Towards Faster Givens Rotations Based Power System State Estimator

A. Pandian K. Parthasarathy Senior Member, IEEE
Department of Electrical Engineering
Indian Institute of Science
Bangalore, INDIA
email: pandian@suraksha.ee.iisc.ernet.in

S. A Soman
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai, INDIA
email: soman@ee.iitb.ernet.in

Abstract

Numerically stable and computationally efficient Power System State Estimation (PSSE) algorithms are designed using Orthogonalization (QR decomposition) approach. They use Givens rotations for orthogonalization which enables sparsity exploitation during factorization of large sparse augmented Jacobian. Apriori row and column ordering is usually performed to reduce intermediate and and overall fills. Column ordering methods, usually based on Minimum Degree Algorithm (MDA), have matured. However, their exists a significant scope for improving the quality of row ordering. This paper introduces a new row ordering technique for Givens rotations based power system state estimators. The proposed row processing method (VPAIR) requires a shift, from conventionally used row oriented QR decomposition implementation to a column oriented QR decomposition implementation. It is demonstrated that, the proposed column oriented QR decomposition algorithm which uses MDA for column ordering and VPAIR for row ordering can lead to a.milch faster PSSE. These aspects are justified by simulations on large power systems.

1 Introduction

PSSE problem can be formulated as an unconstrained non-linrar least squares problem:

PE-104-PWRS-0-08-1998 A paper recommended and approved by the IEEE Power System Analysis, Computing and Economics Committee of the IEEE Power Engineering Society for publication in the IEEE Transactions on Power Systems. Manuscript submitted May 26, 1997; made available for printing August 14, 1998.

$$\min \frac{1}{2} ||z - f(x)||_2^2 + \frac{\rho}{2} ||0 - c(x)||_2^2 \quad (1)$$

where:

- $c(x)$ - Non-linear Equations vector for zero injections ($p \times 1$)
- z - Measurement vector ($m-p \times 1$)
- x - State vector consisting of node voltages and angles($n \times 1$)
- $f(x)$ - Non-linear Equation vector for measurement, vector z ($m-p \times 1$)
- ρ - Large penalty for enforcing zero injection constraints

At each iteration k of PSSE, following linearized Least Square (LS) problem is solved.

$$\min \left|\left| [W^{-1}] \left[z - f(x^k) - \left[\frac{\partial f(x^k)}{\partial x}\right]^T \Delta x^k \right] \right|\right|_2^2 + \frac{\rho}{2} \left|\left| \left[\frac{\partial c^k}{\partial x}\right]^T \Delta x^k + c(x^k) \right|\right|_2^2 = \left|\left| [W^{-1}] \left[\Delta \hat{z}^k - \hat{H}^k \Delta x^k\right] \right|\right|_2^2 \quad (2)$$

Where:

$$\hat{H}^k = \left[\begin{array}{c} \frac{\partial f(x^k)}{\partial x} \\ \frac{\partial c(x^k)}{\partial x} \end{array} \right]^T \quad and \quad \Delta \hat{z}^k = \left[\begin{array}{c} z - f(x^k) \\ -c(x^k) \end{array} \right]$$

Large sparse LS solver is the heart of a power system state estimator. LS problem **(2)** can be solved by various methods[1] which include Normal Equations (NE) approach, QR decomposition approach and method of Peters and Wilkinson.

