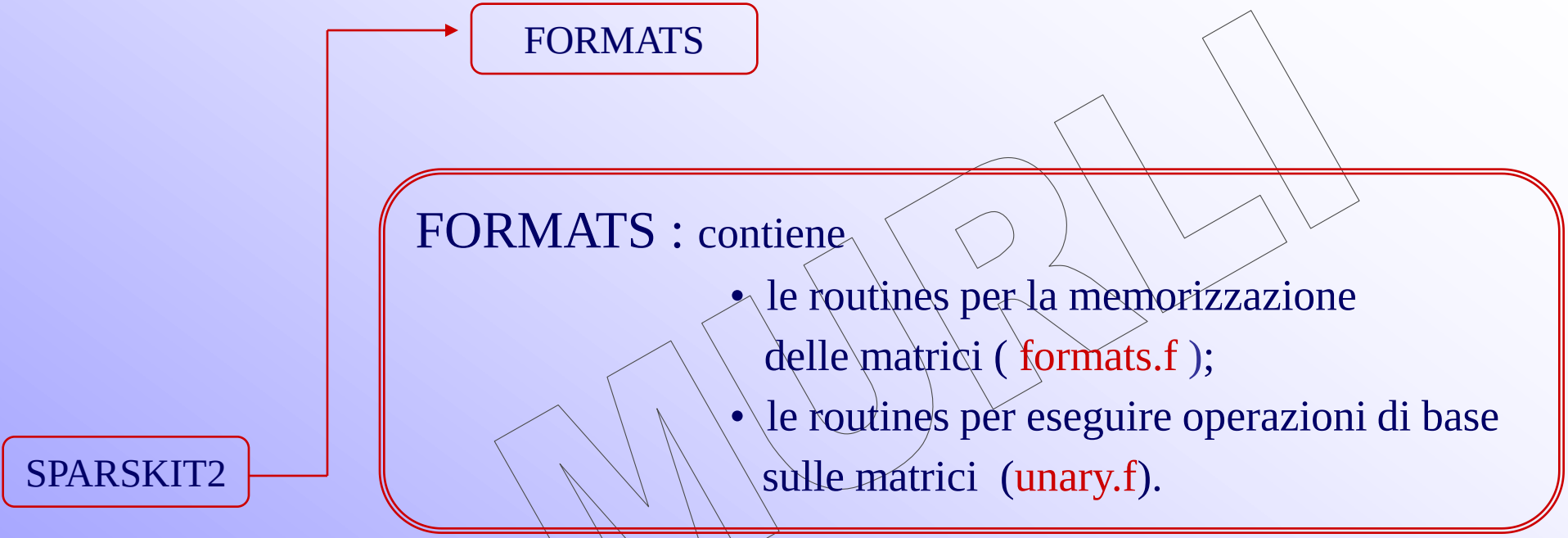
Introduzione alla libreria

SPARSKIT

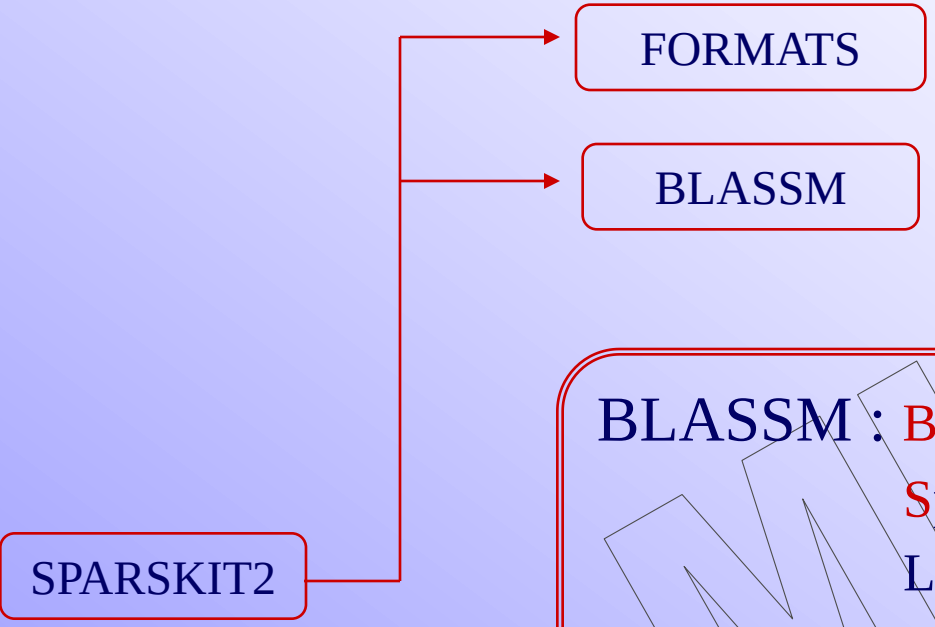
Che cos'è SPARSKIT ?

SPARSKIT è una libreria software per la risoluzione di problemi con matrici sparse.

