TELKOMNIKA, Vol.14, No.4, December 2016, pp. 1321~1328
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v14i4.4196

Particle Filtering Approach for GNSS RAIM and FPGA Implementation

Ershen Wang*[1], Fuxia Yang[2], Gang Tong[3], Pingping Qu[4], Tao Pang[5]
[1,2,4,5]Shenyang Aerospace University, Shenyang 110136, China
[3]Liaoning General Aviation Key Laboratory, Shenyang 110136, China
[1]Key Laboratory of Intelligent Waterway Transport of Ministry of Transport, Dalian 116026, China
*Corresponding author, e-mail: wes2016@126.com

Abstract

The integrity monitoring system, as an integral part of aviation navigation system for global navigation satellite system (GNSS), should detect and isolated Failures or faults caused by system failures to maintain the integrity of the GNSS. The pseudorange residual noise of navigation satellites does not completely follow the Gaussian distribution, the performance of traditional filtering algorithms (such as the Kalman filtering) may be reduced due to non-Gaussian noise. The particle filter algorithm has great advantage to dealing with the nonlinear and non-Gaussian system. in this paper, the particle filter algorithm is applied to GNSS receiver autonomous integrity monitoring(RAIM) to detect the fault of navigation satellite. Firstly, Log likelihood ratio (LLR) testing is established; and then, the consistency between the state estimation of the main particle filter and the auxiliary particle filter is checked to determine whether the navigation satellite has failed; finally, the novel RAIM algorithm is undertaken by field programmable gate array (FPGA), the modules of the proposed RAIM algorithm is implemented. The effectiveness of the proposed approach is illustrated in a problem of GPS (Global Positioning System) autonomous integrity monitoring system, the algorithm and its implementation can be embeded in GNSS receiver.

Keywords: *Global Navigation Satellite System (GNSS), Receiver Autonomous Integrity Monitoring (RAIM), particle filter, Global Positioning System (GPS), Field Programmable Gate Array (FPGA)*

1. Introduction

Fault detection (FD) is usually detected by real-time sensor measurements. In various fields, real-time detection of faults plays an important role in an estimated system. For a stochastic system, the fault detection method is mostly based on a linear system, assuming that noise and interference obey a Gaussian distribution. In this regard, the Kalman filter (KF) is often chosen as the state estimator [1-4]. State estimation of nonlinear systems under non-Gaussian noise is a difficult problem to solve. Usually, It is difficult to analyze and express the optimal solution. Sub-optimal solutions can be approximated to some form, such as Extended Kalman Filtering (EKF). Currently, Particle filtering is widely concerned due to its ability to handle nonlinear system or measurement noise with any probability distribution [5].

Fault detection and exclusion play a vital role in practical applications, especially in Global Navigation Satellite Systems (GNSS) such as aircraft and missile navigation. Failure to detect and exclude failures that may cause risks to the accuracy and integrity of GNSS will result in a continuous risk of navigation systems and significant losses. Fault detection and exclusion mainly considers the probability of satellites sending incorrect navigation signals to users. Integrity means that the system can promptly provide users with warnings about when they should not be used. The response time of the GPS control segment usually exceeds 15 minutes, so it is necessary to apply effective RAIM in advance to cope with sporadic satellite pseudorange error[6-10]. Actually, GNSS measurement errors do not fully follow the Gaussian distribution. Conventional snapshot RAIM are difficult to detect faults, and various filtering algorithms such as the KF presume that the measurement error and noise follow a Gaussian distribution. If the assumptions are incorrect, the performance of algorithm will also be degraded. The particle filter algorithm is suitable for nonlinear system or observation noise with

Received June 17, 2016; Revised November 7, 2016; Accepted November 23, 2016

any probability distribution and can be used to detect the integrity of GNSS [11-15].

In this paper, the principle of particle filter is introdused at first. The LLR method based on particle filter and the common methods of fault detection and isolation are introduced. Then, the GPS system's system and measurement equations are described. The next section, the GPS RAIM system and its availability are given by numerical simulation. Finally, the RAIM algorithm based on FPGA is given, and describe the conclusion.