QR decomposition based PSSE implementations have been found to be numerically stable [1] as they use unitary transformations arid satisfactorily handle numerical ill-conditioning encountered in PSSE. However. numerical stability is obtained at cost of additional computations. Unlike NE approach which- directly computes Cholesky factorization of matrix $\hat{H}^T \hat{H}$, QR decomposition involves elementary rotation operations on matrix $\boldsymbol{H}$ which create

0885-8950/99/$10.00 © 1998 IEEE

intermediate fills. Conceptually, *QR* decomposition algorithms annihilate non zero elements below the diagonal of matrix $\hat{H}$. Intermediate fills do not appear in final upper triangular matrix R and are annihilated at intermediate steps itself. However, they do increase the amount of computations. It is well known that sparsity can be exploited in QR factorization by using Givens rotations. The processing sequence of rows and columns of matrix H affects the computational effort required in factorization. Therefore, efficient column and row ordering methods are used to reduce the computational effort in factorization. Matrix R obtained by *QR* decomposition and NE methods have identical structure. Hence, ordering algorithms like Minimum Degree Algorithm (MDA) used for ordering columns in NE method are also used for column ordering in *QR* factorization[2]. Though the final structure of matrix R obtained in NE method and Givens rotations method are identical, computational efforts required in Givens rotations is proportional to the fill ins (both intermediate and fills in matrix R) generated during the rotations.

Popular PSSE implementations are based on LS solver by George and Heath[2] . George and Heath's algorithm is a Row Oriented Processing (ROP) approach, wherein rows are sequentially processed. Algorithm generates a sequence of upper triangular matrices $R', R^2, \ldots, R^m$ where R^m is upper triangular matrix R corresponding to the QR decomposition of H. The algorithm uses simplified row and column ordering methods *npriori* to decomposition for reducing intermediate fills and the number of non zeros in matrix R. Vempati et *al.*[3] proposed further enhancements like square root free factorization. They report that column ordering by MDA and row ordering by numbering rows in an ascending order of maximum column subscript significantly reduce computation time. In case of a tie due to two or more rows having same maximum column subscript, the row which has least sum of column subscript is processed first. An added advantage of row oriented processing is that any new measurement can be easily incorporated by simple processing of an additional row.

This paper proposes an alternate Column Oriented Processing (COP) algorithm for Givens rotations based PSSE. It is shown that more efficient row and column ordering methods can be developed within framework of column oriented sparse *QR* decomposition. In fact, our simulations (section *5)*, show **9.75** times reduction in execution time and approximately 15 times reduction in number of Givens rotations and intermediate fills with respect to ROP approach for a *1044* node practical system.

2 Column Oriented QR Decomposition

Column oriented QR decomposition algorithm processes columns of matrix $\hat{H}$ sequentially for computing upper triangular matrix R. This algorithm produces a sequence of upper trapezoidal matrices $R(1\times n), R(2\times n), \ldots, R(n\times n)$ where $R(n \times n)$ corresponds to matrix R obtained by *QR* factorization of $\hat{H}$. At i^{th} step of algorithm, all non zeros below the diagonal of i^{th} column are annihilated. For example, in the first step of algorithm, non zero elements below (1,1) in first column of H are annihilated by selectively applying Givens rotations. In the second step non zero elements of second column below (2,2) are annihilated. Finally, at n^{th} step all non zeros below (n,n) in n^{th} column are annihilated and matrix H is transformed to upper triangular matrix R appended with $m - n$ null rows.

Illustration of column oriented rotations:
The sparse column oriented *QR* decomposition is now illustrated for a sparse 4X3 matrix A. Initial matrix has a structure as shown below:

$$A = \begin{pmatrix} X & & X \\ X & X & \\ & X & \\ X & X & \end{pmatrix}$$

Matrix A is transformed into R in three major steps as follows.

Step # 1: Sequence of Givens rotations {(1,1) & *(2.1))* and {(1,*1)*& (4,1)} are applied to annihilate elements (2,1) and (4,1) below diagonal (1,1) of first column. Thus, this step involves operations on row pairs (1,2) and (1,4). Note that, rotations create three fills represented by ⊗.

$$A_1 = \begin{pmatrix} X & \otimes & X \\ & X & \otimes \\ & X & \\ & X & \otimes \end{pmatrix}$$

Step # 2: In this step, the non zeros below diagonal (2,2) of second column of matrix A_1 are annihilated. This involves Givens rotations operations on row pairs (2,3) arid (2,4). Note that an additional fill is also created in this step.

