ISSN:0975-9646

Inverse Orchestra: An approach to find faster solution of matrix inversion through Cramer's rule

Shweta Agrawal , Rajkamal
School of Computer Science & Information Technology,
DAVV, Indore (M.P.), India

Abstract: An Orchestrator coordinates and controls computations at parallel and sequential computing nodes. Matrix inversion is the need of many scientific applications. The paper presents a design and implementation of an Orchestrated framework named InverseOrchestra. The InverseOrchestra is finding the faster large matrix inversion through Orchestration. The InverseOrchestra is using Cramer's rule for finding inverse of generalized matrix. The extension of framework named InverseOrchestraE also has been presented. It has been proved theoretically and practically that InverseOrchestra is much faster than Conventional approach and InverseOrchestraE is faster than InverseOrchestra.

1. INTRODUCTION:

Matrix manipulation often requires in science and engineering. There are many areas like signal processing, communications, parameter optimization, which include the problems which requires solving to matrix inversion. The matrix inversion is avoided by most numerical analyst [1]. This is because inversion is normally more time consuming, and less stable. However, in some practical situations the matrix inversion is compulsorily required. The application domain of matrix inversion includes: Wiener and Kalman filtering [2], all similarity transformations, statistics [3], eigenvalue-related problems [4], super- conductivity computation, in power engineering [5] etc.

There are a variety of methods for matrix inversion. Many parallel algorithms for matrix inversion and related problems (LSE, memory multiplication and determinant) have been proposed [1] [6] [7]. In practice, the most used algorithms for solving inverse of a matrix are based on Gaussian elimination with pivoting, block Gaussian elimination, and their modifications.

The paper presents the orchestrated [8-11] framework for matrix inversion through Cramer's rule. The reason behind choosing the Cramer's rule is that it provides determinant of matrix and adjugate [12] matrix of input matrix also. There are numerous applications of adjugate matrix [13, 14] also.

Let the input matrix A is a nonsingular real square matrix [15], Then by Cramer's rule $A^{-1} = adj\ A / |A|$

The organization of paper is as follows: Section 2 is describing the conventional approach to find the matrix inversion, Section 3 is presenting the InverseOrchestra, and Section 4 is showing the extension of InverseOrchestra, Section 5 consists of implementation procedure, section 6 is showing results and section 7 is giving the conclusion of this paper.

2. CONVENTIONAL APPROACH TO FIND INVERSE OF MATRIX THROUGH CRAMER'S RULE:

To represent the conventional approach two functions determinant (f_1) and minor (f_2)) have been made. We are assuming that time taken by both the functions are same.

For finding the inverse of a given matrix the invoking sequences of functions are shown in Table1. In Table 1 a sequel is showing the invoking and responding tine of any function.

In sequel 1 the f_1 will be called for A, The responding time of f_1 is Δt, so it will respond at $t_1+\Delta t$.

After getting response from f_1, f_2 will be invoked at t_2, in equation form it can be written as

$$t_2 = t_1 + \Delta t \qquad (1)$$

At t_3, f_1 will be invoked for getting determinant of M_{11}. t_3 can be evaluated by following time relation