2. Particle Filter Algorithm

In this section, a brief explanation of a PF will be given. PF, also known as the sequential Monte Carlo method, is a statistical filtering method based on the Monte Carlo method and recursive Bayesian estimation. It approximates the required probability density function (PDF) by swarms of points in the state space. These points are called particles. PF is a nonlinear filter method for nonlinear and non-Gaussian system, which is that choosing swarms of limited weighted points to form the discrete random measure to describe the posterior probability distribution of the state PF is easily implemented in different nonlinear system because it is independent of the system model [16-19].

The dynamic state space model is used in the PF algorithm, and the state model is employed to describe the dynamic system model. PF algorithm was first proposed by Gordon and Salmond in 1993, and was known as SIR (Sampling importance resampling). Its dynamic state space model is as follows:

The state function:

$$X_k = f(X_{k-1}, \mu_{k-1}) \tag{1}$$

The measurement function:

$$Y_k = g(X_k, V_k) \tag{2}$$

Where X_k is a state vector at instant *k*, Y_k is measurement vector at instant *k*, μ_k and V_k are process noise vector and measurement noise, respectively, both are random errors and independent of each other. $f(\cdot,\cdot)$ is state transition function. $g(\cdot,\cdot)$ is measurement function.

The steps of standard PF algorithm can be summarized as follows:

(1) Initialization. calculating the prior probability $p(x_0)$, and generates the initial set of N samples/particles, and that is $\{x_0^i\}_{i=1}^{N_s}$. And the weight of each particle is $1/N_s$.

(2) State prediction: the weight of sampling N particles is $\{w_{k-1}(i): i = 1, 2, ..., N\}$ from the PDF of system noise w_{k-1}. Using these samples to generate new swarm of $\{x_{k|k-1}(i): i = 1, 2, ..., N\}$, which is approximated the predicted PDF $p(x_k \mid Y_{k-1})$.

Where,

$$x_{k|k-1}(i) = f_{k-1}(x_{k-1}(i), w_{k-1}(i)) \tag{3}$$

(3) Sate update. Assigning each particle of $x_{k|k-1}(i)$ the weight of $w_k(i)$ for $i = 1, 2, ..., N$, after the measurement of y_k is attained. The weights are calculated as follow:

$$w_k(i) = \frac{p(y_k \mid x_{k|k-1}(i))}{\sum_{j=1}^{N} p(y_k \mid x_{k|k-1}(i))} \tag{4}$$

Then, the normalized weight of the i th particle is calculated as follow:

$$w_k^i = w_k^i / \sum_{i=1}^{N_s} w_k^i \tag{5}$$

The least square estimate of the unknown parameter x can be shown to be:

$$\hat{x}_k = \sum_{i=1}^{N_s} w_k^i x_k^i \tag{6}$$

(4) Resampling. after a few iterations, most of the particle's weight will become very small until it becomes 0, and only a small amount of particle weight will become particularly large, resulting in particle degradation. We rely on the method of resampling. After resampling, the new set of particles is:

$$\{x_{0:k}^{i^*}, i = 0,1\text{L } N_s\}$$

(5) Estimation. Using the state function $f(\cdot,\cdot)$ to estimate the next state parameter x_{k+1}^i.

(6) Return to step 2, then $k = k+1$. And the steps of prediction, updation and resampling are executed and the iteration recursively applied at each time k .

3. RAIM Based on Particle Filter Algorithm

In life-related navigation application services, integrity issues are very important. Once an error occurs, the consequences can be very serious. Therefore, Faults must be detected as quickly as possible. In this paper, we design the FDI system for GPS signal integrity monitoring. RAIM can use the redundant information of satellites to detect and isolate the fault satellites.

GPS state and Observation Equation can be shown as follows [20].

$$X_k = F_{k-1} X_{k-1} + \mu_{k-1} \tag{7}$$

$$\rho^i(\text{k}) = R^i(\text{k}) + c * \Delta\delta^i + T^i(\text{k}) + I^i(\text{k}) + E^i(\text{k}) + v^i(\text{k}) \tag{8}$$

