VI. CONCLUSION

We have shown that dilution of precision terms for relative positioning, using double-difference processing of GPS satellite signals, are bounded by the corresponding dilution of precision terms for point positioning. This result is valid for the case of four satellites, with a common reference satellite. An example is provided which shows that such a common reference is required. It remains to extend our result to an arbitrary number of satellites. Simulations conducted to date have yielded no counterexamples for the case of $n > 4$ satellites.

RICHARD O. NIELSEN
Rockwell International Corporation
3370 Miraloma Avenue
Anaheim, CA 92803-3105

REFERENCES

[1] Hofmann-Wellenhof, B., Lichtenegger, H., and Collins, J. (1992)
Global Positioning System Theory and Practice.
New York: Springer-Verlag, 1992.

[2] Chaffee, J., and Abel, J. (1994)
GDOP and the Cramer-Rao bound.
In *Proceedings of the IEEE PLANS*, Las Vegas, NV, Apr. 11–15, 1994, 663–668.

A Fast Weighted Bayesian Bootstrap Filter for Nonlinear Model State Estimation

In discrete-time system analysis, nonlinear recursive state estimation is often addressed by a Bayesian approach using a resampling technique called the weighted bootstrap. Bayesian bootstrap filtering is a very powerful technique since it is not restricted by model assumptions of linearity and/or Gaussian noise. The standard implementation of the bootstrap filter, however, is not time efficient for large sample sizes, which often precludes its utilization. We propose an approach that dramatically decreases the computation time of the standard bootstrap filter and at the same time preserves its excellent performance. The time decrease is realized by resampling the prior into the posterior distribution at time instant k by using sampling blocks of varying size, rather than a sample at a time as in the standard approach. The size of each block resampled into the posterior in the algorithm proposed here depends on the product of the normalized weight determined by the likelihood function for each prior sample and the sample size N under consideration.

Manuscript received June 15, 1995; revised July 22, 1996.

IEEE Log No. T-AES/33/1/01108.

I. INTRODUCTION

The extended Kalman filter (EKF) has been one of the most popular approaches to recursive nonlinear state estimation problems [1, 3, 5]. According to this approach, the nonlinear state transition and observation equations are linearized about the current state, and with the system of equations linearized, the Kalman filter algorithm is applied. However, because of the linearization process, the filter may diverge from the correct state-space trajectory, even when the system and/or observation noises are low. This potentially poor performance in some nonlinear state estimation applications has led to the introduction of a Bayesian approach using a weighted bootstrap methodology [2].

The weighted bootstrap filter takes a set of random samples from the probability density function (pdf), conditioned on the measurements, describing the state vector distribution at the discrete time instant k. The state samples are then propagated through the system state model, and the resulting sample distribution is updated using the weighted bootstrap as presented in [6]. The advantage of this approach over the EKF is that no linearization of the equations is performed, so divergence is not a problem. Moreover, the Bayesian bootstrap filter does not restrict the class of system models due to analytical tractability. However, a drawback of this filter, as given in [2], is the computational load of the algorithm in the update stage. Although the update portion of the algorithm is very simple, it is very time consuming when, for accurate estimates, at each time instant large sample sizes (i.e., a few hundred) are required.

The time required to run the bootstrap filter becomes important when we have to apply the filter in real time, or we are faced with the problem of selecting a model from a set of candidates. In these applications, the standard bootstrap filter may become intractable. By contrast, the fast weighted bootstrap proposed here provides great reduction in computation time. The reduction is achieved by using a special resampling scheme, which is an averaging-type procedure. It is based on the expected number of times each prior sample should appear in the posterior at each time instant. The justification for using an averaging procedure is that, had a Monte Carlo simulation been run for some time instant using a given set of prior samples and the data summarized by averaging, the result would be nearly identical to the approach proposed here.

This paper proceeds by first reviewing the Bayesian bootstrap as presented in [2]. This is followed by a description and derivation of our fast bootstrap method. Finally, we conclude with an example that provides performance comparison of the proposed fast and standard bootstrap filters.

