The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

A Novel Approach for GPS/INS Integration using Recurrent Neural Network with Evolutionary Optimization Techniques

N.Sivasankari
Master Student, Dept. of EEE,
Regional Centre of Anna University, Tirunelveli,
Tamilnadu, India.
nshankari_81@yahoo.com

M.Malleswaran
Assistant Professor, Dept. of ECE,
Regional Centre of Anna University, Tirunelveli,
Tamilnadu, India.
malleshaut@gmail.com

***Abstract*—Integration of Global Positioning System (GPS) and Inertial Navigation System (INS) has been extensively used in aircraft applications like autopilot, to provide better navigation, even in the absence of GPS. Even though Kalman Filter (KF) based GPS/INS integration provides a robust solution to navigation, it requires prior knowledge of the error model of INS, which increases the complexity of the system. Hence Neural Networks (NN) based GPS/INS integration are available in literature. But the NN based solutions have problems such as convergence and inaccuracy. To get better convergence ability the Recurrent Neural Network like Jordan Neural Network is proposed. Normally Back propagation Algorithm (BPA) is used to train the Recurrent Neural Network. But BP algorithm has disadvantages such as slow convergence rate and inaccuracy due to local minima. To overcome these problems, Evolutionary Algorithms like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) trained Jordan Neural Network is proposed to get better position accuracy of the target. In this work, GPS/INS integration based on neural networks like Back Propagation Neural Network (BPNN) and Jordan Neural Network using BPA, GA and PSO are also analyzed and their performance parameters are compared.**

Keywords-Recurrent Neural Network (RNN); Jordan Neural Network; Genetic Algorithm (GA); Particle Swarm Optimization (PSO).

I. INTRODUCTION

The present navigation systems mainly rely on GPS (Global Positioning system) to provide accurate position and velocity information.GPS is capable of providing accurate information to unlimited number of users. Though GPS is accurate and the accuracy does not degrade with time, it suffers from its own drawbacks and errors. The major drawback related to GPS is satellite signal blockage in urban canyons. On the other hand an INS (Inertial Navigation System) is a self contained system that incorporates three accelerometers and three angular velocity components with respect to previous position of the vehicles. The sensors constantly monitor the vehicle's linear accelerations and rotation rates. In general, an IMU (Inertial measurement unit), which incorporates three axis accelerometers and three axis gyroscopes can be used as position and altitude monitoring devices. However the accuracy of INS decreases with time due to wear and tear of mechanical sensors that exhibit long term error growth [12].

Considering the drawback of both the navigation systems, integrating GPS/INS data provides better performance in comparision with either a GPS or an INS standalone system. For instance, GPS position values can be used to update INS and improve its long term accuracy. On the other hand, INS can always provide position information which can be used during GPS outages. Accurate position can be estimated using INS values, provided the exact error difference between GPS and INS values are known before there is an outage of GPS signal. INS is also capable of providing position and altitude information at higher data rates [13]. Conventional GPS/INS integration is accomplished by Kalman Filter (KF). Even though it works well for both linear and non-linear trajectory, it requires the prior knowledge of the position error estimates of INS which is not always possible and also raises the complexity of the system due to the requirement of large memory. The inadequacies of KF can be overcome by using AI (Artificial Intelligence) algorithms and techniques.

In this paper Recurrent Neural Network (RNN) based GPS/INS integration method with evolutionary learning algorithms like GA and PSO are proposed. Jordan network is proposed due to its ability of converging. Faster learning of this network can be achieved through GA and PSO algorithms. So that better computational efficiency can be achieved. The Jordan network based GPS/INS integration

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

using evolutionary algorithm is the novelty of the proposed method. The performance parameters of feed forward network like Back Propagation Neural Network (BPNN) and RNN like Jordan Network using BPA, GA and PSO are also compared. This paper is organized as follows. Section II deals with proposed architecture for GPS/INS integration using Recurrent Neural Network, Section III deals about Neural Networks like BPNN and Jordan network with their architecture, Section IV deals with weight optimization algorithms like BPA, GA and PSO, Section V shows the experimental setup and simulation results and section VI deals with conclusion of this paper.

