Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 5 (2023)

Parallel Algorithm for Hermite Interpolation on Hypercube Inter Connection Network

Rasheswari B. Ray[1], Alok Ranjan Tripathy[2]

Abstract The paper focuses on a faster and more parallel algorithm for N-point. Hermite interpolation on a dimensional hypercube with $P = 2(n)$ processors. The algorithm is adapted to the scenario when the number of processors $P < N$ and N is divisible by P. The paper focuses on the parallel Hermite interpolation algorithm for the hypercube network. The algorithm exhibits $(N - 1)O(\log N)$ time for execution.

1. Introduction

In numerical analysis, interpolation is used to predict the value of a function at a point from the known values of function at some tabulated points. In scientific and engineering fields interpolation is pervasively used in various fields of science and engineering ranging from space research, stock predictor, air traffic monitoring, etc., predicting the trajectory of rockets, monitoring air traffic at airports, forecasting the paths of the missile during lunch, curve fitting, etc. There are many interpolating polynomials, such as Lagrange interpolating polynomials, Newton interpolating polynomials, Hermite interpolating polynomials, Bessel's interpolating polynomials, etc. Among them, the Hermite interpolation polynomial fits N points to a polynomial of degree $(2N - 1)$. In contrast, other interpolating polynomial fits N Points to a polynomial of degree $(N - 1)$, thus providing better accuracy than others [1]. Recently Ray et al., [21] gave an efficient algorithm for Hermite Interpolation on uniprocessor setup. For N points, their sequential interpolating polynomial P and their algorithm uses less number of floating points operation than the conventional interpolating algorithm [13]. For random sets of data points, they also verified the speed of their algorithm is 164 times faster than the conventional interpolation algorithm [10].

The advent of system-on-chip (SOC), network on chip (NOC) for multicore systems has pushed the VLSI technology for designing efficient on-chip interconnection networks for high-performance computing. There are many interconnections on-chip networks used for high-performance computing. Some widely used are hypercube [22], sonograph, calligraphy, K-ary network, cross cube [1], the folded cross cube [2] to hyper cube [22], etc. Among them, the hypercube is an interconnection network that has many nice properties such as vertex symmetric, edge symmetric, and low diameter. Many parallel algorithms have been developed in this network in the last two decades [5] [6] [7]. Some of them are Lagrange interpolation hypercube [23], a similar algorithm for FFT [22] motivated by the efficient sequential Hermite interpolation algorithm [20]. In this work the parallel version of the efficient sequential algorithm on a hypercube interconnection network is developed.

Hypercube is an interconnection network that is edge symmetric and vertex symbol. Many parallel algorithms have been developed over the last decade [14] [15]. In 2006 on hypercube network, a parallel algorithm for Lagrange interpolation [24] was developed by C.P.Katti [13] which uses less number of floating point operations for the evaluation N-point interpolation formula. The efficient Hermite interpolating polynomial developed by Ray et al. [21] uses a similar approach to increase the efficiency of the Hermite interpolation algorithm in a sequential setup.

In short, the contributions in this paper are summarized below.

- Developed an efficient parallel Hermite interpolation algorithm on a hypercube interconnection network.
- Studied its computational statistics,
- Compared its running time with existing parallel Hermite interpolation algorithm on an extended

Fibonacci cube [20].

The rest of the paper is organized as follows : Section 2 discusses the Hermite Interpolation and relevant prerequisites for the development of a parallel algorithm for Hermite interpolation in a hypercube interconnection network. Section 3 presents a parallel algorithm for Hermite interpolation in Hypercube. Section 4 discusses the cost analysis of the proposed algorithm. Section 5 elaborates the time estimation. The performance evaluation has also been studied in Section 6 followed by conclusion in Section 7.

Hypercube Interconnection Network

A binary n-cube multiprocessor called as hypercube interconnection network [8] [9]. The network consists of $N = 2^n$ processors that are connected with an N-dimensional binary cube. The node of a cube is denoted as a processor which has a direct communication path with other N neighboring processors. The processor is identified with 2^n distinct n-bit binary address. Figure 1 shows the different hypercube structures with different values of n as 1,2,3,4.

