6}(k1 + 2k2 + 2k3 + k4) \\ k_{12}(i+1) &= k_{12}(i) + \frac{1}{6}(l1 + 2l2 + 2l3 + l4) \\ k_{22}(i+1) &= k_{22}(i) + \frac{1}{6}(m1 + 2m2 + 2m3 + m4) \end{aligned}$$

where

$$\begin{aligned} k1 &= h * \Phi_{11}(k_{11}, k_{12}, k_{22}) \\ l1 &= h * \Phi_{12}(k_{11}, k_{12}, k_{22}) \\ m1 &= h * \Phi_{13}(k_{11}, k_{12}, k_{22}) \\ k2 &= h * \Phi_{11}(k_{11} + \frac{k1}{2}, k12 + \frac{l1}{2}, k22 + \frac{m1}{2}) \\ l2 &= h * \Phi_{12}(k_{11} + \frac{k1}{2}, k12 + \frac{l1}{2}, k22 + \frac{m1}{2}) \\ m2 &= h * \Phi_{13}(k_{11} + \frac{k1}{2}, k12 + \frac{l1}{2}, k22 + \frac{m1}{2}) \\ k3 &= h * \Phi_{11}(k_{11} + \frac{k2}{2}, k12 + \frac{l2}{2}, k22 + \frac{m2}{2}) \\ l3 &= h * \Phi_{12}(k_{11} + \frac{k2}{2}, k12 + \frac{l2}{2}, k22 + \frac{m2}{2}) \\ m3 &= h * \Phi_{13}(k_{11} + \frac{k2}{2}, k12 + \frac{l2}{2}, k22 + \frac{m2}{2}) \\ k4 &= h * \Phi_{11}(k_{11} + k3, k12 + l3, k22 + m3) \\ l4 &= h * \Phi_{12}(k_{11} + k3, k12 + l3, k22 + m3) \\ m4 &= h * \Phi_{13}(k_{11} + k3, k12 + l3, k22 + m3). \end{aligned}$$

3.2. **Runge-Kutta Butcher Method.** RK-Butcher algorithm is nominally considered as a sixth order since it requires six functions evaluation. The accuracy of this algorithm is better than other algorithms. Even though it seems to be sixth order method, it is a fifth order method only.
RK-Butcher method is explained for a system of three first order ODEs with three variables.

$$\begin{aligned} k_{11}(i+1) &= k_{11}(i) + \frac{h}{90}(7k1 + 32k3 + 12k4 + 32k5 + 7k6) \\ k_{12}(i+1) &= k_{12}(i) + \frac{h}{90}(7l1 + 32l3 + 12l4 + 32l5 + 7l6) \\ k_{22}(i+1) &= k_{22}(i) + \frac{h}{90}(7m1 + 32m3 + 12m4 + 32m5 + 7m6) \end{aligned}$$

where

$$
\begin{aligned}
k1 &= \Phi_{11}\Big(k_{11}, k_{12}, k_{22}\Big) \\
l1 &= \Phi_{12}\Big(k_{11}, k_{12}, k_{22}\Big) \\
m1 &= \Phi_{13}\Big(k_{11}, k_{12}, k_{22}\Big) \\
k2 &= \Phi_{11}\Big(k_{11}, k_{12}, k_{22}\Big) + h * \frac{l1}{4} + h * \frac{m1}{4} \\
l2 &= \Phi_{12}\Big(k_{11} + h * \frac{k1}{4}, k_{12} + h * \frac{l1}{4}, k_{22} + h * \frac{m1}{4}\Big) \\
m2 &= \Phi_{13}\Big(k_{11} + h * \frac{k1}{4}, k_{l2} + h * \frac{l1}{4}, k_{22} + h * \frac{m1}{4}\Big) \\
k3 &= \Phi_{11}\Big(k_{11}, k_{12}, k_{22}\Big) + \frac{h * l1 + h * l2}{8} + \frac{h * m1 + h * m2}{8} \\
l3 &= \Phi_{12}\Big(k_{11} + \frac{h * k1 + h * k2}{8}, k_{12} + \frac{h * l1 + h * l2}{8}, k_{22} + \frac{h * m1 + h * m2}{8}\Big) \\
m3 &= \Phi_{13}\Big(k_{11} + \frac{h * k1 + h * k2}{8}, k_{12} + \frac{h * l1 + h * l2}{8}, k_{22} + \frac{h * m1 + h * m2}{8}\Big) \\
k4 &= \Phi_{11}\Big(k_{11}, k_{12}, k_{22}\Big) - \frac{h * l2}{2} + (h * l3) - \frac{h * m2}{2} + (h * m3) \\
l4 &= \Phi_{12}\Big(k_{11} - \frac{h * k2}{2} + h * k3, k_{12} - \frac{h * k2}{2} + h * l3, k_{22} - \frac{h * m2}{2} + h * m3\Big) \\
m4 &= \Phi_{13}\Big(k_{11} - \frac{h * k2}{2} + h * k3, k_{12} - \frac{h * k2}{2} + h * l3, k_{22} - \frac{h * m2}{2} + h * m3\Big) \\
k5 &= \Phi_{11}\Big(k_{11}, k_{12}, k_{22}\Big) + \frac{(3 * h * l1) + (9 * h * l4)}{16} + \frac{(3 * h * m1) + (9 * h * m4)}{16} \\
l5 &= \Phi_{12}\Big(k_{11} + \frac{(3 * h * k1) + (9 * h * k4)}{16}, k_{12} + \frac{(3 * h * l1) + (9 * h * l4)}{16}, \\
&\qquad k_{22} + \frac{(3 * h * m1) + (9 * h * m4)}{16}\Big) \\
m5 &= \Phi_{13}\Big(k_{11} + \frac{(3 * h * k1) + (9 * h * k4)}{16}, k_{12} + \frac{(3 * h * l1) + (9 * h * l4)}{16}, \\
&\qquad k_{22} + \frac{(3 * h * m1) + (9 * h * m4)}{16}\Big)
\end{aligned}
$$

