# Common Lisp Implementation of FastICA

**Roman V. Belavkin (R.Belavkin@mdx.ac.uk)**

**School of Computing Science,**
**Middlesex University, London NW4 4BT, UK**

**April 12, 2005**

# BLIND SOURCE SEPARATION (or ICA)

$\boldsymbol{X} = (x_1, \ldots x_m)^T, \boldsymbol{Y} = (y_1, \ldots, y_n)^T$

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$$

Assuming:

1. $P(x_1, \ldots, x_m) = P(x_1) \cdots P(x_m)$
2. $\forall i$ but one $P(x_i)$ are non–Gaussian

Find $\boldsymbol{A} \approx \boldsymbol{W}^{-1}$ such that

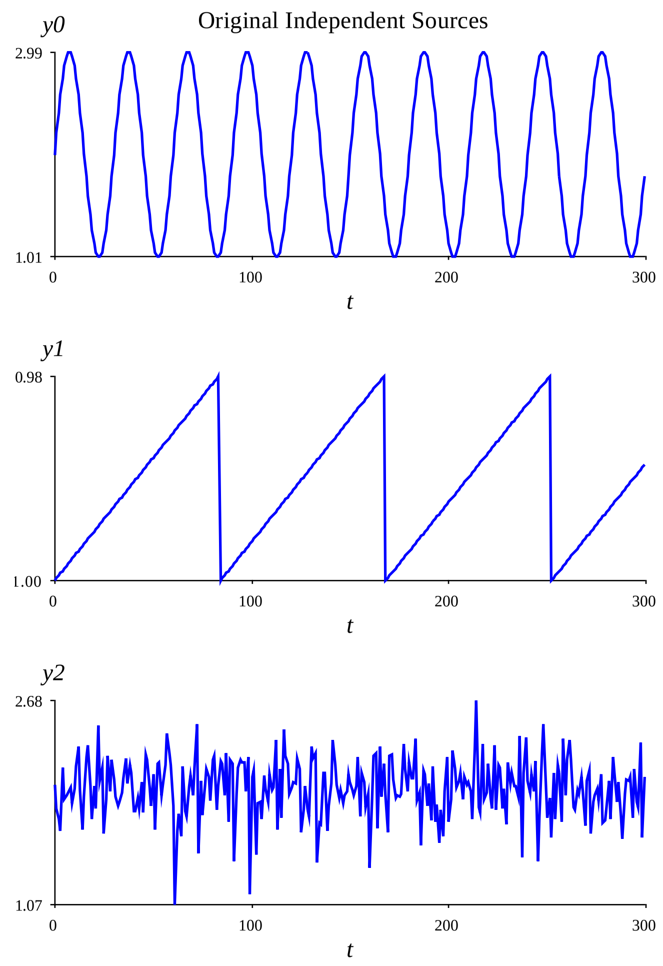
$$\boldsymbol{A}\boldsymbol{Y} \approx \boldsymbol{X}$$

# Common Lisp Implementation

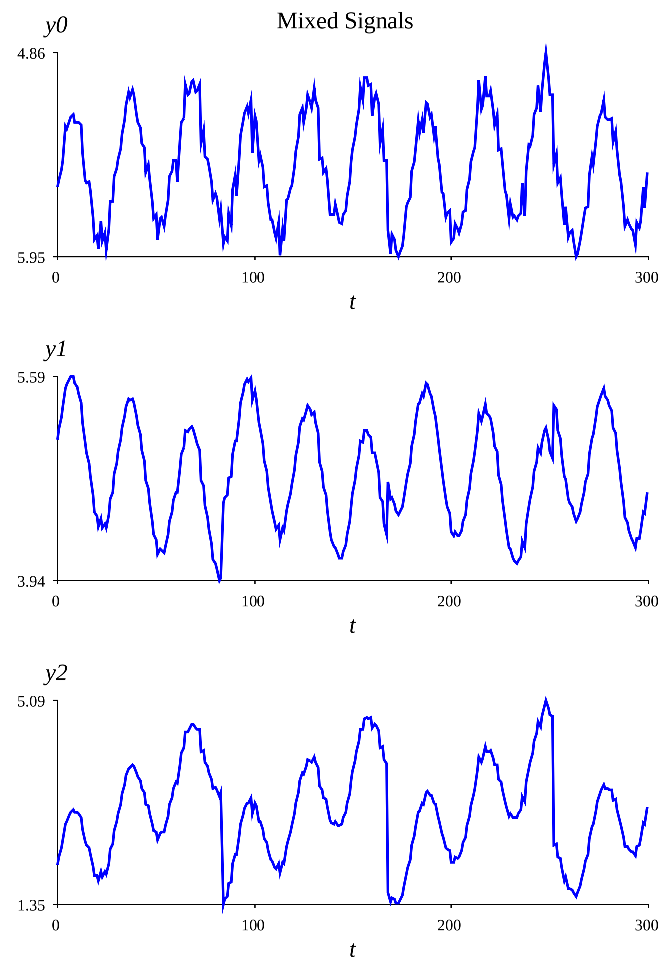
ICA.lisp (5K) + PCA.lisp (8K) + Matrices.lisp (10K) = 23K of code:

```
(defun one-unit-fica (X &key ......)
  (do* ((m (num-rows X))
        (w (find-one-weight X nil)
           (find-one-weight X ws))
        (ws (list w) (push w ws)))
       ((= (length ws) m)
        (let ((W (make-array (list m m))))
          (dotimes (i m W)
            (dotimes (j m)
              (setf (aref W i j)
                    (aref (nth j ws) i 0)))))))))
```

