The Fourth International Symposium on
Personal, Indoor and Mobile Radio
Communications (PIMRC'93)
Yokohama, Japan, September 8-11, 1993

A Novel Algorithm for Dynamic Updating of Decorrelator Coefficients in Mobile DS-CDMA

S.S.H Wijayasuriya, G.H Norton and J.P McGeehan
University of Bristol
Centre for Communications Research
Queens Building, University Walk
Bristol BS8 1TR, United Kingdom
Email *hans@uk.ac.bristol.comms-research*, Fax : +44 272 255265

1 Introduction

The merits of decorrelating solutions to the near-far problem have been established [1]. Decorrelating solutions are clearly optimal in mobile radio environments especially when applied together with RAKE diversity combining [2]. The Sliding Window Decorrelating Algorithm (SLWA) as proposed initially in [1] is based on finite sequence length decorrelation and hence has a much lower recomputation cost compared with the standard decorrelator [3]. Here we investigate the computational requirements of a decorrelating solution and propose a novel algorithm for the dynamic updating of the interference cancellation coefficients. The resulting architecture lends itself to fully parallel ASIC implementation thus making finite sequence length decorrelation comparable to sub-optimal techniques proposed in the literature [4, 5] in complexity.

2 Finite Sequence Length Decorrelation

Central to the implementation of a decorrelating algorithm is the solution of the linear system [1]. It is important to split the solution into two components. The first is the inversion or LU decomposition of the system matrix. The second component involves repeated solution of the system (using the already computed matrix inverse or decomposition) for different RHS (corrected matched filter output vectors in SLWA) vectors. The former operation needs to be performed on a relatively infrequent basis. Recomputation is required if the timing configuration changes significantly, dynamic selection of multi-paths is applied or voice activity is to be exploited. The cost of performing repeated solutions however, determines the cost of each iteration of the algorithm and has to conform to the overall processing delay requirements of the system.

We first examine the per-iteration (symbol interval) operations. Iterations are implemented inexpensively using a simple Zero Force Equaliser structure - see Figure (1). Each user (or multi-path) being decorrelated is assigned a receiver similar to a zero force equaliser. The tap weights for the equaliser are derived from the inverse cross correlation matrix. It is clear that

$$d_k(i) = \Re^{-1}_{k,i}.\mathbf{X} \tag{1}$$

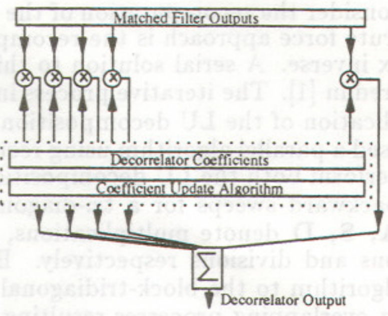


Figure 1: Zero Forcing Decorrelator

where $d_k(i)$ is the decorrelator output for the i^{th} symbol interval of the k^{th} user, $\Re^{-1}_{k,i}$ is the k^{th} row within the i^{th} block of the inverse of $\Re$, and $\mathbf{X}$ is the corrected matched filter vector. If the total number of paths/users being decorrelated is K and the window size N, the degree of parallelism will ideally be NK. However it is conceivable that lower orders of parallelism may be used with the rows of the inverse matrix being cycled through the filter taps. It must be noted that specialised high speed ZF equaliser IC technology is readily available making the decorrelator the simplest interference cancellation technique to implement on a per-iteration basis.

2.1 Competing Techniques

The decorrelator has a low per-iteration cost due to its standard zero-forcing operation. This makes the effect of the number of users on the processing delay negligible in the light of currently available zero-force IC equaliser technology. In contrast CDMA-IC [4] is inherently a serial solution based on successive cancellation of users proposed by Viterbi [6]. A channel gain table needs to be maintained and updated in response to the dynamic nature of the mobile radio channel. The bulk of the processing is done on a per-iteration basis, and additional users, or multi-path combining represent additional steps in a serial process.