Organizzazione di SPARSKIT

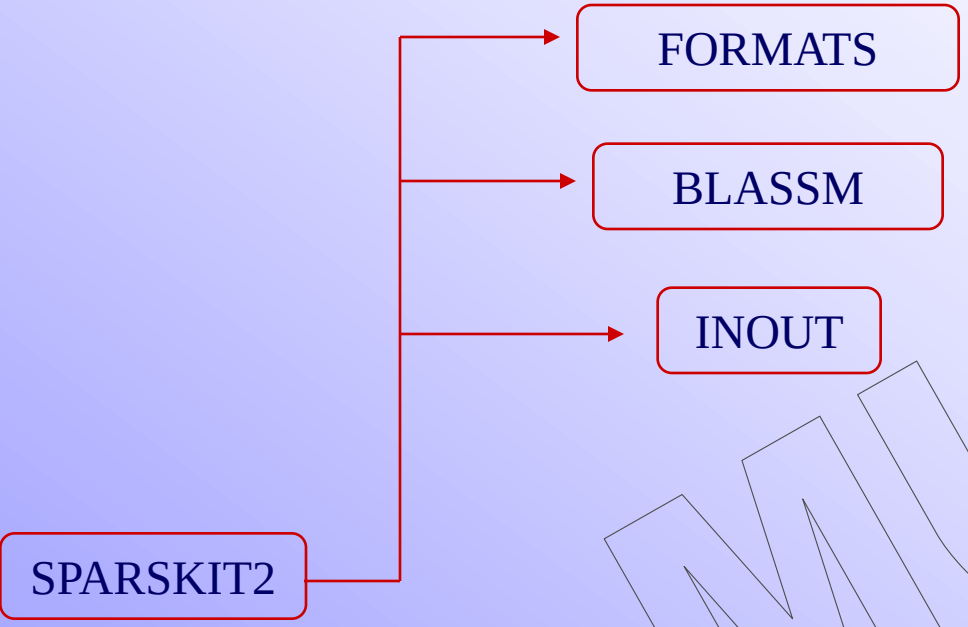


Organizzazione di SPARSKIT



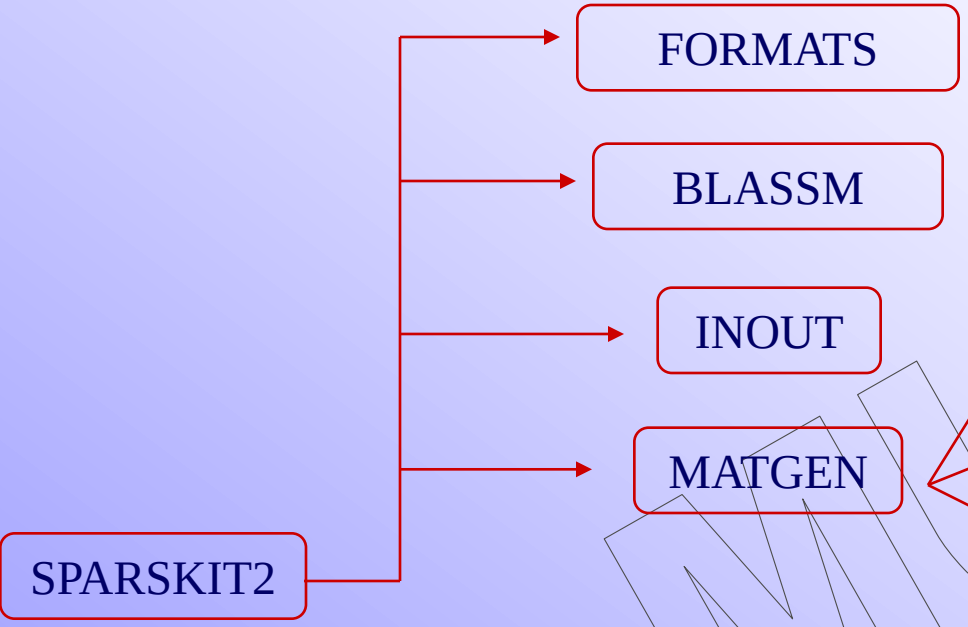
BLASSM : Basic Linear Algebra Subprograms with Sparse Matrices (Routine per l'Algebra Lineare di Base per Matrici Sparse).
Contiene due moduli: blassm.f e matvec.f.

Organizzazione di SPARSKIT



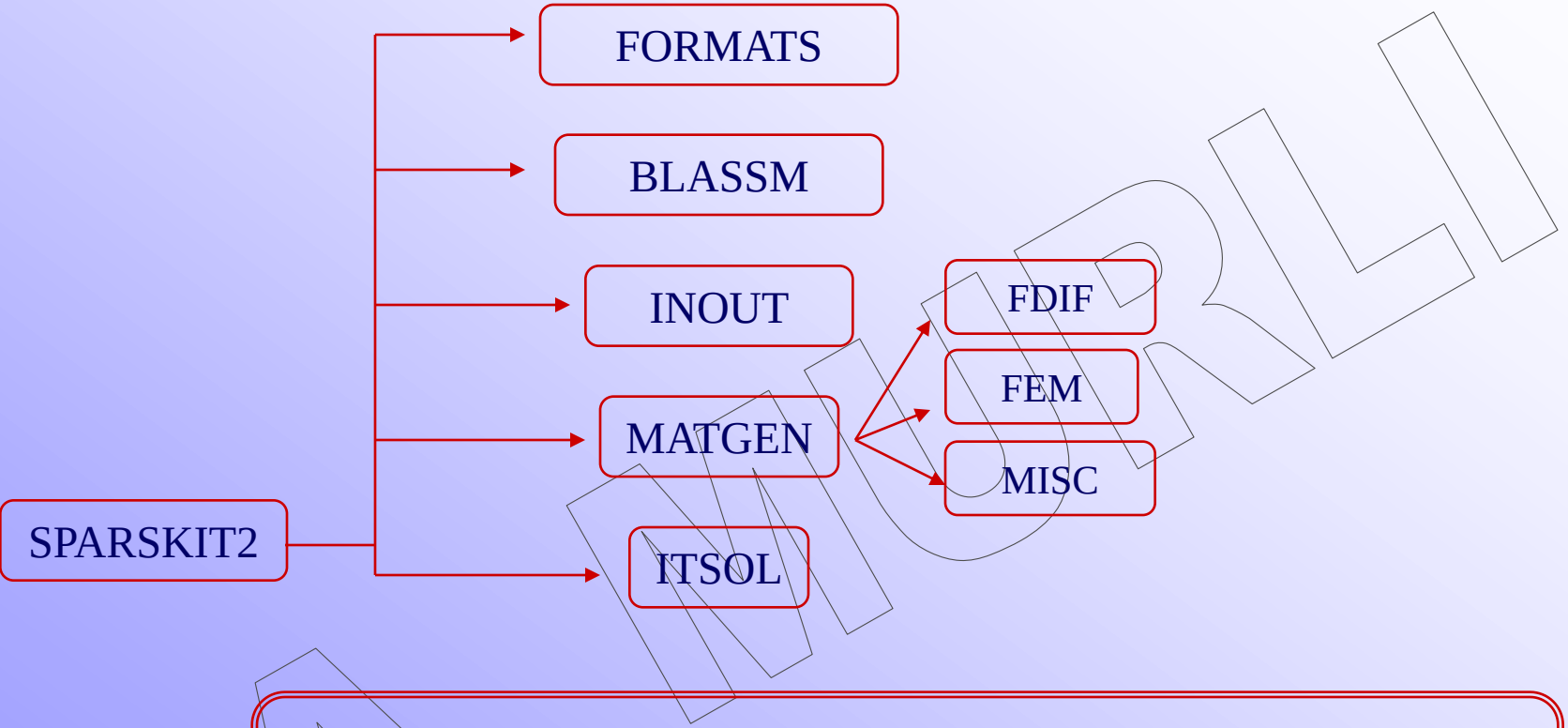
INOUT : contiene le routines per l'input e l'output dei dati (modulo inout.f).