II. THE BAYESIAN BOOTSTRAP FILTER

In this discussion we are concerned only with the problem of estimating the state vector of a discrete-time system. The system model is assumed to have the form

$$x_{k+1} = f_k(x_k, w_k) \tag{1}$$

where $f_k : \mathcal{R}^n \times \mathcal{R}^m \to \mathcal{R}^n$ is the state transition function, and $w_k \in \mathcal{R}^m$ is a sequence of independent zero mean random vectors. The noise vector w_k is assumed independent with the state vector x_l, for $l \leq k$, and the pdfs of the w_ks are known. At the times under consideration, the measurements $y_k \in \mathcal{R}^p$ are related to the state vector via the observation equation

$$y_k = h_k(x_k, v_k) \tag{2}$$

where $h_k : \mathcal{R}^n \times \mathcal{R}^r \to \mathcal{R}^p$ is the measurement function, and $v_k \in \mathcal{R}^r$ is another sequence of independent zero mean random variables. These noise samples are independent with the state vector samples as well as the system equation noise samples, and their pdfs are also assumed known. A further assumption is that the initial pdf, $p(x_1 \mid D_0) \equiv p(x_1)$, of the state vector is available together with the functional forms of the state transition function and observation functions for all k. The available information at time instant k is the set of measurements denoted by $D_k = \{y_1, y_2, \ldots, y_k\}$. The end result is that the posterior pdf for state estimation at time instant k conditioned on the measurements, is given by Bayes' rule, i.e.,

$$p(x_k \mid D_k) = \frac{p(y_k \mid x_k)p(x_k \mid D_{k-1})}{p(y_k \mid D_{k-1})}. \tag{3}$$

The detailed derivation of the above equation is given in [2, sect. 2]. From this formulation the bootstrap filter algorithm can be described as follows.

Let $x_k^*(i)$, $i = 1,2,\ldots,N$, be the set of prior samples for instant k. These samples are passed through an *update* process, which uses the measurement y_k, and each prior sample $x_k^*(i)$ to evaluate the normalized likelihood for each prior sample according to

$$q_i = \frac{p(y_k \mid x_k^*(i))}{\sum_{j=1}^{N} p(y_k \mid x_k^*(j))}. \tag{4}$$

We then define a discrete distribution over $\{x_k^*(i)\}$ with probability mass q_i associated with each $x_k^*(i)$. The distribution is resampled N times to generate the set of samples $\{x_k(i)\}$, so that for $j = 1,2,\ldots,N$, the probability $P\{x_k(j) = x_k^*(i)\} = q_i$. The result of the update procedure is that the samples $\{x_k(i)\}$ are distributed as the required pdf, $p(x_k \mid D_k)$.

Next, the $\{x_k(i)\}$ are passed through a prediction process, where we use the system transition equation to generate a set of predicted samples $\{x_{k+1}^*(i)\}$ by calculating

$$x_{k+1}^*(i) = f_k(x_k(i), w_k(i)). \tag{5}$$

The samples $w_k(i)$ are generated from the pdf of the system noise $p(w_k)$ so that there is one sample for each of the N samples in the set $\{x_k(i)\}$. The $\{x_{k+1}^*(i)\}$ are then passed similarly through the update and prediction processes as $\{x_k^*(i)\}$. The update and prediction processes are repeated until the desired number of time samples has been processed. Note that the algorithm is initiated by generating N samples $x_1^*(i)$, where $i = 1,2,\ldots,N$, from the known prior $p(x_1)$.

The resampling portion of the update process is implemented by drawing a random sample u_i from the uniform $(0,1]$ distribution. When

$$\sum_{j=0}^{M-1} q_j < u_i \leq \sum_{j=0}^{M} q_j \tag{6}$$

where $q_0 = 0$, we choose $x_k(i) = x_k^*(M)$ for making up the posterior.

