International Journal of Computer Engineering in Research Trends
Multidisciplinary, Open Access, Peer-Reviewed and fully refereed
Research Paper Volume-5, Issue-4 ,2018 Regular Edition E-ISSN: 2349-7084

Speed Enhancement of QFT Bound Generation using GPU

Pallavi d Pawar, Mukesh D Patil, Vishwesh A. Vyawahare

Dept. of Electronics & Telecom, Electronics
Ramrao Adik Institute of Technology, Nerul
pawarpallavi216@gmail.com
vishwesh.vyawahare@rait.ac.in
Ramrao Adik Institute of Technology, Nerul
mukesh.rait@gmail.com.

Available online at: http://www.ijcert.org

Received: 18/April /2018, Revised: 19/April /2018, Accepted: 24/April /2018, Published: 27/April/2018

Abstract: In control theory, Quantitative Feedback Theory (QFT) developed by Issac Horowitz, has gained a lot of popularity. Many researchers have proposed a method to generate the bounds that from the literature it is observed that generation of bound takes a lot of time for online design of the controller. It is necessary to speed up the computation of bound generation. This paper exhibits the parallel computing power of the GPU (Graphics Processing Unit) in the area of QFT. In this paper, GPU based approach is proposed to speed up the computation of stability bound. By using MATLAB parallel computing toolboxes, GPU computational power can be easily accessed with the minimum knowledge of GPU architecture, MATLAB code can be executed on the GPU. In order to achieve faster execution of QFT bound generation, NVIDIA GPU with the support of MATLAB parallel computing toolbox is used in this work. Performance comparison of the algorithm for sequential implementation on CPU and parallel implementation on GPU is carried out. This work analyzes the relative performance of GPU vs CPU. In this paper, GPU based approach proposed for significant speedup in the computation of bound using QFT and it is observed that GPU provides speedup two to three times as compared to the CPU.

Keywords: Quantitative Feedback Theory, QFT bounds, QFT Toolbox, Parallel computing toolbox (PCT), Graphics Processing Unit

1. Introduction

In control theory, Isaac Horwitz has developed the Quantitative Feedback theory in 1963. QFT is a frequency domain method. Horwitz observed that the feedback is required only if there is plant parameter uncertainty present in the feedback either because of dynamic or external disturbances. QFT is highly transparent method because the stability and performance criteria are always visible during the designing process. QFT utilizes Nicholas chart to achieve a desired robust design [1-2].

QFT is a graphical loop shaping procedure used for the control design of either Single Input Single Output (SISO) or Multiple Input Multiple Output (MIMO) uncertain systems including the nonlinear and time varying cases [3].

There are different methods for the generation of QFT bounds. Dark colored and Petersen [4], Fialho et al. [5], and Zhao and Jayasuriya [6] created calculations for producing QFT bound for interim plants. Geometry based methods for bound generation is best suitable for plants having wide uncertainty structures [7 9].

© 2018, IJCERT All Rights Reserved https://doi.org/10.22362/ijcert/2018/v5/i4/v5i406

The algorithms developed by Chait and Yaniv [1], Borghesani [10], and Zheng and Rodrigues, Chai, and Hollot

[11] solve quadratic inequalities generated from plant template points to compute the bound at a fixed frequency and controller phase [10-12]. Then, the whole QFT bound at the frequency is computed by a controller phase sweeping procedure. It is noted that the QFT bounds computed from plant template points may be too coarse to meet the given robustness specifications [11-12]. Although one can increase the number of points for approximating the plant templates to increase the accuracy of the computed QFT bounds, the relation between the numbers of the plant template points and the accuracy of the computed QFT bounds is obscure. Therefore, it can lead to an unfavorable trade-off between the computational burden and the accuracy of the computed QFT bounds. To overcome the problem, Nataraj and Sardar [13] and Nataraj [14] developed algorithms which can generate inner and outer enclosures of exact QFT bounds from interval plant templates generated by the algorithm developed by Nataraj and Sardar [15].

Gutman et al. [16] and Yang, Shih-Feng [17] developed a recursive grid algorithm to compute QFT bounds. The algorithm first grids the Nichols chart with a coarse grid size. Then, a coarse boundary of the QFT bound is generated by computing, at the grid points, the values of a function associated with the controller, the desired frequency specification, and the plant template.