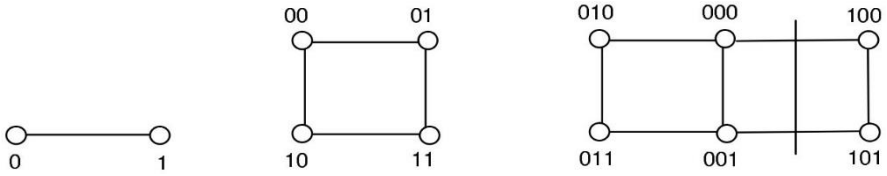
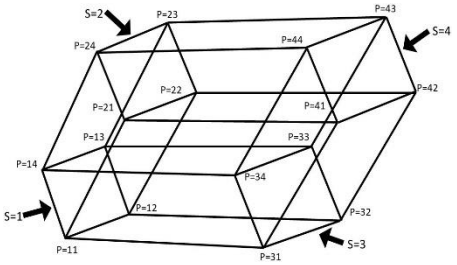


Figure 1: Hypercubes.

Hypercube is one of the topologies for interconnection networks [16]. It was developed by S.B. Aker & Krishnamurthy [3]. Figure 2 shows a hypercube of four dimensions. So, an n-dimension hypercube has a 2^n number of nodes. In view of its regularity and symmetric structure (vertex symmetric and edge Asymmetric). It is widely used as an interconnection network for multi-computer systems many parallel algorithms are developed in this interconnection network [17] [19].

Figure 2: Four-dimensional Hypercube



2.1. Hermite Interpolation

The Hermite interpolating polynomial [11] x for the data set $(x_0, y_0, y_0') \ldots (x_{N-1}, y_{N-1}, y_{N-1}')$ is a polynomial of degree $2N - 1$.

$$H_{2N-1}(x) = \sum_{i=0}^{N-1} \left[1 - 2(x - \mathrm{x}_i)\, l_i'(x_i)\right] l_i^2(x)^2 + \sum_{i=0}^{N-1} (x - x_i) l_i^2(x) y_i' \quad (1)$$

$l_i(x)$ is the Lagrange polynomial in x and $H_{2N-1}(x_i) = y_i, H'_{2N-1}(x_i) = y_i'$

for $i = 0, 1, \ldots N - 1$

From (1) it is obtained as,

Where $h_i(x) = \left[1 - 2(x - \mathrm{x}_i)\, l_i'(x_i) y_i + (x - \mathrm{x}_i) y'_i\right]$,

$$l_i(x) = \frac{(x-x_0)\,(x-x_1)\,\ldots\ldots\,(x-x_{i-1})(x-x_{i+1})\,\ldots..(x-x_{N-1})}{(x_i-x_0)\,(x_i-x_1)\,\ldots\ldots\,(x_i-x_{i-1})(x_i-x_{i+1})\,\ldots..(x_i-x_{N-1})}$$

and

$$l'_i(x_i) = \sum_{j=0,j\neq i}^{N-1} \frac{1}{(x_i-x_j)}$$

for $i = 0, 1, \ldots\ N-1$.

Ray et al. [21] developed an efficient sequential algorithm for Hermite interpolation and compare its running time with its conventional counterpart. The efficient sequential algorithm is 164 times faster than the conventional Hermite Interpolation algorithm.

In the following, the conventional and efficient Hermite interpolation algorithms are presented. To develop the proposed parallel algorithm the given Lemma is used.

1.1.1 Lemma 1

$$l'_i(x_i) = \sum_{j=0,j\neq i}^{N-1} \frac{1}{(x_i-x_j)}$$

for $i = 0, 1, \ldots\ \mathrm{N}-1$.

Parallel algorithms for Hermite interpolation were discussed on Fibonacci cube [12] on star graph by Akers et al. [3][4]. The computational statistics for the parallel algorithm for $H_{2N-1}(x)$ discussed in [3] as follows.