There has been an increased level of interest in Minimum Mean Squared Error (MMSE) algorithms for interference rejection due to their modest demands on information regarding interfering users. It is thus ideal for a de-centralised solution e.g. at the mobile handset. The training of such algorithms is likely to represent a significant system overhead (on the mobile to base link) due to independent variation of the multi-user signals. Training requirements (frequency of training and adaptation algorithms which are critical factors for a mobile application) have not yet been investigated in the literature. A large part of the processing once again translates to a per-iteration processing load. The disadvantage of the decorrelator lies with the recomputation overhead. The recomputation time is critical in order to avoid deterioration of overall system performance due to inaccurate correlation data. However the recomputation can be performed off-line so as not to interrupt the normal functioning of the system.

2.2 Recomputation of Linear System

We now consider the recomputation of the linear system. A brute force approach is the recomputation of the matrix inverse. A serial solution to this problem is considered in [1]. The iterative process involves the serial application of the LU decomposition. Stone [7] has proposed a parallel algorithm using recursive doubling to perform both the LU decomposition and the forward/backward sweeps for a tri-diagonal matrix. Let **M, A, S, D** denote multiplications, additions, subtractions and divisions respectively. By extending this algorithm to the block-tridiagonal case (and identifying overlapping processes resulting in further reductions in serial computation time) we make an estimate of the operations involved as given below.

$$\begin{aligned}\text{LU Cost} &: ((4N-3)K^3+\frac{K^3}{2})\mathbf{M}+((4N-3)K^3\\ &+\frac{K^3}{2})(N-1)K^2)\mathbf{A}+\frac{NK^2}{2}\mathbf{D}\\ //\text{LU Cost} &: 5.Log_2N.K^3\mathbf{M}+5.Log_2N.K^3\mathbf{A} \qquad (2)\end{aligned}$$

Comparing the parallel and serial solutions in equation (2), we conclude that there is no advantage in parallelising the algorithm unless $N >> K$. Since this is not met in practice, we investigate a different strategy.

3 Filter Coefficient Updating Algorithm

We propose a novel algorithm for updating the filter coefficients in response to changes in the correlation coefficients. using a fully parallel architecture. The architecture lends itself to a direct trade-off between parallelism and recomputation time. The algorithm is based on the Sherman-Morrison Formula [8] and exploits the sparse nature of the block tri-diagonal system matrix. We consider a situation where the code synchronisation point of one user changes to such an extent that there results a significant change in the cross-correlation between this user and one or more other users in the system. The 'significance' is clearly determined by system tolerance, capacity and signal to noise ratio, and can be determined by a subsidiary system operating off-line or in parallel with the main algorithm. We refer to Figure (2) in the following discussion. A change with respect to one user or path will result in changes in the elements of a single row (and column) in the matrices $R(0)$, $R(1)$ and $R(-1)$.

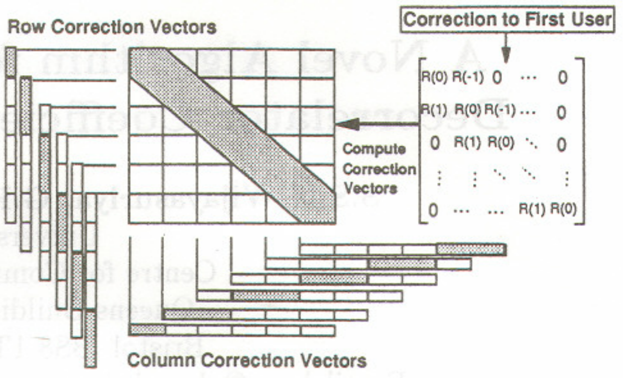


Figure 2: Correction Vectors - Example

It will only be in the worst case that all the cross correlation coefficients related to the user concerned will change significantly. From the structure of $\Re$, and the fact that it is made up of the matrices $R(0)$, $R(-1)$ and $R(1)$, a change to the timing of a particular user will result in both row-wise and column-wise corrections. N rows and an equal number of columns need to be corrected. For clarity we consider only row-wise corrections. Let $\mathbf{v}$,$\mathbf{u}$ be NK vectors. Let $\mathbf{v}$ be a row correction vector containing the 'corrections' to the row of $\Re$ being considered. The sparsity of the vector $\mathbf{v}$ will depend on the number of coefficients affected. In the worst case these vectors will have only $2K$ contiguous non-zero elements due to the sparsity of the block tridiagonal matrix and the fact that $R(-1)$ and $R(1)$ are lower and upper triangular matrices respectively (see Figure 2). We note that