II. PROPOSED ARCHITECTURE FOR GPS/INS INTEGRATION USING RECURRENT NEURAL NETWORK

The architecture of GPS/INS integration scheme is shown in Figure 1, in which we have introduced any one of the Neural Networks (NN) like BPNN and JORDAN network for training the INS data using evolutionary algorithms. The NN can be operated in both the training mode as well as the prediction mode. The NN receives the data from INS mechanization that contains three position components along the East direction (longitude), the North direction (latitude), and the Vertical direction (altitude). The desired outputs are provided by GPS, the neural network then accumulate the acquired navigation knowledge by updating the synaptic weights whenever the GPS signals are available. As long as GPS signals are available, the learning process continues to reduce the estimation error in order to obtain the optimal network parameters.

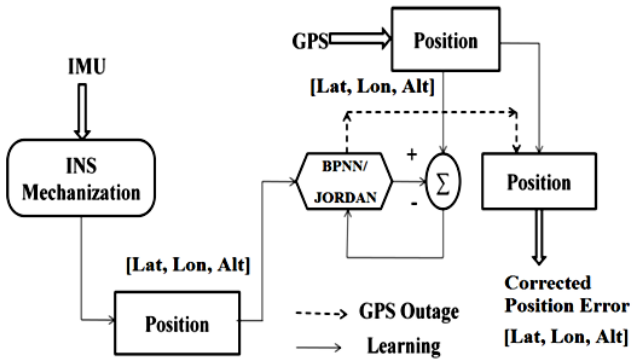

Figure 1. Proposed architecture for GPS/INS integration

In Neural Networks, the synaptic weights should be updated during the navigation process to adapt the network to the latest dynamic condition whenever the GPS signal is available. The training algorithm modifies the network parameters to minimize the mean square error. In case of the GPS outage, the latest updated weights are applied to provide real time prediction. The importance of Jordan network for GPS/INS integration is usually related to its ability to continuously adapt its structure to the application.

III. NEURAL NETWORKS

In this paper, two different neural networks have been analyzed. First the Back Propagation Neural Network which comes under the feed forward neural network is analyzed and the remaining one is the Jordan network which comes under the recurrent type of neural network.

A. *Back Propagation Neural Network Architecture*

BPNN is a multi-layer artificial neural network that uses extend gradient-descent based delta-learning rule and provides computationally efficient method for changing the weights to learn a set of input, output samples, thereby proving good responses to the input [4]. The architecture of BPNN is shown in Figure 2.

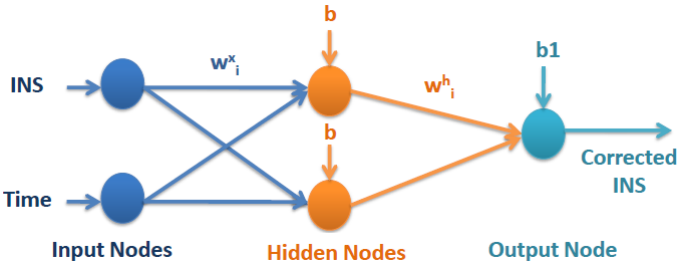

Figure 2. Back Propagation Neural Network Architecture

B. *Jordan Neural Network*

Jordon network shown in Figure 3 is a kind of recurrent network having feedback from the output value. At a specific time t, the previous output value and the current input are used as the inputs to the hidden node. After obtaining the output for a given set of inputs, the output value is sent back through the recurrent links to the context units and saved there for the next training (at time step t+l) [9]. The context units are used only to memorize the previous activations of the output unit.