$$A_2 = \begin{pmatrix} X & X & X \\ & X & X \\ & & \otimes \\ & & X \end{pmatrix}$$

Step # 3: Finally, in this step the non zero (4,3) in column-3 is annihilated by applying Givens rotation to row pair **(3.1).**

$$A_3 = \begin{pmatrix} X & X & X \\ & X & X \\ & & X \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

In column oriented deconiposition, a column once processed is not, required in later stages. At i^{th} step, no intermediate fills can be created in columns $1, \ldots, (i-1)$ which have been already processed. The elements to be annihilated in column i can have row indices $i, \ldots, m$. *i.e.* rows $1, \ldots (i-1)$ are not used in steps $j \geq i$. Thus at every step size of *active* sub matrix to be processed is reduced by a row and a column. In fact rows 1 to $(i-1)$ at i^{th} step constitute first $i-1$ rows of matrix R. This can be contrasted with row oriented processing where complete matrix R is updated at each step which corresponds to annihilation of a row.

3 Ordering Strategies

Column ortitring in column oriented approach can be done either on the fly or apriori to factorization. Duff[4] developed an approach for column ordering which accounts for intermediate fills also. In this approach, at i^{th} step, an unprocessed column j is selected for rotations if number of non zeros below the element (i,j) is minimum. In Duff's column ordering approach the final structure of R also depends on intermediate fills.

In George & Heath's algorithm: column ordering strategy is based on applying MDA to structure of $\hat{H}^T\hat{H}$. This reduces number of fills in matrix R. Structure of matrix R does not depend upon row ordering. Therefore, column ordering is done independent, of row ordering. Also, column ordering is performed apriori to row ordering as row ordering depends upon the column subscripts of non zeros. Also in George and Heath's algorithm, pivot is chosen as diagonal of partially developed matrix R. At i^{th} step, an element $\hat{H}(i,j)$ is annihilated by choosing $R^{i-1}(j,j)$ as pivot. Note that this may introduce intermediate fills in i^{th} row of matrix $\hat{H}$ which have to be annihilated subsequently. It can be observed that only matrix R^{i-1} and i^{th} row of matrix $\boldsymbol{H}$ are modified. The intermediate fills depend mainly upon sequencing of rows (row-ordering). There is only a single degree of freedom in sequencing of rows as pivot element is always fixed to be the diagonal element of matrix R. Such an approach of sequencing can be looked upon as a *fixed pivot* row ordering scheme Duff[4] developed a fixed pivot strategy for COP approach wherein the sparsest row is chosen as pivot row and succeeding rows are processed based on minimizing fills in pivot row.

Variable pivot strategies permit multiple pivot choices. For example, in step #1 of column oriented approach, element (2,1) can be annihilated by selecting the pivot (4,1) instead of (1,1) as shown. Getieraiizing, at i^{th} step, if there are r number of rows to be processed in a given column *i.e.* $(r-1)$ non zeros in that column are to be zeroed, then, first zeroing can be achieved by applying Givens rotations on anyone of possible ${}^rC_2$ (total number of combinations in r rows choosing 2 rows at a time) row pairs. This freedom can be exploited to reduce *local* computations by choosing a favorable row pair. Thus; variable pivot strategies can reduce computations in comparison to fixed pivot strategies. In fixed pivot based implementation illustrated in section 2, 5 rotations corresponding to following row pairs (1,2), (1,4), (2,3), (2,4) and (3,4) were required. Instead if the variable pivoting scheme is used with following sequence of rotations (2,4),(1,2), **(3.4)** and (2,3) only 4 rotations are required thereby saving a rotation (20% saving). In practice, improvements in overall execution time will depend upon saving in computations and additional overheads incurred in implementing variable pair strategy.