$$[\mathbf{u}\otimes\mathbf{v}]_{i,j}=\mathbf{u}[i].\mathbf{v}[j] \qquad (3)$$

if $\mathbf{u}$ is the unit vector $\mathbf{e}_j$, then the correction vector can be applied (added) to the j^{th} row of $\Re$ as follows

$$\Re \longrightarrow \Re + A \qquad (4)$$

where

$$A=\mathbf{u}\otimes\mathbf{v}$$

The new inverse matrix is then $(\Re+A)^{-1}$. Note that

$$\begin{aligned}&(\Re+A)^{-1}\\ &= (I+\Re^{-1})^{-1}\Re^{-1} \qquad (5)\\ &= (I-\Re^{-1}A+(\Re^{-1}A)^2-(\Re^{-1}A)^3+\cdots)\Re^{-1}\end{aligned}$$

From the associativity of outer and inner products, and replacing A by $\mathbf{u}\otimes\mathbf{v}$ we have

$$(\Re+\mathbf{u}\otimes\mathbf{v})^{-1}=\Re^{-1}-\Re^{-1}.\mathbf{u}\otimes\mathbf{v}.\Re^{-1}(1-\beta+\beta^2-\beta^3+\cdots) \qquad (6)$$

where

$$\beta=\mathbf{v}.\Re^{-1}.\mathbf{u}$$

so that

$$(\Re+\mathbf{u}\otimes\mathbf{v})^{-1}=\Re^{-1}-\frac{(\Re^{-1}\mathbf{u})\otimes((\Re^{-1})^T\mathbf{v})}{1+\beta} \qquad (7)$$

Given the matrix $\Re$ and the two vectors $\mathbf{u}, \mathbf{v}$, we need only perform two matrix vector multiplications and a vector dot product as follows

$$\mathbf{z} = \Re^{-1}\mathbf{u} \quad (8)$$

$$\mathbf{w} = (\Re^{-1})^T\mathbf{v} \quad (9)$$

$$\beta = \mathbf{v}.\mathbf{z} \quad (10)$$

then

$$\Re^{-1} \longrightarrow \Re^{-1} - \frac{\mathbf{z} \otimes \mathbf{w}}{1+\beta} \quad (11)$$

Without loss of generality we consider a correction initiated by the first user and the corresponding row in the second block of $\Re$. The edge blocks are special cases (even more sparse) to which we will return. $\mathbf{u}$ is in this case the $(K+1)^{st}$ unit vector and $\mathbf{v}$ is all zero except for the first $2K$ elements. We require only one such correction vector, since the others are easily derived by a cyclic shift. In any case only the non-zero portion is required as will be seen below. Consider each of the equations (8),(9),(10) in turn.

(a) $\mathbf{z} = \Re^{-1}\mathbf{u}$
Since $\mathbf{u}$ is the $(K+1)^{st}$ unit vector, $\mathbf{z}$ is clearly the $(K+1)^{st}$ column of $\Re^{-1}$. We denote x^{th} column of a matrix M by M_x. We then have

$$\mathbf{z} = (\Re^{-1})_{K+1} \quad (12)$$

(b) $\mathbf{w} = \mathbf{v}^T\Re^{-1}$
Since $\mathbf{v}$ has only $2K$ nonzero elements we need consider only $2K$ successive rows of $\Re^{-1}$, in this case beginning with the first row. Define $\mathcal{P}$ to be this submatrix of dimension $2K*NK$ and $\tilde{\mathbf{v}}$ to be a $2K$ vector consisting of the non-zero portion of $\mathbf{v}$. We then have