Organizzazione di SPARSKIT



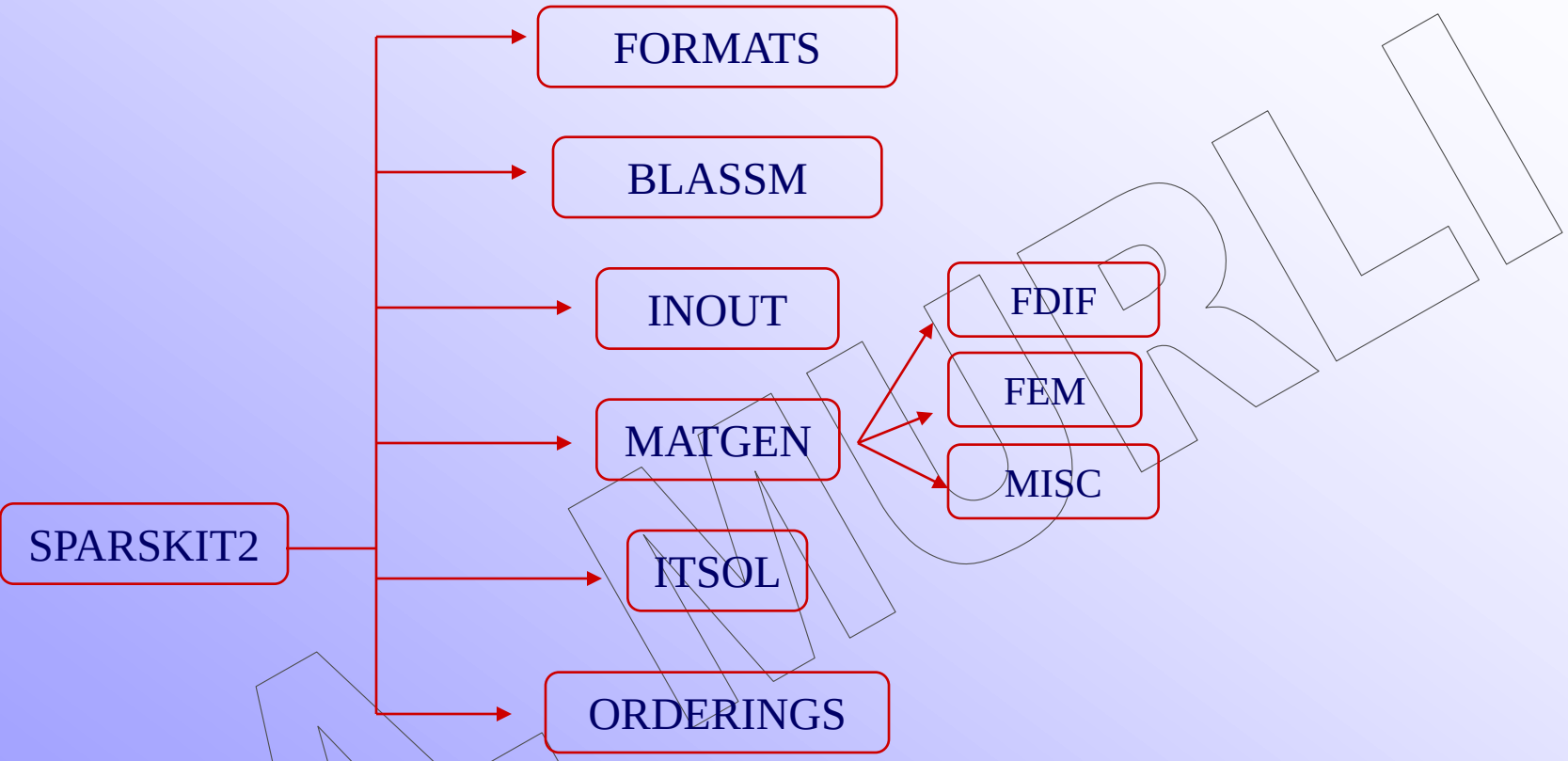
MATGEN : contiene le routines per la generazione delle matrici relative a test case.

Organizzazione di SPARSKIT



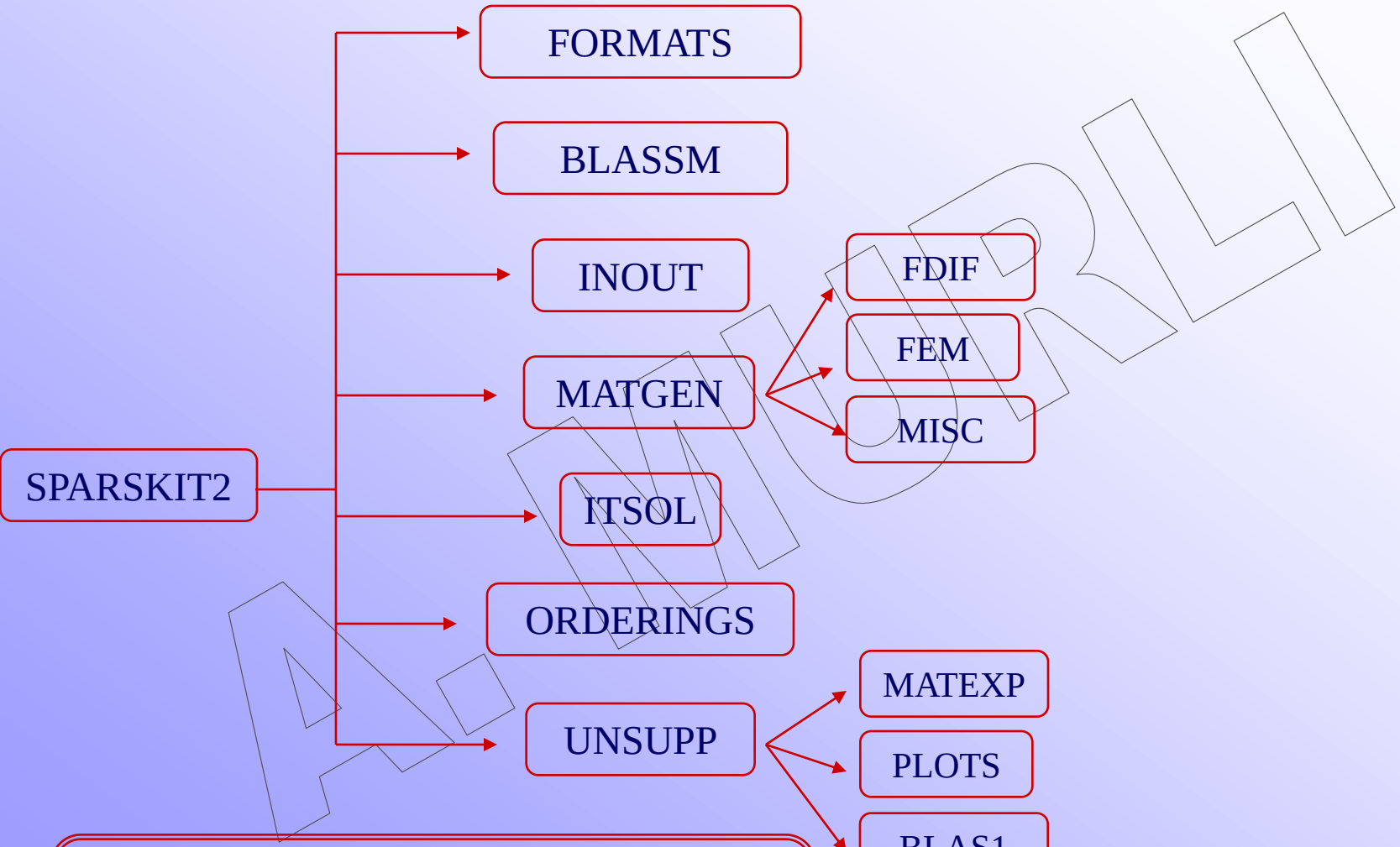
ITSOL : contiene metodi iterativi. Sono implementati i principali metodi iterativi e precondizionatori.