4 **VPAIR** : On Line **Row** Ordering

Gentleman[5] developed a variable pivot strategy for COP approach, wherein zeroing was first performed on sparser row pairs. None of row ordering scheme is complete. since, they often do not come close to globally minimizing intermediate fills. Recently, Robey *et al.*[6] have proposed a new variable pivot strategy VPAIR for sequencing of row pairs. The rows are sequenced based on number of fill ins created during the rotation. This is measured by counting the number of mismatch in non zero pattern of row pair considered for rotation. The row pair which has least fill in creation (best, matched pair) is chosen for rotation. Fill in analysis for choosing the row pair is performed only for next step of rotation. As rotation operation on a row pair changes the structure of both rows, VPAIR ordering is best done online. Hence, ordering becomes dynamic. The tie breaking strategy recommended by Robey *et al.* for VPAIR is based on choosing the sparsest row pair. On a collection of sparse matrix test problems, Robey *et al.* have compared various methods wherein column ordering is either performed by Duff's approach or by MDA. Row ordering based on Gentleman's variable pivot, approach, Duff fixed pivot approach and VPAIR are considered with above column ordering methods. In general, best results are obtained by using MDA for column order-

ing and VPAIR for row ordering. Robey *et al.* [6] have reported that VPAIR row ordering has superior performance compared to other row ordering methods for finite elernent problems.

Column oriented processing logically requires dynamic data structure as fills in structure of intermediate matrices cannot be predicted before hand. However, while allocating memory: heuristics can be used to allocate memory apriori to factorization considering a reasonable safe bound on maximum possible intermediate fills and fills in R. We observed that in PSSE a reasonable bound on memory allocation is 4 times the number non zero elements in original matrix H. Once the memory allocation is clone apriori to factorization, static linked list based data structiire can be easily implemented. In George and Heath's algorithm, implementation is based on static data structure. This assumes that symbolic factorization of matrix $\hat{H}^T\hat{H}$ *i.e* structure of R is already available. Symbolic factorization of matrix $\hat{H}^T\hat{H}$ to obtain structure of matrix R strictly requires a dynamic data structure. But a static linked list, based data structure can be developed by using heuristic to predict the number of non zero elements in R. **A** reasonable bound on memory allocation is 2 times that of non zero elements in matrix $\hat{H}$. In view of above discussion, we can say that both row oriented and column oriented versions of sparse QR decomposition require dynamic data structure, but instead can be implemented using static linked list data structure. However, in George and Heath's scheme, this aspect is more transparent to programmer as it can be shown that there is sufficient space in structure of matrix R and working vector x to accommodate intermediate fills.

5 Results

In this section, we present results for 23 [i]319 and 1044 node systems. These systems have various measurements like, line flows, load injections, zero injections and voltage measurements. For 23, 319 and 1044 bus systems total number. of measurements are 148, 1436 and 7194 respectively. Implementations are done on a IBM RS6000 machine using C++ programming language. Figure (1) shows sparsity structures of matrices H and R for a 1044 node system. The matrices are imported in MATLAB and spy function is used to generate sparsity structures. The row ordering does not affect, sparsity structure of matrix R and hence same sparsity structure is obtained irrespective of the type of row ordering used. We compare and contrast, the performance of power svstem state estimators based on following algorithms. **ROP:** This implementation is same as that of Vempati *et al.* Row oriented processing is used with apriori column and row ordering.