$$t_3 = t_2 + \Delta t \qquad (2)$$

Putting the value of t_2 from time relation 1 to time relation 2

$$t_3 = t_1 + 2\Delta t \qquad (3)$$

$$t_n = t_1 + (n-1)\Delta t \qquad (4)$$

TABLE 1

Sequel Number	Time Instances: Invoke Function	Time Instances: Get Response	f1	f2
1	t_1	$t_1 + \Delta t$	$\|A\|$	
2	t_2	$t_2 + \Delta t$		M_{11}
3	t_3	$t_3 + \Delta t$	$\|M_{11}\|$	
4	t_4	$t_4 + \Delta t$		M_{12}
5	t_5	$t_5 + \Delta t$	$\|M_{12}\|$	
6	t_6	$t_6 + \Delta t$		M_{13}
7	t_7	$t_7 + \Delta t$	$\|M_{13}\|$	
-	-	-	-	-
-	-	-	-	-
k-2	t_{2n^2-2}	$t_{2n^2-2} + \Delta t$		M_{nn-1}
k-1	t_{2n^2-1}	$t_{2n^2-1} + \Delta t$	$\|M_{nn-1}\|$	
K	t_{2n^2}	$t_{2n^2} + \Delta t$		M_{nn}
K+1	t_{2n^2+1}	$t_{2n^2+1} + \Delta t$	$\|M_{nn}\|$	

k is the number of iteration for getting Matrix of minors of *A (M) and* calculated by the relation $k=2n^2$.

The time relation 4 presents that

$$t_{n^2} > t_{n^2-1} > t_{n^2-2} > \cdots > t_4 > t_3 > t_2 > t_1 \qquad (5)$$

Which shows that sequel 2 will be invoked after getting response from sequel 1; sequel 3 will be invoked after getting response from sequel 2. Sequel n will be invoked after getting response from sequel *n-1* and so on. All sequels are sequentially initiated services.

At time $t_{2n^2+1} + \Delta t$ we will get *M and* $|A|$. The A^{-1} will be calculated by using *M* and $|A|$. As *M* and $|A|$ have been calculated, so the time taken by next calculation for finding A^{-1} can be ignored and the time for calculating A^{-1} is given by

$$t_{2n^2+1} + \Delta t \qquad (6)$$

By putting the value of t_n from time relation 4 into 6

$$t_1 + (2n^2)\Delta t + \Delta t \qquad (7)$$

3. PROPOSED FRAMEWORK:

Figure 1 shows the framework for InverseOrchestra. It divides overall work into two services. Determinant service (S_1) and matrix minor service (S_2). The services are running on different nodes. Service S_1 computes the determinant of given matrix; service S_2 computes minor matrix for given row and column number. InverseOrchestra is coordinating software for the sequence of operations of two services. InverseOrchestra interacts with user and manages exceptions also.

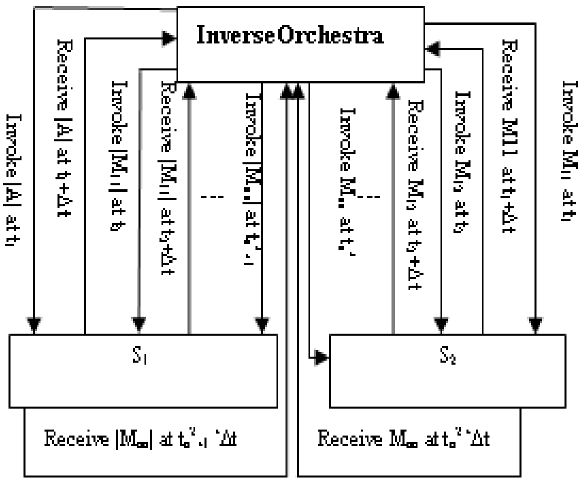


Fig. 1: InverseOrchestra

Timings of the sequences for Orchestration are based on following assumptions: the responding time (Δt) of S_1 and S_2 are same.

The number of iteration for getting Matrix of minors of *A* is j *and* calculated by the relation $j = n^2$.

Table 2 gives the timings of the sequences for computational services. The sequels described by the Table 2 show that S_1 and S_2 are parallel initiated services for different inputs.

