A Review on accelerating scientific computations using the Conjugate Gradient Method

Shreyasee Debnath, Manashwi Tamuli, Ashok Ray
Department of Electronics and Communication Engineering
North Eastern Regional Institute of Science and Technology
Nirjuli, Arunachal Pradesh, India
(shreyasee2013ec, tamulimanashwi25, ashokrayec)@gmail.com

Gaurav Trivedi
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Guwahati, Assam, India
trivedi@iitg.ernet.in

***Abstract*—Conjugate Gradient method is a very efficient iterative method for solving large systems of equations arising from real life scientific computing applications. In this paper we present the Conjugate Gradient method and its variants in brief. We also present a comparative analysis of implementations of this method on various platforms like FPGAs, GPUs etc which are suitable for High Performance Computing.**

Keywords—Conjugate Gradient (CG); Field Programmable Gate Array (FPGA); Graphics Processing Unit (GPU); Reconfigurable Computer (RC); General Purpose Processor (GPP); Central Processing Unit (CPU).

I. INTRODUCTION

Scientific computing applications like genetics, robotics, weather and climate prediction models etc frequently require to solve systems of equations which may be either linear or non linear. The solution of such systems of equations is the most time consuming part of any scientific computing application. Moreover the recent advances in such applications have made them much data intensive and computationally demanding. These factors have triggered the need for accelerating these computations to achieve faster and efficient implementations of such applications. Conjugate Gradient (CG) method is one of the most powerful Krylov subspace iterative methods to rapidly solve large linear systems of equations whose coefficient matrices are positive definite and symmetric [1, 2]. CG method has a good convergence rate, requires minimum amount of memory, is easy to code and each iteration step produces better solution than the previous one [2]. In this review paper we summarize the Conjugate Gradient algorithm and its implementation on various platforms from literatures and perform a comparative analysis in terms of speed.

II. THE CONJUGATE GRADIENT METHOD

In 1952, two great mathematicians, Magnus R. Hestenes and Eduard Stiefel [3] discovered the Conjugate Gradient algorithm to solve large systems of linear equations. This method is well suited to systems of the form

$$Ax = b \tag{1}$$

where x is an unknown vector of dimension $n\times 1$, A is a known coefficient matrix of dimension $n\times n$ which is symmetric (i.e. $A^T=A$) and positive definite (i.e. $x^TAx > 0$ for every non-zero vector x in R^n) and b is a known $n\times 1$ vector. Finding the solution to the system is equivalent to minimizing a scalar, quadratic function of the form

$$f(x) = \frac{1}{2}x^TAx - b^Tx + c \tag{2}$$

where c is a scalar constant [1]. The CG algorithm starts with an initial guess x_0, an initial residue r_0 and an initial search direction $p_0=r_0$. The residual computed at each step

$$r_k = b - Ax_k \tag{3}$$

is perpendicular to previous residuals. The search direction p_k is linear combination of previous search directions. x_k , the solution at step k of iteration, is equal to the solution of the previous step plus constant times the previous search direction. One advantage of this method is that it converges at n steps for n unknowns in n linear equations [2]. The basic elements of CG algorithm are matrix vector multiplication, dot products and linear combination of vectors and all of them exhibit parallelism [4].

Conjugate Gradient Method can be adapted for nonlinear systems as well. Fletcher and Reeves [5] extended the Conjugate Gradient Method for nonlinear systems. A detailed description of linear and nonlinear method has been presented in [6]. Efficiency of Conjugate Gradient method is less for ill-conditioned (i.e. whose condition number is large) matrices which generally occur in real life applications. Condition number of an $n\times n$ matrix A is given by $\kappa(A) = \|A\|\|A^{-1}\|$ with respect to a matrix norm. As the condition number increases, the convergence rate of CG method decreases. To increase the efficiency of CG algorithm preconditioning is used. Suppose that a symmetric and positive definite matrix M approximates matrix A such that $\kappa(M^{-1}A) \ll \kappa(A)$ then M is called preconditioner matrix. The solution to $Ax = b$ can be found by solving $M^{-1}Ax = M^{-1}b$ instead. Preconditioning enhances the convergence rate of CG method by enhancing the spectral property of matrix A [1, 2].