Algorithm 1 Seq_Lagr_Algo_KATT()

input : $x, x_1, x_2, x_N \ldots\ldots\ldots y1, y2, yN$

Output : $F(x)$

Initialize : Product = 1 : $F(x) = 0$

for j = 1 *to* j = N in steps of 1 do :

Product=Product*(x-x_j)

for j = 1 *to* j=N in steps of 1 do :

{

P = 1

for i = N to i> j in steps of 1 do :

{

$a_{ij} = x_j - x_i$

$P = P \times a_{ij}$

}

$for\ i = N\ to\ i < j$ in steps of 1 do :

{

$a_{ij} = -a_{ji}$

$P = P \times a_{ij}$

}

$$F(x) = F(x) + \frac{\Pr o\, duct \times y_i}{(x - x_j) \times P}$$

}

For Lagrange function $l_i\,(x)$ with dataset $x_0, x_1, \ldots. x_{N-1}$, $l_i'\,(x_i)$ is given by

$$l_i'\,(x_i) = \sum_{\substack{j=0 \\ j \neq i}}^{N-1} \frac{1}{(x_i - x_j)}$$

$$T_1^{EH} = \left[(2N-2) + \frac{N(N+5)}{2} + (N^2 + 2N - 1) + N^2\right]\alpha = \left[\frac{5N^2}{2} + \frac{13N}{2} - 3\right]\alpha = \frac{5N^2}{2}$$

1.1.2 Theorem 1

The main phase of the Hermite interpolation algorithm involves $(2N+3)$ multiplications, $N-$ additions, $2N$-subtractions, and N- divisions. N denotes the number of processors of Hypercube $HC(n)$, and n is the degree of the Hypercube. From the algorithm, the overall construction and statistics for computation for parallel Hermite interpolation algorithm on extended Hypercube are $\left(\frac{N}{2}+3\right)$ data communications, $(2N+3)$ multiplications, N-additions, $2N$-subtractions and N-divisions.

If we use startup time σ and communication time per unit γ in seconds. Then$\left(\frac{N}{2}+3\right)$ data communications shall take $(\sigma+\gamma)\left(\frac{N}{2}+3\right)$ second. Here the total number of arithmetic operations

= $(2N+3)$ multiplications + N (additions) + $2N$ (subtractions) + N (divisions) = $(6N+3)$. Since α unit time is the time in sec requested per arithmetic operations, so the time taken is equal to $(6N+3)\alpha$ second.

Thus the total running time of Hermite Interpolation in HC with N processors = data communication time + time for arithmetic operations

$= (\sigma+\gamma)\left(\frac{N}{2}+3\right) + (6N+3)\alpha$ ……..(1)

Equation (1) is the statistics of the main phase.

The final stage accumulates the partial results stored as access N processors. The final phase of the Algorithm uses $logN$ steps to accumulate [18]. In step-1 there are $\left(\frac{N}{2}\right)$ data communications and $\left(\frac{N}{2}\right)$ additions. In step -2 there are $\left(\frac{N}{2^2}\right)$data communication time and $\left(\frac{N}{2^2}\right)$ additions. For $LogN$, step there are $\left(\frac{N}{2^{\log N}}\right)$data communications and $\left(\frac{N}{2^{\log N}}\right)$additions. Thus at the final phase the total no of data communications

$$= \left(\frac{N}{2}\right) + \left(\frac{N}{2^2}\right) + \ldots\ldots + \left(\frac{N}{2^{\log N}}\right)$$

$$= \left(\frac{N}{2}\right)\left(1 + \left(\frac{N}{2}\right) + \left(\frac{N}{2^2}\right) + \ldots\ldots + \left(\frac{N}{2^{\log N}}\right)\right)$$

$$= \left(\frac{N}{2}\right)\left(\frac{1 - \frac{1}{2^{\log N}}}{\frac{1}{2}}\right)$$

$$= N\left(1 - \frac{1}{N}\right)$$

$$= \frac{N(N-1)}{N}$$

$$= (N-1)$$

Number of additions

$$= \left(\frac{N}{2}\right) + \left(\frac{N}{2^2}\right) + \ldots\ldots + \left(\frac{N}{2^{\log N}}\right) = (N-1)$$