Bailey and Hui [18] and Donald J. Ballance et al. [19] considered the case where the uncertain parameters in the numerator and denominator are independent and affine. When the uncertain parameters in the numerator and denominator are dependent and affine, Fu [20], Bartlett [21] and Donald J. Ballance et al. [19] claim the boundary of the templates are only generated by the edges of parameter box.

Accomplishing an effective robust outlined includes various strides: particular of the control issue, plant demonstrates information, usage of the theoretical plan, reproduction, and system under real working conditions. One of the principle presumptions of QFT is that the uncertainty in the plant that is being controlled can be evaluated somehow. This is regularly accomplished by the utilization of a transfer function with parameters that differ somehow.

Figure 1 shows the two degree of freedom feedback system. In which T.F P(s) is a set which represents uncertainty of plant. G(s) is a compensator and F(s) is filter which is to be synthesized for making system robustly stable [11].

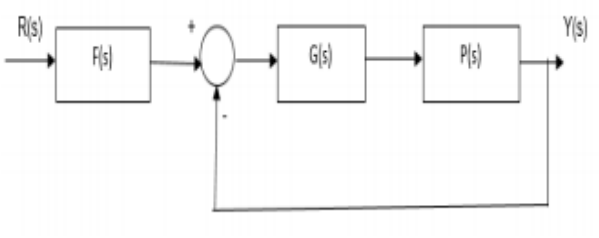

Fig. 1 Two Degree of Freedom [9]

The open loop transmission function is L(s) = G(s) P(s) and nominal loop transfer function is $Lo\ (s) = G(s)P_o(s)$ Close-loop performance problem are given as [9, 22]

1. Robust stability margin

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leq Ws$$

2. Robust tracking performance

$$|T_L(j\omega)| \leq \left|\frac{F(j\omega)L(j\omega)}{1+L(j\omega)}\right| \leq |T_U(j\omega)|$$

3. Robust input disturbance rejection performance

$$\left|\frac{P(j\omega)}{1+L(j\omega)}\right| \leq |Wd_i(\omega)|$$

4. Output disturbance rejection performance

$$\left|\frac{1}{1+L(j\omega)}\right| \leq |Wd_o(\omega)|$$

Where Ws is the stability margin specification, $TL(j\omega)$ and $TU(j\omega)$ are the lower and upper tracking performance specifications, while di and do are the input and output rejection performance specifications [23]. In QFT method, the design of controller online. Time spent in the computation of such complex bounds is financially expensive. There is need to speed up the QFT bound generation process which will improve the speed of controller design. Many strategies are being pursued to keep the computation time within reasonable but at the end of the day, the size of problems which can be tackled is ultimately limited by the capabilities of the available hardware. Using a faster machine reduces the computation time, but since CPU clock-speed development has hit a wall, only parallel computing promises increased performance in the future. Parallel computing requires dedicated parallel architectures. An example is a Multiple-Instruction Multiple Data (MIMD) cluster. Such clusters are very expensive and in general only large institutions can afford them. An interesting alternative parallel architecture is the

© 2018, IJCERT All Rights Reserved

Graphical Processing Unit (GPU). In this approach, parallel processing power is available to individual users on standalone machines, or as special nodes in large compute clusters. In this work, we are attempting to speedup bound generation process by using GPU. In this work, the GPU computation will be carried out by using Parallel Computing Toolbox (PTC) of MATLAB. This paper outline as follows: Section II contains information about GPU computing using MATLAB. Section III introduces the overview of GPU toolboxes for MATLAB. Section IV mainly focus on the whole designing process of QFT. In the section, V gives the information about CPU and GPU information and we present the result for Template and stability bound. Section VI concludes the work.

2. GPU Computing With Mat lab

A Graphics Processing Unit (GPU) is an electronic equipment circuit created to rapidly process and change memory to quicken the computational speed of the machine and provide a speed up for calculation of the huge amount of input data. GPUs with their exceedingly parallel structure give considerably speedier handling of extensive information pieces and applications, for example, picture and video preparing, in contrast with ordinary CPUs. A Graphics Processing Unit (GPU) is a unique processor, enhanced for various and quicker figurings [24].