If N is large, this process takes a very long time to complete, and the problem is further exacerbated when a large number of samples is simulated over many time instants. However, examining the structure of the resampling process, a faster method is possible which can still generate useful results. The faster method is based on the expected number of times each prior sample should appear in the posterior. It is described in the next section.

III. DERIVATION OF THE FAST BOOTSTRAP

As mentioned above, the main limitation of the weighted bootstrap as given in [2] is in the computational load for its implementation. Most of it comes from the resampling of the prior distribution into the posterior. The current proposal is to decrease the execution time of the weighted bootstrap by generating samples into the posterior in groups rather than one at a time as in the weighted bootstrap. The groups are generated based on the expected number of times each value in the prior should be resampled to the posterior. Simulations have been performed comparing the two resampling approaches and are presented in the next section using a highly nonlinear model from [4], which was used in [2] to show the superior performance of the weighted bootstrap filter over the EKF.

The weighted bootstrap provides a possibly small, but finite, probability of resampling any value $x_k^*(i)$ for $i = 1,2,\ldots,N$, from the prior into the posterior. The probability of resampling a particular $x_k^*(M)$ is q_M. To implement this, the value of $x_k^*(M)$ is selected each time a random variable U uniformly distributed on $(0,1]$ satisfies

$$\sum_{j=0}^{M-1} q_j < U \leq \sum_{j=0}^{M} q_j. \tag{7}$$

Thus, the probability of selecting $x_k^*(M)$ is the same as U lying in the interval bounded as shown above. Equivalently, this probability can be expressed as U lying in the range,

$$0 < U \leq q_M. \tag{8}$$

Therefore, the probability of selecting a particular $x_k^*(M)$ on a single trial is as claimed. In addition, the probability of not selecting $x_k^*(M)$ in a single trial is $1 - q_M$. Now, with a sequence of N trials, the weighted bootstrap resampling procedure can be analyzed as a sequence of Bernoulli trials. Cast in this form, the probability of "success" on a single trial (i.e., selecting an $x_k^*(M)$) is $p = q_M$, and the probability of "failure" is $q = (1 - q_M)$. The probability of selecting the value $x_k^*(M)$ exactly L times in N trials is given by the binomial distribution of order N. Thus, the expected number of times any prior sample $x_k^*(i)$ should appear in the posterior is Nq_i. This is the key to the fast algorithm.

In our algorithm, at a fixed time instant k, we pick one of the N prior samples from $\{x_k^*(i)\}$, say $x_k^*(M)$, at random by assigning a sampling probability of $1/N$ for each $x_k^*(i)$ and place $[Nq_M]$ samples of the $x_k^*(M)$ value into the posterior, rather than generating one sample into the posterior at a time as in the weighted bootstrap approach. The symbol $[\cdot]$ means the largest integer not exceeding the contents. This method is repeated until a total of N samples have been generated. Then the resampling is stopped, and the posterior samples are projected ahead using the given system model. The resample and projection scheme is repeated until the desired number of observed data has been processed.

As with most statistical simulation algorithms, the proposed fast algorithm has some difficulty in handling low probability events known as outliers. Briefly, if at a particular time instant k for some is, the product of N and q_i is not at least one, then those $x_k^*(i)$s in the prior cannot generate any samples into the posterior using the $[\cdot]$ criterion. The weighted bootstrap on the other hand, guarantees each prior sample a chance, no matter how small, of being selected in the posterior. This difficulty can be handled in two ways depending on the needs of the modeler. In either case selecting a large N decreases this problem, and some experimentation may be necessary to determine what "large N" means.

In any bootstrap method, of course, there is no guarantee that outliers even exist in a prior distribution sample of fixed size. However, with an N large enough, statistically a few outliers will eventually appear, and therefore a mechanism to handle them is necessary. For the fast algorithm, one solution is to choose an N large enough so that $[Nq_i] \geq 1$ for almost every $x_k^*(i)$ in the prior. However, no matter how large N is selected, there can be outliers still ignored when, dependent on y_k, the product of N and the normalized resampling probability q_i of some states is less than one. Hence, relative to the repeated application of the weighted bootstrap at that time instant, given the same set of prior samples, the low probability states are underrepresented in the posterior.