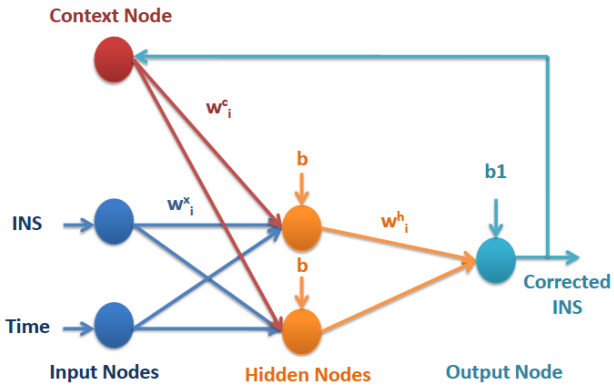

Figure 3. Jordan Neural Network Architecture

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

IV. WEIGHT OPTIMIZATION ALGORITHMS

A. Back Propagation Algorithm (BPA)

In BPA algorithm, the weights and the bias of the output node are updated using gradient- descent based delta-learning rule as follows.

$$w_i^h(new) = w_i^h(old) + \Delta w_i^h \quad (1)$$

$$b1(new) = b1(old) + \Delta b1 \quad (2)$$

Similarly the weights and the bias of the hidden node is updated using gradient- descent based delta-learning rule as follows

$$w_i^x(new) = w_i^x(old) + \Delta w_i^x \quad (3)$$

$$b(new) = b(old) + \Delta b \quad (4)$$

The weight and bias correction terms in equation (1), (2), (3) and (4) can be written as

Weight correction terms:

$$\Delta w_i^h = \alpha \delta_k y_i \text{ and } \Delta w_i^x = \alpha \delta_i z_i \quad (5)$$

Bias correction terms:

$$\Delta b1 = \alpha \delta_k \text{ and } \Delta b = \alpha \delta_i \quad (6)$$

In equations (5) & (6) the term α is the learning rate δ_k and δ_i are the error coefficients at the output and the hidden units respectively.

B. Genetic Algorithm (GA)

Genetic Algorithm is an intelligent exploitation of random search used in optimization problems. Hence it can be used for RNN weight optimization. GA starts at multiple random points (initial population) when searching for a solution. Each solution is then evaluated based on the objective function. Once this has been done, solutions are then selected for the second generation based on how well they perform. After the second generation is drawn, they are randomly paired and the crossover operation is performed. This operation keeps all the weights that are included in the previous generation but allows for them to be rearranged. This way, if the weights are good, they still exist in the population. The next operation is mutation, which can randomly replace any one of the weights in the population in order for a solution to escape the local minima. Once it is completed the generation is ready for evaluation and the process continues until the best solution is found [10, 11].

The objective function (evaluation function) is used to provide a measure of how individual solutions have performed in solving the problem. It takes a chromosome as input and produces an objective value as a measure to the chromosome's performance. To maintain uniformity fitness function is needed to transform the objective value to a fitness value. Fitness value is a quality value which is a measure of the reproductive efficiency of chromosomes. In GA, fitness is used to allocate reproductive traits to the individuals in the population and thus act as some measure of goodness to be maximized. The fitness function can be defined as

$$F(x) = \frac{1}{1 + f(x)} \quad (7)$$

Alternate representation of fitness function to transform the objective function to get the fitness value F(i) as below.

$$F(i) = \frac{P(x_i)}{\sum_{i=1}^{N} P(x_i)} \quad (8)$$

In equation (8), x_i is the i[th] chromosome, $P(x_i)$ is the probability that i[th] chromosome being populated, and N is the total number of chromosomes. The population is then operated by three main operators; reproduction (selection), crossover and mutation to create a new population of points.

Mutation introduces variations into the chromosome. This variation can be global or local. It adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. Having a mutation probability newly generated off springs is mutated at each position in the chromosome.

C. Training of BPNN and Jordan Network using GA

Training of and BPNN and Jordan using GA start with randomly generated initial population having a number of chromosomes (i.e. random weight for NN) as shown below.

w_1^{INS}	w_2^{INS}	w_1^{Time}	w_2^{Time}	w_1^h	w_2^h

Figure 4. Chromosome formed for BPNN using GA

w_1^{INS}	w_2^{INS}	w_1^{Time}	w_2^{Time}	w_1^c	w_2^c	w_1^h	w_2^h

Figure 5. Chromosome formed for Jordan network using GA

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

Based on the error produced in the training of BPNN and Jordan, fitness function is defined. The fitness of each individual chromosome is calculated. Based on the fitness value, a pair of chromosomes is selected in a way that they are fit for the combination process. By doing crossover and mutation new chromosomes are generated [5].