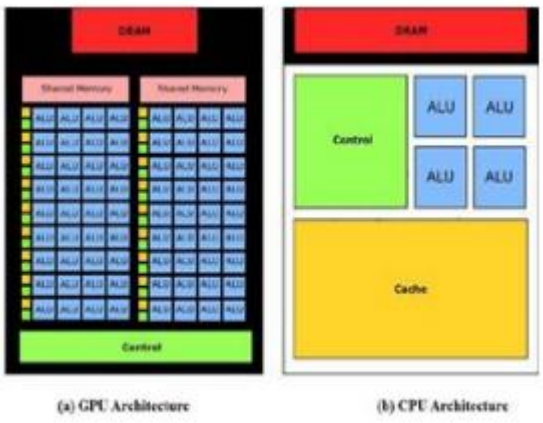

Fig. 2 CPU Vs GPU [25]

A Central Processing Unit (CPU) is a generally helpful processor it can on a crucial level do any figuring, yet not by any means in a perfect shape for any given calculation. GPU has large number of cores to process parallel workloads beneficially. In this work, we are utilizing the mixture GPU CPU system in which CPU will be treated as primary processor and the GPU is known as the Device secondary processor. When we run an application code on the machine. The CPU as fundamental processor choose which computational-intensive some portion of the program. This part really takes a vast measure of computational time and because of which its data set can be executed in parallel which supports the NVIDIA GPU. MATLAB has inbuilt Run and Time command with the assistance of that we come to comprehend which part of the application code is expending additional time in this way, that part can be parallelized. This part is then exchanged to the GPU for quickening parallel processing, while whatever is left of the program is executed successively by the CPU memory, which will be accessible in the MATLAB workspace and for further computations. In the GPU layout, where instructions are carried out in parallel, the largest amount of surface area is spent on arithmetic units (ALUs). This is the reason a GPU has a high computational throughput at the cost of a higher latency. This fundamental difference is reflected in the Chip designs as we can see from. In the initial days, the GPUs faster and efficient parallel processing ability was mainly used for 3D video game rendering. Presently, those some abilities have found large use in fields like oil and gas exploration, mathematical and scientific research, control designs, modelling of financial and stock markets.

3. Overview for GPU Toolboxes for Mat lab

A. Jacket toolbox The benefits of this toolbox are that capacities accessible can be utilized like some other MATLAB work. A capacity to be needed keep running on GPU, the variable and additional information should be as a jacket characterized GPU information sort. At the point when GPU variable is utilized, the estimation is done on the GPU straightforwardly [24].

B. GPUMat GP-you group created the GPUmat toolbox. This toolbox is completely free under GNU permit, and anybody can use it. The GP-you Gathering offers support on creating GPyou construct programming with respect to the request, and also to modify our effectively existing product in function of the client demands [24].

C. Parallel Computing Toolbox (PCT) Parallel Computing toolbox is MATLAB own toolbox. In the scientific calculation, MATLAB is visualization software which contains signal processing, numerical analysis, graphical display and matrix operation. The main advantages of parallel computing are:

- A significant amount of time and money are saved.
- Larger problems can be solved.
- Achievement of parallelism.
- Usage of non-local resources.

Parallel computing toolbox helps us to solve MATLAB computational engines for parallel computing, associated with the cores in a multi-core machine. It accelerates the code by running on multiple MATLAB workers, using distributed arrays and map reduce. Without changing the

© 2018, IJCERT All Rights Reserved

code, we can run a similar application on a PC bunch or a grid computing service [24].

4. Step for Bound Generation Using QFT

The design procedure to design controller using QFT [4]:

- Converting time domain specification into the frequency domain specification.
- Generating plant sets.
- Obtain plant template.
- Select nominal plant transfer function.
- Obtain the stability bound.
- Determine the input disturbance bounds.
- Determine the output disturbance bounds.
- Obtain tracking bounds.
- Grouping of the bounds.
- Intersection of the bounds.
- Loop shaping.
- Pre-filter Designing.

Consider the Transfer function

$$P(s) = \frac{K}{s(s+a)}, K = 0{:}0.5{:}10, a = 0{:}0.5{:}10$$

Converting time domain specification into the frequency domain specification

- Upper tracking Model

$$T_{RU} = \frac{19.753}{s + 2 \pm j3.969}$$

- Lower tracking model

$$T_{RL} = \frac{120}{(s+3)(s+4)(s+10)}$$

Plant Template: Templates are nothing but the magnitude verses phase plot of plants set P(s). Figure 3 shows the plant template for different frequencies. Every point speaks about the frequency response of single plant out of the complete set of plant [26].