Data communication cost $(\sigma + \gamma)(N-1)$

Number of addition cost $= (N-1)\alpha$

Computational overhead of final stage $= (\mathfrak{6} + \gamma)(N-1) + (N-1)\alpha$ …… (2)

Adding (1) & (2) this total complexity

$$= (\sigma + \gamma)\left(\frac{N}{2} + 3\right) + (6\mathrm{N} + 3)\alpha + (\sigma + \gamma)(N-1) + (N-1)\alpha$$

$$= (\sigma + \gamma)\left(\frac{3}{2}N + 2\right) + (7N + 2)\alpha$$

Algorithm 2 Efficient Hermite Interpolation Algorithm

This section presents Efficient_Hermite_Interpolation_Algorithm( ) to implement equation efficiently.

$(x, x_1, x_2, \ldots\ldots x_N, y_1, y_2, \ldots\ldots y_N, x)$

{

Input : $x_1, x_2, \ldots\ldots x_n, y_1, y_2, \ldots\ldots y_n, x$

Output : s //Value of polynomial at x

Initialize: $Prod = 1, s = 0, b_{ij}, a_{ij}$

//Evaluation of $\pi^2(x)$

For $j = 1$ *to* n

$Prod = Prod * (x - x_j)$

End for

$Prod = Prod * prod$ $//\pi^2(x)$

For $j = 1$ *to* n

$p = 1, q = 0$

$for\ i = n\ to\ i > j$

$a_{ij} = x_j - x_i$ $//denominator\ of\ l_i(x)$

$b_{ij} = \frac{1}{a_{ij}}$

$q = q + b_{ij}$ $//evaluation\ of\ l_i'(x)$

$p = p * a_{ij}$

End for

for i = 1 to i < j

$a_{ij} = -a_{ij}$

$b_{ij} = -b_{ij}$

$p = p * a_{ij}$

$q = q + b_{ij}$

end for

$p = p * p$ //$evaluation\ of\ (\pi'(x))^2$

$$S = S + \frac{(y_j X(1 - 2 * q) + y_j')}{(x - x_j) * p}$$

Output $s * prod$

}

Parallel Algorithm for Hermite Interpolation in Hypercube

In the following, a parallel algorithm of the efficient sequential algorithm is discussed in [21] on n dimensions algorithm of hypercube networks developed. In this work, the number of the processor of the hypercube as the degree of the Hermite interpolating polynomial is taken. The motivation behind taking the degree of the hypercube is to compare the algorithm with the parallel algorithm for Hermite interpolation developed in FFC [23]. The case for the degree of the Hermite polynomial larger than the number of processors of the hypercube is the future scope of our research.

Let $P = N$ be the degree of the Hermite interpolation polynomial where $P = 2^n (n \geq 3)$, n being the dimension of the hypercube.

As discussed in C.P. Katti, the P processor is divided into S-subcubes starting from $S = 2^{n-2}$, and S is the subcube of dimension two.

Let P_{il} be the processor $l \leq i \leq 4$

Then each subcube $1 \leq i \leq s$ has four processors $P_{il}, 1 \leq i \leq 4$. Each $P_{il}, 1 \leq i \leq s, 1 \leq i \leq 4$ is initialize to the data points $X_{(4i-1)} + l, Y_{(4i-1)} + l, Y'_{(4i-1)} + l$, and x where $(x_0, y_0, y_o) \ldots\ldots (x_n, y_n, y_n)$ are N-tabulated points for the Hermite interpolating polynomial $P_{2N-1}(x)$discuss in sec 2.1. The algorithm discussed is the parallel version of the efficient hermite interpolation algorithm.

Step 1 :

Every processor $P_{il}, 1 \leq i \leq 4$, in multi-node broadcast [8] is the subcube $i (1 \leq i \leq l)$ that receives the data point $x_{4(i-l)} + J, 1 \leq i \leq 4, y_{4(i-l)} + l$ and x

The multi node broadcast is done in parallel for all i and l $(1 \leq i \leq S);\ (1 \leq l \leq 4)$.