Another solution is to consider $\max([Nq_M], 1)$ as the number of times of including in the posterior the selected value $x_k^*(M)$. Thereby, even those values which are outliers can be selected at least once into the posterior. However, there is a danger here because if N is too small, some outliers can be overrepresented in the posterior by selecting them much more often than their normalized probability indicates.

A possibility to overcome the above problems is a combination approach. This entails that each prior sample $x_k^*(i)$ with a q_i such that $[Nq_i] = 0$, is "tagged" when it is selected the first time so it cannot be selected again at a particular time step. This guarantees that each sample from the prior below $1/N$ probability can only appear at most once in the posterior. The removal of the outliers from the prior population of N samples once selected is such a small effect when N is chosen properly, that hypergeometric arguments are not necessary when considering the resampling procedure proposed [7].

Due to the random selection of the samples from the prior, it is possible to generate more than N samples in the posterior. When this happens, the posterior sample vector is truncated to the first N selected. Random selection of the prior sample index provides several benefits. First, the process of selecting values is unbiased as long as the random number generator is good. That is, no index in the prior distribution sample vector is more likely to be selected than any other. So every sample has an equal chance of selection, although they will not contribute an equal number of samples to the posterior. Second, a deterministic selection criteria for selecting indexes cannot guarantee exactly when the N points will be achieved. Thus, it seems random sampling of the prior distribution sample vector is justified.

IV. SIMULATION RESULTS

In this section, we compare the performance of the proposed fast bootstrap algorithm to that of the standard bootstrap method. For the purpose of simulation, the highly nonlinear scalar model from

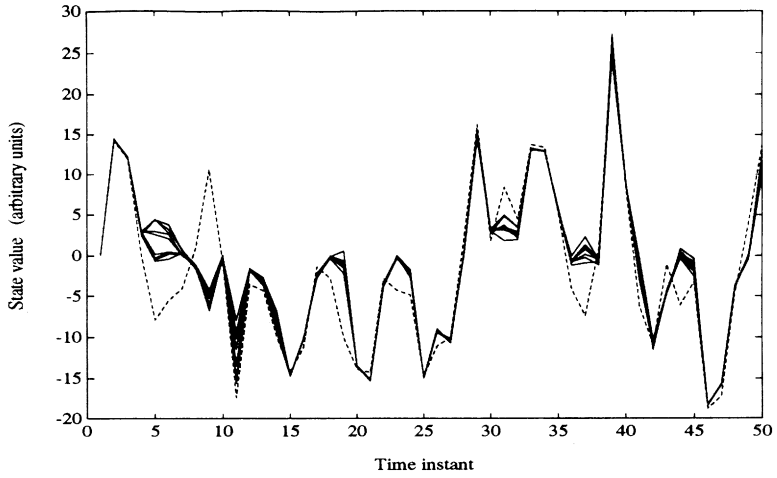

Fig. 1. Performance of standard bootstrap filter. Dashed line denotes true system states, solid line denotes estimates of 9 different trials.