TABLE I. COMPARISON OF SPEED OF CG SOLVER IMPLEMENTED ON VARIOUS PLATFORMS

Year	Reference	Platform	Device	Algorithm	MFLOPS	Execution Time (s)
1988	[4]	CPU	Alliant FX/8	CG	8.8	113.1
				PCG	9.1	6.6
1992	[18]	CPU	Intem T800	CG	-	128
2000	[42]	CPU	Pentium 4	CG	425	-
2001	[17]	FPGA	Virtex II 6000-4	CG	1500	-
2005	[24]	FPGA	VirtexII-6000	LNS CG	1.1×10^3	-
2006	[25]	FPGA	Virtex4-25	Rational CG	1.5×10^3	-
	[29]	RC	SRC-6	CG	-	5.20
2007	[20]	GPP	Intel Xeon 5140 Woodcrest	CG	-	1.59
		GPU	NVIDIA GeForce 8800 GTX	CG	-	1.22×10^{-3}
2008	[22]	FPGA	SRC MAPstation	CG	418	-
	[23]	FPGA	VirtexII-6000	CG	5×10^3	-
		FPGA	Virtex5-330	CG	35×10^3	-
	[40]	FPGA	Stratix EP1S80	CG	1760	-
2009	[21]	multi-GPU	NVIDIA GeForce 8800 GTS	CG	13.85×10^3	-
	[43]	hybrid supercomputer	SRC6	CG	1290×10^3	-
2011	[28]	FPGA	Altera Aria II	CG	-	less than 1
2012	[31]	GPU	NVIDIA Tesla C2050 GPU	PCG	20×10^3	0.1
2013	[29]	FPGA	ALTERA STRATIX IV 530 GX.	BiCGSTAB	-	60×10^{-9}

CG-Conjugate Gradient, PCG- Preconditioned Conjugate Gradient, BiCGSTAB- Bi Conjugate Gradient Stabilized, LNS- Logarithmic Number System, CPU- Central Processing Unit, FPGA- Field Programmable Gate Arrays, GPU- Graphics Processing Unit, RC- Reconfigurable Computer, GPP- General Purpose Processor.

In [7] a generalized CG method for non symmetric systems is described. Reference [8] provides an extension to the Fletcher and Reeves CG method by applying it to unconstrained optimal control problems. The computational results of [8] show that CG method has superior convergence rate as compared to steepest descent and has higher stability than second variational techniques for such problems. Preconditioned Conjugate Gradient method based on trust regions has been applied to large scale optimization problems where trust region is the region where we trust the approximation [9]. A paper on necessary and sufficient conditions for the CG method to exist [10] shows that a class of matrices A, CG(s), exists for which the CG method is already known and for which the system $Ax = b$ can be solved by s-term CG method. A theorem according to [10] states that for a system $Ax = b$ an s-term CG method exists if and only if 1) the degree of minimal polynomial of A is less than or equal to s, or 2) the adjoint of A is a polynomial of degree less than or equal to $s–2$ in A is satisfied. To adapt the CG algorithm for nonlinear systems, the iterative procedure must be restarted occasionally, every n or $(n+1)$ iterations. Without restart the rate of convergence is linear. Restart procedures for CG method are discussed in [11]. A nonlinear CG method with a strong global convergence property where the line search satisfies the standard Wolfe conditions [12, 13] is proposed in [14]. A new nonlinear CG method with efficient line search and guaranteed descent is proposed and analyzed in [15]. A new line search which satisfies the approximate Wolfe conditions is highly accurate. A generalized CG algorithm for solutions of nonlinear systems of equations derived from elliptic partial differential boundary value problems is discussed in [16].