Step 2

For all i and $l, 1 \leq i \leq S, 1 \leq l \leq 4$ compute parallel

$$A_{il} = \prod_{j=1, j \neq l}^{4} (X_{4(i-1)} + l - X_{4(i-1)} + J)$$

$$B_{il} = \prod_{j=1}^{4} (X - X_{4(i-l)} + J)$$

$$L_{il} = \sum_{j=1, j\neq l}^{4} = \frac{1}{X_{4(i-1)} + l - X_{4(i-1)} + J}$$

Step 3

For every i and $l, 1 \leq i \leq s; 1 \leq l \leq 4$ let the processor P_{il} transmit the product's value. B_{il} and $X_{4(i-1)+l}$ to each processor P_{il} $1 \leq l \leq s, k \neq i$. (For all l and $i, 1 \leq l \leq 4; 1 \leq i \leq s$ P_{il} receives product B_{il} and $X_{4(i-1)+l}$ from P_{il} $1 \leq k \leq s, k \neq i$). This process is performed in parallel for $1 \leq i \leq s; 1 \leq l \leq 4$

Step 4

$$A_{kl}^{i} = \prod_{j=1}^{4} \left(X_{4(k-i)} + l - X_{4(i-1)} + J\right)$$

For all l_i $, 1 \leq l \leq 4, 1 \leq i \leq s, (k \neq i)$

$$L_{kl}^{i} = \sum_{j=1}^{4} \left(X_{4(k-i)} + l - X_{4(i-1)} + J\right)$$

Step 5

For l and i present, $1 \leq l \leq 4, 1 \leq i \leq s$ processor P_{il} transfers A_{kl}^{i} to $P_{kl}, 1 \leq k \leq s, k \neq i$. (Alternatively, for l and i, $1 \leq l \leq 4, 1 \leq i \leq s$ processor P_{il} receives A_{il}^{k} from each $P_{kl}, 1 \leq k \leq s, k \neq i$). Again, it will be performed in parallel for all i and $l, 1 \leq i \leq s, 1 \leq l \leq 4$.

Step 6

$$A_{il}^{*} = A_{il} \prod_{k=1, k\neq i}^{s} A_{il}^{k}$$

$$B_{il}^{*} = B_{il} \prod_{k=1, k\neq i}^{s} B_{kl}$$

$$L_{il}^{*} = L_{il} \sum_{k=1, k\neq i}^{s} L_{il}^{k}$$

Step 7

$$C_{il} = \frac{B_{il}^{*}}{x - x_{4(i-1)+l})A_{il}^{*}}$$

Step 8

$$F(x) = \sum_{\mathrm{i}} \sum_{l} C_{il}$$

Cost Analysis of Algorithm

Step 1 = 3γ

Step 2 = $6\alpha + 6\alpha + 6\alpha = 18\alpha$

Step 3 = $2(s-1)\gamma$

Step 4 =$7(s-1)\alpha * 2 = 14\alpha\,(s-1)$

Step 5 = $(logp - 2)\left(\frac{p}{8}\right)\gamma$

Step 6 = $2(s-1)\alpha + (s-1)\alpha = 3(s-1)\alpha$

Step 7 = 5α

Step 8 = $(\alpha+\gamma)logp$

$$T_p = \big(23 + 17(s-1)\big)\alpha + \big(3 + 2(s-1) + (log\rho - 2)(p/8)\big)\gamma + \theta + (\alpha+\gamma)logp$$

Estimation of time

Any arithmetic operation such as addition, multiplication, or division is assumed to take $\alpha -$ unit of time. For communication operation, it takes $\gamma + ml$ to transfer words from one processor to any of its P neighbors, where the communication start-up time in seconds is δ and the element transfer time in seconds per word is γ . It is assumed like simultaneous transfers along various links are allowed. A node can receive almost one packet along its incident link at any point without deadlock. The total time for solving Hermite interpolation for the show in each uniprocessor environment using the conventional and efficient algorithm is discussed in section 2.

