Genetic Algorithm Sequential Monte Carlo Methods For Stochastic Volatility And Parameter Estimation

Robert Smith* and Muhammad Shakir Hussain**

***Abstract*- Particle filters are an important class of online posterior density estimation algorithms. In this paper we propose a real coded genetic algorithm particle filter (RGAPF) for the dual estimation of stochastic volatility and parameters of a Heston type stochastic volatility model. We compare the performance of our hybrid particle filter with a parameter learning particle filter present in literature. Our algorithm out performs this algorithm for both the volatility and parameter estimation.**

Index Terms: particle filters, stochastic volatility models, recombination operators, genetic algorithms

I. Introduction

The famous Black-Scholes model [2] was the starting point of a new financial industry and has been a very important pillar of all option trading since. One of its core assumptions is that the volatility of the underlying asset is constant. It was realized early that one has to specify a dynamic on the volatility itself to get closer to the market behaviour. There are mainly two aspects making the fact apparent. Considering historical evolution of volatility by analysing time series data one observes erratic behaviour over time. Secondly, backing out implied volatility from daily traded plain vanilla options, the volatility changes with strike. The most common realisations of this phenomenon are the implied volatility smile or skew. The natural question arises how to extend the Black-Scholes model appropriately. [9]

The variance-gamma model reflects this with its 'g' parameter. Low values of 'g' correspond to a low arrival rate for information and a low volatility; high values of 'g' correspond to a high arrival rate for information and a high volatility.

Manuscript received March 18, 2012; Revised April 14, 2012.

* Dr. Robert Elliot Smith is a senior research fellow at University College London, Department of Computer Science, London, United Kingdom,WC1E 6BT. Email: rob.smith@cs.ucl.ac.uk

** Muhammad Shakir Hussain is a PhD candidate in the Department of Computer Science, at University College London, United Kingdom. Email: m.hussain@cs.ucl.ac.uk

An alternative to the variance-gamma model is a model where the process followed by the volatility variable is specified explicitly. Suppose first that we make the volatility parameter in the geometric Brownian motion a known function of time. The risk neutral process followed by the asset price is then:

$$dS = (r - q)Sdt + \sigma(t)Sdz \qquad (1)$$

Where S is the price process, r is the drift, q is the dividend yield and σ(t) is the time varying volatility process.

The Black-Scholes formulas are correct provided that the variance rate is set equal to the average variance rate during the life of the option. Equation (1) assumes that the instantaneous volatility of an asset is perfectly predictable. In practice volatility varies stochastically. This has led to the development of more complex models with two stochastic variables; the stock price and its volatility.

The stochastic volatility estimation requires filtered estimates to account for estimation errors. Many authors have considered the problem of simulation based filtering with known parameters. A common approach is to use particle filtering methods, see for example Gordan, Salmond and Smith [7], Liu and Chen [10], and Pitt and Shepherd [12].

In 2001, Liu and West proposed a particle filtering algorithm for dual estimation of states and parameters. Their algorithm built upon the works of Gordon [5] in which be proposed the concept of 'artificial evolution'. A brief discussion of 'artificial evolution' and the dual state and parameter estimation algorithm of Liu and West are also given in this paper.

The similarities between evolutionary algorithms and particle filters have been observed by many researchers [8]. In 2005, Patrigo, Sanchez, Gionikellis and Montemayor [13] combined particle filters with population based meta heuristics for visual articulated motion tracking. Uosaki and Hatanaka [15] proposed their evolution strategies particle filter (ESP) in 2007, and applied it to a fault detection algorithm with favourable results. In their algorithm, they added an evolution strategies layer into a generic particle filter. In 2011, a general frame work for applying population based meta heuristics to particle filters was proposed by Patrigo [13].

In this paper we propose a real coded genetic algorithm particle filter (RGAPF) for the dual estimation of volatility and parameters. We apply our particle filtering algorithm on an Euler discretization of the Heston model. The performance of our algorithm is compared with the performance of the state and parameter estimation algorithm given by Liu and West in [11]. In order to quantify performance, we use the traditional measure of particle filter performance: The root mean square error (RMSE). This measure of performance has been used extensively in literature [1].

