Parallel and Adaptive Reduction of Hyperspectral Data to Intrinsic Dimensionality

Tarek El-Ghazawi

Department of Electrical and Computer Engineering

The George Washington University

tarek@seas.gwu.edu

Sinthop Kaewpijit (GMU)

Jacqueline Le Moigne (NASA GSFC)

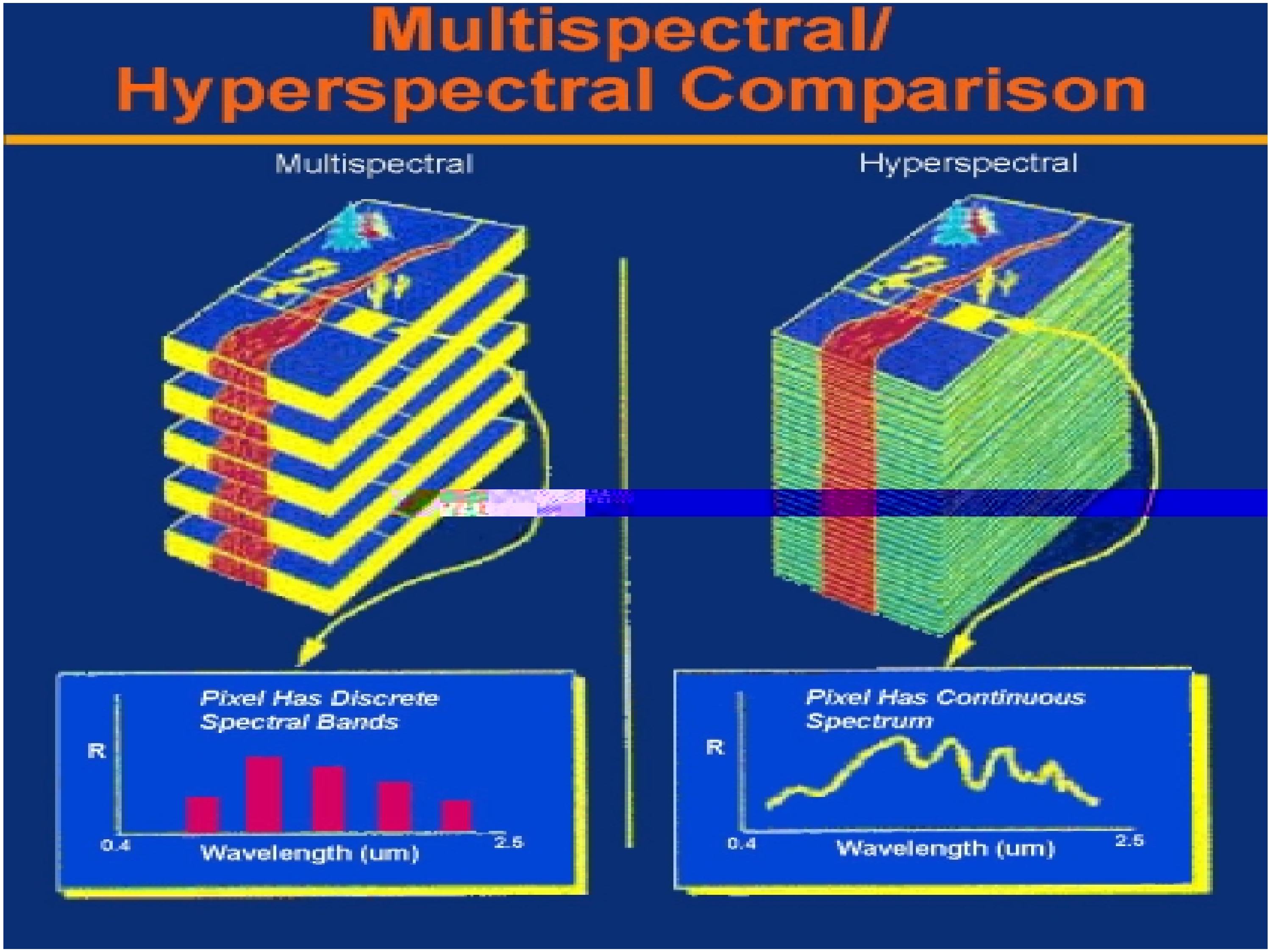
CLUSTER2001, Newport Beach, CA.