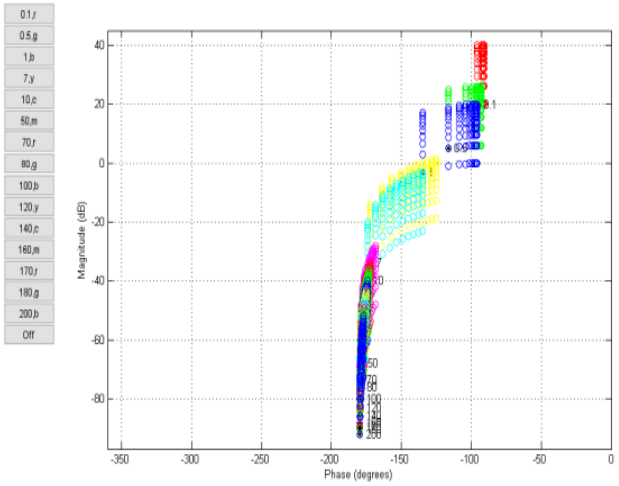

Fig. 2 Template

Stability bounds: Given the plant template, QFT converts the closed loop magnitude specification into magnitude and phase constrains. These constrain are call QFT bounds. The bounds are computed by moving each plant template around the Nichols chart to fulfil the control specification. At each point, the nominal plant is marked on the Nichols chart [28] and Fig. 3 shows the stability bound.

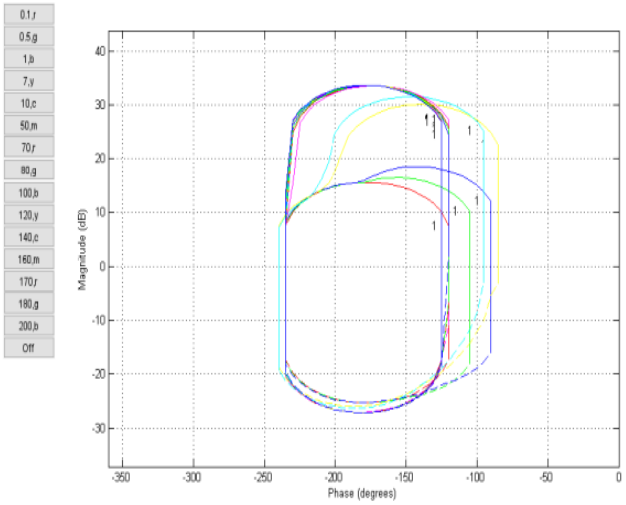

Fig. 3 Stability Bound

5. Design Methodology

For the GPU implementation bound generation using Quantitative feedback theory using MATLAB, the following steps were followed [27]. The CPU and GPU specification are listed in table I.

© 2018, IJCERT All Rights Reserved

Step 1: Sequential code is written to generate the bounds on CPU.
For eg. : Write a sequential code in MATLAB to square every element of the vector.
$a = [2\ 5\ 7\ 8];$
$for\ i = 1{:}4;$
$a(i) = a(i)^2{:}$
Step 2: Writing vectorized code on CPU.
For eg.: Vectorized code for above example:
$a = [2\ 5\ 7\ 8];$
$a.^2\ 2;$
This code will work on all the four elements at the same time. Thus, as a result the vectorized codes are faster and less time to consume than sequential codes.

Step 3: After that vectorized code is transferred on GPU.
For eg.: GPU code for above example
$a = gpuArray([2\ 5\ 7\ 8]);$
$a = a.^2\ 2;$

Step 4: Comparing the time required for execution of code on CPU and GPU.

TABLE I
GPU SPECIFICATIONS

Model	NVIDIA Geforce 940M
Total Graphics Memory	2048 MB
No. of cores	384
Clock Rate	1071 MHz

TABLE II
CPU SPECIFICATIONS

Model	Intel,i5-5200U
Total Graphics Memory	3 MB
No. of cores	2
Clock Rate	2.5 GHz

Profiler: Profiler gives us the analysis of the time, which part of code takes the maximum time, for that we use Run and Time feature of MATLAB. Profile summary Report: The profile summary report gives us the information about how much time taken by each function and provides summary statistics for each function called. The profile summary report as shown in the Fig. 5.1.

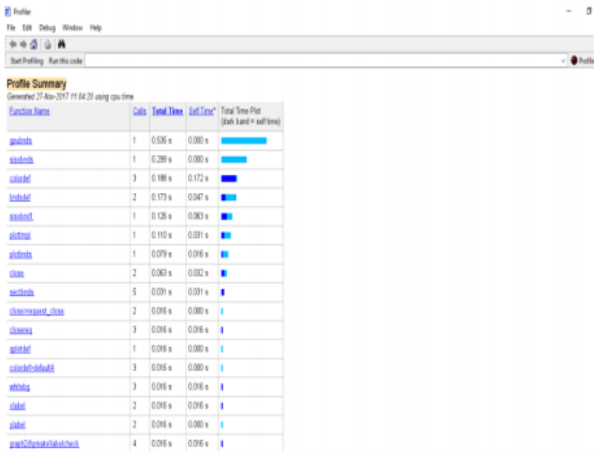

Fig. 5 Profile summary Report.