A brief discussion of particle filters, evolutionary computation and the algorithm proposed by Liu and West is given in the following sections.

II. Stochastic Volatility

The Heston model to approximate the stochastic volatility is given by the following stochastic differential equation:

$$dX(t)/X(t) = \sqrt{V(t)}dW_X(t) \quad (2)$$

$$dV(t) = k\left(\theta - V(t)\right)dt + \varepsilon\sqrt{V(t)}dW_V(t) \quad (3)$$

Where k, ε, θ are strictly positive constants, and where W_X and W_V are scalar Brownian motions in some probability measure; we assume that $dW_X(t).dW_V(t) = \rho dt.$ Where the correlation ρ is some constant in [-1, 1]. *X(t)* represents an asset price process.
The Euler discretization of the above stochastic differential equation is given by:

$$\ln\hat{X}(t+\Delta) = \ln\hat{X}(t) - \frac{1}{2}\hat{V}(t)^{+}\Delta + \sqrt{\hat{V}(t)^{+}}Z_X\sqrt{\Delta} \quad (4)$$

$$\hat{V}(t+\Delta) = \hat{V}(t) + k\left(\theta - \hat{V}(t)^{+}\right)\Delta + \varepsilon\sqrt{\hat{V}(t)^{+}}Z_V\sqrt{\Delta} \quad (5)$$

Where $\hat{X}(t),$ is the observed price process, and $\hat{V}(t)$, is the volatility process that has to be estimated.
For a dual estimation process, the parameters k, ε, θ will also be estimated on line along with the stochastic volatility.

III. .Problem Formulation

To define the estimation problem, consider a dynamic system represented by the state sequence
$\{x_k, k \in \boldsymbol{N}\}$ where the temporal evolution is provided by the state equation:

$$x_k = f(x_{k-1}, v_{k-1}), \quad (6)$$

where '*f*' is a nonlinear function and **N** is the set of natural numbers, $\{v_k, k \in \boldsymbol{N}\}$ is the process noise sequence. The state process is hidden, but we are provided with online measurements of the observation process that is given by the observation equation:

$$z_k = h(x_k, n_k) \quad (7)$$

Where h is a non-linear function of x_k. The objective is to recursively estimate x_k whenever we obtain a new measurement of z_k. At the same time, the parameters of f also need to be estimated recursively.
In the stochastic volatility context, equation (2) gives the hidden state process while equation (3) gives the observation process.

A. Bayesian Filtering

Consider that we know apriori the state and the observation equations, i.e. $x_{k+1} = f(x_k, v_k)$, which can also be assumed to be sampled from $x_{k+1} \sim p(x_{k+1}|x_k)$, because of the random noise v_k. Similarly, the observation equation for his hidden process is$y_k = f(x_k, w_k)$. Which can assumed to be sampled form $p(y_k|x_k)$. The values generated by the state equation are hidden and we only have the observation values visible.

The Bayesian solution to computing the posterior distribution $p(x_k|y_{1:k})$ of the state vector given past observations is given by the general Bayesian update recursion:

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})} \quad (8)$$

$$p(y_k|y_{1:k}) = \int p(y_k|x_k)p(x_k|y_{1:k-1})dx_k \quad (9)$$

$$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k})dx_k \quad (10)$$

The above result is the cornerstone in non-linear Bayesian filtering. The first equation follows directly from Bayes law, and the other two follow from the law of total probability. The first equation corresponds to a measurement update, the second is normalization constant and the third corresponds to a time update [6].

B. Particle Filtering Equations

A generic particle filter uses the above Bayesian measurement and time update equations to predict the posterior distribution function (pdf) [6]. The posterior distribution is estimated using N particles $\{x^i\}_{i=1:N}$. Each particle is a potential estimate of the state vector. The particles are assigned weights based on their relative fitness compared to each other.