$$\mathbf{w} = \tilde{\mathbf{v}}^T\mathcal{P} \quad (13)$$

Next consider the term
(c) $\beta = \mathbf{v}^T\Re^{-1}\mathbf{u}$
This is clearly a scalar constant for the particular row operation. Using the previous notation for a column of a matrix

$$\begin{aligned} \beta &= \mathbf{v}^T(\Re^{-1})_{K+1} \\ &= \tilde{\mathbf{v}}^T\mathcal{P}_{K+1} \quad (14) \end{aligned}$$

from (13) it is clear that

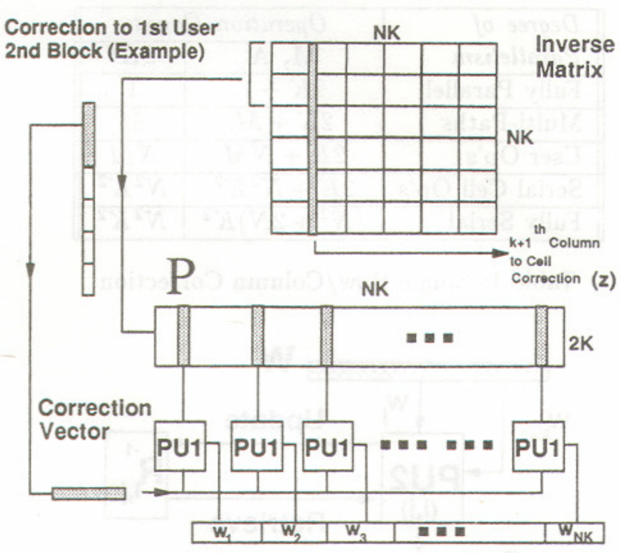
$$\beta = \mathbf{w}_{K+1} \quad (15)$$

We have hence expressed the terms in equations (8),(9),(10) in terms of rows and columns of the inverse system matrix $\Re^{-1}$, and the elements of the correction vector.

3.1 Estimation of Operation Counts

The basic computational element (cell) is the correction to a single filter coefficient. We will now consider this operation and estimate the number of computations required. Since $\mathbf{z}$ is the $(K+1)^{st}$ row of $\Re^{-1}$, $\mathbf{w}$ from (13) is given by

$$\mathbf{w}_j = \sum_{i=1}^{2K} \tilde{\mathbf{v}}_i.\mathcal{P}_j(i) \quad (16)$$



Correction to 1st User 2nd Block (Example)
NK
Inverse Matrix
NK
k+1th Column to Cell Correction (z)
P
NK
2K
Correction Vector
PU1
PU1
PU1
PU1
w1
w2
w3
wNK

Figure 3: Parallel Correction Set-up

let

$$Q = \mathbf{z} \otimes \mathbf{w}$$

Then

$$\begin{aligned} Q_{i,j} &= \mathbf{z}_i.\mathbf{w}_j \\ &= \Re^{-1}_{i,K+1}.\sum_{l=1}^{2K} \tilde{\mathbf{v}}_l\mathcal{P}_j(l) \quad (17) \end{aligned}$$

This operation is computationally identical to zero-forcing (multiply and sum) and can be performed fully in parallel. However if performed serially, the operation count is given by

$$C_1 = (2K+1)\mathbf{M} + 2K\mathbf{A} \quad (18)$$

Substituting in (11) from (15), (17) we have

$$\Re^{-1}_{i,j} \longrightarrow \Re^{-1}_{i,j} - \frac{Q_{i,j}}{1+\mathbf{w}_{K+1}} \quad (19)$$

Denoting the operation count by C_2

$$C_2 = 1\mathbf{A} + 1\mathbf{S} + 1\mathbf{D} \quad (20)$$

The total cost of a single correction is then

$$\begin{aligned} C_3 &= C_1 + C_2 \\ &= (2K+1)\mathbf{M} + (2K+1)\mathbf{A} + 1\mathbf{S} + 1\mathbf{D} \quad (21) \end{aligned}$$