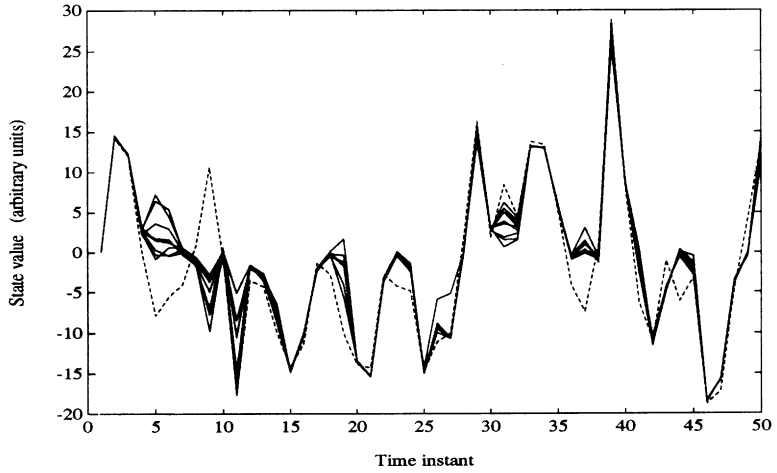

Fig. 2. Performance of fast bootstrap filter. Dashed line denotes true system states, solid line denotes estimates of 9 different trials.

[4 and 2] is used to generate the observed data. We chose this model because its nonlinearity in both measurement and observation equations produces difficulties with standard estimation approaches, in particular the EKF. Further, the modality of the likelihood function depends on the sign of the observed data. The model is described by

$$x_k = 0.5x_{k-1} + \frac{0.25x_{k-1}}{1 + x_{k-1}^2} + 8\cos(1.2(k-1)) + w_k \qquad (9)$$

$$y_k = \frac{x_k^2}{20} + v_k D$$

where $w_k \sim N(0,10)$, $v_k \sim N(0,1)$, and $x_0 = 0.1$. The assumption of Gaussian distribution of w_k and v_k is not a requirement. It is made here merely for comparison with previously published work. Using Monte Carlo methods, both bootstrap algorithms were repeatedly run many times and the results are summarized in Figs. 1–4. Figs. 1 and 2 show the performance "spread" of the standard and fast bootstraps. For clarity, only 9 runs are shown. In each case note that the repeatability between realizations is in general quite good, both, amongst the realizations of each algorithm and between the algorithms. We used $N = 500$ at each time instant and applied the $[Nq_M]$ criterion to determine the number of times to add $x_k^*(M)$ into the posterior. This number of points per time instant provided simulations that give repeatable data within a reasonable tolerance. In Fig. 3, the individual run data are summarized by displaying the sample mean (ensemble average) performance for the estimated state value (minimum mean-square error (MMSE)) of both algorithms against the true nonlinear states, versus time instant. We show that both bootstrap methods on average track each other very well. When at some time instants the fast bootstrap deviates from the true state value, so does the well-accepted standard bootstrap. Fig. 4 shows the mean-square error (MSE) of the estimates of the two bootstrap approaches at each time instant. Clearly, the fast algorithm provides state estimates nearly as good as the standard method. Additional experiments with other models have shown similar agreement between the standard and

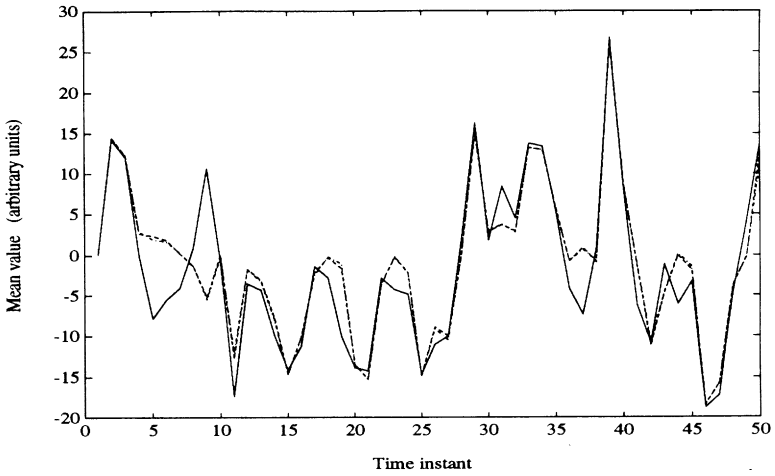

Fig. 3. Trajectories of state estimate mean values (dashdot line is standard bootstrap filter, dashed line is fast bootstrap filter, solid line is true states).