The Bayesian filtering equation in case of N particles of a particle filter are modified as follows:

$$p\left(x^i_{1:k+1}|y_{1:k}\right) = p\left(x^i_{k+1}|x^i_{1:k}, y_{1:k}\right)p(x^i_{1:k}|y_{1:k})$$
$$= w^i_{k|k}p(x^i_{k+1}|x^i_k) \quad (11)$$

Hence:

$$p(x_{1:k+1}|y_{1:k}) \approx \sum_{i=1}^{N} w^i_{k|k}p(x^i_{k+1}|x^i_k)\delta(x_k - x^i_k) \quad (12)$$

Now consider the case where sampling from $p(x_{k+1}|x_k)$ is not computationally feasible, the concept of importance sampling is used.

$$p(x_{1:k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)\, p(x_k|y_{1:k}) dx_k$$

$$= \int q(x_{k+1}|x_k) \frac{p(x_{k+1}|x_k)}{q(x_{k+1}|x_k)}\, p(x_k|y_{1:k}) dx_k$$
(13)

Thus;

$$p(x_{k+1}|y_{1:k}) = \sum_{i=1}^{N} \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i)} w_{k|k}^i \delta(x_{1:k+1} - x_{1:k+1}^i)$$
(14)

IV. Evolutionary Algorithms

Evolutionary algorithms are stochastic population based metaheuristics that have been successfully applied to many real and complex optimization problems [14]. Evolutionary algorithms are based on the notion of competition. They are based on the evolution of a population of individuals. An objective function associates a fitness value with every individual indicating its suitability to the problem. At each iteration using some selection procedure, individuals are selected and variation operators are applied that updates the population. The process is iterated until a stopping criterion is reached.

A. Genetic algorithms

Genetic algorithms have been developed by John Holland in the 1990s to understand the adaptive processes of natural systems [3]. They have been applied to optimization and machine learning in the 1980s [3]. Traditionally genetic algorithms use binary representation of the population of individuals. After selection, a crossover (recombination) and mutation operators are applied to the individuals for search space exploration. A generic template for a genetic algorithm is shown below:

Template of a Genetic algorithm

```
Generate(P(0)); /* initial population */
t = 0;
While not Termination_Criterion(P(t)) Do
Evaluate(P(t));
P'(t) = Selection(P(t));
P'(t) = recombination(P'(t));
P'(t) = mutation(P'(t));
P(t + 1) = Replace(P(t), P'(t));
t = t + 1;
End While and Output Best individual found
```

B. Real coded genetic algorithm

Genetic algorithms (GA) are a very popular class of evolutionary algorithms [3]. Traditionally GAs are associated with binary representation, where the population was coded into binary by using some coding mechanism.

Real coded genetic algorithms do not use a coding scheme to code the population. The recombination and mutation operators are applied on the population directly. The theory of convergence of real coded genetic algorithms was given by Goldberg in [4].

Many different types of real recombination operators are found in literature. We discuss the arithmetic recombination operators that we have used in our experiments.

1) Arithmetic recombination

The arithmetic recombination operator attempts to average the elements of the parent. Given two parents $x_1$and x_2, the arithmetic recombination operator creates an off spring o using the weighted average:

$o_i = \alpha x_1 + (1 - \alpha)x_2$ (15)

Where o_i is the offspring created.

2) Real Mutation operator

In case of real coded genetic algorithms, the mutation operator can be viewed as a random perturbation or noise that is randomly added to a component making up the chromosome. In our experiments we used of the Gaussian mutation operator.

The Gaussian mutation adds random perturbations, sampled from the distribution N(0,σ). Where σ is the standard deviation of the mutation operator [14].

V. Dual state and parameter estimation.

The need for more general algorithms that deal simultaneously with both fixed model parameters and state variable is especially pressing [11].

Gordon et al. in 1993 [5] introduced the idea of adding additional random disturbances or roughening penalties to sampled state vectors in an attempt to deal with sample degeneracy. Extending this idea to fixed model parameters leads to a synthetic method of generating new sample points for parameters. This ad-hoc idea is similar to using a Gaussian mutation in evolution strategies literature.

Consider a parameter vector θ_t in a model at time t. At time t+1 add an independent zero-mean normal increment to the parameter.

That is:

$$\theta_{t+1} = \theta_t + \varepsilon_{t+1}$$

$$\varepsilon_{t+1} \sim N(\theta, \sigma)$$

The key motivating idea is that the artificial evolution provides the mechanism for generating new parameter values at each time step. Liu and West proposed a kernel method with location shrinkage for this parameter estimation problem. The shrinkage pushes samples $\theta_t^{(j)}$values towards their mean, $\bar{\theta}_t$ before adding a small degree of 'noise' implied by the normal kernel.