The RAIM based on PF algorithm is modularly designed by field programmable gate array (FPGA). FPGA can perform larger scale calculation at the same time, and can execute the algorithm in parallel, which has a great advantage for computing swarms of particles in PF algorithm for shorter time. It has the limitation for FPGA to process floating point numbers, so the soft core processer of Microblaze is employed to FPGA. The RAIM algorithm is decomposed into two parts, the logical part and the soft-core part. The soft-core for the algorithm also consists of two parts, the part of weight calculation and the part of log-likelihood ratio (LLR) calculation. The reason is that the module of weight calculation has exponential operation, and the module of LLR contains logarithm operation. It could be easier to implement by using the soft core on FPGA.The logical part for the algorithm includes five parts: the random noise generation module, the particle state calculation module, the resample module, the fault detection and the fault isolation modules, which include lots of matrix and summation computation. It could improve the computing speed of the program on FPGA. After meeting the precision, the floating point numbers should be transformed into fixed-point numbers for FPGA to give full play in computation.

The structure of the algorithm on FPGA is as follows.

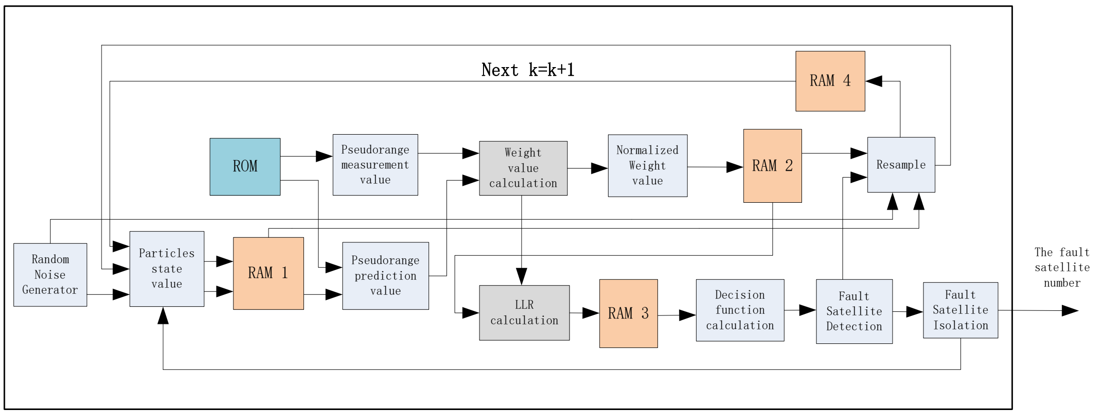

Next k=k+1
RAM 4
ROM
Pseudorange measurement value
Weight value calculation
Normalized Weight value
RAM 2
Resample
Random Noise Generator
Particles state value
RAM 1
Pseudorange prediction value
LLR calculation
RAM 3
Decision function calculation
Fault Satellite Detection
Fault Satellite Isolation
The fault satellite number

Figure 1. The structure of the algorithm on FPGA

The RAIM algorithm flow is described as follows:

(1) Generate the random noise. The random numbers are generated in the random noise generator and will be used as initial particles in the module of particle state value and as random numbers in the resample module.

(2) Calculate out the particle state. The initial particles are sent into the transition function in the particle state value module to get the particle prediction value that will be stored into RAM1 module.

(3) Predict the pseudorange value. The pseudorange measurement value is gotten from ROM module as the true value in the module of pseudorange measurement. The pseudorange prediction module will read the particle state prediction value from RAM1 module and the position coordinates and the clock error of the i-th satellite from ROM module. And the pseudorange function will calculate pseudorange prediction of the i-th satellite.

(4) Calculate the weight. There will be in two ways after the weight computation in the particle weight calculation module. One way is to update the particles, and the other one will conduct the fault detection, and the two can be parallelly executed, which could save the time of fault detection.

(5) Particles update. The process of particle update will enter the resampling module, which is to eliminate the particles that have small weight and to concentrate on particles with large weight to store into RAM4 module.

(6) Fault detection and isolation. In the process of fault detection, it firstly calculates the LLR in the soft core and the LLR function is stored into RAM3 module. Then the next module detects if the satellites have fault. If there is no fault, the resampling module updates the particles directly at next time $k = k+1$. If the fault has been detected, firstly the alarm will be issued, and the module of fault satellite isolation will determine the number of the fault satellite as the output result and then isolate the fault satellite in the navigation solution. After then the resampling module updates the particles to the next time k=k+1.Then go back to step 2.