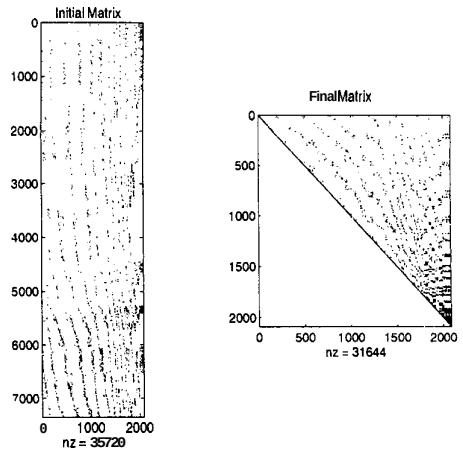

Figure 1: Initial and Final matrix structure for 1044 node system

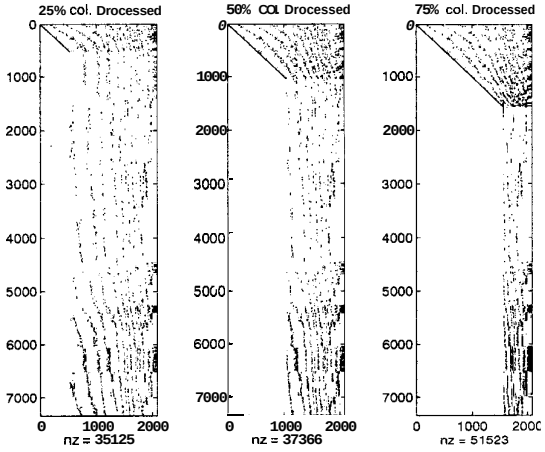

Figure 2: Intermediate sparsity structure for 1044 bus system for COP1

MDA is used for column ordering and rows are ordered in ascending order of maximum column subscript with minimum sum of column subscripts as tie breaking strategy. **COP1:** This implementation is based on column oriented processing with apriori row and column ordering methods as used In ROP.

COP2: This implementation is also based on column oriented processing. Unlike fixed pivot implementations, COP2 uses variable pivot strategy VPAIR for ordering rows. Columns are ordered apriori using MDA.

5.1 COP2 'VsCOP1

Figures (2,3) show sparsity structures of intermediate matrices for 1044 node system using implementations COP1