TABLE 2

Sequel Number	Time Instances		Invoke S_1 for	Invoke S_2 for
	Invoke Service at time	Get Response at time		
1	t_1	$t_1 + \Delta t$	$\|A\|$	M_{11}
2	t_2	$t_2 + \Delta t$	$\|M_{11}\|$	M_{12}
3	t_3	$t_3 + \Delta t$	$\|M_{12}\|$	M_{13}
4	t_4	$t_4 + \Delta t$	$\|M_{13}\|$	M_{14}
5	t_5	$t_5 + \Delta t$	$\|M_{14}\|$	M_{21}
-	-	-	-	-
-	-	-	-	-
j	t_{n^2}	$t_{n^2} + \Delta t$	$\|M_{nn-1}\|$	M_{nn}
j+1	t_{n^2+1}	$t_{n^2+1} + \Delta t$	$\|M_{nn}\|$	

At time $t_{2n^2+1} + \Delta t$ InverseOrchestra will get *M and* $|A|$. The A^{-1} will be calculated by using *M* and | A|. As *M* and $|A|$ have been calculated, so the time taken by next calculation for finding A^{-1} can be ignored and the time for calculating A^{-1} is given by

$$t_{n^2+1} + \Delta t \qquad (8)$$

By putting the value of t_n from time relation 4 into 8

$$t_1 + (n^2)\Delta t + \Delta t \qquad (9)$$

4. EXTENSION IN PROPOSED FRAMEWORK:

In InverseOrchestra we have assumed that time taken by both the services (S_1 and S_2) are same. But at the time of implementation it has been observed that time taken by S_1 is much more than S_2.The extended version InverseOrchestraE of InverseOrchestra has been presented which orchestrate the four services(three determinant and one minor)for getting inverse of a matrix through Cramer's rule. The InverseOrchestraE is initiating all services in parallel.

The InverseOrchestraE is based on the assumption that it can get three minors from S_2 in the responding time of S_1.

Table3 is showing the sequence of service invoking and response. Here αt is the responding time of S_2 . S_{1a}, S_{1b} and S_{1c} are three determinant services and Δt is the responding time of these three determinant services.

In sequel 1 at time t_1 InverseOrchestraE will invoke S_2 for getting M_{11} and S_{1a} for getting |A|.As Δt is >= $3\alpha t$, so S_2 will respond at T1+ αt, but S1a won't respond, InverseOrchestraE will invoke S2 at $t_1 + \alpha t$ for M_{12} and will get response at $t_1 + 2\alpha t$, again in same sequel InverseOrchestraE will invoke S_2 for M_{13} at $t_1 + 2\alpha t$ and will get response at $t_1 + 3\alpha t$, S_{1a} will respond at $t_1 + \Delta t$.

After sequel 1 InverseOrchestra will have M_{11}, M_{12} and M_{13}, so in sequel 2 at time t_2 it will invoke S_2 for M_{14}, S_{1a} for $|M_{11}|$ and S_{1b} for $|M_{12}|$ and S_{1c} for $|M_{13}|$. It means it is initiating four services in parallel. S_2 will generate three minors (M_{14}, M_{15} and M_{16}) in the responding time of S_1.

Table 3

Sequel Number	Time Instances		Invoke S_2 for	Time Instances		Invoke S_{1a} for	Invoke S_{1b} for	Invoke S_{1c} for
	Invoke S_2 at time	Get Response From S_2 at time		Invoke S_1 at time	Get response From S_1 at time			
1	t_1	$t_1 + \alpha t$	M_{11}	t_1	$t_1 + \Delta t$	$\|A\|$		
	$t_1 + \alpha t$	$t_1 + 2\alpha t$	M_{12}					
	$t_1 + 2\alpha t$	$t_1 + 3\alpha t$	M_{13}					
2	t_2	$t_2 + \alpha t$	M_{14}	t_2	$t_2 + \Delta t$	$\|M_{11}\|$	$\|M_{12}\|$	$\|M_{13}\|$
	$t_2 + \alpha t$	$t_2 + 2\alpha t$	M_{15}					
	$t_2 + 2\alpha t$	$t_2 + 3\alpha t$	M_{16}					
	-	-	-	-	-	-	-	-
	-	-	-					
	-	-	-					
j	$t_{\lfloor n^2/3 \rfloor} + \beta - 1$	$t_{\lfloor n^2/3 \rfloor} + \beta - 1 + \alpha t$	M_{nn-2}	$t_{n^2/3} + \beta - 1$	$t_{2n^2/3} + \beta - 1 + \Delta t$	$\|M_{nn-5}\|$	$\|M_{nn-4}\|$	$\|M_{nn-3}\|$
	$t_{\lfloor n^2/3 \rfloor} + \beta - 1 + \alpha t$	$t_{\lfloor n^2/3 \rfloor} + \beta + 2\alpha t$	M_{nn-1}					
	$t_{\lfloor n^2/3 \rfloor} + \beta + 2\alpha t$	$t_{\lfloor n^2/3 \rfloor} + \beta + 3\alpha t$	M_{nn}					
j+1				$t_{n^2/3} + \beta + 1$	$t_{n^2/3} + \beta + 1 + \Delta t$	$\|M_{nn-2}\|$	$\|M_{nn-1}\|$	$\|M_{nn}\|$