4. Implement of RAIM Algorithm Based on FPGA

This part will show several modules that have been implemented of the RAIM algorithm based on FPGA by Verilog.

4.1. Random Noise Generator

The random noise generator is used to provide the initial particles and be regarded as the reference numbers in the resampling module. Hence the generator has two output. As the non-Gaussian noise is not easy to implement in simulation on FPGA, so the noise can be represented by the random numbers in the RAIM algorithm. To simply generate random numbers, the linear feedback shift register (LFSR) is used in this module. The LFSR consists of two parts. One is the shift register, and the another is the feedback net.

The stream of data produced by the shift register is determined by the current state. And the output sequence is periodical because of the finite number of possible states in the register. The maximum cycle of the output sequence is 2^{n-1}. However, an LFSR with a well-selected feedback function can generate a randomly displayed bit sequence with a long period. In this paper, the n is set the number 10, and we can design m pseudo-random number generators, and output the result of the summation of generators. The m in this module is 8, and the initial value, which is called the seed, is different from each another. Figure 2 shows the simulation of the random noise generator.

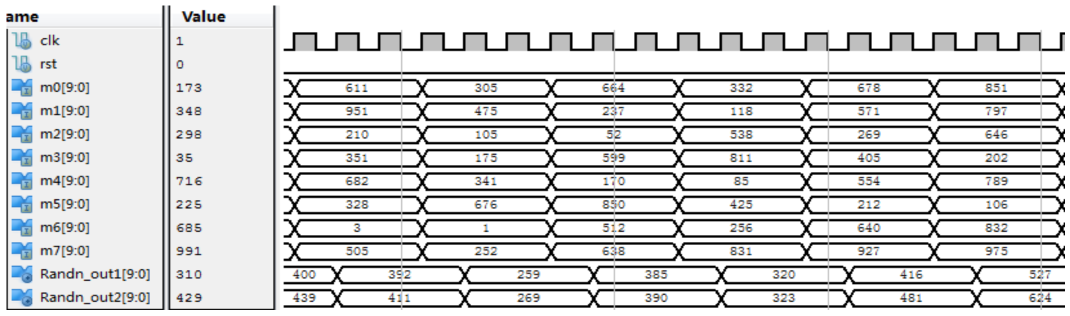


Figure 2. The simulation of random noise generator

As illustrated in Figure 2, the Randn_out1 and Randn_out2 give the two output ports respectively. The rst is the reset signal and the clk is the clock signal. The m0, m1…,m7 present the 8 pseudo-random number generators, each of which generates the pseudo-random numbers. Importing the two channels generated random numbers to Matlab program and then calculate the distribution of each channel random numbers.

4.2. Fault Detection Module

In this module, it should judge whether the fault detection function β_k exceeds detection the threshold τ. If $\beta_k > \tau$, it means that a fault has occurred and the module could get the fault occurrence time t. If $\beta_k < \tau$, it means there is no fault in GPS navigation system. The simulation result of this module is shown in Figure 3.

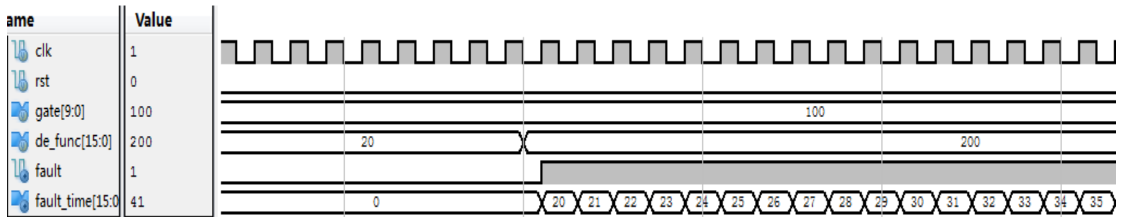


Figure 3. The simulation result of fault detection module

In simulation, the threshold is set τ=100, that is, the internal signal gate is valued 100. The rst is the reset signal, and the clk is the clock signal. The de_func is an input port of the value of fault detection function β_k to compare with the threshold. The port of fault is the enable signal port, which means that if the fault is detected, then the output of fault siganal is equal 1. The fault_time port outputs the time of fault occurrence. From Figure 3, it is shown that the there is no fault occurred until the time is 20, when the de_func is 200 at the time of 20 and exceeds the τ. Then, the output of fault signal is set to 1 in order to indicate that the system has detected the fault.