Time Complexity of Algorithm 3

The communication time for steps $1, 3,$ and 5 is 3δ, $2(s-1)\delta$ and $(\log P - 2)\left(\frac{P}{8}\right)\delta$ is computed using the above notation. The computation time for executing steps 2, 4, 6 and 8 are 18α, $14\alpha(s-1), 3(s-1)\alpha, 5\alpha$ and $(\alpha\delta)logP$. Therefore the total time T_P when $P = N$ us given by

$$T_p = ((18 + 5 + 14(s-1) + 3(s-1))\alpha + (3 + 2(s-1) + (\log P - 2)\left(\frac{P}{8}\right)\delta + \sigma + (\alpha\delta)\log P$$

$$= \big(23 + 17(s-1)\big)\alpha + 3 + 2(s-1) + (\log P - 2)\left(\frac{P}{8}\right)\delta + \sigma + (\alpha\delta)\log P$$

$$= (17s+6)\alpha + 2s + 1 + (\log P - 2)\left(\frac{P}{8}\right)\delta + \sigma + (\alpha\delta)\log P$$

Note that $s = 2^{n-2}$, $p = 2^n$ (number of processor)

Thus $s = \frac{p}{4}$

$$T_p = \left(17 + \frac{p}{4} + 6\right)\alpha + \left(\frac{2p}{4} + 1 + (\log P - 2)\left(\frac{P}{8}\right)\right)\delta + \sigma + (\alpha\delta)\log P$$

$$= \left(\frac{17p}{4} + 6\right)\alpha + \left(\frac{p}{2} + 1 + (\log P - 2)\left(\frac{P}{8}\right)\right)\delta + \sigma + (\alpha\delta)\log P$$

Putting $p = N$

$$T_p = \left(\frac{17N}{4} + 6\right)\alpha + \left(\frac{N}{2} + 1 + (\log N - 2)\left(\frac{N}{8}\right)\right)\delta + \sigma + (\alpha\delta)\log N$$

Whereas its sequential algorithm, as stated in C.P.Katti, runs with complexity $T_1 = 3N^2$

Performance Evaluation

The performance measure of the proposed parallel algorithm defines speed up factor of a similar algorithm by comparison with its serial counterpart as $S_P = \frac{T_1}{T_P}(\geq 1)$ and efficiency as $E_P = \frac{S_P}{P}(\leq 1)$. The parameter of Intel/860 as given in [9].

Start up time $(\delta) = 1.0X10\text{^}\{-4\}\ \sec s$

Element transfer time $(\gamma) = 1.0X10^{-6}Secs$

Multiplication time $(\alpha) = 1.0X10\hat{}\{-7\}Secs$

3. Conclusion

The paper focuses on the parallel algorithm for Hermite Interpolation in the Hypercube. The algorithm consist of different phases. This algorithm uses $\frac{N}{2}$ steps for data communication and addition in the initial stage whereas it take $(N-1)\log(N)$ times for data communication in the final phase. So it performs better in hypercube as compared to the work discussed in [18].

References

[1] Adhikari, N.; Tripathy, C. R., "Extended Crossed Cube: An Improved Fault Tolerant Interconnection Network", *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, vol., no., pp.86,91, 25-27 Aug. 2009

[2] Adhikari, N.; Tripathy, C. R., "The Folded Crossed Cube: A New Interconnection Network for Parallel Systems", vol.4, no.3, pp.43-50, 2010.

[3] Akers, S.B.; Harel, D.; Krishnamurthy, B., "The Star Graph: An Attractive Alternative to the *n*-Cube", *Proceeding of International Conference on Parallel Processing*, pp. 393-400, 1987.

[4] Dally, William J.; Towles, Brian., "Route packets, not wires: On-chip interconnection networks". In DAC '01: *Proceedings of the 38th Conference on Design Automation*, pp. 684-689, 2001.

[5] Dally, William J.; Seitz, C.L., "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *Computers, IEEE Transactions on*, vol. C-36, no.5, pp. 547, 553, May 1987.

[6] Dally, William J.; Towles, B., "Principles and Practices of Interconnection Networks", *Morgan Kaufmann Publishers, San Francisco, California*, USA, 2004 .

[7] Dally, William J.; Seitz, C.L., "The Torus Routing Chip", *Journal of Distributed Computing*, vol. 1, no. 3, pp. 187-196, 1986.