D. Particle Swarm Optimization (PSO)

In this paper, the RNN is trained by Particle Swarm Optimization (PSO) which is also called as swarm propagation learning. PSO applies the concept of social interaction to problem solving based on the movement and intelligence of swarms. It uses a number of particles that constitute a swarm moving around in the search space looking for the best solution. The modification of the particle's position can be mathematically expressed as

$$V_i^{k+1} = wV_i^{k+1} + C_1 Rand_1(..) * (Pbest_i - S_i^k) + C_2 Rand_2(..) * (Gbest - S_i^k) \quad (9)$$

Where w is the inertial weight, V_i^{k+1} is the velocity and S_i^k is the position of agent i at iteration k, C_i is acceleration constant, $Rand_i$ is the random number distributed between 0 and 1, $Pbest_i$ is the pbest of agent, $Gbest$ is the gbest of the group.

$$S_i^{k+1} = S_i^k + V_i^{k+1} \quad (10)$$

Pbest updation is given by

$$Pbest_i(k+1) = \begin{cases} Pbest_i(k) & \text{If } f(S_i(k+1) > Pbest_i(k+1) \\ S_i^{k+1} & \text{Otherwise} \end{cases} \quad (11)$$

Gbest updation is given by

$$Gbest(k+1) = \min(Pbest_i(k+1)) \quad (12)$$

In PSO, balance between the global and local search can be achieved through the inertial weight factor (w) [7, 8].

V. EXPERIMENTAL SETUP AND RESULTS

The sample values for GPS and INS are extracted from FDC (Flight Dynamic Control) toolbox supported by Matlab software and the simulated trajectory is generated as shown in figure 6. INS values for the given trajectory are generated after the initial parameter values of accelerometer and gyroscope sensors are set. Then the corrected INS values are predicted by comparing the INS values with the corresponding GPS values. When we predict the corrected INS values, we predict the Latitude, Longitude and Altitude separately.

The proposed RNN-based GPS/INS integration module is examined and analyzed during training and prediction mode of operations using Matlab. The parameters associated with the RNN module using GA and PSO training are given in Table 1, change through the update procedure to achieve the correct results. Training is performed for latitude, longitude and altitude components. The actual output that is corrected INS value for latitude, longitude and altitude components are found. Over the whole trajectory, no natural GPS absences are detected and thus the corrected INS position of each sample was predicted. In order to evaluate the presentation of the proposed method, GPS signal outages are purposely initiated. The precision of the predicted positions are then compared with the results acquired from GPS solution. In case of a GPS outage, the recently updated weights are functional to present the real time prediction [1,2].

Table1. Implementation Parameters for GA and PSO

GA		PSO	
Parameter	Values	Parameter	Values
Population Size	20	Population Size	20
Crossover	Two Point Crossover	Inertia weight	0.4
Crossover Probability	0.5	c1(acceleration constant)	2
Mutation	Random	c2(acceleration constant)	2
Mutation Probability	0.2	r1(random number)	1
		r2(random number)	1

During the training of neural networks, network over-fitting problem has been encountered. Network over-fitting is a classical machine learning problem. It usually occurs, when the network captures the internal local patterns of the training data set rather than recognizing the global pattern of the data set. It is important to realize that the specification of the training samples is a critical factor in producing a neural network output which is capable of making correct response. Two procedures have been evaluated to overcome the problem of over-fitting namely, early stopping and regularization. In this RNN model, early stopping procedure was used to solve the network over-fitting problem. The aim of early stopping is to mimic the prediction of future individuals from the present population. The regularization method is utilized to avoid the over-fitting problem and to optimize the RNN model. This is known to be a very

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

desirable procedure when the scaled conjugate gradient descent method is adopted for training.

In this paper, early stopping criterion procedure is used to model the INS position error accurately. During the training stage, the module performs the function of understanding the input/output mapping. On the other hand, artificial GPS absences are intentionally introduced to the simulated trajectory in order to test its ability to predict the INS errors and provide reliable INS position information accurately. A time of absence of GPS signals at subsequent intervals are selected at different locations on the trajectory path based on the consideration of GPS jamming and multipath error. The CPU training time is used to measure the efficiency of convergence and the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are used to measure the accuracy. To measure how successful fitting is achieved between target and prediction, the R-square statistic measurement is employed. A value closer to 1 indicates a better fit.