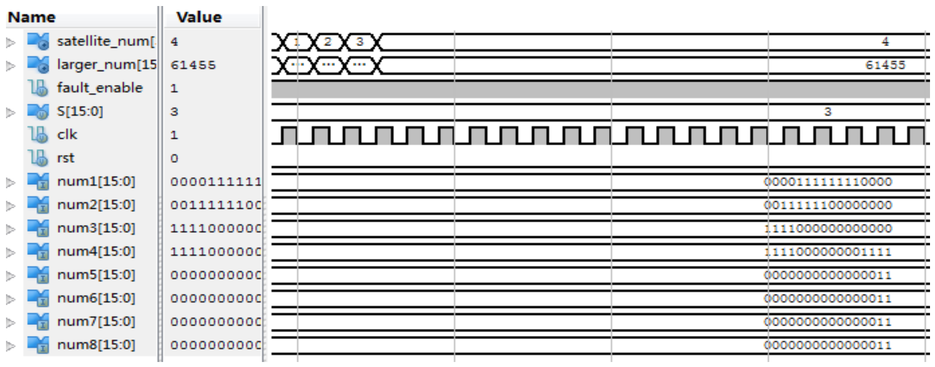
4.3. Fault Isolation Module

The fault satellite could be found in this module. The fault isolation function can be expressed as:

$$satnum = \arg\max_{1\le q\le Q} S(q)_{t_a}^{k} \tag{9}$$

Where, Q=m+1, and m is the number of visible satellites, $S_{t_a}^{k}$ is the cumulative LLR of the main particle filter and the auxiliaries particle filter, where ta is a time where the satellites' fault is detected. And $satnum$ is the number of the fault satellite. According to the function, the fault isolation is achieved by getting the maximum value of all cumulative LLRs.

The simulation of the fault isolation module is shown in Figure 4.



Name	Value
satellite_num[	4
larger_num[15	61455
fault_enable	1
S[15:0]	3
clk	1
rst	0
num1[15:0]	0000111111
num2[15:0]	0011111100
num3[15:0]	1111000000
num4[15:0]	1111000000
num5[15:0]	0000000000
num6[15:0]	0000000000
num7[15:0]	0000000000
num8[15:0]	0000000000

Figure 4. The simulation of the fault isolation module

As illustrated in Figure 4, S is the input of cumulative LLR function. It can be shown as followed:

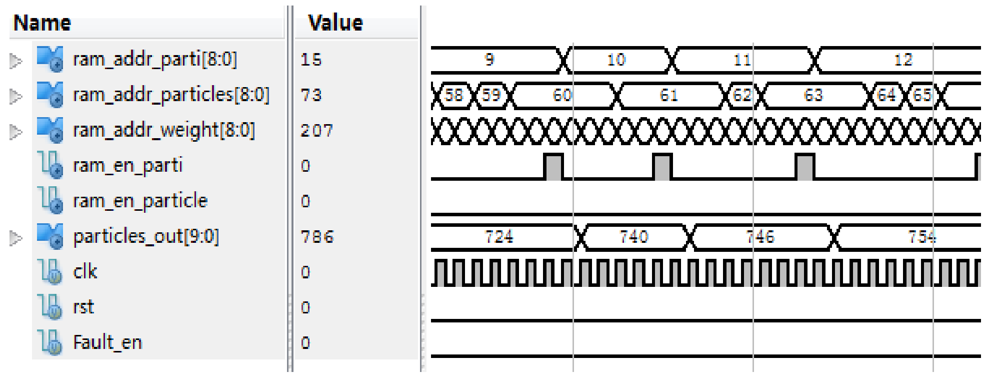
$$S_j^k(q) = \sum_{i=j}^{k} \ln \frac{\frac{1}{N}\sum_{i=1}^{N} \tilde{w}_k^{q}(i)}{\frac{1}{N}\sum_{i=1}^{N} \tilde{w}_k^{main}(i)} \tag{10}$$