Organizzazione di SPARSKIT



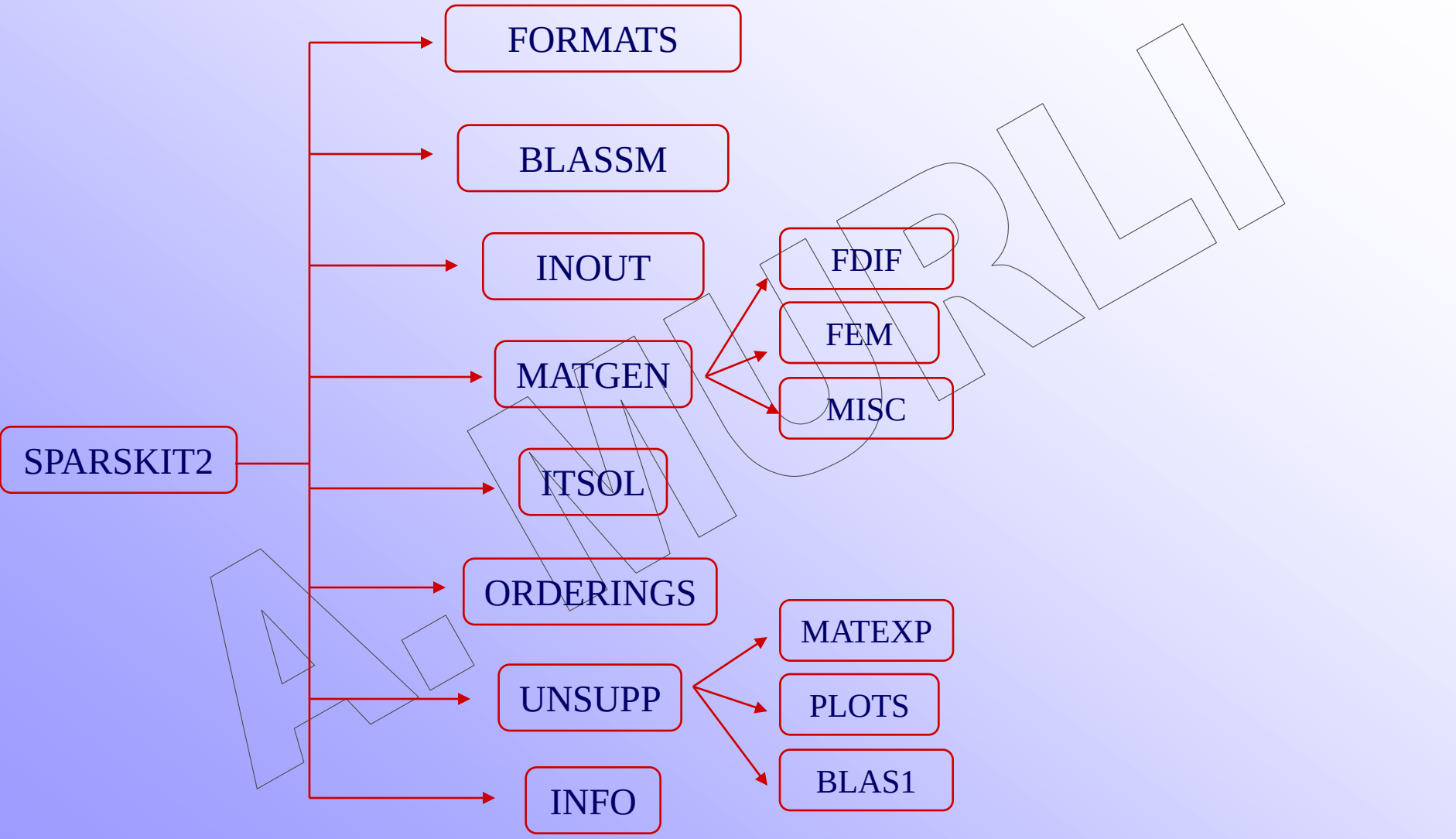
ORDERINGS : contiene le routines per il riordino degli elementi utilizzando tre distinti metodi (levset.f , color.f , ccn.f).

Organizzazione di SPARSKIT



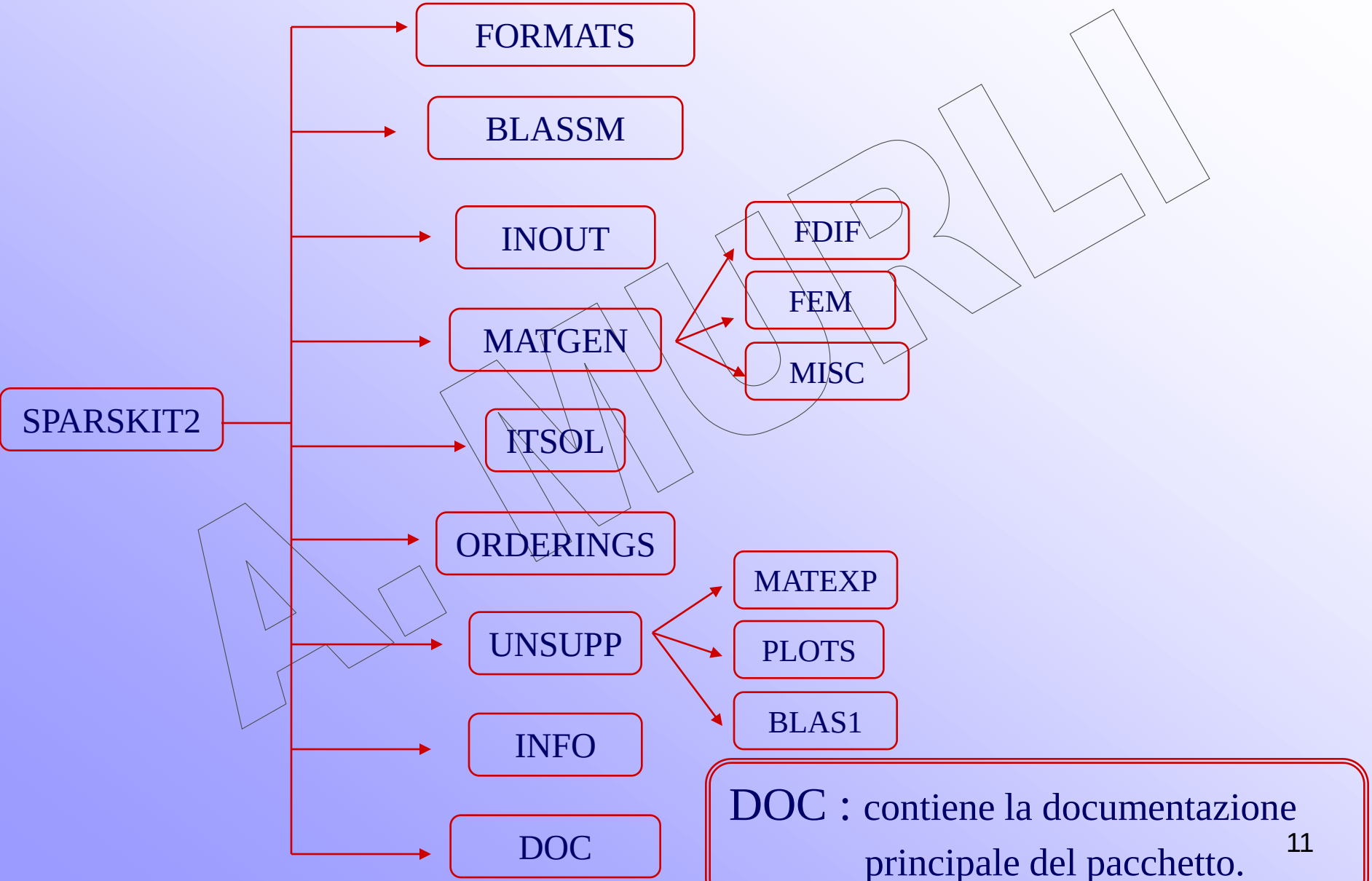
UNSUPP : contiene routines
e drivers di supporto.