[8] Fujii, H; Yasuda, Y; Akashi, H; Inagami, Y.; Koga, M.; Ishihara, O.; Kashiyama, M.; Wada, H.; Sumimoto, T., "Architecture and performance of the Hitachi SR2201 massively parallel processor system", *Proceeding of 11th International Parallel Processing Symposium*, pp.233-241,1997.

[9] Hsu, W.J., "Fibonacci Cubes-A New Interconnection Topology", *IEEE Trans. Parallel and Dis- tributed Systems*, vol.4, no.1, pp.3-12, Jan.1993.

[10] Intel Corporation, "A Touchstone DELTA system description", *Intel Supercomputer Systems Division*, February 1991.

[11] Jain M.K., Iyengar S. R. K., Jain R. K., "Numerical Methods for Scientific and Engineering Computation" *3e, Wiley Eastern Limited, 2006.*

[12] Jie Wu, "Extended Fibonacci Cubes", *Parallel and Distributed Systems, IEEE Transactions on*, vol.8, no.12, pp.1203,1210, Dec 1997.

[13] Katti C.P.; Kumari, R., A New Parallel Algorithm for Lagrange Interpolation on a Hypercube. Computers and Mathematics with Applications, vol. 51, Issue. 6-7 pp. 1057,1064, Mar 2006.

[14] Mohanty, S.P.; Patro, S. N.; Patra, N; Ray, B. N. B., "Parallel algorithm for solving system of linear equation by Jacobi method on Star graph", *Proceeding of the National conference on Advanced Data Computing Communications and Security, S.V. Institute. of Computer Studies*, pp. 305-309, Jul. 2007.

[15] Mohanty, S.P.; Ray, B. N. B.; Patro, S. N.; Patra, N, "Parallel computation for Hermite interpolation on k-ary n-cube", *Proceeding of 21st National convention of computer engineer and national seminar on advances in soft computing, organized by Institute of Engineers, Bhubaneswar*, pp.141-148, 2007.

[16] Mohapatra P., "Wormhole routing techniques in multicomputer systems", *ACM Computing Surveys*, vol. 30, no. 3, pp. 375-411, 1998.

[17] Ray, B. N B; Tripathy, A.R., "Extended Hypercube with Cross-Connection - A New Interconnection Fault Tolerant Network for Parallel Computers", *Advance Computing Conference, 2009. IACC 2009. IEEE International* , vol., no., pp.513,518, 6-7 March 2009.

[18] Ray, B.N.B, Tripathy, A.R & Mohanty, S.P. Parallel hermite interpolation on extended Fibonacci cubes. International Journal of Computer Applications. 54 (17). pp 36-41. 2012.

[19] Ray, B. N. B.; Tripathy, Alok Ranjan; Mohanty, S. P., "A New Interconnection Network for Parallel Computers", *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, vol., no., pp.19,24, 25-27 Aug. 2009.

[20] Ray, B. N. B.; Tripathy, Alok Ranjan; Mohanty, S. P., "Parallel Hermite Interpolation on Extended Fibonacci Cubes", International Journal of Computer Applications, vol.54, issue 17, pp.36-41, Sept. 2012

[21] Ray, Rasheswari & Tripathy, Alok & Ray, B.N.B., ``Efficient Algorithm for Hermite Interpolation.", 19th OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, pp.32-36, Dec 2021.

[22] Tripathy, A.R." Development of Parallel Algorithms for Graph and Numerical Problems using Parallel Interconnection Network. Ph.D Thesis, Dept of Computer Science, Utkal University, Vani Vihar, Bhubaneswar, pp.173, Mar. 2014.

[23] Tripathy, A.R.; Ray, B. N. B., Parallel Algorithm for Solving the System of Simultaneous Linear Equations by Jacobi Method on Extended Fibonacci Cubes. Advance Computing Conference, IACC. pp.961-967, Feb. 2013.

[24] Zelina, Ioana, "Parallel Lagrange Interpolation on Extended Fibonacci Cube", studia univ. Babes-bolyai, informatica, vol. L, no.1, pp.105-110, 2005.