CONTINUOUS LYAPUNOV DYNAMICAL SYSTEMS – ARTIFICIAL NEURAL NETWORK APPROACH

Venkata Sundaranand Putcha¹ §, G.V. Deepika

¹,²Center for Mathematical Sciences - DST
C.R. Rao Advanced
Institute of Mathematics, Statistics and Computer Science
(AIMSCS), UOH Campus, Prof. C.R. Rao Road, Gachibowli
Hyderabad, 500 046 (A.P), INDIA

Abstract: In this paper, an artificial Neural Network is constructed to solve the initial value problem associated with continuous Lyapunov dynamical system after developing variation of parameters formula for the inhomogeneous system. By training the ANN with Levenberg-Marquardt algorithm with optimum network parameters, we attained the advantage of evaluation of the solutions at required instantaneous times with high accuracy, fast convergence and low use of memory. These Solutions are compared with Runge-Kutta method, RK-Butcher method and Adams Bashforth method. Numerical examples are presented for the developed methodology.

AMS Subject Classification: 34B05, 34B15, 65L05, 34L30, 92B20
Key Words: Lyapunov systems, continuous dynamical system, feed forward artificial neural networks, Levenberg-Marquardt algorithm

1. Introduction

Many engineering systems, such as mechanical systems, electrical circuits and chemical reaction kinetics are modeled by coupled differential equations that can’t be transformed into ordinary differential equations. Matrix Lyapunov

Received: June 19, 2013

§Correspondence author

© 2013 Academic Publications, Ltd.
url: www.acadpubl.eu
systems arise in stability of the linear as well as non-linear systems and also in the quadratic optimal control problems. Hence solving matrix Lyapunov equations is very important in system theory and control engineering. Many matrix differential equations which arise in various practical problems are non-linear in nature and finding exact solutions is not possible. The origin of Matrix Lyapunov differential equations goes back to the general problem of the stability of motion discussed in the doctoral thesis of Alexander Mikhailovitch Lyapunov [15]. In 1991, by expressing the closed form solution of the homogeneous matrix Sylvester differential equation

\[
\frac{dX(t)}{dt} = A(t)X(t) + X(t)B(t),
\]

where \(A(t)\), \(B(t)\) and \(X(t)\) are \(n^{th}\) order square matrices whose elements are real functions defined on \(R\), in terms of two fundamental matrix solutions of the systems

\[
\frac{dX(t)}{dt} = A(t)X(t) \quad \text{and} \quad \frac{dX(t)}{dt} = B^*(t)X(t).
\]

Murthy et al [16] developed variation parameters formula for the inhomogeneous matrix Sylvester differential equation

\[
\frac{dX(t)}{dt} = A(t)X(t) + X(t)B(t) + F(t),
\]

where \(F(t)\) is a square matrix of order \(n\) whose elements are real functions defined on \(R\) and also for the non-linear matrix Sylvester differential equation

\[
\frac{dX(t)}{dt} = A(t)X(t) + X(t)B(t) + F(t, X(t)),
\]

where \(F : I \times R^{n \times n} \rightarrow R^{n \times n}\), is continuous, \(I\) is any any sub interval of \(R\).

In 1992, Murthy, K N, Howell, G W and Sivasundaram, S [17] solved the two and multi point boundary value problems associated with the linear as well as non-linear matrix Sylvester differential equations by using fixed point theorems. In all these works the latent condition is the existence of fundamental matrix solutions for the systems \(\frac{dX(t)}{dt} = A(t)X(t)\) and \(\frac{dX(t)}{dt} = B^*(t)X(t)\).

But while modeling many real world problems we can not impose the condition that the choice of \(A(t)\) and \(B(t)\) is such that the resulting integrands presented in [16] and [17] are integrable and there by finding exact solutions is not possible. So we have to go for approximate solutions.

The main aim behind solving a matrix Lyapunov differential equation by using Artificial neural Network is to find the closest possible solution.
An artificial neural network (ANN) system is an abstract mathematical model or computational model inspired by brain structures, mechanisms, and functions[13].