Organizzazione di SPARSKIT



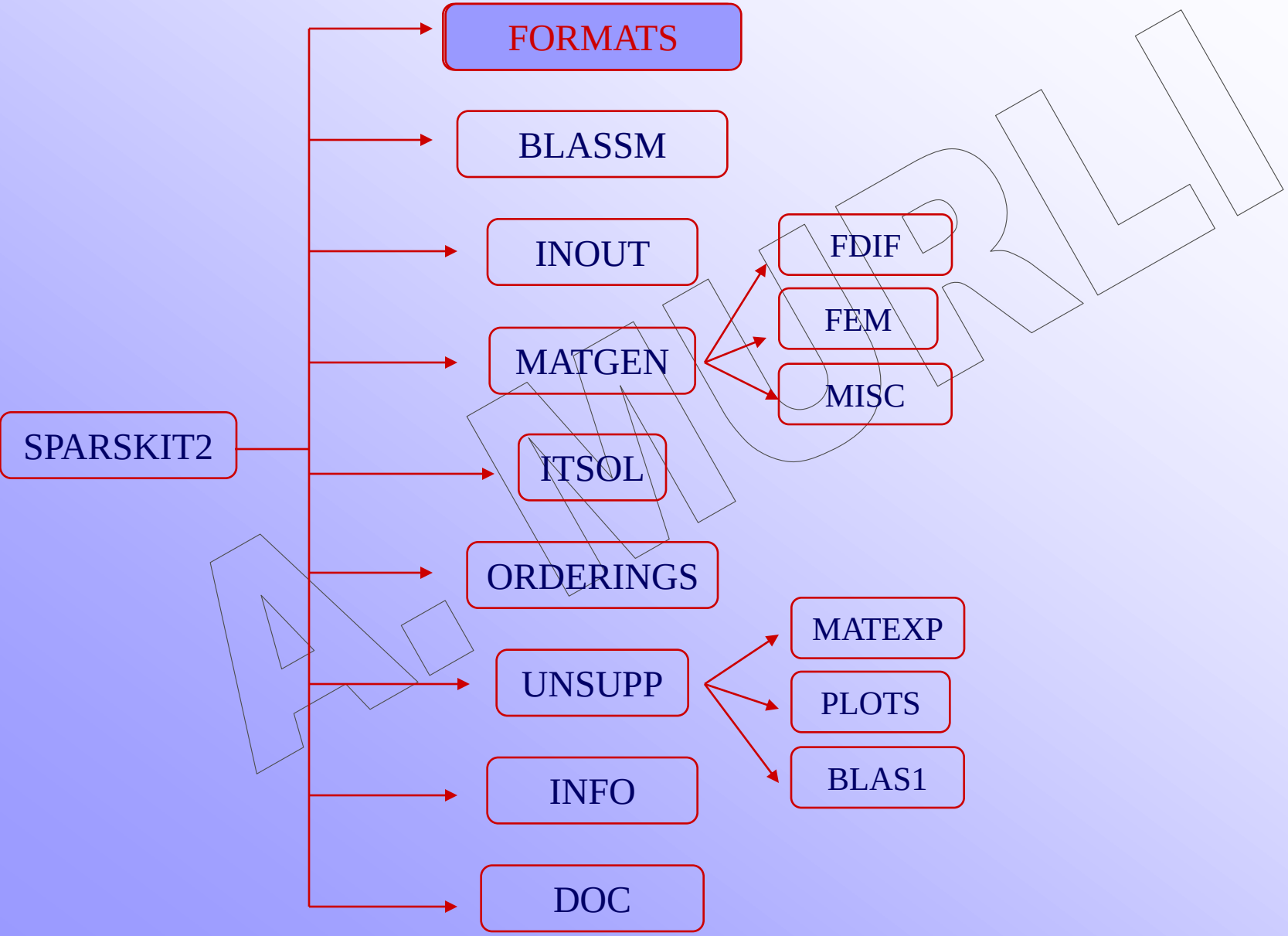
INFO : contiene informazioni utili sulle routines.

Organizzazione di SPARSKIT



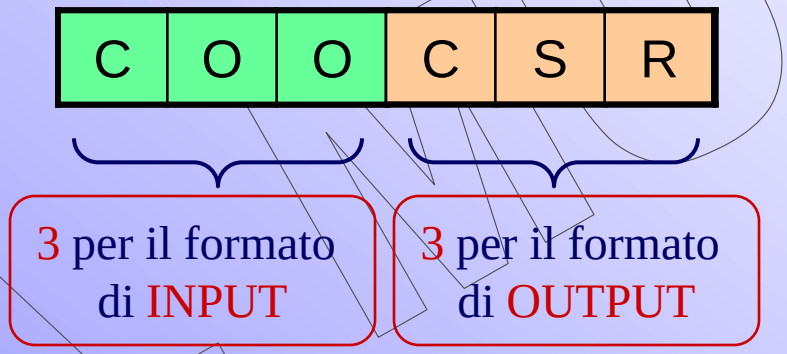
DOC : contiene la documentazione principale del pacchetto.

Dettagli su SPARSKIT



I FORMATS : formats.f

formats.f permette di trasformare il tipo di memorizzazione di una matrice sparsa. Il nome delle routine che tale file include è costituito da 6 lettere:



Analizziamo alcune delle 16 memorizzazioni supportate.