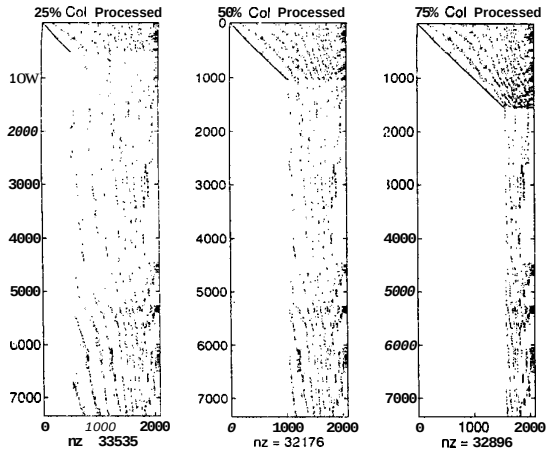

Figure 3: Intermediate sparsity structure for 1044 bus system for COP2

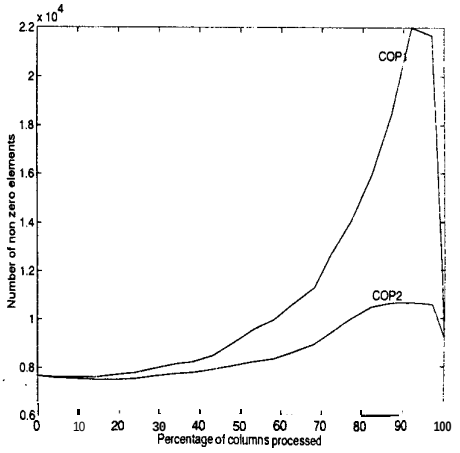

Figure 5: Number of non zero elements for 319 bus system during factorization

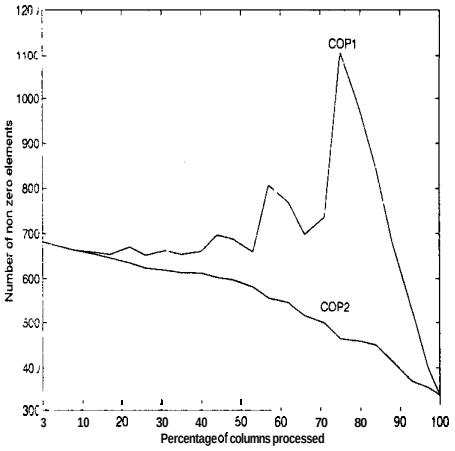

Figure 4: Nuniber of non zero elements for 23 bus system during factorization

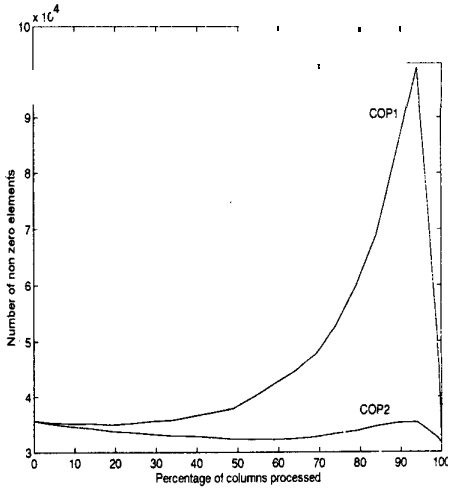

Figure 6: Number of non zero elements for 1044 bus system during factorization

and COP2. For 23, 319 and 1044 node systems, with 75 % processing of columns, COP1 has 115.84%, 34.69 % and 56.62% more non zeros compared to COP2. Clearly. COP2 requires less flops for obtaining matrix R. At any stage of processing, better performance of COP2 in comparison to COP1 can be attributed to direct on line effort of COP2 to minimize intermediate fills. Approach COP2 exploits advantages due to variable pivots. This can be contrasted with COP1 which uses fixed pivot strategy with apriori row ordering. Figures (4, 5 & 6) show number of non zero elements being recorded at every 5% processing of columns for 23, 319 and 1044 node systems. For all systems, the curves show a steep increase in non zero elements at intermediate steps of COP1 as compared to COP2. The nature of curve of non zero elements at various stages of processing for COP1 can be explained as follows:

Annihilation of an element (i,j) by rotation with element (j,j) introduces non zeros corresponding to union of sparsity structures of rows i and j of active sub matrix. The new active sub matrix is thus less sparse than active matrix before rotation. An upper bound on the number of non zeros in intermediate matrices depends upon the size of active sub matrix and number of rows of matrix R created so far. This upper bound on number of non zeros in intermediate steps equals product of number of rows and columns of active sub matrix plus number of non zeros in matrix $R(j \times n)$. Note that processing of j^{th} column creates corresponding j^{th} row of matrix R. Every

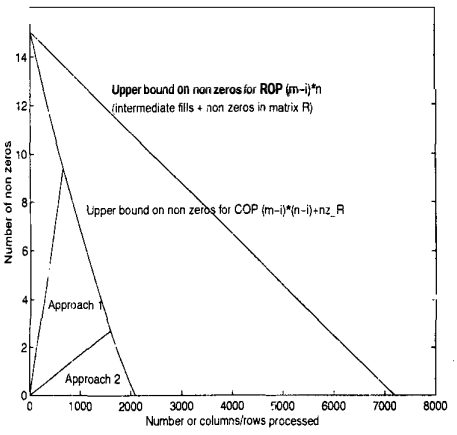

Figure 7: Upper bound on non zeros with various approaches for 1044 node system

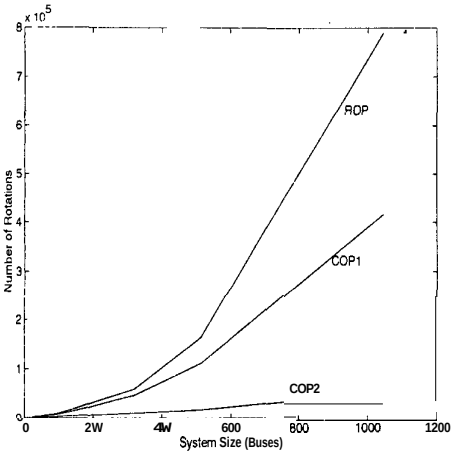

Figure 8: Number of Rotations required for various system?