Where, a new data would be imported in per clock cycle. In equation (10), $\tilde{w}_k^{q}$ is the weight of the auxiliaries particle filter, and $\tilde{w}_k^{main}$ is the weight of the main particle filter, N is the number of particles. The satellite_num is the port to output the number of the fault satellite. The larger_num is the port to output the maximum value of the cumulative LLR function, which is proportional to the number of the fault satellite. num1, num2, …, num8 are the cumulative LLR function of eight auxiliary PFs respectively, the rst is the reset signal and the clk is the clock signal. As is shown in Figure 4, this module is effective and can work steadily, and the output of the fault_enable signal is equal to 1, which means the fault has been detected, and the number of the fault satellite is 4, which should be isolated during the following navigation solution.

4.4. The Resampling Module

In the particle filter algorithm, the variance of the importance weights will increase after a few iterations. However, the weight of most particles is getting smaller and smaller. This degeneracy means that updating particles whose are approximately to $p(x_k | Y_{1:k})$ is almost 0 cost a lot of computational effort. Only a few of particles' weight is extremely large.

The purpose of resampling is to optimize the particles with their weight. It is an effective way to decrease the invalid samples and increase the valid particles by abandoning the small weight particles and keeping the large weight ones. In this algorithm, the system resampling method is used. The number of the resampling particles is N, and the algorithm includes two cycles, the inner loop and the outer loop, both cycles' number are N. At the beginning of resampling, it will choose a random number $g \in (0,1)$ from the random noise generator, and make the summation of weights, the initialization of Sum is equal to zero. And then, it will enter the inner loop. The result of each inner cycle is that the corresponding summation of particle weight will accumulate. The value of Sum is compared with g in every inner loop. If $Sum < g$, it shows that the weight of current particle is small, and the particle should be abandoned. After then, according to the iteration times of inner loop, it should determine whether to continue staying in the inner loop or to be in next outer loop by comparing i with N. If $Sum \geq g$, it indicates that the weight of current particle is large, and the particle should be taken out. After then, according to the iteration times of outer loop, it should determine whether to be in next cycle or to complete the system resampling by comparing j with N. The particles after resampling should be stored and the weight of all stored particles is $1/N$. The simulation result of this module can be shown as follows in Figure 5.



Name	Value
ram_addr_parti[8:0]	15
ram_addr_particles[8:0]	73
ram_addr_weight[8:0]	207
ram_en_parti	0
ram_en_particle	0
particles_out[9:0]	786
clk	0
rst	0
Fault_en	0

Figure 5. The simulation result of the resampling module

In the simulation result, the signal of fault_en is the input port of enable signal. The signal of particles_out is the output of the particles being resampled, the signal of ram_en_parti is the enable signal of RAM4 module to store the particles from the signal port of particles_out, and the ram_addr_parti is the address bus of RAM4 module, the ram_en_particle is the enable signal of RAM1 moduole that stores the particles to be resampled, and also the ram_addr_particles is the address bus of RAM1 module, the ram_addr_weight is the address bus of RAM2 module that stores the weight that corresponds the particles in RAM1 module. The resampling particles from the signal port of particles_out enter next cycle.

5. Conclusion

The RAIM algorithm for GNSS is based on PF, and the fault detection is accomplished by the log-likelihood ratio test. In addition, the paper implements some modules of the proposed RAIM algorithm by FPGA, and the main modules of the fault detection, the fault isolation and the resampling are simulated and implemented. Then, the proposed algorithm implemented on FPGA is summarized as six steps including generating the random noise, calculating the particle state, predicting the pseudorange, calculating the weight, updating the particles, Detect and isolate faults. The simulation results demonstrate that the satellite failure of GNSS is successfully detected and isolated. The proposed method has good reference value to the study BeiDou satellite navigation RAIM and improve the positioning reliability of BeiDou receiver. The RAIM algorithm implemented on FPGA for fault detection and isolation will be optimized to improve the computation efficiency of future work.

Acknowledgements

This study is supported by the National Natural Science Foundation of China (Nos. 61571309 and 61101161),the Liaoning BaiQianWan Talents Program and the Open Program of Key Laboratory of Intelligent Waterway Transport of Ministry of Transport and the Fundamental Research Funds for the Central Universities (No.3132016317).