Architectures are still being developed and the best way to use ANNs for a wide field of problems is not well studied. Activation functions which minimize the error are the vital tools for ANNs method of solving differential equations. In recent years, the study of dynamical behavior of recurrently connected neural networks has attracted attention of researchers for the wide range of applications [4, 5, 6, 3]. In 1990, Lee and Kang [14] used Hopfield-type nets to solve differential equations. Isaac Lagaris used ANN to solve non-homogeneous ordinary and partial differential equations [7] and eigenvalue problems [8].

Hornik et al. [12] investigated function approximation by feed forward neural networks. In 1999, S Haykin [10] observed that a feed forward neural network with a single hidden layer is capable of approximating a function and its derivatives to an arbitrary level of accuracy. Feed forward ANN is an alternative method for approximating solutions to differential equations. Here we use back propagation algorithm which provides the iterative approach for minimizing the error of this approximation. The problem of neural network learning can be seen as a function optimization problem, where we are trying to determine the best network parameters (weights and biases) in order to minimize network error. Levenberg-Marquardt algorithm gives better results when compared with optimization techniques [11] from numerical linear algebra. In this paper we consider the homogeneous matrix Lyapunov differential equation

\[ \frac{dX(t)}{dt} = A(t)X(t) + X(t)A^*(t), \] (4)

where \(A(t)\) and \(X(t)\) are \(n^{th}\) order square matrices whose elements are real functions defined on \(R\) and construct the solution of the initial value problems associated with the inhomogeneous matrix Lyapunov differential equation

\[ \frac{dX(t)}{dt} = A(t)X(t) + X(t)A^*(t) + F(t), \] (5)

where \(F(t)\) is a square matrix of order \(n\) whose elements are real functions defined on \(R\) and non-linear matrix Lyapunov differential equation

\[ \frac{dX(t)}{dt} = A(t)X(t) + X(t)A^*(t) + F(t, X(t)), \] (6)

where \(F : I \times R^{n \times n} \rightarrow R^{n \times n}\), is continuous, \(I\) is any any sub interval of \(R\) containing \(t_0\) satisfying the initial condition

\[ X(t_0) = X_0 \] (7)
This paper is organized as follows. In Section 2, we present the existence of solutions of the continuous Lyapunov dynamical system. Section 3, is concerned with the Levenberg Marquardt algorithm. Numerical examples demonstrating the ANN approach to solutions of continuous Lyapunov dynamical system are presented Section 4.

### 2. Continuous Lyapunov Dynamical System

**Theorem 1.** If $\phi(t, t_0)$ is the fundamental matrix solutions of $\frac{dX(t)}{dt} = A(t)X(t)$ then any solution $X(t)$ of the homogeneous Singular Matrix Sylvester System (4) is of the form $\phi(t, t_0)C\phi^*(t, t_0)$, where $C$ is an arbitrary constant square matrix and $*$ indicates the transpose of the matrix.

**Proof.** We seek a solution of (4) in the form $X(t) = \phi(t, t_0)K(t)$ where $K(t)$ is a square matrix of order $n$, whose elements are functions defined on $R$. Then

$$
\frac{d(\phi(t, t_0)K(t))}{dt} = A(t)(\phi(t, t_0)K(t)) + (\phi(t, t_0)K(t))A^*(t)
$$

$$
\iff \frac{dK(t)}{dt} = K(t)A(t)
$$

$$
\iff \frac{dK^*(t)}{dt} = A^*(t)K(t).
$$

Since $\phi(t, t_0)$ is a fundamental matrix solutions of

$$
\frac{dX(t)}{dt} = A(t)X(t),
$$

it follows that there exists a constant square matrix $C_1$ of order $n$ such that $K^*(t) = \phi(t, t_0)C_1$ and $X(t) = \phi(t, t_0)C\phi^*(t, t_0)$, (where $C = C_1^*$).

**Theorem 2.** Any solution $X(t)$ of the non-homogenous Singular Matrix Sylvester System (5) is of the form $\phi(t, t_0)C\phi^*(t, t_0) + \overline{X}(t)$ where $\overline{X}(t)$ is a particular solution of (5).

