# Joint Approximate Diagonalization of Eigenmatrices

La JADE di un insieme di matrici quadrate consiste nella ricerca di una base ortonormale che diagonalizza quanto possibile le matrici dell’insieme. Quando tutte commutano, la diagonalizzazione è perfetta; altrimenti è sempre possibile ottimizzare un criterio di diagonalità congiunta.  
Quando tutte le matrici dell’insieme sono “pressochè esattamente congiuntamente diagonalizzabili” questo approccio definisce anche qualcosa come l’”autospazio medio” dell’insieme di matrici.  
Ecco un algoritmo per la JADE di un insieme di matrici, reali o complesse, autoaggiunte o non, siccome il metodo di Jacobi può esser facilmente esteso: abbiamo trovato un trucco attraverso il quale gli angoli di Givens ottenuti nelle rotazioni planari del metodo di Jacobi possono essere velocemente ottenuti.

## Matlab Implementation

**function** **[** V **,** D **]** **=** joint\_diag**(**A**,**jthresh**)**

% Joint approximate of n (complex) matrices of size m\*m stored in the

% m\*mn matrix A by minimization of a joint diagonality criterion

%

% Input :

% \* the m\*nm matrix A is the concatenation of n matrices with size m

% by m. We denote A = [ A1 A2 .... An ]

% \* threshold is an optional small number (typically = 1.0e-8).

%

% Output :

% \* V is an m\*m unitary matrix.

% \* D = V'\*A1\*V , ... , V'\*An\*V has the same size as A and is a

% collection of diagonal matrices if A1, ..., An are exactly jointly

% unitarily diagonalizable.

%

% The algorithm finds a unitary matrix V such that the matrices

% V'\*A1\*V , ... , V'\*An\*V are as diagonal as possible, providing a

% kind of `average eigen-structure' shared by the matrices A1 ,...,An.

% If the matrices A1,...,An do have an exact common eigen-structure ie

% a common orthonormal set eigenvectors, then the algorithm finds it.

% The eigenvectors THEN are the column vectors of V and D1, ...,Dn are

% diagonal matrices.

%

% The algorithm implements a properly extended Jacobi algorithm. The

% algorithm stops when all the Givens rotations in a sweep have sines

% smaller than 'threshold'.

%

% In many applications, the notion of approximate joint

% diagonalization is ad hoc and very small values of threshold do not

% make sense because the diagonality criterion itself is ad hoc.

% Hence, it is often not necessary in applications to push the

% accuracy of the rotation matrix V to the machine precision.

%

% PS: If a numrical analyst knows `the right way' to determine jthresh

% in terms of 1) machine precision and 2) size of the problem,

% I will be glad to hear about it.

%

%

% This version of the code is for complex matrices, but it also works

% with real matrices. However, simpler implementations are possible

% in the real case.

**[**m**,**nm**]** **=** size**(**A**);**

B **=** **[** 1 0 0 **;** 0 1 1 **;** 0 **-**i i **]** **;**

Bt **=** B**'** **;**

Ip **=** zeros**(**1**,**nm**)** **;**

Iq **=** zeros**(**1**,**nm**)** **;**

g **=** zeros**(**3**,**nm**)** **;**

g **=** zeros**(**3**,**m**);**

G **=** zeros**(**2**,**2**)** **;**

vcp **=** zeros**(**3**,**3**);**

D **=** zeros**(**3**,**3**);**

la **=** zeros**(**3**,**1**);**

K **=** zeros**(**3**,**3**);**

angles **=** zeros**(**3**,**1**);**

pair **=** zeros**(**1**,**2**);**

G **=** zeros**(**3**);**

c **=** 0 **;**

s **=** 0 **;**

%% Init

V **=** eye**(**m**);**

encore **=** 1**;**

**while** encore**,** encore**=**0**;**

**for** p**=**1**:**m**-**1**,** Ip **=** p**:**m**:**nm **;**

**for** q**=**p**+**1**:**m**,** Iq **=** q**:**m**:**nm **;**

%% Computing the Givens angles

g **=** **[** A**(**p**,**Ip**)-**A**(**q**,**Iq**)** **;** A**(**p**,**Iq**)** **;** A**(**q**,**Ip**)** **]** **;**

**[**vcp**,**D**]** **=** eig**(**real**(**B**\*(**g**\***g**')\***Bt**));**

**[**la**,** K**]** **=** sort**(**diag**(**D**));**

angles **=** vcp**(:,**K**(**3**));**

**if** angles**(**1**)<**0 **,** angles**=** **-**angles **;** **end** **;**

c **=** sqrt**(**0.5**+**angles**(**1**)/**2**);**

s **=** 0.5**\*(**angles**(**2**)-**j**\***angles**(**3**))/**c**;**

**if** abs**(**s**)>**jthresh**,** %%% updates matrices A and V by a Givens rotation

encore **=** 1 **;**

pair **=** **[**p**;**q**]** **;**

G **=** **[** c **-**conj**(**s**)** **;** s c **]** **;**

V**(:,**pair**)** **=** V**(:,**pair**)\***G **;**

A**(**pair**,:)** **=** G**'** **\*** A**(**pair**,:)** **;**

A**(:,[**Ip Iq**])** **=** **[** c**\***A**(:,**Ip**)+**s**\***A**(:,**Iq**)** **-**conj**(**s**)\***A**(:,**Ip**)+**c**\***A**(:,**Iq**)** **]** **;**

**end**%% if

**end**%% q loop

**end**%% p loop

**end**%% while

D **=** A **;**

**return**