I FORMATS : formats.f

Alcune rappresentazioni di una matrice sparsa :

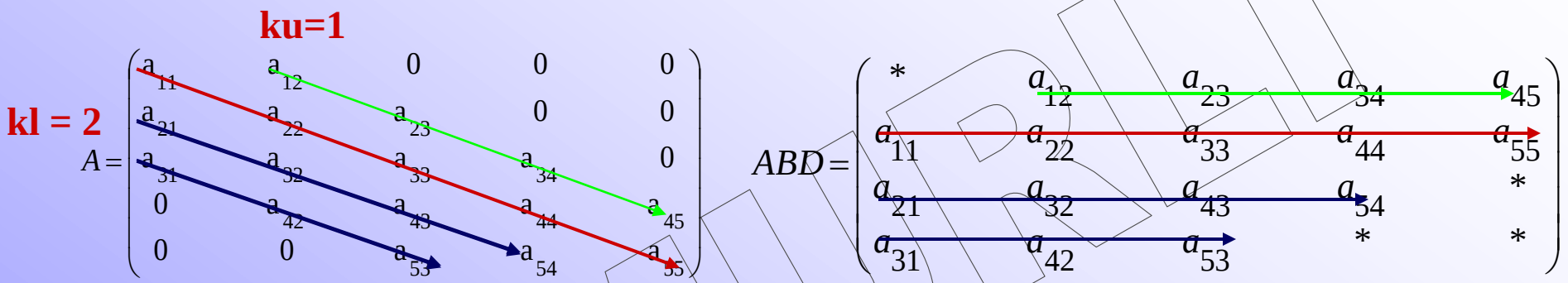
• DNS (Dense format) :
sono memorizzati tutti gli elementi della matrice sparsa.

• BND (Banded Linpack format) :
memorizzazione utilizzata da Linpack per le matrici a banda.

BND format

Assegnata la matrice a banda $A \in R^{mxn}$, con ku diagonali superiori e kl diagonali inferiori.

Esempio: m=n=5, kl=2 , ku=1 $ABD \in R^{4X5}$



Si memorizza A in modo più compatto in un array bidimensionale $ABD \in R^{(kl+ku+1)xn}$;
Le colonne di A sono memorizzate lungo le colonne di ABD e le diagonali di A sono memorizzate nelle righe di ABD

Osservazioni :

- Lo schema è opportuno utilizzarlo quando kl,ku<<min(m,n)
- a_{ij} è memorizzato in ABD(ku+1+i-j,j) con $\max(1,j-ku) \leq i \leq \min(m,j+kl)$

I FORMATS : formats.f

Alcune rappresentazioni supportate :

• DNS (Dense format) :
sono memorizzati tutti gli elementi della generica matrice sparsa.

• BND (Banded Linpack format) :
memorizzazione utilizzata da Linpack per le matrici a banda.

• CSR (Compressed Sparse Row format) :
è la rappresentazione di base utilizzata da SPARSKIT.

CSR format

Assegnata la matrice sparsa $A \in R^{nxm}$ e con nnz elementi non nulli.

Esempio: m=n=5, nnz=12

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

Memorizziamo la matrice A nei 3 vettori AA, JA, IA:

- Il vettore di reali AA contiene gli nnz elementi non nulli di A memorizzati per righe;
- Il vettore di interi JA contiene l'indice di colonna degli elementi a_{ij} di A memorizzati in AA. La dimensione di JA è nnz;
- Il vettore di reali IA contiene il puntatore al primo elemento di ogni riga in AA e JA. Quindi il valore di IA(i) coincide con la posizione iniziale in AA e JA della i_esima riga. La lunghezza di IA è n+1 con IA(n+1)=IA(1)+nnz

AA	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
JA	1	4	1	2	4	1	3	4	5	3	4	5
IA	1	3	6	10	12	13						

IA(j+1)-IA(j)?

I FORMATS : formats.f

Alcune rappresentazioni supportate :

• DNS (Dense format) :
sono memorizzati tutti gli elementi della generica matrice sparsa.

• BND (Banded Linpack format) :
memorizzazione utilizzata da Linpack per le matrici a banda.

• CSR (Compressed Sparse Row format) :
rappresentazione di base utilizzata da SPARSKIT : memorizzazione degli elementi non nulli per righe .

• CSC (Compressed Sparse Column format) :
memorizzazione degli elementi non nulli per colonne.

CSC format

Assegnata la matrice sparsa $A \in R^{nxm}$ e con nnz elementi non nulli.

Esempio: m=n=5, nnz=12

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

Memorizziamo la matrice A nei 3 vettori AA, JA, IA:

- Il vettore di reali AA contiene gli nnz elementi non nulli di A memorizzati per colonne;
- Il vettore di interi JA contiene l'indice di riga degli elementi a_{ij} di A memorizzati in AA. La dimensione di JA è nnz;
- Il vettore di interi IA contiene il puntatore al primo elemento di ogni colonna in AA e JA. Quindi il valore di IA(i) coincide con la posizione iniziale in AA e JA della i_esima colonna. La lunghezza di IA è n+1 con IA(n+1)=IA(1)+nnz

AA	1.	3.	6.	4.	7.	10.	2.	5.	8.	11.	9.	12.
JA	1	2	3	2	3	4	1	2	3	4	3	5
IA	1	4	5	7	11	13						

I FORMATS : formats.f

Alcune rappresentazioni supportate :

• DNS (Dense format) :
sono memorizzati tutti gli elementi della generica matrice sparsa.

• BND (Banded Linpack format) :
memorizzazione utilizzata da Linpack per le matrici a banda.

• CSR (Compressed Sparse Row format) :
rappresentazione di base utilizzata da SPARSKIT: memorizzazione degli elementi non nulli per righe .

• CSC (Compressed Sparse Column format) :
memorizzazione degli elementi non nulli per colonne.

• COO (Coordinate format) :
la memorizzazione degli elementi non nulli di una matrice sparsa in qualunque ordine .

COO format

Assegnata la matrice sparsa $A \in R^{nxm}$ e con nnz elementi non nulli.

Esempio: m=n=5, nnz=12

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

Memorizziamo la matrice A nei 3 vettori AA, JR, JC di dimensione nnz:

- Il vettore di reali AA contiene gli nnz elementi non nulli di A memorizzati in qualunque ordine;
- Il vettore di interi JR contiene l'indice di riga di ciascuno degli nnz elementi memorizzati in AA;
- Il vettore di interi JC contiene l'indice di colonna di ciascuno degli nnz elementi memorizzati in AA;

AA	12.	9.	7.	5.	1.	2.	11.	3.	6.	4.	8.	10.
JA	5	3	3	2	1	1	4	2	3	2	3	4
IA	5	5	3	4	1	4	4	1	1	2	4	3

I FORMATS : formats.f

Alcune rappresentazioni supportate :

• DNS (Dense format) :
sono memorizzati tutti gli elementi della generica matrice sparsa.

• BND (Banded Linpack format) :
memorizzazione utilizzata da Linpack per le matrici a banda.