III. HIGH PERFORMANCE COMPUTING USING CONJUGATE GRADIENT METHOD

The systems of equations arising from real life problems are generally sparse in nature. Sparse Matrix Vector Multiplication is the most time consuming portion of iterative algorithms like CG. General Purpose Processors (GPPs) do not deliver their peak performance for such computationally intensive problems. In fact only 10–20% of their peak performance is delivered to solve sparse systems of equations [17]. Field Programmable Gate Arrays (FPGAs), with their reconfigurable hardware, and Graphics Processing Units (GPUs), with their high performance floating point hardware, have become potential platforms for accelerating scientific computations in this era of High Performance Computing. This section presents a brief overview of some of the implementations of CG method on these platforms found in literatures and performs a comparative analysis in terms of speed. Table I compares some of the implementations.

The implementation of classical CG method as well as preconditioned CG method on Alliant FX/8, which is a multivector processor with two level memory hierarchy, is shown in [4]. It shows that preconditioning affects parallelism and number of iterations must be reduced for the preconditioner M to be effective. CG method for power system simulation on parallel computer, whose processing nodes are based on Inmos T800 processor, is implemented in [18]. A sparse matrix Conjugate Gradient solver implemented in NVIDIA GeForce FX GPU [19] shows that 120 matrix multiplications take place per second and the vector inner product executes at 3400 sum reductions per second. Implementation of CG algorithm on Intel Xeon 5140 Woodcrest General Purpose Processor (GPP) and NVIDIA

GeForce 8800 GTX GPU has shown the effectiveness of this algorithm on multi-core systems [20]. Reference [21] shows a high speedup of CG solver implemented with 4 NVIDIA GeForce 8800 GTS GPU cards. A power efficient FPGA implementation of double precision CG solver using MAPstation hardware platform from SRC Computers, Inc., shows that with same performance as compared to contemporary processors FPGA based systems can operate with 30 times slower clocks [22]. The implementation shows that for problem size N=8000 CG achieves a rate of 418 MFLOPS and projects that FPGA outperforms CPU for problem sizes exceeding N=46656. A deeply pipelined CG algorithm for dense matrices implemented on VirtexII-6000 and Virtex5-330 FPGAs demonstrate the significant speed up of FPGAs over high end CPUs [23]. An implementation of CG solver with Logarithmic Number System (LNS) on VirtexII-XC2V6000 is shown in [24] and with rational arithmetic system on Virtex4-25 is shown in [25]. A Network on Chip (NoC) architecture based FPGA implementation of CG solver on Virtex5-XC5VLX110T is reported in [26]. In [27] a high level language, Impulse C, is used to implement a preconditioner on Virtex5-LX330 with CG solver implemented on CPU. Reference [28] shows that FPGA implementation of CG solver for large systems of dense matrices can achieve 30X speedup as compared to software implementations of CPU. Reference [29] shows the FPGA implementation of BiConjugate Gradient Stabilized (BiCGSTAB) solver for video processing applications. The BiCGSTAB implemented on ALTERA STRATIX IV 530 GX executes at a rate of 0.06ms/iteration as compared to 5ms/iteration for MATLAB code on CPU. Reconfigurable Computers (RC) combine GPPs and FPGAs. Reference [30] presents the implementation of CG algorithm with SPARSKIT software base on SRC-6 Model T210-0 MAPStation which is a Reconfigurable Computer. This implementation has been found to be 1.3 times faster than software only implementations. Numerical weather prediction models largely depend on iterative solvers to solve the set of equations used to model the atmosphere. A matrix free implementation of preconditioned CG method is discussed in [31]. Its uses a NVIDIA Tesla C2050 GPU for the implementation which achieves faster result as compared to a CPU. A Matrix free implementation of preconditioned CG method on NVIDIA Fermi M2090 GPU is presented in [32]. The results show that for single precision the achieved speedup is 17X while for double precision is 25X as compared to CPU software implementations. Experimental results of [33] shows that a Virtex4VLX160 FPGA achieves 40X speedup as compared to a software implementation for the preconditioned CG algorithm. In [34] a reconfigurable hardware approach is used to accelerate a preconditioned CG algorithm. In this work four accelerator models called Vectis is used with the host CPU, Intel Xeon E5-2670. Each accelerator board consists of two FPGAs, Virtex-6XC6VSX475T for computation and Virtex-6XC6VLX75T for interface. The algorithm for seismic wave simulation was executed in this model and was found to be twice as faster as compared to one Intel Xeon E5-2670 core. Application Robustification, which converts applications to numerical optimization problems, is a promising technique to reduce the power consumption of processors. It uses iterative solvers like CG to solve those optimization problems. A hardware accelerator of CG algorithm for this technique is presented in [35] and is found to reduce the execution time of one iteration of the algorithm by 96% as compared to a CPU without the accelerator. In [36] some optimizing techniques such as storing only the nonzero elements of the vector used, using shared buses and more memory interfaces to enhance the performance of FPGA based solvers are proposed. A mixed precision fully pipelined CG solver with iterative refinement is proposed in [37]. The evaluation for FPGA test performed on Xilinx ISE 7.1 showed that this solver is far more efficient for FPGA implementation than plain CG algorithm to obtain fast and accurate solution for Partial Differential Equations (PDEs). A distributed implementation of CG algorithm on multiple Altera Stratix III EP3SE110 FPGA devices connected in a ring configuration is presented in [38]. The ring configuration was chosen to achieve maximum inter FPGA communication bandwidth. The implementation achieves a speed of 302 Giga Operations Per Second (GOPS). In [39] hardware libraries that can be ported and scaled to match the performance of any FPGA for iterative solvers were proposed. The performance of this design was tested on Stratix III 3SL340F1760C3 FPGA and was found to be 2.2 times faster than a single Intel Xeon 5160 core. The power consumption was also found to be 5 times less. Implementation of double precision CG solver on hybrid supercomputers is presented in [40].