Mean Absolute Error:

$$MAE = \frac{1}{m} * \sum_{i=1}^{m} |y_i - ay_i| \quad (13)$$

R-Square:

$$R - Square = 1 - \left[\frac{\sum_{i=1}^{n} (y_i - ay_i)^2}{\sum_{i=1}^{n} (y_i - ay_i)} \right] \quad (14)$$

Root Mean Square:

$$RMSE = \sqrt{\frac{1}{m} * \sum_{i=1}^{m} (y_i - ay_i)^2} \quad (15)$$

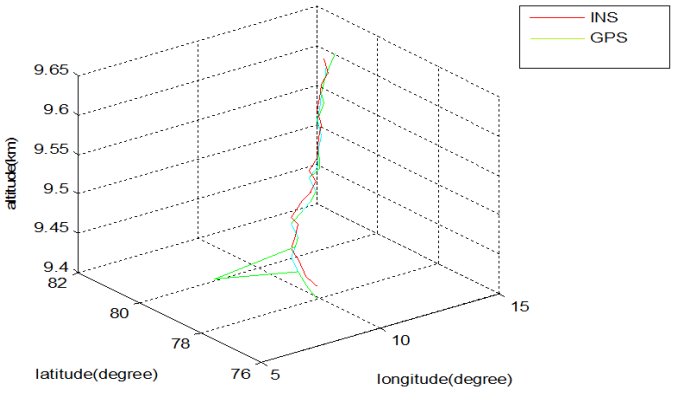

Figure 6. Simulated Trajectory

Where the term 'y_i' denotes target value and the term 'ay_i' represents computed neural network output value and the term 'm' denotes the number of samples.

The BPA error performance for BPNN and Jordan network for trajectory are given in figure 7 and 8 respectively. GA error performance curve for BPNN and Jordan network for trajectory are given in figure 9 and 10 respectively. Similarly, PSO error performance curve for BPNN and Jordan for trajectory are given in figure 11 and 12 respectively. The performance values obtained for latitude, longitude and altitude when the network is trained using BPA, GA and PSO algorithms for trajectory are given in Tables 2, 3 and 4 respectively.