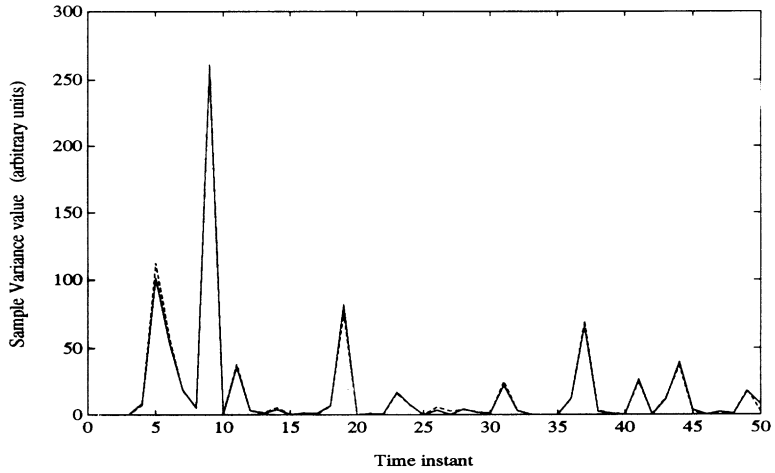

Fig. 4. Comparison of MSE of state estimates between standard bootstrap filter (solid line) and fast bootstrap filter (dashed line).

fast methods. Regarding the execution speed, we have found that the fast algorithm, using MATLAB software running on a 486 PC, reduces the execution approximately by a factor of 20.

V. CONCLUSION

In this paper, we have proposed a bootstrap filter that requires much less computation than the standard bootstrap filter. The decrease in computation is achieved by resampling the posterior in groups of samples instead of one at a time. The simulation results are very encouraging. They show that the fast bootstrap produces comparable results to those of the standard filter. The main advantage of the fast method is the time reduction for model simulation and its suitability for real-time applications. Further work on this technique should be performed to investigate its applicability to other nonlinear models and study its performance in more detail. An important issue is to determine the conditions when the posterior state distribution collapses and remains a set with few truly distinct values.

ACKNOWLEDGMENT

This work has been supported by the National Science Foundation under Award No. MIP-9506743.

EDWARD R. BEADLE
PETAR M. DJURIĆ
Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794-2350

REFERENCES

[1] Brown, R., and Hwang, P. (1992)
Introduction to Random Signals and Applied Kalman Filtering (2nd ed.).
New York: Wiley, 1992.

[2] Gordon, N. J., et al. (1993)
Novel approach to nonlinear/non-Gaussian Bayesian state estimation.
IEE Proceedings, Pt. F, **140**, 2 (1993), 107–113.

[3] Kay, S. M. (1993)
Fundamentals of Statistical Signal Processing.
Englewood Cliffs, NJ: Prentice Hall, 1993.
[4] Kittagawa, G. (1987)
Non-Gaussian state-space modeling of nonstationary time series.
Journal of the American Statistical Association, **82**, 400 (1987), 1032–1063.
[5] Ljung, L., and Söderström, T. (1987)
Theory and Practice of Recursive Identification.
Cambridge, MA: MIT Press, 1987.
[6] Smith, A. F. M., and Gelfand, A. E. (1992)
Bayesian statistics without tears: A sampling-resampling perspective.
The American Statistician, **46**, 2 (1992), 84–88.
[7] Stuart, A., and Ord, K. (1994)
Kendall's Advanced Theory of Statistics.
New York: Wiley, 1994, ch. 5.

Adaptive Beamforming Based on the Conjugate Gradient Algorithm

Adaptive filtering using a version of the conjugate gradient (CG) method which does not involve matrix inversions and is hence computationally attractive has been presented in [1]. The method, which uses a time average over a suitably chosen window in order to generate the required gradients, has been used in [2] for the design of an adaptive beamformer. The algorithm essentially uses time diversity to obtain improved performance. We consider a modification which essentially combines spatial and time diversity to obtain an algorithm for adaptive beamforming which has potential application in cellular communication systems. Simulation results are presented to demonstrate the performance of the algorithm.