October 8-11, 2001

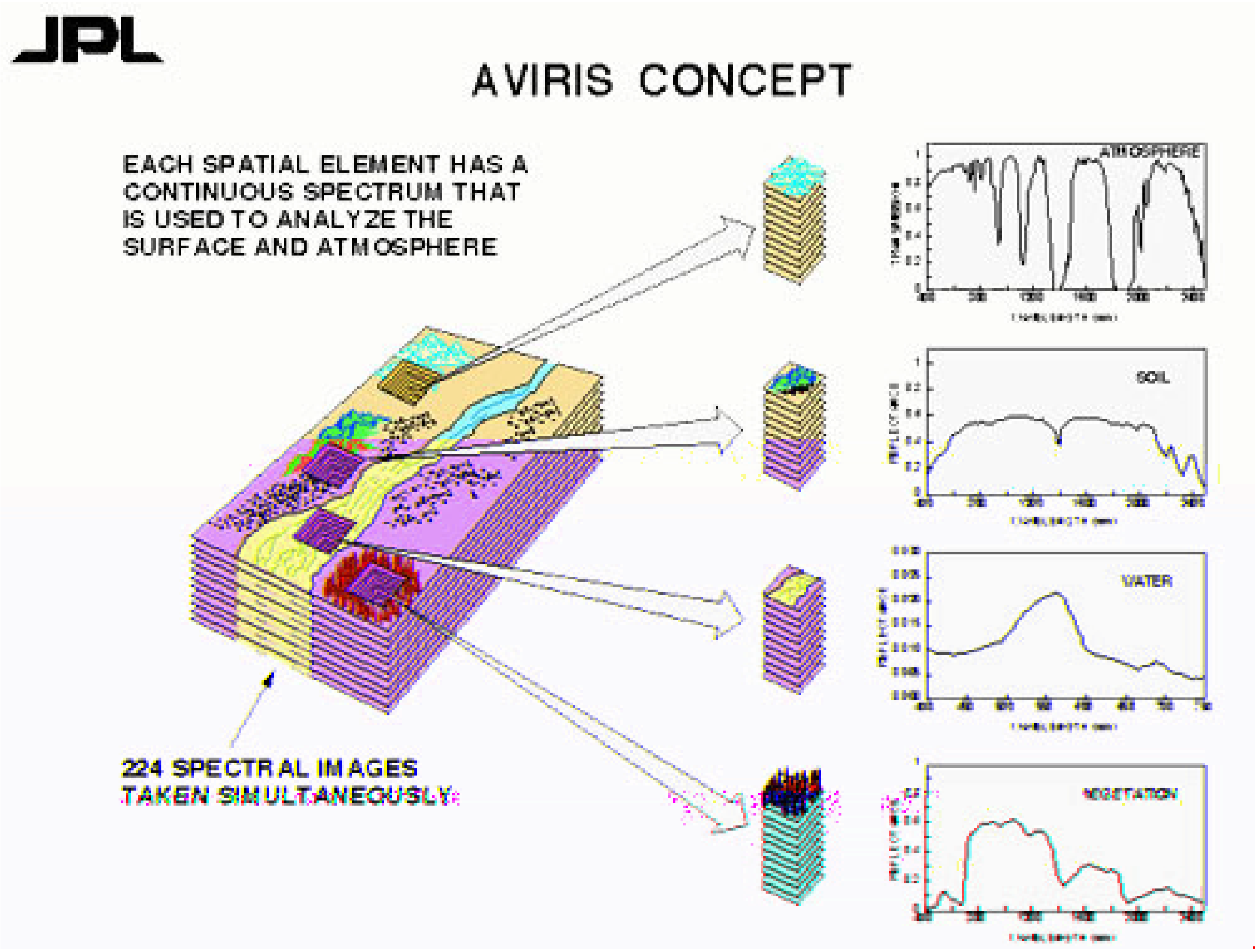
AGENDA

- Introduction to Hyperspectral Imagery
- Overview of Dimension Reduction and PCA
- Information-content based PCA
- An Adaptive Algorithm for PCA
- Experimental Results
- Conclusion