The same sequence will be followed in next sequels.. For getting M the total iterations will be $n^2/3 + \beta$, Where $\beta = 0$, for n is such that $n\%3=0$; and $\beta=1$ for $n\%3 \neq 0$. $j= n^2/3 + \beta$. At time $t_{n^2/3} + \beta + 1 + \Delta t$ InverseOrchestraE will get *M and* $|A|$. The A^{-1} will be calculated by using M and $|A|$. As M and $|A|$ have been calculated, so the time taken by next calculation for finding A^{-1} can be ignored and the time for calculating A^{-1} is given by

$$t_{n^2/3} + \beta + \Delta t \qquad (10)$$

By putting the value of t_n from time relation 4 into 10

$$t_1 + (n^2/3 - 1)\Delta t \quad + \beta + 1 + \Delta t \quad (11)$$

From time relation 7, 9 and 11 it is clear that time taken by Conventional approach is approximately double from InverseOrchestra and time taken by InverseOrchestraE is approximately 1/3 from the conventional approach.

5. Implementation:

The framework InverseOrchestra and InverseOrchestraE have been developed using Remote Method Invocation (RMI) and extensive use of multithreading in java. For invoking the services from remote machines the RMI has been used. For initiating services in parallel or sequentially the use of multithreading has been done.

Fig. 2 is showing the implementation structure of InverseOrchestra. Two servers RMIServer for S_1 and RmiServerCo for S_2 have been developed. RMIClient is playing the role of InverseOrchestra. It has stub object of the servers and servers have the skeleton object of RMIClient. The stub object consists of an identifier of the remote object to be used, an operation number describing the method to be called and the marshalled parameters. It sends all these information to the server. The job of skeleton object at the server side are unmarshalling the parameters, calling the desired method on the real object lying on the server, capturing the return value or exception of the call on the server, again marshalling the return value, sending a package consisting of the value in the marshalled form back to the stub on the RMIClient. At RMIClient Two thread have been developed to call two servers in parallel or in sequential manner. For managing the sequence of operations synchronized methods have been used.

The experimental setup was on single machine, two local hosts were made for calculating the results. The configuration of machine was intel (R) core (TM) 2 Duo CPU with 2 GB RAM and 2.93 GHz clock frequency.

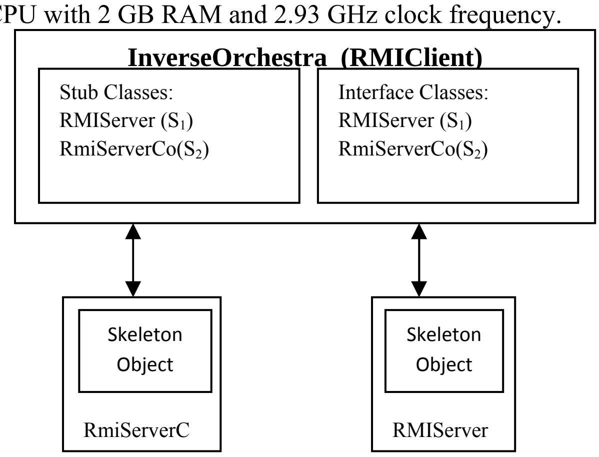


Fig2. InverseOrchestra Implementation

Fig. 3 is showing the implementation structure for InverseOrchestraE. Four servers RMIServer for S_{1a}, RMIServer_2 for S_{1b}, RMIServer_3 for S_{1c} and RmiServerCo for S_2 have been developed. RMIClient is playing the role of InverseOrchestraE.