Clearly most of C_1 is the overhead cost incurred for any one row or column correction. We now define an algorithm for parallel implementation of a row or column correction (as before we consider the general case excluding the edge blocks). Let a Processing Unit **PU1** perform an operation of the form

$$\sum_{j=1}^{2K} \tilde{\mathbf{v}}_j.\mathbf{p}_j$$

where $\mathbf{p}$ is a column of $\mathcal{P}$. By using NK degree parallelism at this stage the serial operation count is

Degree of Parallelism	*Operation Count*	
	M, A	**S,D**
Fully Parallel	$2K+1$	1
Multi-Paths	$2K+M$	M
User Op's	$2K+NM$	NM
Serial Cell Op's	$2K+N^2K^2$	N^2K^2
Fully Serial	$(N^2+2N)K^2$	N^2K^2

Table 1: Single Row/Column Correction

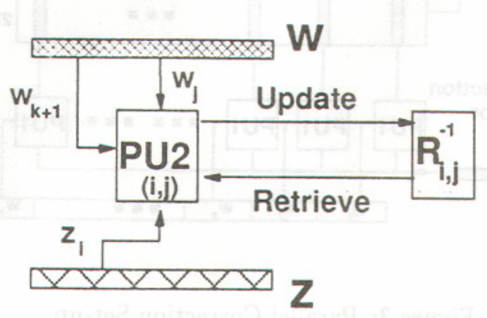


Figure 4: Single Cell Correction

2KM+2KA. The single cell correction which follows is identical to that in equations (17),(19) and the operations involved are **1M + 1A + 1S + 1D**. Varying degrees of parallelism can be used at this stage depending on ergonomic restrictions, cost and/or state of the art ASIC technology. A fully parallel version would involve all cells being computed simultaneously. The total serial operation count for a row/column correction is then identical to (21). We note that there is no specific need to locate all cells performing (19) on the same device. Full parallelism is hence feasible by dividing the row or column space over several identical IC's. Configurations for reduced parallelism would be to perform cell corrections to rows corresponding to the multi-paths of the same user, all operation for a single user or rows/columns in a single block, sequentially. Let $K = SM$ where S is the number of users and M the multi-paths being combined. We first tabulate the operations for a single row correction (Table 1). Clearly we have to repeat the operation discussed above $2N$ times. We must however bear in mind the reduced operation count for the edge blocks. The two edge corrections have a total of $3K$ non-zero elements. We tabulate the total operation count in Table 2.

The correction algorithm given below is for the fully parallel cell computation. When an interval of the form $[1 \leq j \leq N]$ appears after a statement, that statement is assumed to be executed simultaneously

Degree of Parallelism	*Operation Count*	
	M, A	**S,D**
Fully Parallel	$2K(2N-1)+2N$	$2N$
Multi-Paths	$2K(2N-1)+2NM$	$2NM$
User Op's	$2K(2N-1)+2N^2M$	$2N^2M$
Serial Cell Op's	$2K(2N-1)+N^3K^2$	$2N^3K^2$
Fully Serial	$(2N-1)2NK^2+N^3K^2$	$2N^3K^2$

Table 2: Total Matrix Correction

for all indices in that interval. The mathematical expression being executed is enclosed in {.}. We assume the correction applies to the k^{th} user/path.