**Proof.** It can be easily be verified that $\phi(t, t_0)C\phi^*(t, t_0) + \overline{X}(t)$ is a solution of (5). Now to prove that every solution is of this form. Let $X(t)$ be any solution of (5) and $\overline{X}(t)$ be a particular solution of (5). Then it can be easily verified that $X(t) - \overline{X}(t)$ is a solution of (4). Hence from Theorem1 we have $X(t) = \phi(t, t_0)C\phi^*(t, t_0) + \overline{X}(t)$.  

\[\square\]
**Theorem 3.** Any particular solution of the non-homogeneous Singular Matrix Lyapunov System (5) is of the form \( X(t) = \int_{t_0}^{t} \phi(t, s)C\phi^*(t, s)ds \)

**Proof.** The general solution of the homogeneous Matrix Lyapunov System (4) is of the form \( X(t) = \phi(t, t_0)C_2\phi^*(t, t_0). \) Let \( C_2 \) be a function of \( t \) defined on \( R. \) Let us impose the condition that \( X(t) \) satisfies (5). Then it follows that

\[
\phi(t, t_0) \frac{dC_2(t)}{dt} \phi^*(t, t_0) = F(t).
\]

That is

\[
C_2(t) = \int_{t_0}^{t} \phi(t_0, s)F(s)\phi^*(t_0, s)ds.
\]

**Theorem 4.** Any solution \( X(t) \) of the initial value problem associated with the inhomogenous Matrix Lyapunov System (5) satisfying the initial condition \( X(t_0) = X_0 \) where \( X_0 \) is a given square matrix of order \( n \) is of the form

\[
X(t) = \phi(t, t_0)X_0\phi^*(t, t_0) + \int_{t_0}^{t} \phi(t, s)F(s)\phi^*(t, s)ds
\]

(8)

**Theorem 5.** Any solution \( X(t) \) of the initial value problem associated with the non-linear Matrix Lyapunov System (6) satisfying the initial condition \( X(t_0) = X_0 \) where \( X_0 \) is a given square matrix of order \( n \) is of the form

\[
X(t) = \phi(t, t_0)X_0\phi^*(t, t_0) + \int_{t_0}^{t} \phi(t, s)F(s, X(s))\phi^*(t, s)ds
\]

(9)

### 3. Method

Feedforward neural networks which are the most popular architectures due to their structural flexibility, good representational capabilities and availability of a large number of training algorithm (Haykin, 1999). Multi Layer Perceptron and Radial Basis Function networks are two kinds of feed-forward neural network with different transfer functions. In this approach we use radial basis feedforward neural network. Artificial neural network representing the system of differential equations is constructed.

The architecture consists of input layer, one hidden layer and output layer of artificial neurons connected in a feed forward node. Output of each neuron
is produced by computing the inner product of the input and its appropriate weight vector and thereby passing the result through the non-linear tan-sigmoid functions.

We feed the network with input data that propagating through layers from input layer to output layer. This state of the network is called relax state. In the train state of the network the adjustable parameters of the networks (weights and biases) are tuned by a proper learning algorithm to minimize the energy function of the network. The learning supervised rule that modifies the weights of the connections according to the input patterns that are presented in the levenberg Marquardt algorithm. During the training network Sum Squared Error is calculated for the initial inputs and weights are adjusted iteratively according to Sum Squared Error.

3.1. Levenberg Marquardt Algorithm

The Levenberg-Marquardt algorithm is a very simple, but robust, method for approximating a function. Basically, it consists in solving the equation:

\[(J^TJ + \lambda I)\delta = J^TE.\] (10)

Here \(J\) is the Jacobian matrix for the system, \(\lambda\) is the Levenberg’s damping factor, \(\delta\) is the weight update vector that we want to find and \(E\) is the error vector containing the output errors for each input vector used on training the network. The \(\delta\) tell us by how much we should change our network weights to achieve a (possibly) better solution. The \(J^TJ\) matrix can also be known as the approximated Hessian.