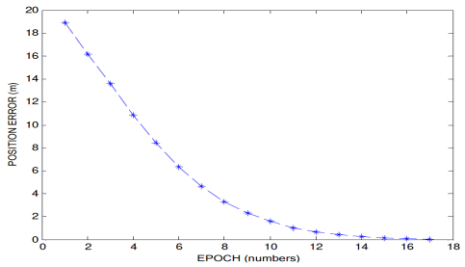

Figure7. BPNN Error by BPA optimization for trajectory

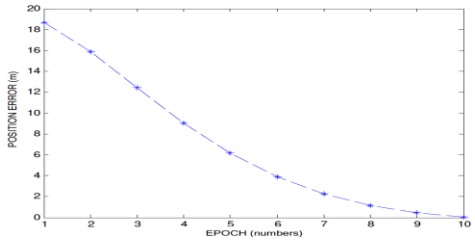

Figure 8. Jordan Error by BPA optimization for Trajectory

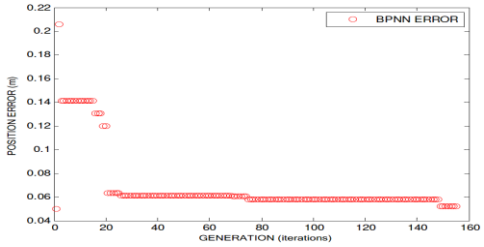

Figure 9. BPNN Error by GA optimization for Trajectory

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

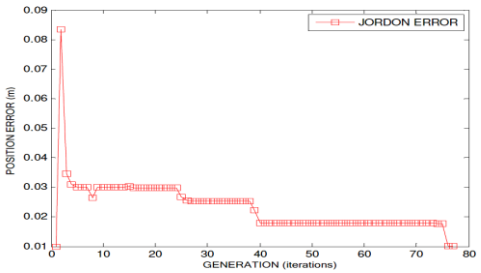

Figure10. Jordan Error by GA optimization for Trajectory

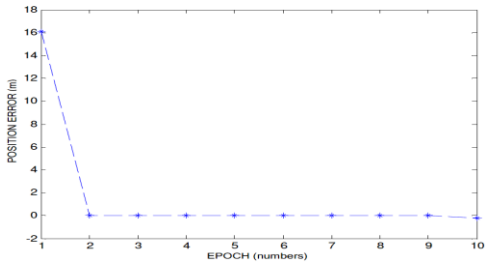

Figure 11. Performance curve for BPNN-PSO for Trajectory

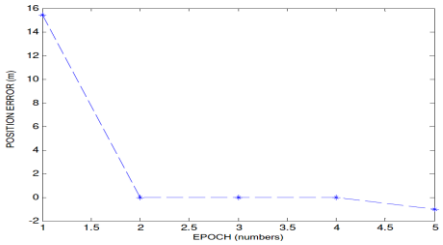

Figure 12. Performance curve for Jordan-PSO for Trajectory

Table 2. Performance Analysis for Latitude using BPA, GA and PSO in BPNN and Jordan network for Trajectory

Algorithm	Parameter	Latitude	
		BPNN	JORDON
BPA	MAE (m)	0.0270	0.0142
	R-square	0.9803	0.9856
	RMSE (m)	0.0220	0.0132
	Training time	12 sec	9 sec
GA	MAE (m)	0.019	0.0037
	R-square	0.9811	0.9854
	RMSE (m)	0.0164	0.0037
	Training time	7 sec	3 sec
PSO	MAE (m)	0.0195	0.0025
	R-square	0.9899	0.9953
	RMSE (m)	0.0094	0.0012
	Training time	5 sec	1 sec

Table 3. Performance Analysis for Longitude using BPA, GA and PSO in BPNN and Jordan network for Trajectory

Algorithm	Parameter	Longitude	
		BPNN	JORDON
BPA	MAE (m)	0.0292	0.0132
	R-square	0.9818	0.9870
	RMSE (m)	0.0148	0.0122
	Training time	12 sec	9 sec
GA	MAE (m)	0.0191	.0042
	R-square	0.9820	.9899
	RMSE (m)	0.0157	0.0033
	Training time	8 sec	2sec
PSO	MAE (m)	0.0187	0.0028
	R-square	0.9829	0.9970
	RMSE (m)	0.0153	0.0015
	Training time	5 sec	1 sec

Table 4. Performance Analysis for Altitude using BPA, GA and PSO in BPNN and Jordan network for Trajectory

Algorithm	Parameter	Altitude	
		BPNN	JORDON
BPA	MAE (m)	0.0281	0.0159
	R-square	0.9797	0.9958
	RMSE (m)	0.0211	0.0145
	Training time	12 sec	9 sec
GA	MAE (m)	0.0195	.0059
	R-square	0.9813	.9876
	RMSE (m)	0.0190	0.0019
	Training time	8sec	2 sec
PSO	MAE (m)	0.0183	0.0048
	R-square	0.9887	0.9958
	RMSE (m)	0.0101	0.0030
	Training time	5 sec	1 sec

From the simulation results, the Jordon network with PSO training is better than BPNN when accuracy and efficiency is concerned. But with the R-square value both the networks with PSO training provides better learning capability. From Tables 2, 3 and 4 the Jordon network with

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.97 e-ISSN: 2251-7545

PSO weight optimization provides better results. Among Back propagation learning, Genetic learning and Swarm propagation learning, Swarm propagation learning gives faster learning. The Learning capability of BPNN is poor when compared to Jordan network. In Jordan network, context layer receives information from output layer leading to the quick convergence of network which receives information from outer layer for context layer. The processing time of Recurrent Neural Network may be varied from one epoch to other epoch, in a selected system. The training time taken by non-linear trajectory is more than linear trajectory in all the selected neural networks.