consecutive step reduces the size of active sub matrix by a row and a column. Thus, upper bound on number of non zeros decreases from mn (step 0) at beginning to nz_R (step n), where nz_R is number of non zero elements in matrix R. In initial stages of processing, number of non zeros is far less than the corresponding upper bound. As processing progresses number of non zeros increases while upper bound reduces. A more realistic bound on overall maximum nmnber of non zeros can be obtained from intersection of curve of growth of non zeros and curve of upper bound of non zeros (Figure 7). As maximum number of non zeros cannot exceed the upper bound discussed earlier, number of non zeros should reach a maximum at some intermediate step and then reduce, finally, resulting in matrix R.

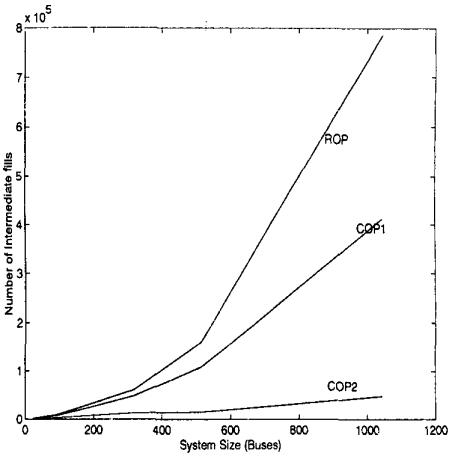

Figure 9: Number of intermediate fills for various systems

If slope of curve 1 representing the growth of non zeros at each step for a given approach 1 is high, it intersects curve of upper bound in earlier stages of processing (Figure 7). This implies that maximum possible number of non zeros in active sub matrix is large and hence overall computational effort(flops) may be more. On the other hand, if growth of intermediate fills in an approach (say approach 2) is well controlled, the point of intersection with the curve of upper bound is delayed and hence, maximum possible non zeros is reduced. In illustration of figure (7), we have assumed for simplicity a constant rate of increase of non zeros at all stages. In case of approach COP2, variable pivot strategy (VPAIR) is used in such a fashion so as to minimize intermediate fills for each annihilation. Hence, the rate of growth of non zeros is small and the maximum number of non zeros are reduced resulting in better performance of COP2 in comparison to alternate method COP1.

5.2 COP Vs ROP

The anatomy of annihilating an element (i,j) is identical in ROP and COP approaches. However, next non zero to be eliminated in ROP corresponds to next non zero in row i after annihilation of (i,j). In case of COP next non zero to be annihilated corresponds to a non zero element of column j. In case of COP1, all non zero elements in column j are annihilated using the fixed pivot (jj). While in case of COP2, the choice of non zero to be annihilated and its pivot corresponds to row pair that introduces least number of intermediate fills. Both COP1 and ROP use fixed pivot strategies. Assuming for simplicity and without loss of generality that, rows in ROP are eliminated in decreasing order of row index *i.e.* row to be eliminated first is numbered last, it can be seen that active sub matrix at i^{th}

step corresponds to first $m-i$ rows with all n columns. Thus the upper bound on number of non zeros in ROP after i^{th} step equals $(m-i) \times n$. This bound is clearly larger than $(m-i) \times (n-i)$ bound (neglecting contribution of matrix R) for COP. This suggests that with similar mechanisms (ordering and pivoting strategy) to control growth of non zeros (intermediate fills), COP approach should have better performance than corresponding ROP approach (see fig 7). In fact, such a behavior is observed in simulations in PSSE problems (see Figures 8 and 9) where it is observed that COP1 and COP2 outperform ROP. Actual performance also depends upon overheads involved. In case of COP2, this overhead corresponds to finding the best pair that gives least number of intermediate fills. If at a given step, r is the column degree of first column in active sub matrix then this requires ${}^{r}C_{2}$ evaluations of intermediate fills (0 flops). Overall execution time for PSSE using ROP, COP1 and COP2 for various systems (of sizes 23, 89, 319, 514 and 1044) are illustrated in figure (10). Figure clearly shows superior performance of COP2 in comparison to COP1 and ROP.