MULTISPECTRAL vs. HYPERSPECTRAL DATA



IMAGING SPECTROSCOPY

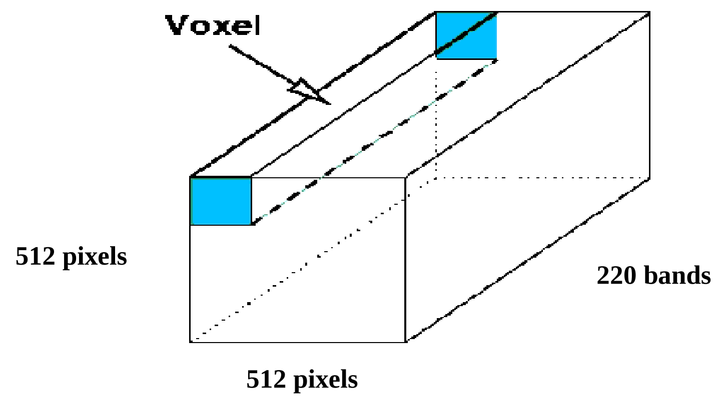


[redacted]

PRINCIPAL COMPONENT ANALYSIS (PCA)

- One of the most popular techniques for dimension reduction
- A rotational transformation in which the data can be represented without correlation in a new-coordinate system.
- Often used as a preprocessing step for classification
- Produces contrast-stretched pixel values and used for enhancement prior to visual interpretation

PCA Computations



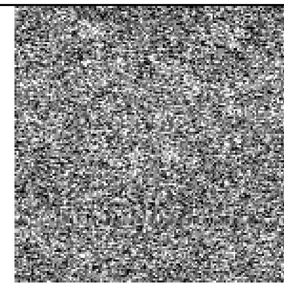
- Find mean position (mean vector/voxel)
- Compute the covariance matrix of an image
- Determine eigenvalues and eigenvectors
- Form the principal components (PCs)

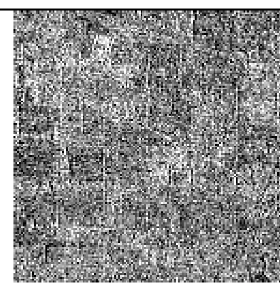
PCA Computations

- Compute the mean voxel
- Compute the covariance matrix
  - Unbias each voxel by subtracting the mean
  - Compute the cross-product of each unbiased voxel with its own transpose producing a matrix
  - Sum up all matrices
  - Divide by the number of voxels minus 1
- Compute the Eigen Problem for the Covariance Matrix– can be done in two main different ways!
- Form the components
  - Each component is produced by multiplying the voxels of the original image by one given eigen vector

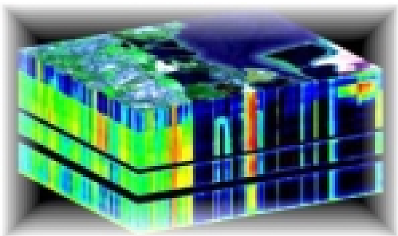
EXAMPLE
Original Bands and PCs

(AVIRIS Data, The Portion of IindianPines'92) --DS02

 Visible	 Near Infrared	 Water absorption band

 PC1	 PC10

PCA AND INTRINSIC DIMENSIONALITY



PCs	Eigenvalues	PTDV (%)	CUPV (%)
1	23026166.00	54.63970	54.63970
2	16925160.00	40.16238	94.80208
3	1037322.62	2.46150	97.26
4	403824.12	0.95825	98.22184
5	251390.68	0.59653	98.81837
6	129284.06	0.30678	99.12515
7	79201.42	0.18794	99.31309
8	61343.96	0.14556	99.45866
9	42339.22	0.10046	99.55913
10	30403.65	0.07214	99.63127
•	•	•	•
•	•	•	•
•	•	•	•

IC~99%

PTDV = Percent of Total Data Variation
CUPV = CUmulative Percent Variation

SPEEDING PCA COMPUTATIONS

- Determine (automatically) and calculate PCs only till desired level of information contents (intrinsic dimensionality)
- Parallel processing on clusters
  - Distribute work, but
  - Not all computations that can be distributed should be distributed!
    - Pay attention to poor communications on clusters

COMPUTING THE EIGEN VALUES: HOETELLING POWER METHOD

Compute eigenvalue and corresponding eigenvector one by one starting from the largest eigenvalue