• CSR (Compressed Sparse Row format) :
rappresentazione di base utilizzata da SPARSKIT : memorizzazione degli elementi non nulli per righe

• CSC (Compressed Sparse Column format) :
memorizzazione degli elementi non nulli per colonne

• COO (Coordinate format) :
memorizzazione degli elementi non nulli di una matrice sparsa in qualunque ordine

⋮

I FORMATS : formats.f

Analizziamo in dettaglio alcune delle 31 routine contenute in formats.f

- CSRDNS : converte il formato dal CSR al DNS
- DNSCSR : converte il formato dal DNS al CSR
- COOCSR : converte il formato dal COO al CSR
- COICSR : converte il formato dal COO al CSR effettuandolo *in-place*.
- CSRCOO : converte il formato dal CSR al COO
- CSRCSC : converte il formato dal CSR al CSC, effettuando una trasposizione della matrice iniziale.

⋮

Esempio 1 : **dnscsr**

subroutine dnscsr (nrow, ncol, nzmax, dns, ndns, a, ja, ia, ierr)

Parametri di input:

nrow – numero di righe della matrice assegnata

ncol – numero di colonne della matrice assegnata

nzmax – massimo numero di elementi non zero della matrice assegnata; sarà la dimensione degli array a e ja

dns – array di dim. `nrow x ncol` contenente, in input, la matrice in formato denso

ndns – prima dimensione dell'array dns

Parametri di output:

a – array dei valori non nulli della matrice assegnata nel formato CSR (primo vettore della rappresentazione)

ja – array delle colonne degli elementi non nulli della matrice assegnata nel formato CSR (secondo vettore della rappresentazione)

ia – array di puntatori; terzo vettore della rappresentazione della matrice assegnata nel formato CSR

ierr – indicatore di errore; ierr=0 esecuzione corretta, ierr=i l'esecuzione si è fermata nell'analisi della riga i, poiché non c'è spazio sufficiente nei vettori a, ja, ia (come definito dal parametro nzmax).

Esempio 1 : **dnscsr**

subroutine dnscsr (nrow, ncol, nzmax, dns, ndns, a, ja, ia, ierr)

dns =

1.	0.	2.	0.	0.
0.	3.	0.	4.	0.
0.	0.	0.	5.	6.
0.	7.	0.	0.	8.
0.	0.	0.	0.	9.

In INPUT

nrow = 5
ncol = 5
nzmax = 9
ndns = 5

In OUTPUT

a =

1.	2.	3.	4.	5.	6.	7.	8	9.

ja =

1	3	2	4	4	5	2	5	5

ia =

1	3	5	7	9	10

ierr = 0

Esempio 2 : **csrdns**

subroutine csrdns (nrow, ncol, a, ja, ia, dns, ndns, ierr)

Parametri di input:

nrow – numero di righe della matrice assegnata

ncol – numero di colonne della matrice assegnata

a – array dei valori non nulli della matrice assegnata nel formato CSR (primo vettore della rappresentazione)

ja – array delle colonne degli elementi non nulli della matrice assegnata nel formato CSR (secondo vettore della rappresentazione)

ia – array di puntatori; terzo vettore della rappresentazione della matrice assegnata nel formato CSR

ndns – prima dimensione dell'array dns

Parametri di output:

dns – array dove allocare la matrice in formato denso, dns(ndns,*)

ierr – indicatore di errore; ierr=0 esecuzione corretta, ierr=i l'esecuzione si è fermata nell'analisi della riga i, poiché ha riscontrato un identificativo per la colonna maggiore del parametro ncol

Esempio 2 : **csrdns**

subroutine csrdns (nrow, ncol, a, ja, ia, dns, ndns, ierr)

a =

1.	4.	6.	9.	2.	2.	8.	5	7.

ja =

1	2	1	2	3	4	2	3	5

ia =

1	3	6	7	9	10

In INPUT

nrow = 5
ncol = 5
ndns = 9

In OUTPUT

ierr = 0

dns =

1.	4.	0.	0.	0.
6.	9.	2.	0.	0.
0.	0.	0.	2.	0.
0.	8.	5.	0.	0.
0.	0.	0.	0.	7.

I FORMATS : unary.f

Le routine contenute nel file unary.f eseguono operazioni di base sulle matrici. Analizziamone alcune:

- SUBMAT : estrae una sottomatrice quadrata o rettangolare da una matrice sparsa. Sia la matrice in Input che quella in Output sono nel formato CSR. La routine è in-place
- COPMAT : copia una matrice nel formato CSR in un'altra anch'essa in formato CSR.
- GETELM : è una funzione il cui valore di ritorno è l'elemento a_{ij} per ogni coppia (i,j) assegnata. Come parametro di ritorno abbiamo anche l'indirizzo dell'elemento negli array A e JA
- GETDIA : estrae la diagonale della matrice assegnata. Si può scegliere di non modificare la matrice di input o azzerare tutti i suoi elementi diagonali.

I FORMATS : unary.f

- CPERM : effettua una permutazione delle colonne della matrice assegnata A, ovvero calcola la matrice B=A·Q, con Q matrice di permutazione.
- RPERM : effettua una permutazione delle righe della matrice assegnata A, ovvero calcola la matrice B=P·A, con P matrice di permutazione.
- RETMX : restituisce l'elemento massimo in valore assoluto per ciascuna riga della matrice assegnata A.
- INFDIA : calcola il numero di elementi non zero di ciascuna delle 2n-1 diagonali della matrice assegnata. Si noti che la prima diagonale considerata è quella denominata –n costituita dal solo elemento di input $a_{n,1}$ mentre l'ultima è quella denominata n, costituita dal solo elemento $a_{1,n}$
- RNRMS : calcola le norme delle righe della matrice assegnata. Le norme $\| \|_1$, $\| \|_2$ ed $\| \|_\infty$ sono supportate.

⋮

Esempio 1 : **copmat (ao ← a)**

subroutine copmat (nrow, a, ja, ia, ao, jao, iao, ipos, job)

Parametri di input:

nrow – numero di righe della matrice assegnata

a – array dei valori non nulli della matrice assegnata nel formato CSR (primo vettore della rappresentazione)

ja – array delle colonne degli elementi non nulli della matrice assegnata nel formato CSR (secondo vettore della rappresentazione)

ia – array di puntatori; terzo vettore della rappresentazione della matrice assegnata nel formato CSR

ipos – intero, indica la posizione dove copiare il primo elemento negli array ao e jao; ovvero iao(1) = ipos

Parametri di output:

ao – array dove viene copiato l'aray a

jao – array dove viene copiato l'aray ja

iao – array dove viene copiato l'aray ia

job – intero, indicatore del lavoro. Se job ≠ 1 i valori non sono stati copiati

Esempio 1 : copmat (ao ← a)

subroutine copmat (nrow, a, ja, ia, ao, jao, iao, ipos, job)

In INPUT

nrow = 5
ipos = 1

a =	1.	4.	6.	9.	2.	2.	8.	5	7.
ja =	1	2	1	2	3	4	2	3	5
ia =	1	3	6	7	9	10			

In OUTPUT

job = 1

ao =	1.	4.	6.	9.	2.	2.	8.	5	7.
jao =	1	2	1	2	3	4	2	3	5
iao =	1	3	6	7	9	10			

Esempio 2 : **infdia**

subroutine infdia (n, ja, ia, ind, idiag)