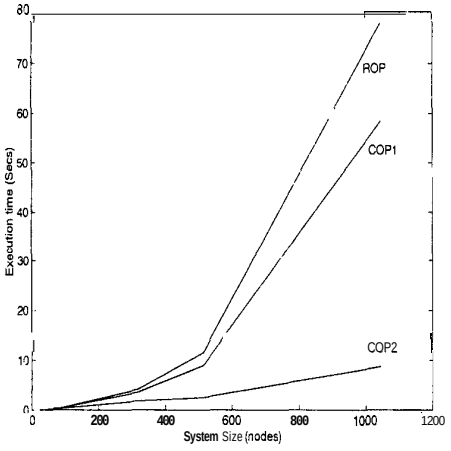

Figure 10: Total execution time for various systems

6 Conclusions

This paper explored various ways of enhancing the performance of Givens rotations based Power System State Estimator. A strong case was argued for shifting the PSSE implementation from ROP approach to COP approach. Column oriented processing permits variable pivot selection which in turn leads to a direct control on growth of intermediate fills (VPAIR ordering). In case an additional measurement has to be processed then the original matrix can be viewed as $\begin{bmatrix} R \\ 0 \end{bmatrix}$ appended with row for measurement to be processed. In such a case, at most one element per column has to be annihilated. Hence, processing of additional measurement becomes identical to that in ROP.

7 References

[1] L. Holten, A. Gjelsvik, F. F. Wu, et al., "Comparison of different methods for state estimation", *IEEE Transactions on Power Systems*, Vol.3, No.4, Pages:1798-1806,1988.

[2] Alan George and M. T. Heath, "Solution of sparse linear least squares problems using givens rotations", *Linear Algebra and its Applications* , Vol.34, pages:69-83,1980

[3] N. Vempati, I W Slutsker and W F Tinney, "Enhancement to givens rotations for power system state estimation", *IEEE Transactions on Power Systems*, Vol.6, No.2. pages:842-849, 1991

[4] I. S. Duff, "Pivot selection and row ordering in Givens reduction on sparse matrices", *Computing*, Vol.13, pages:239-248, 1974

[5] W. M. Gentleman, "Row elimination for solving sparse linear systems and least squares problems", *Lecture Notes in Mathematics, 506*, Springer-Verlag, New York, pages: 122-133, 1975

[6] Thomas H. Robey and Deborah L. Sulsky, "Row ordering for a sparse QR decomposition", *SIAM Journal of Matrix analysis & applications*, Vol.15, No. 4, pages:1208-1225, 1994

[7] A. A. A. El-Ela, " Fast and accurate technique for power system state estimation", *IEE Proceedings - C*, Vol. 139, No. 1, pages:7-12, 1992

8 Biography

A. Pandian received his M.Tech. in Computer Engineering from Mysore University, INDIA an? is presently working for his Ph.D in Indian Institute of Science (IISc), INDIA. His research interest are in real time state estimation, sparse matrix computations and sparse optimization problems.

S. A. Soman received M.E & Ph.D in Electrical Engineering from IISc, INDIA. Presently, he is working as Faculty at Department of Electrical Engineering, Indian Institute of Technology, Bombay, INDIA. His research interests include power system state estimation, reactive power control and optimization, sparse matrix computations, practical optimization and parallel computing

K. Parthasarathy is Professor at Electrical Engineering Department, IISc, INDIA. His research interests include computer aided protection, control of power systems and energy management systems.