Suppose $p(\theta_{t+1}/D_t)$ is the posterior distribution of the parameters. Then the parameter update is given by $p(\theta_{t+1}/D_t) \approx \sum_{j=1}^{N} w_t^{(j)} N(\theta_{t+1}|m_t^{(j)}, h^2 V_t)$

Where, $m_t^{(j)} = a\theta_t^{(j)} + (1-a)\bar{\theta}_t$

$$a = \sqrt{1-h^2}$$

V_t is the variance and h is chosen as a slowly decreasing function on N. The pseudo code of the state and parameter learning algorithm follows:

```
1. For each j = 1, …N, identify the prior point estimates of $(x_t, \theta)$ given by $(\mu_{t+1}^{(j)}, m_t^{(j)})$ where
       $\mu_{t+1}^{(j)} = E(x_{t+1}|x_t^j, \theta_t^{(j)})$
   may be computed from the state evolution density and $m_t^{(j)} = a\theta_t^{(j)} + (1-a)\bar{\theta}_t$ is the kernel location.
2. Sample a new parameter vector $\theta_{t+1}^{(k)}$ from the normal component of kernel density, namely
       $\theta_{t+1}^{(k)} \sim N(\cdot\,|m_t^{(k)}, h^2 V_t)$
3. Sample a value of the current state vector $x_{t+1}^{(k)}$ from the system equation
       $p(\cdot\,|x_t^{(k)}, \theta_{t+1}^{(k)})$.
4. Evaluate a corresponding weight
       $w_{t+1}^{(t)} \propto \frac{p(y_{t+1}|x_{t+1}^{(t)}, \theta_{t+1}^{k})}{p(y_{t+1}|\mu_{t+1}^{(k)}, m_t^{(k)})}$
5. Repeat steps (2) – (4) a large number of times to produce a final posterior approximation $(x_{t+1}^{(k)}, \theta_{t+1}^{(k)})$ with weights $w_{t+1}^{(k)}$, as required.
```

For further details of the algorithm, refer to [11].

VI. Real Coded Genetic Algorithm Particle Filter (RGAPF)

We propose a state and parameter estimating particle filter that has a real coded genetic algorithm (RGA) layer. The RGA layer replaces the resampling method in the particle filter. Park, Hwang, Rou and Kim in 2007 showed that a genetic algorithm layer can be used to solve the sample depletion problem. In our proposed algorithm the resampling method is replaced by the RGA layer. The importance weights assigned to a particle is a measure of its relative fitness in the particle population [6]. The assigned weight is used as a measure of the fitness landscape of the parameters. Genetic algorithms are stochastic optimization algorithms. In this scenario they provide us with an optimized estimate of the particles.

In the genetic algorithm layer we use an arithmetic recombination operator and an annealed Gaussian mutation operator. The annealed Gaussian mutation is similar to the random perturbation that are used in artificial evolution, however the standard deviation σ decays with time *t*.

$$\theta_{t+1} = \theta_t + \varepsilon_{t+1}$$

$$\varepsilon_{t+1} \sim N(\theta, \sigma)$$

The arithmetic recombination operator attempts to average the elements of the parent. Given two parents $x_1$and x_2, the arithmetic recombination operator creates an off spring o using the weighted average:

$$o_i = \alpha x_1 + (1-\alpha)x_2$$

Where α is a constant, and x_{1i} and x_{2i} are particles that were selected for recombination.

```
Choose a proposal distribution $q(x_{k+1}^i|x_k^i, y_{k+1})$, resampling strategy and the number of particles N. The pseudo code of our proposed algorithm is given below.
Initialization: Generate $x_1^i \sim p_{x_0}, i = 1, \ldots, N$ and let initial weights to be 1/N.
For loop k = 1, 2…end of observations.

Start:

    1. Measurement and Time Update: For i = 1,2,…, N
           $w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k|x_k^i)$
       Where the normalization weight is given by:
           $c_k = \sum_{i=1}^{N} w_{k|k-1}^i \, p(y_k|x_k^i)$

    2. Estimation:
       The filtering density is approximated by
           $p(x_{1:k}|y_{1:k}) = \sum_{i=1}^{N} w_{k|k}^i \, \delta(x_{1:k+1} - x_{1:k+1}^i)$
       Apply Recombination & Mutation:

    3. IF ($y_k$ is not last observation)
       k = k + 1
       Go to step 1.
       Else End For loop.
```