$$
\begin{aligned}
k6 &= \Phi_{11}\Big(k_{11}, k_{12}, k_{22}\Big) \\
&\quad -\Big(\frac{3 * h * l1 - 2 * h * l2 - 12 * h * l3 + 12 * h * l4 - 8 * h * l5}{7}\Big) \\
&\quad -\Big(\frac{3 * h * m1 - 2 * h * m2 - l2 * h * m3 + 12 * h * m4 - 8 * h * m5}{7}\Big)
\end{aligned}
$$

$$
\begin{aligned}
l6 &= \Phi_{12}\Big(k_{11} - \frac{3*h*k1 - 2*h*k2 - 12*h*k3 + 12*h*k4 - 8*h*k5}{7}, \\
&\quad k_{12} - \Big(\frac{3*h*l1 - 2*h*l2 - 12*h*l3 + 12*h*l4 - 8*h*l5}{7}\Big), \\
&\quad k_{22} - \Big(\frac{3*h*m1 - 2*h*m2 - l2*h*m3 + 12*h*m4 - 8*h*m5}{7}\Big)\Big) \\
m6 &= \Phi_{13}\Big(k_{11} - \Big(\frac{3*h*k1 - 2*h*k2 - 12*h*k3 + 12*h*k4 - 8*h*k5}{7}\Big), \\
&\quad k_{12} - \Big(\frac{3*h*l1 - 2*h*l2 - 12*h*l3 + 12*h*l4 - 8*h*l5}{7}\Big), \\
&\quad k_{22} - \Big(\frac{3*h*m1 - 2*h*m2 - l2*h*m3 + 12*h*m4 - 8*h*m5}{7}\Big)\Big).
\end{aligned}
$$

3.3. **Neural Network Solution.** In this approach, a new feedforward neural network is used to solve the system of differential equations (10). A neural network architecture is shown in figure 1.

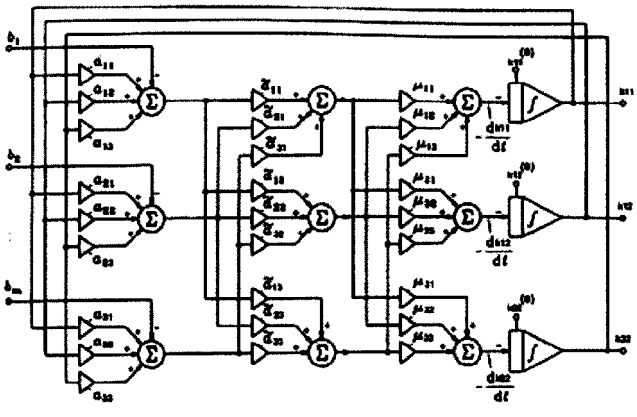

FIGURE 1. Neural Network Architecture

The architecture consists of 3 layers of artificial neurons which are connected in a feed forward node. The network consists of integrators and adders with associated connecting weights. The connection weights denoted by $a_{ij}(=\tilde{a}_{ij})$ and μ_{ij} are the coefficients of the third order matrices $\mathcal{A}$ and μ.