RMIClient have four threads two manage the sequence of operations. An array of minors of size 3 has been made at the RMIClient. It can store three minors in the mean time of determinant calculation. It diverts the calculated minor to any of the server whichever is free.

The experimental setup was on three machines, one local host for S_2 has been made. The configuration of machines was intel (R) core (TM) 2 Duo CPU with 2 GB RAM and 2.93 GHz clock frequency.

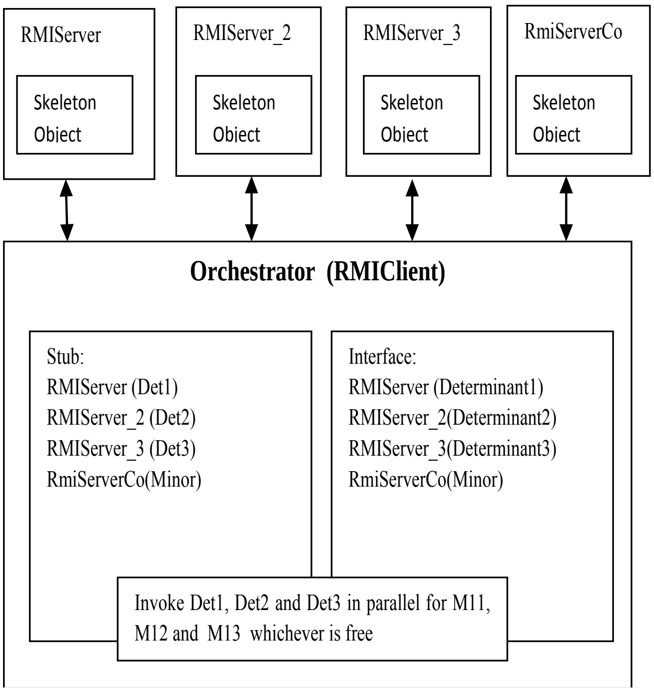


Fig. 3 InverseOrchestraE Implementation

6. RESULTS:

Table 4 is showing the speed of two frameworks InverseOrchestra, and InverseOrchestraE with reference to Conventional approach. From Table 4 it is clear that average speedup of InverseOrchestra is more 3 from the conventional approach and average speedup of InverseOrchestraE is more than 20 from the conventional approach. From table it is also clear that the response of InverseOrchestraE was very good up to 50×50 matrix, after that the timing calculation of minor gets apparent and RMIClient has to wait for minors to calculate the determinant. By increasing number of minor servers the response of InverseOrchestraE can be improved further. Chart 1 is showing the comparison between three approaches. Chart2 and Chart3 are showing the comparison of conventional approach with InverseOrchestra and InverseOrchestraE respectively.

Table 4

S.No.	Matrix Size	Speedup (Conventional/ InverseOrchestra)	Speedup (Conventional/ InverseOrchestraE)
1	10×10	3.567	42.812
2	20×20	3.389	49.873
3	30×30	3.918	41.168
4	40×40	2.662	19.575
5	50×50	3.717	18.823
6	60×60	3.847	13.725
7	70×70	3.519	8.880
8	80×80	2.945	5.716
9	90×90	3.101	4.790
10	100×100	3.722	4.332
Average Speedup		3.439	20.969