3.1.1. Computing the Jacobian

The Jacobian is a matrix of all first-order partial derivatives of a vector-valued function. In the neural network case, it is a \(N \times W\) matrix, where \(N\) is the number of entries in our training set and \(W\) is the total number of parameters (weights + biases) of our network. It can be created by taking the partial derivatives of each output in respect to each weight, and has the form

\[
\begin{bmatrix}
\frac{\partial F(x_1, W)}{\partial W_1} & \cdots & \frac{\partial F(x_1, W)}{\partial W_w} \\
\frac{\partial F(x_N, W)}{\partial W_1} & \cdots & \frac{\partial F(x_N, W)}{\partial W_w}
\end{bmatrix}.
\]

Here \(F(x_i, W)\) is the network function evaluated for the \(i^{th}\) input vector of the training set using the weight vector \(w\) and \(w_j\) is the \(j^{th}\) element of the weight.
vector $W$ of the network. In traditional Levenberg-Marquardt implementations, the Jacobian is approximated by using finite differences. However, for neural networks, it can be computed very efficiently by using the chain rule of calculus and the first derivatives of the activation functions. Levenberg-Marquardt consists in solving (1) with different values of $\lambda$ until the sum of squared error decreases. So, each learning iteration (epoch) will consist of the following basic steps:

1. Compute the Jacobian matrix (by using finite differences or the chain rule);
2. Compute the error gradient $g = JtE$;
3. Approximate the Hessian using the cross product Jacobian (eq. 3) $H = JtJ$;
4. Solve $(H + \lambda I)\delta = g$ to find $\delta$;
5. Update the network weights $w$ using $\delta$;
6. Recalculate the sum of squared errors using the updated weights;
7. If the sum of squared errors has not decreased, Discard the new weights, increase $\delta$ using $v$ and go to step 4. Otherwise decrease $\lambda$ using $v$ and stop.

4. Numerical Examples

**Example 1.** Consider a linear continuous Lyapunov dynamical system with

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad F(t) = \begin{bmatrix} \frac{e^t}{(1+t)^2} & -\frac{e^t}{1+t^2} \\ \frac{e^t}{1+t^2} & -\frac{e^t}{(1+t)^2} \end{bmatrix}, \quad \text{and} \quad X(t) = \begin{bmatrix} w(t) & x(t) \\ y(t) & z(t) \end{bmatrix},$$

satisfying the initial condition

$$X(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

**Example 2.** Consider a non-linear continuous Lyapunov dynamical systems with

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad X(t) = \begin{bmatrix} w(t) & x(t) \\ y(t) & z(t) \end{bmatrix},$$
and

\[ F(t, X(t)) = \begin{bmatrix} t + x_1^2 e^{x_1} \sin^2 x_1 & 1 + t^2 + x_2^2 \cos x_2^2 \\ 1 - t^3 - x_3^3 \cos x_3^2 & t - x_4^3 \tan x_4 \end{bmatrix}, \]

satisfying the initial condition

\[ X(0) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}. \]

The numerical solutions of linear (Figure 1.) and non-linear (Figure 2) continuous Lyapunov dynamical Systems considered in example 1 and example 2 are calculated by using RK, RK Butcher, Adams Bashforth with RK as well as RK Butcher for initial evaluations and by training ANN with radial basis for these four methods.

5. Conclusion

An artificial Neural Network is constructed to solve the initial value problem associated with continuous Lyapunov dynamical system. Constructed ANN is trained by Levenberg-Marquardt algorithm with optimum network parameters. The examples demonstrated that by taking solutions obtained by Adams
Figure 2: Solution curves – non-linear continuous Lyapunov dynamical system

Bashforth method with RK Butcher values for initial evaluations as target solutions improved accuracy as well as convergence of solutions when compared to Runge-Kutta method, RK-Butcher method and Adams Bashforth method with RK method for initial evaluations.

Acknowledgments

This work is supported by DST-CMS project Lr.No. SR/S4/MS:516/07, Dt.21-04-2008 and the support is gratefully acknowledged.

References