Given a real symmetric matrix $A \in R^{n \times n}$ and initial estimate $q^{(0)}$ of the principal eigenvector of *A*, the power method proceeds as follows:

for k : **0,1,…**(until $\left|q^{(k+1)} - q^{(k)}\right| \approx 0$) do

$p := Aq^{(k)}$

$q^{(k+1)} := p / \|p\|$

end

$q^{(k)}$ converge to the principal eigenvector e_1 of *A*

Source: Diamantaras, 1996

COMPUTING THE EIGEN VALUES: The Jacobi Method

Compute all eigenvalues and corresponding eigenvectors at once

Given a real symmetric matrix $A \in R^{n \times n}$. Nullify at each iteration k the biggest off diagonal element of A by multiplying from both sides by a properly chosen orthogonal matrix $J^{(k)}$ **:**

for k **:** 1,2,…N (until the off-diagonal elements are all zero) do

identify largest off-diagonal element a_{ij} of A

construct matrix $J^{(k)}$

$A := J^{(k)T} A J^{(k)}$

end

The eigenvector matrix is $J^{(1)}J^{(2)}\cdots J^{(N)}$

The eigenvalues are the diagonal elements of the final matrix A

Sort the eigenvalues in descending order

IC-BASED PCs

The sum of eigenvalues is equal to the trace of the covariance matrix

$$\sum_{i=1}^{N} \lambda_i = Trace\,(A)$$

IC-BASED ALGORITHM

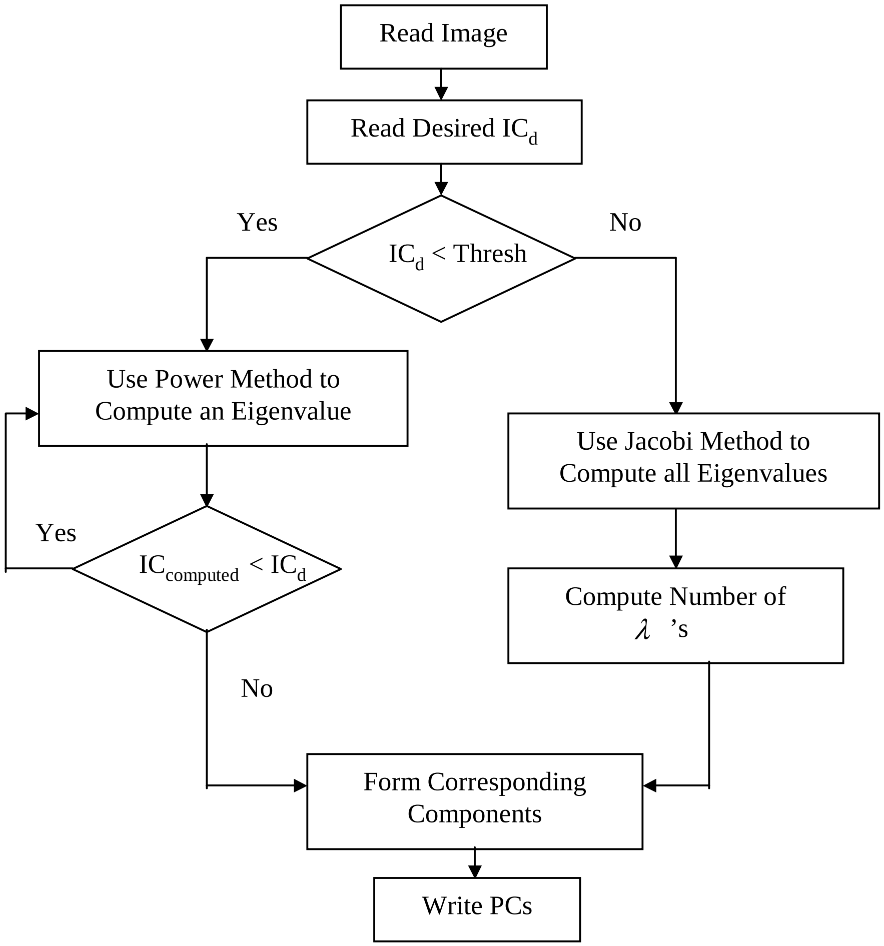
```
Specify threshold /* in PERCENT */
Get Trace /* Summation of diagonal of covariance matrix */
IC = 0
while (IC<threshold) {
        Get eigenvalue and eigenvector /* Using Power Method
                and Matrix Deflation */
        IC += eigenvalue/Trace*100
}
Form Principal Components that correspond to computed
eignevectors
```

[redacted]

NOTES ON IC ALGORITHM AND EIGENPROBLEM COMPUTATION

- IC algorithm can be slightly modified to work with Jacobi like methods
- Power Method works well when less number of PCs is needed
- Jacobi works better for larger number of PCs
- One can adaptively select among the two

THE ADAPTIVE PCA ALGORITHM



PARALLEL PCA

- Master
  - Read hyperspectral data from disk
  - Compute mean vector
  - Broadcast mean vector to all processors
  - Distribute data voxels across processors
  - Receive the integrated Covariance Matrix from workers
  - Compute Eigen problem and broadcast eigen vectors (slightly different in case of Power Method)
  - Gather PCs
  - Write PCs back to disk

PARALLEL PCA cont.