Each neuron produces its output by computing the inner product of its input k_{ij}, $i,j = 1,2,3$ and its appropriate weight vector and possibly by passing the result through the nonlinear tansigmoid functions. The learning supervised rule that modifies the weights of the connections according to the input patterns that are presented in the Levenberg-Marquardt algorithm [12]. During the training the weights and biases of the network are iteratively adjusted till the following error quantity becomes zero.

$$E = \sum_{i,j=1}^{3} \Big((\dot{k}_{ij}) - \phi_{ij}(t,(k_{ij}))\Big)^2 = 0$$

t	k_{11}	k_{12}	k_{22}
0.2	-0.242057	-0.237297	-0.532972
0.4	-0.572630	-0.509861	-1.336832
0.6	-1.007613	-0.698837	-2.267480
0.8	-1.624486	-0.684054	-3.076698
1.0	-2.583961	-0.428650	-3.656955
1.2	-4.083993	0.059739	-4.070754
1.4	-6.293067	0.780564	-4.419200
1.6	-9.259361	1.724925	-4.767664
1.8	-12.808259	2.833062	-5.133368
2.0	-16.530706	3.981502	-5.495683

TABLE 1. Solution by RK-Method

This algorithm is fastest method for training moderate-sized feedforward neural networks. The learning rule that was implemented with a PC, CPU 1.7 GHz which has unit roundoff $\simeq 1.1 \times 10^{-4}$ in MATLAB for the neuro computing approach to solve MRDE (9) for the linear system (1).

3.3.1 Neural network Algorithm

Step 1: Feed the random input vector $k_{ij}, \quad i,j = 1,2,3.$
Step 2: Initialize randomized weight matrix and bias in each layer.
Step 3: Computing the inner product in each neuron and pass through tansigmoidal function.
Step 4: Pass the output of the third layer through integrator with initial values.
Step 5: Repeat the neural network training until the following error function becomes zero.

$$E = \sum_{i,j=1}^{n} \left(\dot{k}_{ij} - \phi_{ij}(t, k_{ij})\right)^2 = 0$$

4. Numerical Example

Consider the MRDE (9) in which

$$A = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, R = 1, Q = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

The numerical solution of MRDE were calculated and displayed respectively in the following tables 1, 2 and 3 using the RK-method, RK-Butcher methd and the Neural Networks approach. The solution of MRDE and the deviation of RK solution is displayed in Fig. 2,3 and 4. The computation time for neural network solution is 1.6330sec. whereas the RK method is 2.1930sec. Hence the neural solution is faster than RK method.

t	k_{11}	k_{12}	k_{22}
0.2	-0.258775	-0.238316	-0.487425
0.4	-0.627679	-0.517358	-1.220125
0.6	-1.112443	-0.721680	-2.112111
0.8	-1.750488	-0.716365	-2.953969
1.0	-2.663129	-0.453504	-3.596852
1.2	-4.023458	0.041253	-4.057855
1.4	-5.982123	0.740360	-4.427505
1.6	-8.604144	1.622707	-4.774734
1.8	-11.807094	2.643066	-5.124072
2.0	-15.331792	3.716645	-5.466076

TABLE 2. Solution by RK-Butcher Method

t	k_{11}	k_{12}	k_{22}
0.2	0.2927	-0.2818	-0.4961
0.4	-0.6045	-0.5491	-1.2291
0.6	-1.0612	-0.6632	-2.1458
0.8	-1.7266	-0.6554	-2.9159
1.0	-2.6888	-0.4907	-3.5424
1.2	-4.0628	0.0525	-4.0501
1.4	–5.9843	0.8098	-4.4640
1.6	-8.5722	1.4832	-4.8107
1.8	-11.8157	2.6431	-5.1257
2.0	-15.3191	3.7166	-5.4655