Reference [41] shows that GPUs have more advantages as compared to FPGAs. With the advent of General Purpose GPUs (GPGPUs), they have outperformed FPGAs in the field of scientific computing applications. GPUs are easier to program and are less expensive than FPGAs. Moreover FPGAs suffer from low memory bandwidth but they can outperform GPUs by scaling their memory bandwidth to match that of GPUs.

IV. Conclusion

In this paper a brief description of the Conjugate Gradient algorithm and its variants have been presented. Also various implementations of CG and their comparisons have been shown. It can be seen that implementations on FPGAs and GPUs are much more efficient than software implementations on CPUs. So they are particularly useful for High Performance Computing.

References

[1] J. R. Shewchuk, “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain”, August, 1994, Carnegie Mellon University, Pittsburgh, PA.

[2] Elena Caraba, “Preconditioned Conjugate Gradient Algorithm”, Louisiana State University & Agricultural and Mechanical College, March 2008.

[3] M. R. Hestenes and E. Stiefel, “Methods of Conjugate Gradients for Solving Linear Systems”, Journal of Research of the National Bureau of Standards, Vol. 49, No.6, December 1952.

[4] Ulrike Meier and Ahmed Sameh, “The behavior of conjugate gradient algorithms on a multivector processor with a hierarchical memory”, Journal of Computational and Applied Mathematics Volume 24, Issues 1–2, November 1988, pp 13–32.

[5] R. Fletcher and C. Reeves, “Function minimization by conjugate gradients”,1964, Computer Journal Volume 7, Issue 2 pp. 149–154.

[6] J Nocedal, SJ Wright, “Numerical Optimization”, 2006 – Springer

[7] David M. Young, Kang C. Jea, "Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods", Linear Algebra and its Applications Volume 34, December 1980, pp 159–194.

[8] L.S. Lasdon, S.K. Mitter, A.D. Waren, "The Conjugate Gradient Method for Optimal Control Problems", IEEE Transactions on Automatic Control, Volume 12 , Issue 2, pp 132–138.