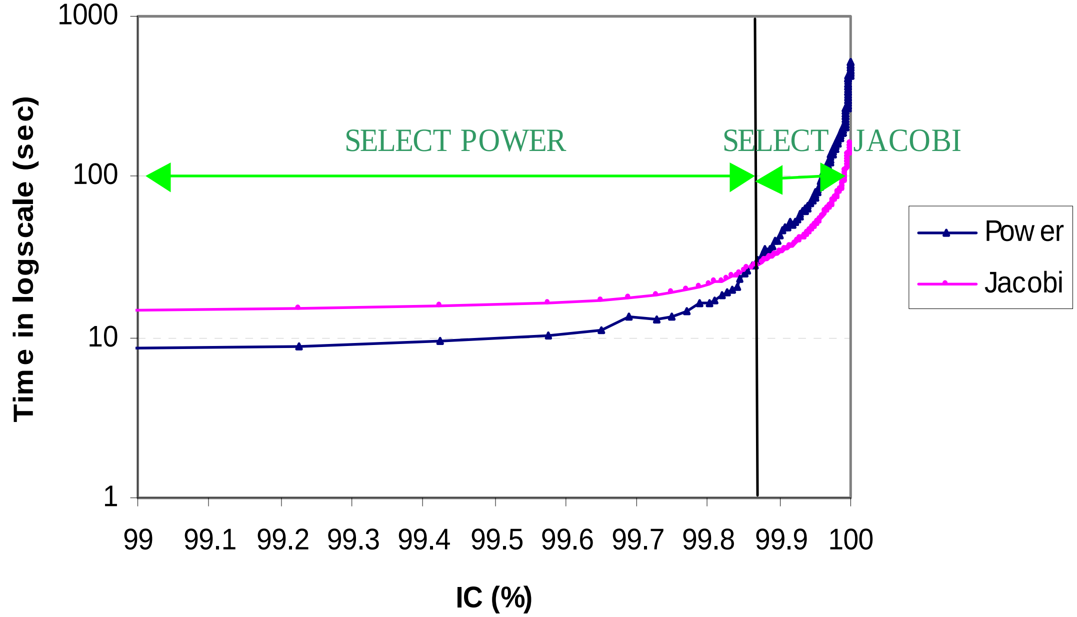
- Workers
  - Receive local share of voxels from Master
  - Partially compute covariance matrix (for local voxels)
  - Reduce-add to integrate covariance matrix at master
  - Receive broadcast eignenvectors
  - Form (Partially) the PCs
  - Gather PC's into master

EXPERIMENTAL RESULTS: HYPERSPECTRAL DATA SETS USED

HYPERSPECTRAL DATA SETS	NO. OF PIXELS (Spatial Domain)	NO. OF BANDS (Spectral Domain)
DS01	145X145	75 bands
DS02	145x145	220 bands
DS03	614x512	224 bands

SEQUENTIAL ADAPTIVE RESULTS

Switching Decision of the 145 X 145 X 220 Data Set -- DS02
(AVIRIS Data, The Portion of IindianPines'92)



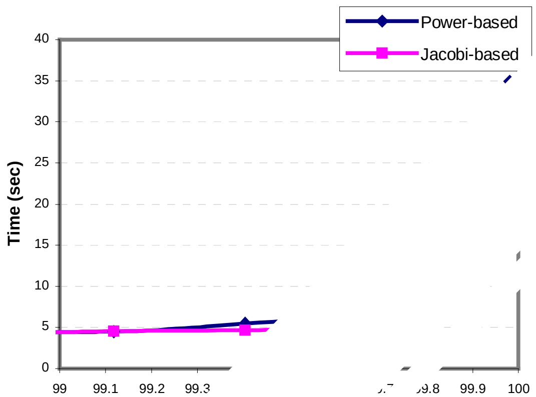
Time and Information Content

PCs	IC (%)	Power-based	Jacobi-based
21	99.85638	26.01 sec	26.64 sec
22	99.86135	27.64 sec	27.18 sec