TABLE 3. Solution by Neural Networks

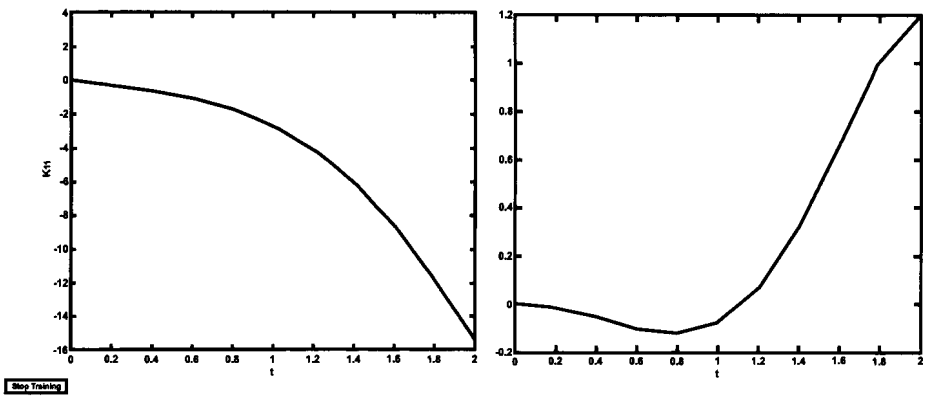

FIGURE 2. Solution curve and error curve for k_{11}

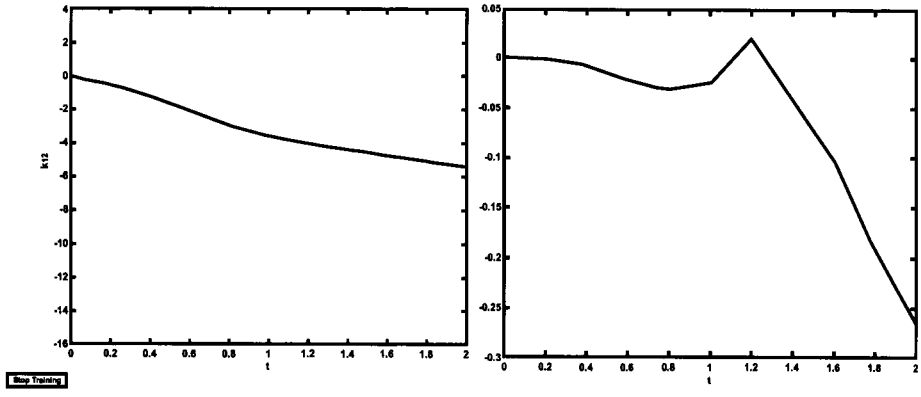

FIGURE 3. Solution curve and error curve for k_{12}

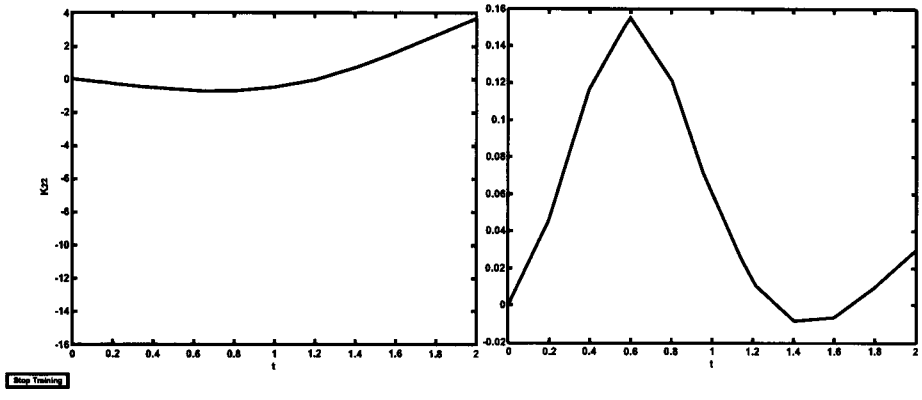

FIGURE 4. Solution curve and error curve for k_{22}

4.1. Remarks

Solving MRDE by neural network is a naval approach. Computation time is very minimum when compared with existing methods stated in the survey paper [3]. High accuracy of numerical solutions compared with traditional methods like Rung-Kutta, RK-Butcher method have been analysed. Once network is trained, it allows instantaneous evaluation of solutions at any desired number of points with less memory occupation. The neural network approach for solving MRDE is different from the previous methods discussed in the survey paper [3] in various directions stated sequentially as follows:

(i) In direct integration method, there is a difficulty to apply for higher order Riccati Differential Equation

(ii) In negative exponential method, the computation of transformation matrix is numerically unreliable in a fixed precession of arithmetic.

(iii) In Davison Maki metho, Pad approximation is used to compute the solution. Accuracy of Pade's solution is not better than this neural network approach

(iv) In ASP(Automatic Synthesis program) matrix iteration procedure for computation of matrix exponential of size $2n \times 2n$ is time consuming.

(v) In a method using an algebric Riccati solution the computation of matrix exponential of size $n \times n$ is also time consuming.