[9] Trond Steihaug, "The Conjugate Gradient Method and Trust Regions in Large Scale Optimization", SIAM Journal on Numerical Analysis, Vol. 20, No. 3 (Jun., 1983), pp. 626-637

[10] Vance Faber and Thomas Manteuffel, "Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method", SIAM Journal on Numerical Analysis,Vol. 21, No. 2 (Apr., 1984), pp. 352-362.

[11] M.J.D. Powell, "Restart Procedures for The Conjugate Gradient Method", Mathematical Programming 1977, Volume 12, Issue 1, pp 241-254.

[12] Philip Wolfe, **"**Convergence Conditions for Ascent Methods", SIAM Review, Volume 11, Issue 2, pp. 226-235 (1969)

[13] Philip Wolfe, **"**Convergence Conditions for Ascent Methods. II: Some Corrections", SIAM Review, 1971, Vol. 13, No. 2 , pp. 185-188.

[14] Y. H. Dai*,* Y. Yuan*,* "A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property", SIAM Journal on Optimization, 1999, Volume 10, Issue 1, pp. 177–182.

[15] William W. Hager*,* Hongchao Zhang, "A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search", SIAM Journal on Optimization Volume 16, Issue 1 pp- 170–192.

[16] P. Concus, G. H. Golub, D. P. O'Leary, "Numerical Solution of Nonlinear Elliptic Partial Differential Equations by a Generalized Conjugate Gradient Method", Computing 1978, Volume 19, Issue 4, pp 321-339.

[17] Michael deLorimier, Andr´e DeHon, "FloatingPoint Sparse MatrixVector Multiply for FPGAs", Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, pp 75-85.

[18] I.C. Decker, D. M. Falcão, E. Kaszkurewicz, "Parallel Implementation of a Power System Dynamic Simulation Methodology using The Conjugate Gradient Method", IEEE Transactions on Power Systems, Volume:7, Issue: 1, pp 458 – 465.

[19] Jeff Bolz, Ian Farmer, Eitan Grinspun, Peter Schröder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid", ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003, Volume 22 Issue 3, July 2003, pp 917-924.

[20] W.A. Wiggers, V. Bakker, A.B.J. Kokkeler, G.J.M. Smit, "Implementing the conjugate gradient algorithm on multi-core systems", IEEE International Symposium on System-on-Chip, 2007, pp 1 - 4

[21] Ali Cevahir, Akira Nukada, and Satoshi Matsuoka, "Fast Conjugate Gradients with Multiple GPUs", Computational Science – ICCS 2009 Lecture Notes in Computer Science Volume 5544, 2009, pp 893-903.

[22] David DuBois, Andrew DuBois, Thomas Boorman, Carolyn Connor, Steve Poole, "An Implementation of the Conjugate Gradient Algorithm on FPGAs", 16th International Symposium on Field-Programmable Custom Computing Machines, 2008. FCCM '08. IEEE .pp. 296 – 297.

[23] Antonio Roldao, George A. Constantinides, "A High Throughput FPGA based Floating Point Conjugate Gradient Implementation for Dense Matrices", ACM Transactions on Reconfigurable Technology and Systems, Volume 3 Issue 1, January 2010.

[24] Owen Callanan, David Gregg, Andy Nisbet, Mike Peardon, "High Performance Scientific Computing using FPGAs With IEEE Floating Point And Logarithmic Arithmetic For Lattice QCD", Proc. Field Programmable Logic and Applications, 2006, pp. 29-35.

[25] V.L.O. Maslennikow, A. Sergyienko, "FPGA Implementation of the Conjugate Gradient Method", Proc. Parallel Processing and Applied Mathematics, 2005, pp. 526-533.

[26] C.-C. Sun, J. Gotze, H.-Y. Jheng, S.-J. Ruan, "Sparse matrix-vector multiplication on network-on-chip", Advances in Radio Science, vol.8, pp 289-294.