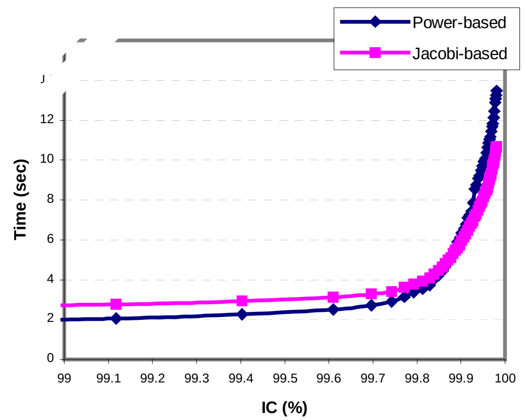
System: Linux-Based Pentium III 600 MHz

PARALLEL ADAPTIVE [illegible] RESULTS – DS01

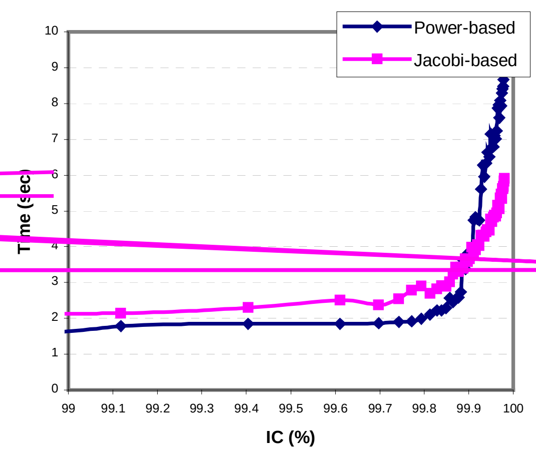
PEs = 2



PEs = 4



PEs = 16



(AVIRIS Data, The Portion of IindianPines'92) – DS01

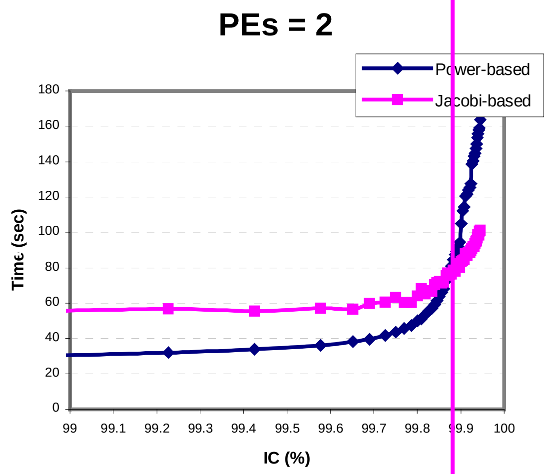
PARALLEL PCA RESULTS – DS01

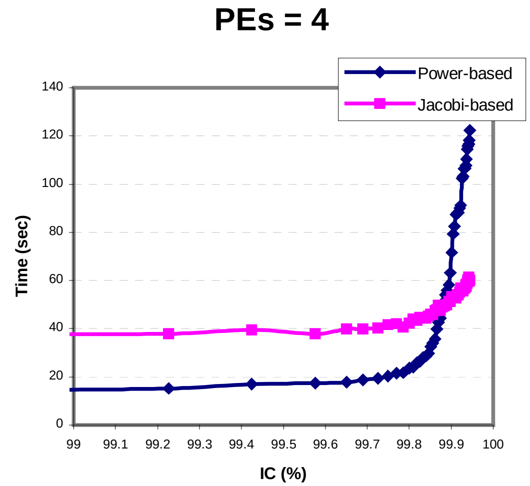
(AVIRIS Data, The Portion of IindianPines'92) – DS01