I. INTRODUCTION

Adaptive beamforming has been an area of active study for many years [3]. While many of its applications have been in the areas of radar and sonar, recently it is being applied to other areas such as cellular communications, in which microwave transmission is necessary. Although beamforming can enhance such systems, many adaptive beamforming algorithms are usually not fast enough for use in real-time applications. Consequently, current research is concerned with developing faster, more accurate methods for adaptive beamforming. Among the several adaptive techniques available for beamforming, the conjugate gradient (CG) method offers a good compromise between speed of convergence and computational complexity.

The CG algorithm basically is an iterative technique for solving the matrix equation $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a known $M \times M$ matrix, $\mathbf{b}$ is a known $M \times 1$ vector, and $\mathbf{x}$ is unknown. In the classical conjugate gradient algorithm, $\mathbf{b}$ is determined by minimizing the functional

$$f(\mathbf{x}) = \frac{\mathbf{X}^T\mathbf{Ax}}{2} - \mathbf{b}^T\mathbf{x}. \tag{1}$$

If $\mathbf{d}_i$ are vectors that are conjugate with respect to $\mathbf{A}$ the solution $\mathbf{x}^*$ can be expressed as

$$\mathbf{x}^* = \sum_{i=0}^{M-1} \alpha_i \mathbf{d}_i \tag{2}$$

where the α_is are scalar constants that are found using the relation

$$\alpha_i = \frac{\mathbf{d}_i^T\mathbf{b}}{\mathbf{d}_i^T\mathbf{Ad}_i}. \tag{3}$$

The vectors $\mathbf{d}_i$ at iteration k are determined as

$$\mathbf{d}_k = -\mathbf{g}_k + \beta_{k-1}\mathbf{d}_{k-1} \tag{4}$$

where $\mathbf{g}_k$s are negative gradient vectors of the functional $f(\cdot)$ and the β_ks are chosen so as to provide conjugacy. For nonquadratic cost functions, the Fletcher-Reeves method uses the Hessian $F(\mathbf{x})$ of $f(\mathbf{x})$ in place of $\mathbf{A}$ in (3) and a line search to minimize the functional $f(\mathbf{x}_k + \alpha_k\mathbf{d}_k)$ [4].

Computation of the Hessian matrix and the line search can be avoided by using the following algorithm proposed in [4]. The steps in the algorithm are as follows.

Step 1 Starting with an initial choice $\mathbf{x}_0$, evaluate the gradient vector $\mathbf{g}_0 = \delta f'(\mathbf{x}_0)$, $\mathbf{y}_0 = \mathbf{c}_0 - \mathbf{g}_0$, and $\mathbf{p}_0 = \delta f'(\mathbf{y}_0)$.

Step 2 For $k = 0, 1, \ldots, M-1$, compute

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k \tag{5}$$

where

$$\alpha_k = \frac{\mathbf{g}_k^T\mathbf{d}_k}{\mathbf{d}_k^T(\mathbf{g}_k - \mathbf{p}_k)} \tag{6}$$

$$\mathbf{g}_{k+1} = \delta f'(\mathbf{x}_{k+1}) \tag{7}$$

$$\mathbf{y}_{k+1} = \mathbf{x}_{k+1} - \mathbf{g}_{k+1} \tag{8}$$

$$\mathbf{p}_{k+1} = \delta f'(\mathbf{y}_{k+1}). \tag{9}$$

Unless $k = M-1$

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k\mathbf{d}_k \tag{10}$$

where

$$\beta_k = \frac{\mathbf{g}_{k+1}^T\mathbf{g}_{k+1}}{\mathbf{g}_k^T\mathbf{g}_k}. \tag{11}$$

Manuscript received November 6, 1995; revised March 25, 1996.

IEEE Log No. T-AES/33/1/01109.