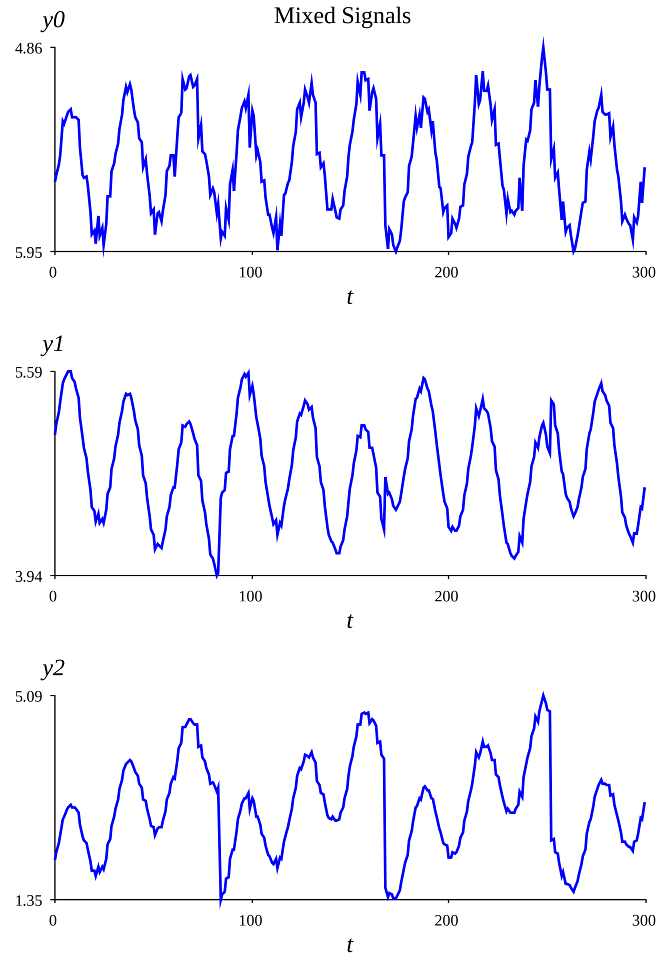
# MIXING SOURCES



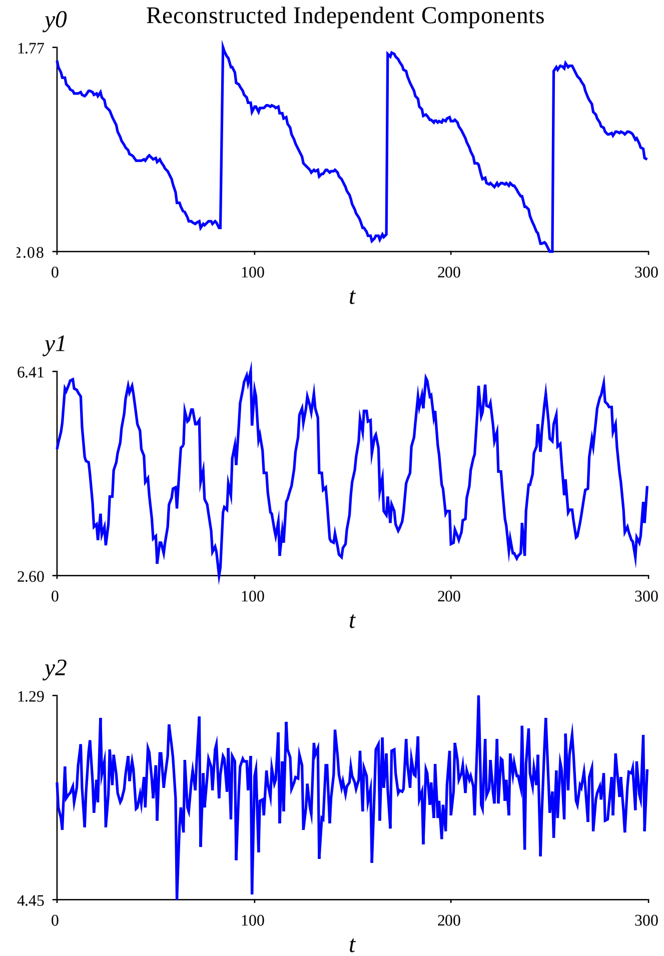

$$\times \boldsymbol{W} =$$



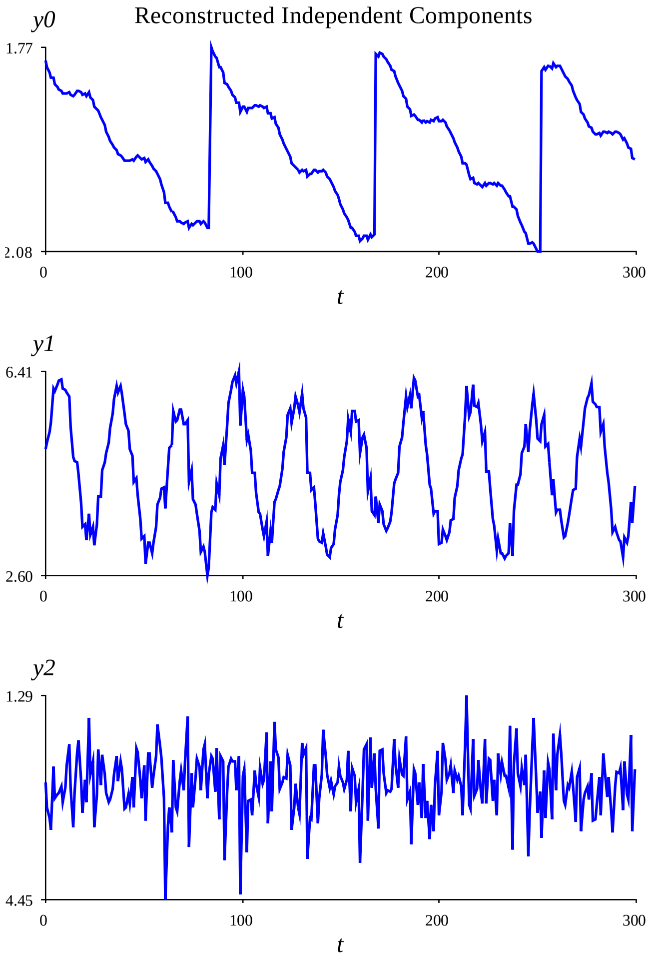
# DEMIXING SIGNALS



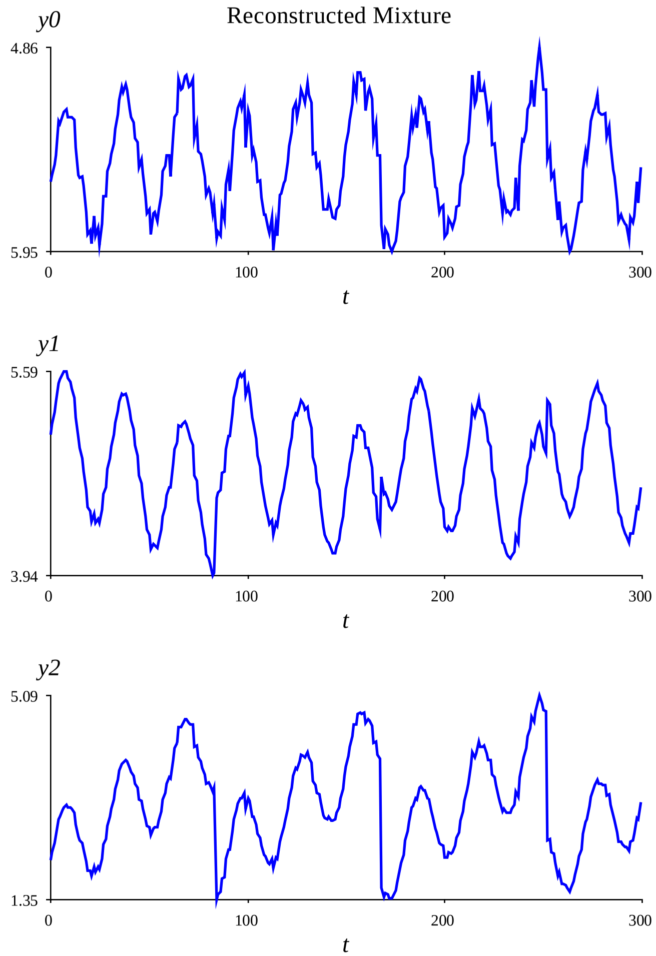

$\times \boldsymbol{A} =$



# RECONSTRUCTING MIXTURE




$$\times \boldsymbol{A}^{-1} =$$



# FUTURE WORK

- Use ICA output as an input to Bayesian learning agents (perception with reduction of dimensions and minimal loss of information):

$$\begin{aligned} \boldsymbol{X} &\in \mathbf{R}^n && \text{perception inputs} \\ \boldsymbol{Y} &\in \mathbf{Z} && \text{set of actions} \\ \boldsymbol{X} \to \boldsymbol{S} &\in \mathbf{R}^m && \text{reduce dimensions (ICA)} \end{aligned}$$

$$\boldsymbol{P}(\boldsymbol{Y} \mid \boldsymbol{S}, U) = \alpha \boldsymbol{P}(\boldsymbol{Y}, \boldsymbol{S}, U)$$

- Encourage MSc students to use ICA in their projects in Business Information Systems programme.