VII. Design of Experiments and Results

We simulate a price process and a volatility process by using equations (4 - 5) using known parameters. The two algorithms are then tested on this volatility process for comparison. The Euler discretization of the Heston stochastic volatility model is given in equation 4-5.

Where $\hat{X}(t)$, is the observed price process, and $\hat{V}(t)$ is the volatility process that has to be estimated. We simulate a volatility and prices process by setting the parameters equal to:

$$k = \theta = \epsilon = 0.5,$$

$$\Delta = 0.001$$

The two algorithms mentioned in this paper are then run 50 times on the simulated prices process, and the average RMSE of the 50 runs is calculated.

The algorithms estimate the volatility process and at the same time learn the static parameters of the state equation.

We calculate the average of the root mean square error (RMSE) as our performance measure of the volatility estimation. The reason for using RMSE as a performance measure is given in [1].

Particle Filter	Root Mean Square Error
PLA (Particle Learning Algorithm)	0.0328
RGAPF	0.0074

Graphically, for a single run, the estimation results obtained from running these two algorithms are shown below:

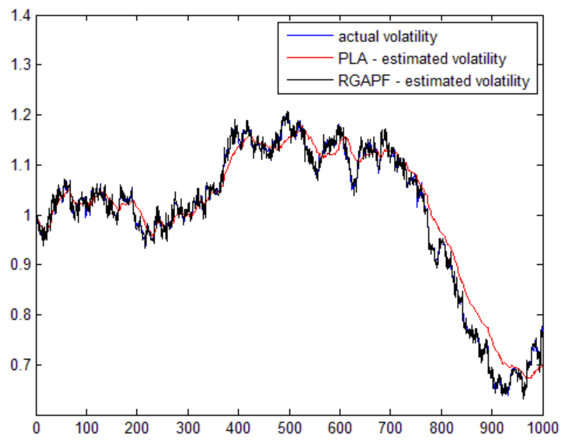

Figure 1. Volatility estimation

In the above figure, the RGAPF estimate (black line) and the actual volatility (blue line) lie close together in comparison to the PLA estimate (red line). The estimation of parameters with time are shown in the next graphs.

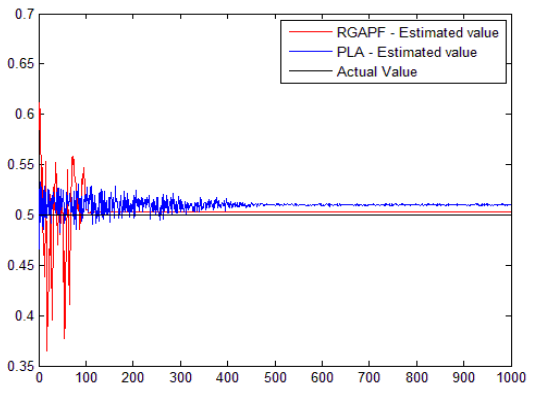

Figure 2. Estimation of 'k'

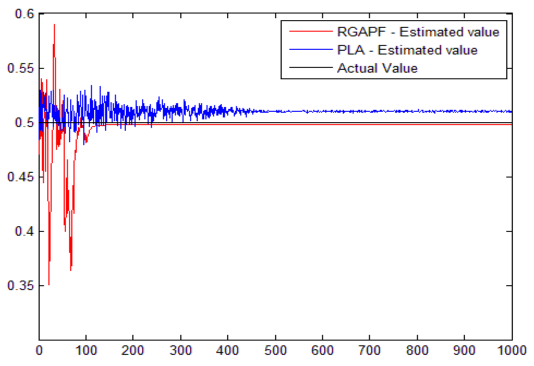

Figure 3. Estimation of 'θ'