Correction Algorithm

```
 BEGIN
Setup row (v), column (u) correction vectors
 FOR  x = rows (r), columns (c)  DO
   Setup$\tilde{v}$ $\tilde{u}$ from r (c) correction vectors
       FOR (i=1  STEP i,  UNTIL N)
         IF(i  NEQ 1)  AND (i  NEQ N)
           Set P to $2K$ rows of $\Re^{-1}$
             start : $((i-2)K+k)^{th}$ x (r/c)
         ELSE
           IF (i  EQ 1)
             Set P to $(K+k-1)$ x (r/c)'s of $\Re^{-1}$
               start :  $((i-1)K+k)^{th}$ x (r/c)
           ELSE
             Set P to $(2K-k)$ x (r/c)'s of $\Re^{-1}$
               start :$((i-1)K+k)^{th}$ x (r/c)
        Set z to $((i-1)K+k)^{th}$ ,$\bar{x}$ (c/r)
        Compute $\mathbf{w}_j$; $[1 \leq j \leq NK]$
          $\left\{\sum_{j=1}^{2K} \tilde{\mathbf{v}}_j.\mathbf{p}_j\right\}$
        Compute $Q_{p,q}$ ; $[1 \leq p,q \leq NK]$
          $\{\mathbf{z}_p.\mathbf{w}_q\}$
        Compute Correction $\Delta_{p,q}$; $[1 \leq p,q \leq NK]$
          $\frac{Q_{p,q}}{1+\mathbf{W}_{(i-1)K+k}}$
        Apply Correction to $\Re^{-1}_{p,q}$; $[1 \leq p,q \leq NK]$
          $\Re^{-1}_{p,q} \longrightarrow \Re^{-1}_{p,q} - \Delta_{p,q}$
 END
```

The statement regarding the triangular structure of the matrices $R(-1), R(1)$ relies on the users being ordered according to their delays. Clearly the condition ceases to hold if the timing of an user changes to such an extent as to change its position in the delay table. It is clear that the above corrections will have to be performed many times over (repeated swapping of users) so as to arrive at a new system matrix corresponding to the modified delay table. From this point of view it is beneficial in the long term to remove the restriction on delay-ordering. $\Re$ remains block tridi-

Degree of Parallelism	*Operation Count*	
	M, A	**S,D**
Fully Parallel	$2K(3N-2)+2N$	$2N$
Multi-Paths	$2K(3N-2)+2NM$	$2NM$
User Op's	$2K(3N-2)+2N^2M$	$2N^2M$
Serial Cell Op's	$2K(3N-2)+N^3K^2$	$2N^3K^2$
Fully Serial	$(3N-2)2NK^2+N^3K^2$	$2N^3K^2$

Table 3: Correction without Delay Ordering

agonal, however $R(-1)$ and $R(1)$ have to be treated as being full matrices. We hence have a contiguous block

Degree of Parallelism	*Operation Count*	
	M, A	**S,D**
Fully Parallel	$2K^2(N+1)+NK$	NK
User Op's	$2K^2(N+1)+N^2K^2M$	N^2K^2M
Serial Cell Op's	$2K^2(N+1)+N^3K^3$	N^3K^3
Fully Serial	$2NK^3(N+1)+N^3K^3$	N^3K^3

Table 4: Matrix Inversion

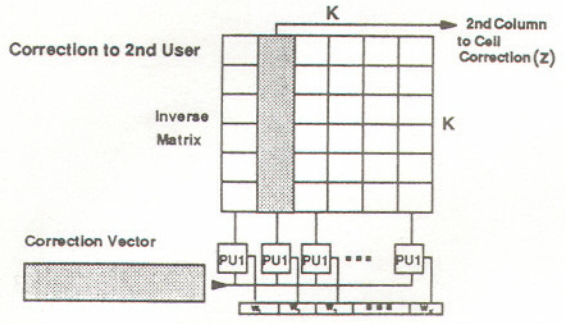


Figure 5: Correction for Sync. Case

Degree of Parallelism	*Operation Count*	
	M, A	**S,D**
Fully Parallel	$2K+2$	2
User Op's	$2K+2M$	$2M$
Serial Cell Op's	$2K+K^2$	K^2
Fully Serial	$3K^2$	K^2

Table 5: Matrix Correction - Synchronous

of $3K$ potentially non-zero elements in the correction vectors. The two 'edge block' vectors have a total of $5K$ elements in the correction vector. The modified operation count is given in Table 3. The system is now completely flexible to changes in timing. A moderate number of new users can be accommodated by maintaining unused 'slots' in the correlation matrices, and 'correcting' them to accommodate the new user. A user leaving the system creates a vacant slot, but does not initiate a correction. An attractive feature of the algorithm is that once in place, the hardware architecture described can be used to compute the matrix inverse in the first instance, or to periodically recompute the matrix to avoid the build up of inaccuracy due to continuous correction. We start with $\Re = \Re^{-1} = I$. NK row corrections are applied to $\Re$ so as to arrive at the desired partial correlation matrix. The operation count for an NK matrix inversion is given in Table 4.