Chart 1

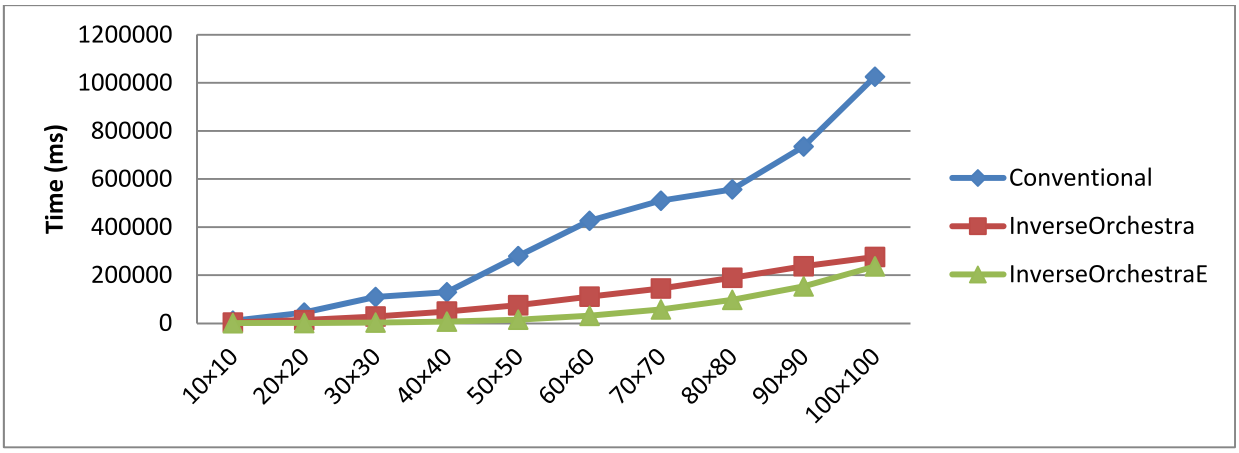


Chart 2

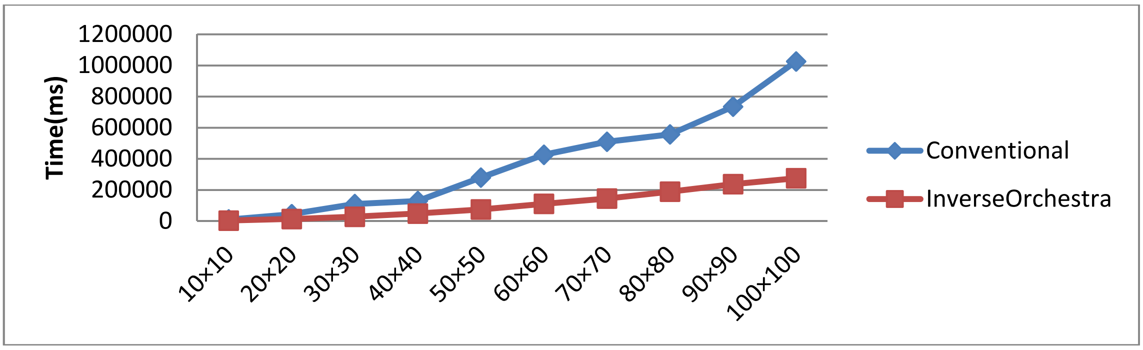
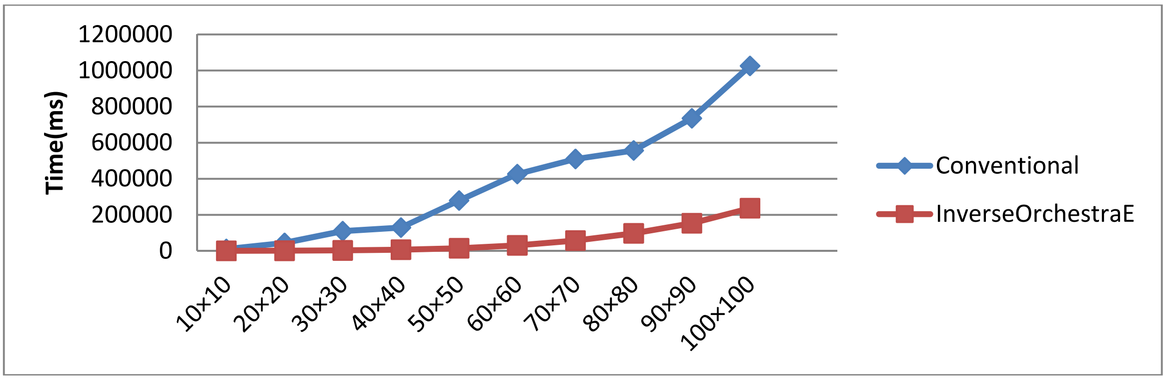