Profile Detail Report: The Profile Detail report shows profiling results for a function that MATLAB called while profiling [27]. The figure 5.2 shows the part which takes maximum time for the execution.

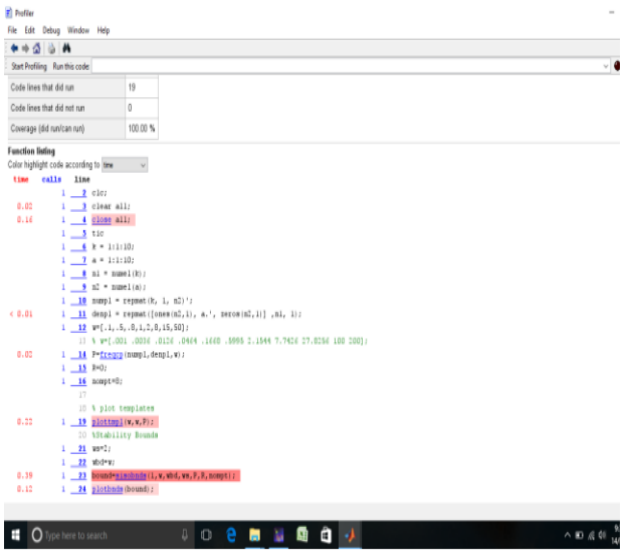

Fig. 6 Profile Report.

TABLE III

Number of frequency	Frequency array	Number of plant set
ω_1	[0.01 0.1 10 50 100]	1805
ω_2	[0.1 0.5 1 5 10 20 30 50 60 70 80 90 100]	4332
ω_3	[0.1 0.5 1 7 10 50 70 80 100 120 130 140 150 160 170]	5415
ω_4	[0. 0.5 1 3 5 10 20 30 40 45 50 60 70 80 90 100 120 150 170]	7220

© 2018, IJCERT All Rights Reserved

TABLE IV
CPU AND GPU PERFORMANCE

No. of frequency	Time(sec)			Speedup = Vectorize /GPU
	CPU		GPU	
	Sequential	Vectorize		
ω_1	0.9142	0.7798	0.3734	2.0884
ω_2	1.7592	1.7468	0.5212	3.3510
ω_3	1.7768	1.7539	0.5206	3.3688
ω_4	1.8752	1.8543	0.5504	3.3690

6. Conclusion

GPU offers significant computational power for programmers to achieve the desired result. In this work, both CPU and GPU implementation of bound computation for QFT were presented. As shown in table IV for $\omega 1$ the CPU is faster than GPU but in that case, considered less number of frequencies so that the information between two frequencies is lost. After that, as increased the number of plant set the time required for GPU is less than CPU. The execution or processing time of any function can be reduced by vectorizing it and running it on the GPU cores. The overall performance of the system improved by vectorization of code and reducing CPU load by using GPU. Parallel Computing Toolbox of MATLAB is very useful for modifying your code to run on GPU. From the results, we can conclude that implementation of bound generation using QFT on GPU speeds up the execution.

References

[1] Chait, Yossi and Yaniv, Oded, ``Multi-input/single-output computer-aided control design using the quantitative feedback theory," International Journal of Robust and Non-linear Control,vol. 3,pp. 47-54,1993. R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] Houpis, Constantine H., Steven J. Rasmussen, and Mario GarciaSanz. Quantitative feedback theory: fundamentals and applications. CRC press, 2005.

[3] Patil, Mukesh D., and Kausar R. Kothawale``Design of robust PID controller for flexible transmission system using quantitative feedback theory (QFT)," Advances in Computing, Communication and Control (2011): 479-485.

[4] Brown, Matthew, and I. R. Petersen,``Exact computation of the Horowitz bound for interval plants,"Decision and Control, 1991., Proceedings of the 30th IEEE Conference on. IEEE, 1991.

[5] Fialho IJ, Pande V, Nataraj PSV, ``Design of feedback system using kharitonov's segment in Quantitative Feedback Theory," Proceeding of thr 1st QFT syposium,Dayton,OH,1992;457-470.