3.2 Synchronous DS-CDMA

In the case of Synchronous DS-CDMA the algorithm collapses to the original Sherman-Morrison Formula. Clearly only $R(0)$ exists and hence only one row operation and one column operation are required. The correction vectors are now complete (potentially no zero elements), but have only K elements. K degree parallelism can be used to compute **w**. The vector **z** is a column of $R(0)$ and equations (8),(9),(10), (15),(17) and (19) hold as before. The cell computations can be performed fully in parallel with only K^2 processing units. The operation count is summarised in Table 5. Clearly the algorithm provides a very fast method of updating the inverse matrix in the synchronous case. Extending the architecture as before to compute the matrix inverse, the operation count is tabulated in Table 6.

4 Conclusions

We have proposed a coefficient correction algorithm for finite sequence length decorrelators. The algorithm is suitable for implementation on a fully parallel ASIC. Competing solutions to the decorrelator have significantly larger per-iteration cost. The computational overhead of the decorrelator lies in the re-computation of the linear system solution, which is performed off-line on an occasional basis. We have proposed a solution to this problem which avoids re-computation. Processing requirements have been evaluated (data transmission overhead has not been considered at this stage) which identify the decorrelator as a potentially attractive centralised solution to the Near-Far problem.

Degree of Parallelism	*Operation Count*	
	M, A	**S,D**
Fully Parallel	K^2K	K
User Op's	K^2+KM	MK
Serial Cell Op's	K^2+K^3	k^3
Fully Serial	$2K^3$	K^3

Table 6: Matrix Inversion - Synchronous

5 Acknowledgements

The authors would like to thank the CCR for the provision of computing facilities, and CVCP UK (Overseas Research Student Award) and AMI Fidelity USA for the financial support of S.S.H Wijayasuriya.

References

[1] S. S. H. Wijayasuriya, G. H. Norton, and J. P. McGeehan, "A Sliding Window Decorrelating Algorithm for Multi-User DS-CDMA Systems," in *Globecomm '92*, (Orlando, USA), Dec 1992.

[2] S. S. H. Wijayasuriya, J. P. McGeehan, and G. H. Norton, "RAKE Decorrelating Receiver for DS-CDMA Mobile Radio Networks," *Electronics Letters*, vol. 29, pp. 395 – 96, August 1993.

[3] R. Lupas and S. Verdu, "Near-Far Resistance of Multi-User Detectors in Asynchronous Channels," *IEEE Trans. on Communications*, vol. 38, pp. 496 – 508, April 1990.

[4] P. Dent, B. Gudmundson, and M. Ewerbring, "A Novel Code Division Multiple Access Scheme Based on Interference Cancellation," in *International Symposium on Personal, Indoor, and Mobile Radio Comunications*, (Boston, USA), Oct 1992.

[5] U. Madhow and M. Honig, "Error Probability and Near Far Resistance of Minimum Mean Squared Error Interference Suppression Scheme for CDMA," in *Globecomm '92*, (Orlando, USA), Dec 1992.

[6] A. Viterbi, "Very Low Rate Convolutional Codes for Maximum Theoretical Performance of Spread Spectrum Multiple Access Channels," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 641 – 649, 1990.

[7] H. Stone, "An Efficient Parallel Algorithm for the Solution of a Tri-diagonal Linear System of Equations," *Journal of the Association for Computing Machinery*, vol. 20, pp. 27–38, Jan 1973.

[8] G. Golub and C. Van-Loan, *Matrix Computations*. The John Hopkins University Press.