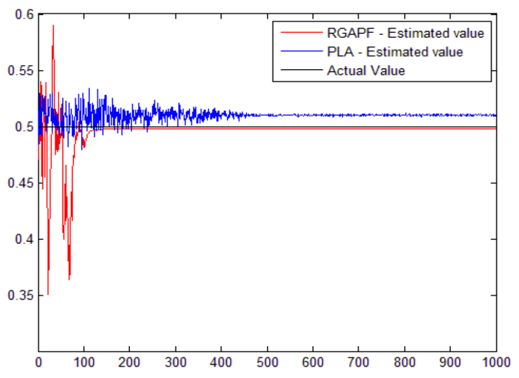

Figure 4. Estimation of 'ε'

VIII. Conclusion

Our algorithm has performed significantly better at estimating both the stochastic volatility and the static parameters of the model. Figures 2-4 show the estimation of the static parameters. The plots show that the RGAPF converge to the actual values of the parameters after only a few iterations, and compared to the algorithm due to Liu and West, the estimate is closer to the actual value. The parameter learning algorithm of Liu and West can be seen as using an annealed mutation operator, i.e. a Gaussian mutation, where the standard deviation of the perturbation decays with time. In RGAPF, along with a decaying (annealed) mutation, we have also added a recombination operator. The importance of the recombination operator in a genetic algorithm framework has been mentioned by many researchers [3]. In the real coded set up, most notably [4] has commented on the added search space exploration capability that recombination operators bring in an optimization problem. We can conclude that the recombination operator may be responsible for the improved performance of the particle filter.

For future experiments, other recombination operators apart from arithmetic recombination should be tried to establish which recombination operator leads to the best possible performance.

REFERENCES

[1] Arulampalam, M.S, Maskell, S., Gordon, N., and Clapp, T., A tutorial on particle filters for online non-linear/non-Gaussian Bayesian tracking. IEEE Transactions on signal processing, Vol. 50, No. 2, February 2002.

[2] F.Black and M.Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81:637-654, 1973.

[3] D.E., Goldberg, (1989*) Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, ISBN 0-201-15767-5

[4] Goldberg, D.E. *Real Coded Genetic Algorithms, Virtual Alphabets and Blocking.* Complex Systems 5(1991) 139-167.

[5] Gordon, N. (1993). Bayesian methods for tracking, PhD thesis, University of London.

[6] Gustafsson, F. *Particle filter theory and practice with positioning applications*. IEEE A&E Systems Magazine vol.25, No.7 July 2010

[7] Gordon, N., Salmond, D., and Smith, A. *A novel approach to nonlinear/non-Gaussian Bayesian state estimation*. In lEE Proceedings on Radar and Signal Processing, vol.140, 1993, 107-113.

[8] Anders M. Johansson and Eric A. Lehmann, Evolutionary Optimization of Dynamics Models in Sequential Monte Carlo Target Tracking, *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 879-894, August 2009.

[9] Christian Kahl. Modelling and simulation of stochastic volatility in finance. Ph.D. dissertation, University of Wuppertal, 2007.

[10] Liu, J.S and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems, Journal of the American Statistical Association 93(443): 1032 – 1044.

[11] Liu, J.and West, M. *Combined parameter and state estimation in simulation-based filtering*. Sequential Monte Carlo in Practice, Springer July 2001, 197-217.

[12] Pitt, M.K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters, Journal of the American Statistical Association 94(446): 590 – 599.

[13]Patrigo, J., Sanchez, A., Gianikellis, K. and Montemayor*, A. Heuristic particle filter: applying abstraction techniques to the design of visual tracking algorithms.* Expert Systems. Wiley. Vol. 28, Issue 1, Pages 49-69, Feburary 2011

[14] Talbi, E. Metaheuristics: From Design to Implmentation. (Wiley series on Parallel and Distributed Computing). Wiley-Blackwell July 2009. 208-213.

[15] Uosaki, K., Kimura, Y. and Hatanaka, T. *Nonlinear State Estimation by Evolutionary Strategies Based Particle Filters*. 0-7803-7804-0 /03/17.00 0 2003 IEEE