[6] Zhao, Yongdong, and Suhada Jayasuriya ``On the generation of QFT bounds for general interval plants," Transactions-American Society Of Mechanical Engineers Journal Of Dynamic Systems Measurement And Control 116 (1994): 618-618.

[7] East, D. J, ``A new approach to optimum loop synthesis," International Journal of Control 34.4 (1981): 731-748.

[8] Longdon, L., and D. J. East, ``A simple geometrical technique for determining loop frequency response bounds which achieve prescribed sensitivity specifications," International Journal of Control 30.1 (1979): 153-158.

[9] Yang, Shih-Feng, ``Generation of QFT bounds for robust tracking specifications for plants with affinely dependent uncertainties,"International Journal of Robust and Nonlinear Control 21.3 (2011): 237-247.

[10] Chait, Yossi, Craig Borghesani, and Yuan Zheng. "Single-loop QFT design for robust performance in the presence of nonparametric uncertainties." Journal of dynamic systems, measurement, and control 117.3 (1995): 420-425.

[11] Rodrigues, J. M., Y. Chait, and C. V. Hollot. "An efficient algorithm for computing QFT bounds." transactions-american society of mechanical engineers journal of dynamic systems measurement and control 119 (1997): 548-552.

[12] Yang, Shih-Feng. "Efficient algorithm for computing QFT bounds." International Journal of Control 83.4 (2010): 716-723.

[13] Nataraj, P. S. V., and Gautam Sardar. "Computation of QFT bounds for robust sensitivity and gain-phase margin specifications." transactions-american society of mechanical engineers journal of dynamic systems measurement and control 122.3 (2000): 528-534.

[14] Nataraj, Palur SV. "Computation of QFT bounds for robust tracking specifications." Automatica 38.2 (2002): 327-334.

[15] Nataraj, P. S. V., and Gautam Sardar. "Template generation for continuous transfer functions using interval analysis." Automatica 36.1 (2000): 111-119.

[16] Gutman, Per-Olof, Mattias Nordin, and Bnayahu Cohen. "Recursive grid methods to compute value sets and Horowitz–Sidi bounds." International Journal of Robust and Nonlinear Control 17.2-3 (2007): 155-171.

[17] Yang, Shih-Feng. "Generation of QFT bounds for robust tracking specifications for plants with affinely dependent uncertainties." International Journal of Robust and Nonlinear Control 21.3 (2011): 237-247.

[18] Bailey, F. N., and C-H. Hui. "A fast algorithm for computing parametric rational functions." IEEE transactions on automatic control 34.11 (1989): 1209-1212.

[19] Ballance, D. J., and G. Hughes, ``A survey of template generation methods for Quantitative Feedback Theory," (1996): 172-174.

© 2018, IJCERT All Rights Reserved

[20] Fu, Minyue. "Computing the frequency response of linear systems with parametric perturbation." Systems & Control Letters 15.1 (1990): 45-52.
[21] BARTLETT, ANDREW C. "Computation of the frequency response of systems with uncertain parameters: a simplification." International Journal of Control 57.6 (1993): 1293-1309.
[22] Patil, Mukesh D., P. S. V. Nataraj, and Vishwesh A. Vyawahare,"Automated design of fractional PI QFT controller using interval constraint satisfaction technique (ICST)," Nonlinear Dynamics69.3 (2012): 1405-1422.
[23] Purohit, Harsh, and P. S. V. Nataraj, ``Optimized and automated synthesis of robust PID controller with quantitative feedback theory," ." Industrial Instrumentation and Control (ICIC), 2015 International Conference on. IEEE, 2015.
[24] Baida Zhang, Shuai Xu, Feng Zhang, Yuan Bi and Linqi Huang,``Accelerating MatLab Code using GPU: A Review of Tools and Strategies,"in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[25] https://www.researchgate.net/figure/270222593_fig1_Fig-1-CPUvs-GPU-Architecture
[26] Chandrima Roy, Kalyankumar Datta and Devmalya Banerjee, ``Quantitative Feedback Theory based Controller Design of an Unstable System,"IJCA Proceedings on International Conference on Communication, Circuits and Systems 2012 iC3S(5):11-15, June 2013.
[27] Amin, Shraddha. "Review On Quantitative feedback Theory (QFT) To Maintain Power System Stabilty." (2014).
[28] Altman, Yair M ``Accelerating MATLAB Performance: 1001 tips to speed up MAT- LAB programs," 2014.

© 2018, IJCERT All Rights Reserved