References

[1] A Abdullah, M Zribi. Sensor-fault-tolerant control for a class of linear parameter varying systems with practical examples. *IEEE Trans. Ind. Electron.* 2013; 60(11): 5239-5251.
[2] S Chen. Kalman filter for robot vision: A survey. *IEEE Trans. Ind. Electron.* 2012; 59(11): 4409-4420.
[3] MB Loiola, RR Lopes, JMT Romano. Modified Kalman filters for channel estimation in orthogonal space-time coded systems. *IEEE Trans. Signal Process.* 2012; 60(1): 533-538.
[4] S Roshany-Yamchi. Kalman filter-based distributed predictive control of large-scale multi-rate systems: Application to power networks. *IEEE Trans. Control Syst. Technol.* 2013; 21(1): 27-39.
[5] Zhiyu Zhu. Particle filtering algorithm and application. Beijing: Science Press. 2010.
[6] Brown RG. A Baseline GPS RAIM Scheme and a Note on Equivalence of Three RAIM Methods. *Journal of the Institute of Navigation*. 1992; 39(3): 301-311.
[7] RM Kalafus. Receiver Autonomous Integrity Monitoring of GPS. *Project Memorandum DOTTSC-FAA-FA-736-1, DOT Transportation System Center.* Cambridge, MA. 1987.
[8] Wang Ershen, Ming Cai, Tao Pang. *A Simple and Effective GPS Receiver Autonomous Integrity Monitoring and Fault Isolation Approach.* In Control Engineering and Communication Technology (ICCECT), 2012 International Conference on. 2012: 657-660.
[9] MS Amin, MI Reaz, SS Nasir. Integrated Vehicle Accident Detection and LocationSystem. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2014;12(1): 73-78.
[10] Shaojun Feng, Shenghai Wang, Washington Ochieng. *A Core Constellation Based Multiple-GNSS Positioning and Integrity Monitoring Algorithm.* In Proc. 27th international technical meeting of the satellite division of the institute of navigation (ION GNSS). Tampa. 2014: 307-314.
[11] E Wang, Q Zhang, T Pang, P Qu, X Li. Research on Particle Filter Based on Neural Network for Receiver Autonomous Integrity Monitoring. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2016; 14(1): 203-210.
[12] Antonio Angrisano, Salvatore Gaglione, Ciro Gioia. *RAIM algorithms for aided GNSS in urban scenario*. Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS). 2012: 1-9.
[13] Mathieu Joerger, Fang-Cheng Chan, Steven Langel, Boris Pervan. *RAIM Detector and Estimator Design to Minimize the Integrity Risk*. In Proc. 25th international technical meeting of the satellite division of the institute of navigation (ION GNSS). Nashville. 2012: 2785-2870.
[14] Cao, Kejing, Yangfeng Hu. *Research on Improved RAIM Algorithm Based on Parity Vector Method*. Information Technology and Applications (ITA), 2013 International Conference on. 2013: 221-224.
[15] Rosihan R, Indriyatmoko A, Chun S, Won DH, Lee YJ, Kang TS, Kim J, Jun HS. *Particle Filtering Approach To Fault Detection and Isolation for GPS Integrity Monitoring.* In Proc. 27th international technical meeting of the satellite division of the institute of navigation (ION GNSS). Fort Worth. 2006: 873-881.
[16] F Gustafsson. Particle filters for positioning, navigation, tracking. *IEEE Trans. Signal Process.* 2002; 50(2): 425-437.
[17] Yin Sha, Xinen Zhu. Intelligent particle filter and its application on fault detection of nonlinear system. *IEEE Trans. Ind. Electron.* 2015; 62(6).
[18] A Shenoy, J Prakash, V Prasad, S Shah, K McAuley. Practical issues in state estimation using particle filters: Case studies with polymer reactors. *J. Process Control*. 2013; 23(2): 120-131.
[19] S Das, A Kale, N Vaswani. Particle filter with a mode tracker for visual tracking across illumination changes. *IEEE Trans. Image Process.* 2012; 21(4): 2340-2346.
[20] Haiying Liu, Huinan Wang, Zhiming Chen. Principles and Applications of Satellite Navigation. Beijing: National Defense Industry Press. 2013; 9: 192-211.