(vi) In square root method accuracy of the solution is not satisfactory.

(vii) In analytic approximation method, the accuracy of solution depends upon accuracy of solution of Algebraic Riccati equation.

(viii) A matrix valued approach the accuracy of solution depends upon the solution of Sylvester equation.

5. Conclusion

The numerical results of the MRDE (9) in the above tables indicate that the neural networks solutions are much more efficient, compared with other well-known RK-method solutions. The long calculus time of the MRDE (9) was avoided by using a neuro-optimal controller based Lenvenberg-Marquardt training algorithm. The efficient approximations of the optimal solution were done with PC, CPU 1.7 GHz in MATLAB which has unit roundoff $\simeq 1.1 \times 10^{-4}$.

Acknowledgements

The authors would like to thank the referee for their valuable comments and suggestions to improve the quality of the paper in the presented form. The work of the first author was supported by the DST, Govt. of India, New Delhi under the grant No. SR/FTP/MA-05/2002.

References

[1] P. Balasubramaniam, J. Abdul Samath, N. Kumaresan, A. Vincent Antony Kumar, Solution of matrix Riccati differential equation for the linear quadratic singular system using neural networks, *Appl. Math. Comput.*, **182** (2006), 1832–1839.

[2] P. Balasubramaniam, J. Abdul Samath, N. Kumaresan, Optimal control for nonlinear singular systems with quadratic performance using neural networks, *Appl. Math. Comput.*, (2006) (in press).

[3] C. H. Choi, A survey of numerical methods for solving matrix Riccati differential equation, *Proceedings of Southeastcon*, (1990), 696–700.

[4] S. W. Ellacott, Aspects of the numerical analysis of neural networks, *Acta Numer.*, **5** (1994), 145–202.

[5] M. Gaiceanu, E. Rosu, A. M. Tataru, Neuro-optimal controller for three-phase induction motor based on Levenberg-Marquardt training algorithm *Proceedings PowerCon. IEEE Conference on Power System Technology,* **1** (2000), 97–102.

[6] F. M. Ham, E. G. Collins, A neurocomputing approach for solving the algebraic matrix Riccati equation, *Proceedings IEEE International Conference on Neural Networks*, **1** (1996), 617 - 622

[7] M. Jamshidi, An overview on the solutions of the algebraic matrix Riccati equation and related problems, *Large scale Systems*, **1** (1980), 167–192.

[8] L. Jodar, E. Navarro, Closed analytical solution of Riccati type matrix differential equations, *Indian J. Pure and Appl. Math.*, **23** (1992), 185–187.

[9] A. Karakasoglu, S. L. Sudharsanan, M. K. Sundareshan, Identification and decentralized adaptive control using neural networks with application to robotic manipulators, *IEEE Transactions on Neural Networks*, **4** (1993), 919–930.

[10] N. Lovren, M.Tomic, Analytic solution of the Riccati equation for the homing missile linear quadratic control problem, *J. Guidance. Cont. Dynamics* , **17** (1994), 619–621.

[11] K. S. Narendra, K. Parathasarathy, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, **1** (1990), 4–27.

[12] M. Norgaard, Neural Network Based System Indentification Toolbox, *Technical Report*, 97-E-851, Department of Automation, Technical University of Denmark, 1997.
[13] M. Razzaghi, A Schur method for the solution of the matrix Riccati equation, *Int. J. Math. Math. Sci.*, **20** (1997), 335–338.
[14] M. Razzaghi, Solution of the matrix Riccati equation in optimal control, *Information Sci.*, **16** (1978), 61–73.
[15] M. Razzaghi, A computational solution for a matrix Riccati differential equation, *Numerische Math.*, **32** (1979), 271–279.
[16] D.R. Vaughu, A negative exponential solution for the matrix Riccati equation, *IEEE Trans Automat. Control*, **14**(1969), 72–75.
[17] P. De. Wilde, *Neural Network Models*, Second ed., Springer-Verlag, London(1997).
[18] J. Wang, G. Wu, A multilayer recurrent neural network for solving continuous time algebraic Riccati equations, *Neural Networks*, **11** (1998), 939–950.
[19] B. Zhang, D. Kannan, A numerical analysis of stochastic neural networks, *Neural Parallel Sci. Comput.*, **8**(3-4)(2000), 209–241.
[20] K. Zhou, Khargonekar, An algebraic Riccati equation approach to H_∞ optimization, *Systems and Control Letters*, **11** (1998), 85–91.