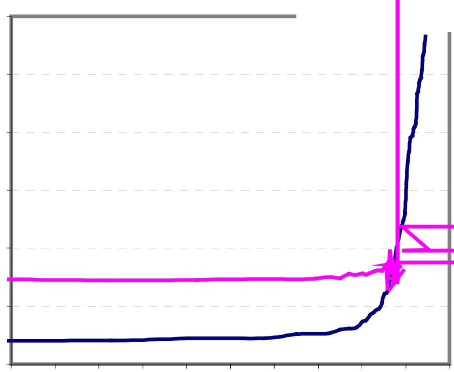
PEs	PCs	IC (%)	Parallel-time (Power)	Parallel-time (Jacobi)
2	3	99.11751	4.482 sec	4.526 sec
	4	99.40379	5.509 sec	4.676 sec
4	18	99.88254	5.303 sec	5.313 sec
	19	99.88767	5.651 sec	5.466 sec
8	22	99.90222	3.645 sec	3.693 sec
	23	99.90674	3.888 sec	3.783 sec
16	23	99.90674	3.859 sec	3.978 sec
	24	99.91116	4.738 sec	3.79 sec

PEs = Number of Processors
PCs = Number of Principal Components
IC = Information Content (in percentage)

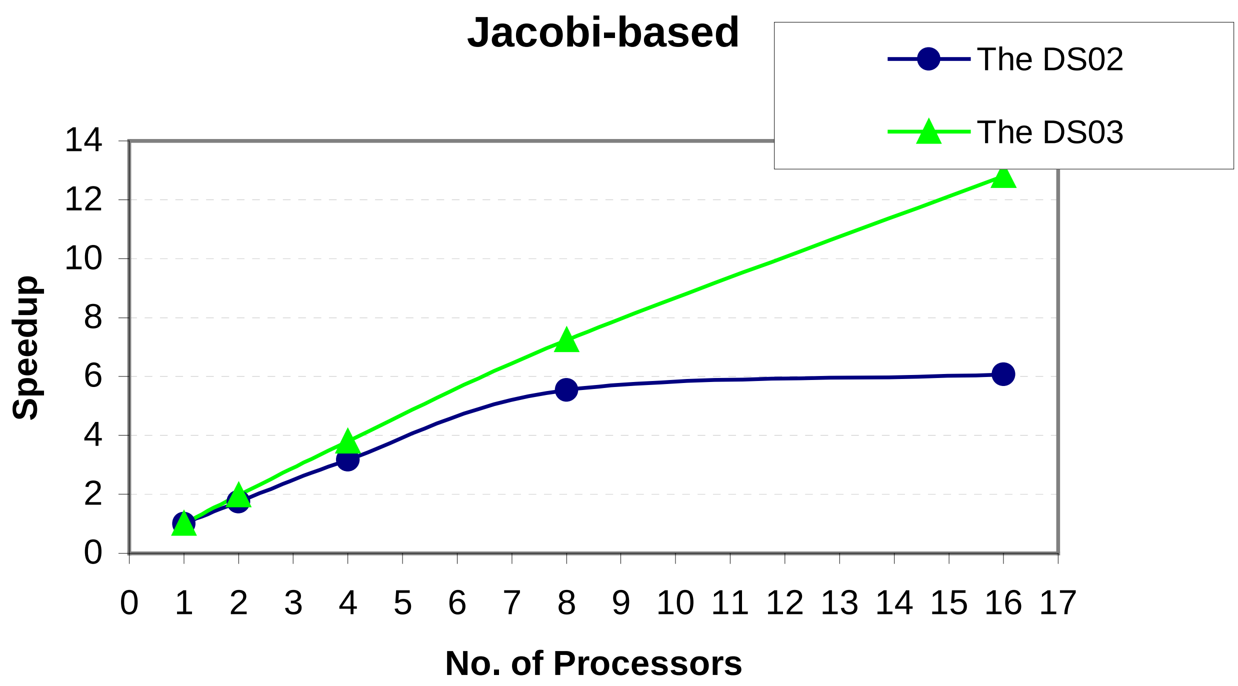
PARALLEL ADAPTIVE PCA RESULTS – DS02







PARALLEL PCA RESULTS (cont.)



Maximum Scalability can be obtained for the DS03 (AVIRIS IndianPines'92) and the DS02 (the portion of DS03) with the Jacobi-based Method

CONCLUSIONS

- The power method is efficient for computing a small number of "aimed-for PCs" based on the user's specified information content
- Jacobi offers a better choice when obtaining a higher number of PCs is of interest
- This behavior is consistent for both sequential and parallel implementations
- An adaptive algorithm can be constructed to switch between the two to provide the most efficient overall execution time

CONCLUSIONS cont.

- The switching point depends upon the data set and machine parameters
- PCA scales reasonably on clusters when all or most PCs are to be formed for relatively large size data using the Jacobi-based method