Chart 3



7. CONCLUSION:

Two frameworks InverseOrchestra and InverseOrchestraE have been presented and implemented. The frameworks are utilizing the property of Orchestration to speed up the matrix inversion. The method used for matrix inversion is Cramer's rule. The implementation is based on RMI and multithreading in java. The experimental results are much closer to theoretically derived time relations. Intermediately calculations done by frameworks such as determinant and adjugate matrix are also very useful in many applications.

REFERENCES:

[1] N.J. Higham , Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, 1996.

[2] Thomas Kailath, Ali H. Sayed, and Babak Hassibi, Linear Estimation, Prentice-Hall, NJ, 2000.

[3] P. McCullagh and J.A. Nelder, Generalized Linear Models, Chapmann and Hall, London, 1989.

[4] Ralph Byers, Solving the algebraic Riccati equation with the matrix sign function, J. Linear Algebra and its Applications, Volume 85, January 1987, pages 267-279.

[5] Jun Qiang Wu; Bose A., Parallel solution of large sparse matrix equations and parallel power flow, *IEEE Transactions on Power Systems*, volume 10, Issue 3, August 1995,pages 1343,1349.

[6] M. Cosnard and D. Trystram, Parallel Algorithms and Architectures, Blackwell North America, 1995.

[7] V. Kumar, A. Grama, A. Gupta, and G. Karypis , Introduction to Parallel Computing. Design and Analysis of Algorithms, The Benjamin/Cummings Publishing Company, 1994.

[8] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi and G. Zavattaro, JOLIE: a Java Orchestration Language Interpreter Engine, Electronic Notes in Theoretical Computer Science (ENTCS), June 2007, pages 19-33,

[9] N. Viswanadham and Kameshwaran,, Orchestrating a Network of Activities in the Value Chain, 5th Annual IEEE Conference on Automation Science and Engineering Bangalore, India, August 22-25, 2009.

[10] Yinong Chen and Xiaoying Bai , On Robotics Application in Service Oriented Architecture, The 28 international IEEE conference on Distributed Computing Systems Workshops,2008,pages 551-556.

[11] Lavanya Ramakrishnan, Jeffrey S. Chase, Dennis Gannon, Daniel Nurmi, Rich Wolski, Deadline-sensitive workflow orchestration without explicit resource control, J. Parallel and Distributed Computing, Volume 71, Issue 3, March 2011, pages 343-353.

[12] G.W. Stewart, On the adjugate matrix, J. Linear Algebra and its applications, Volume 283, Issues 1–3, 1 November 1998, pages 151-164.

[13] Elizabeth A. Cudney, Kioumars Paryani, Kenneth M. Ragsdell , Identifying Useful Variables for Vehicle Braking Using the Adjoint Matrix Approach to the Mahalanobis-Taguchi System, J. Industrial and Systems Engineering Volume . 1, Issue 4, 2008, pages 281-292.

[14] Chen Hui-ru; He Chun-lin, The Properties of Adjoint Matrix, *International Conference on Intelligence Science and Information Engineering (ISIE), 2011, August 2011,* pages 113-114.

[15] M. C. Pease "Methods *of Matrix algebra "Academic* Press, New York, 1965.