[27] Vincent Heuveline, Wolfgang Karl, Fabian Nowak, Mareike Schmidtobreick, Florian Wilhelm, "Employing a High-Level Language for Porting Numerical Applications to Reconfigurable Hardware", Preprint Series of the Engineering Mathematics and Computing Lab (EMCL).

[28] Torstein Habbestad, "An FPGA-based implementation of the Conjugate Gradient Method used to solve Large Dense Systems of Linear Equations", Norwegian University of Science and Technology, August 2011.

[29] Pierre Greisen, Marian Runo, Patrice Guillet, Simon Heinzle, Aljoscha Smolic, Hubert Kaeslin, Markus Gross, "Evaluation and FPGA Implementation of Sparse Linear Solvers for Video Processing Applications", IEEE Transactions on Circuits and Systems for Video Technology, Volume:23 , Issue: 8 , pp 1402 – 1407.

[30] Gerald R. Morris, Viktor K. Prasanna, Richard D. Anderson, "A Hybrid Approach for Mapping Conjugate Gradient onto an FPGA-Augmented Reconfigurable Supercomputer", 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006. FCCM '06, pp 3-12.

[31] Sinan Shi, "GPU Implementation of Iterative Solvers in Numerical Weather Predicting models", The University of Edinburg. 2012.

[32] Eike Mueller, Xu Guo, Robert Scheichl, Sinan Shi, "Matrix free GPU implementation of a preconditioned conjugate gradient solver for anisotropic elliptic PDEs", Numerical Analysis, Cornell University Library.

[33] Jing Hu, "Solution of Partial Differential Equations using Reconfigurable Computing", The University of Birmingham, 2010.

[34] Art Petrenko, "Accelerating an Iterative Helmholtz Solver Using Reconfigurable Hardware", The University Of British Columbia, 2014.

[35] David Kesler, Biplab Deka, Rakesh Kumar, "A Hardware Acceleration Technique for Gradient Descent and Conjugate Gradient", IEEE 9th Symposium on Application Specific Processors (SASP), 2011, pp 94 – 101.

[36] Marcel van der Veen, "Sparse Matrix Vector Multiplication on a Field Programmable Gate Array", University of Twente, 2007.

[37] Robert Strzodka, Dominik Goddeke, "Pipelined Mixed Precision Algorithms on FPGAs for Fast and Accurate PDE Solvers from Low Precision Components", IEEE Proceedings on Field-Programmable Custom Computing Machines, 2006, pp 259-270.

[38] Seyed Behzad Mahdavikhah Mehrabad, "A Multiple-FPGA Parallel Computing Architecture for Real-Time Simulation of Deformable Objects", Mcmaster University, 2009.

[39] Wei Zhang, "Portable and Scalable FPGA-Based Acceleration of a Direct Linear System Solver", University of Toronto, 2008.

[40] Yousef Elkurdi, David Fernández, Evgueni Souleimanov, Dennis Giannacopoulos, Warren J. Gross, "FPGA architecture and implementation of sparse matrix–vector multiplication for the finite element method", Computer Physics Communications 178 (2008), pp 558–57

[41] Yan Zhang, Yasser H. Shalabi, Rishabh Jain, Krishna K. Nagar, Jason D. Bakos, "FPGA vs. GPU for Sparse Matrix Vector Multiply", International Conference on Field-Programmable Technology, 2009. FPT 2009, pp 252-262.

[42] R. Vuduc, J. Demmel, K. Yelick, S. Kamil, R. Nishtala, and B. Lee. "Performance Optimizations and Bounds for Sparse Matrix-Vector Multiply". Proceedings of IEEE/ACM Conference on Supercomputing, November 2002.

[43] David DuBois, Andrew DuBois, Thomas Boorman, Carolyn Connor, "Non-Preconditioned Conjugate Gradient on Cell and FPGA based Hybrid Supercomputer Nodes", 17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009. FCCM '09, pp 201-208.