VI. CONCLUSION

In this paper, we have analyzed GPS/INS integration based neural networks with weight optimization of Genetic and Swarm propagation learning. The results presented in this paper indicate the potential of recurrent neural network in GPS/INS integration. RNN is an empirical and adaptive model in which prior knowledge is not required and the design time is short. The knowledge accumulation of Jordan network is superior when compared to the Back Propagation network. So it provides better performance in non-linear maneuvering trajectories. Among the neural networks analyzed, the Jordan network provides superior performance when error efficiency and position accuracy is considered. From this, we concluded that the Jordan recurrent neural network with PSO weight optimization is well suited method for real time GPS/INS integration for Intelligent Navigation System.

REFERENCES

[1] Kai-Wei Chiang, Yun-Wen Huang, "An intelligent navigator for seamless INS/GPS integrated land vechicle navigation applications", Elsevier, Applied soft computing 8, pp.722-733, 2008.
[2] Kai-Wei Chiang, Aboelmagd Noureldin and Naser El-Sheimy, "A new weight updating method for neural networks based INS/GPS integration architectures," measurement science and technology. 15, 2053-2061, 2004.
[3] S N Sivanandam, S Sumathi, and S N Deepa, "Introduction to Neural Networks using Matlab 6.0", Tata MC Graw-Hill puplication.
[4] Ji-Xiang Dua, De-Shuang Huangc, Guo-Jun Zhangc, Zeng-Fu Wangb, "A novel full structure optimization algorithm for radial basis probabilistic neural networks" , Elsevier, Neurocomputing 70 , 592–596, 2006.
[5] De-Shuang Huang, and Ji-Xiang Du, "A Constructive Hybrid Structure Optimization Methodology for Radial Basis Probabilistic Neural Networks", IEEE Transactions on Neural Networks, Vol. 19, No. 12, December 2008.
[6] D. T. Pham and X. Llu "Training of Elman networks and dynamic system modelling", International Journal of Systems and Science, volume 27, number 2, pages 221,226, 1996.
[7] Min Han, Jianchao Fan, and Jun Wang, "A Dynamic Feedforward Neural Network Based on Gaussian Particle Swarm Optimization and its Application for Predictive Control", IEEE Transactions on Neural Networks, Vol. 22, No. 9, September 2011.
[8] Mohd Saberi Mohamad, Sigeru Omatu, Safaai Deris, and Michifumi Yoshioka, "A Modified Binary Particle Swarm Optimization for Selecting the Small Subset of Informative Genes From Gene Expression Data" IEEE Transactions on Information Technology in Biomedicine, Vol. 15, No. 6, November 2011.
[9] Jeffrey Elman, "Finding Structure in Time", cognitive science, Vol.14, pp.179-21, 1990.
[10] Ioan Ileana, "The optimization of feed forward neural networks structure using genetic algorithms", International Conference on Theory and Applications of Mathematics and Informatics, pp 223-234, 2004.
[11] Zhen-guo che, "feed-forward neural networks training: a comparison between genetic algorithm and back-propagation learning algorithm", International journal of innovative computing, information and control, volume 7, number 10, pp. 5839-5850, october 2011.
[12] G.Dissanayake and Sukkariah, " The aiding of low cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications", IEEE transactions on Robot Automation 17(5) pp 731-747, 2001.

AUTHORS

Mrs. N. Sivasankari received her B.E. Degree in Electronics and Instrumentation from Manonmanium Sundaranar University, Tirunelveli and currently pursuing her M.E Degree in Embedded system Technologies in Regional Centre of Anna University, Tirunelveli. Her current area of interest includes Neural Networks, Fuzzy Systems, Evolutionary Algorithms and Navigation.

Mr. M. Malleswaran received his B.E Degree in ECE and M.Tech. in Communication systems, and currently pursuing his Ph.D. in MIT Campus of Anna University Chennai. Currently, he is working as an Assistant Professor in the Department of Electronics and Communication, Regional Centre of Anna University, Tirunelveli. His current area of interest includes Neural Networks, Fuzzy Systems, Evolutionary Algorithms and Navigation.