Parametri di input:

n – dimensione della matrice assegnata

ja – array delle colonne degli elementi non nulli della matrice assegnata nel formato CSR (secondo vettore della rappresentazione)

ia – array di puntatori; terzo vettore della rappresentazione della matrice assegnata nel formato CSR

Parametri di output:

ind – array di interi di lunghezza 2*n-1. Il k-esimo elemento del vettore ind contiene il numero di elementi non nulli nella diagonale k.

idiag – intero, contiene il numero di elementi non nulli trovati sulle diagonali della matrice assegnata

Esempio 2 : **infdia**

subroutine infdia (n, ja, ia, ind, idiag)

In INPUT

n = 5

ja = | 2 | 3 | 1 | 4 | 5 | 2 | 3 | 5 | 4 | 1 | 4 |

ia = | 1 | 3 | 6 | 9 | 10 | 11 |

In OUTPUT

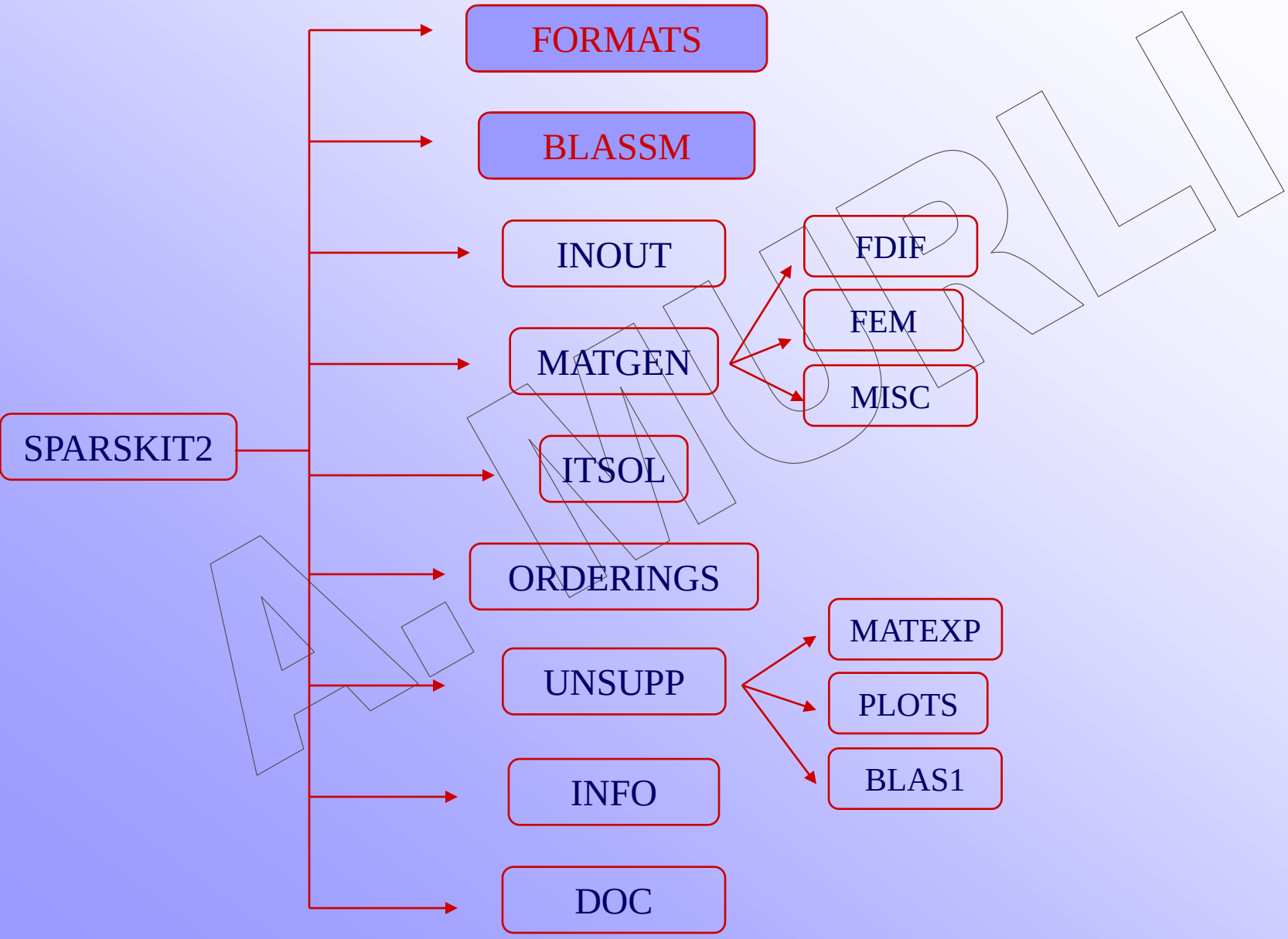
idiag = 11

ind = | 1 | 0 | 0 | 3 | 2 | 1 | 3 | 1 | 0 |

Infatti leggendo l'output usando il formato denso per la matrice assegnata, abbiamo:

0.	3.	2.	0.	0.
1.	0.	0.	2.	3.
0.	6.	8.	0.	7.
0.	0.	0.	1.	0.
3.	0.	0.	8.	0.

Dettagli su SPARKIT



BLASSM : blassms.f

I moduli contenuti in BLASSM eseguono operazioni algebriche di base.
In particolare il modulo blassms.f esegue operazioni che coinvolgono 2 matrici, quali: $C = A + B$, $C = A + \beta B$, $C = AB$, etc..

Analizziamo in dettaglio le 9 routine contenute in blassms.f

- AMUB : calcola il prodotto di due matrici, ovvero $C = A \cdot B$, dove sia A che B sono in formato CSR.
- APLB : calcola la somma di due matrici, ovvero $C = A + B$, dove sia A che B sono in formato CSR.
- APLSB : calcola $C = A + \sigma B$, dove σ è uno scalare e sia A che B sono matrici in formato CSR.
- APMBT : calcola sia la somma $C = A + B^T$ che la differenza $C = A - B^T$
- APLSBT : calcola l'operazione $C = A + sB^T$

BLASSM : blassms.f

- DIAMUA : calcola il prodotto di una matrice diagonale (posta a sinistra) ed una matrice sparsa, ovvero C = D·A , con D matrice diagonale ed A matrice sparsa, entrambe memorizzate in formato CSR.

- AMUDIA : calcola il prodotto di una matrice sparsa ed una matrice diagonale (posta a destra), ovvero C = A·D , con D matrice diagonale ed A matrice sparsa, entrambe memorizzate in formato CSR.

- APLDIA : calcola la somma di una matrice sparsa e di una matrice diagonale (C = A + D)

- APLSCA : *in-place* somma uno scalare alla diagonale di una matrice sparsa, ovvero esegue A = A + σ I, dove σ è uno scalare, A la matrice sparsa ed I la matrice identica.

Esempio 1 : amub (C = A*B)

subroutine amub (nrow, ncol, job, a, ja, ia, b, jb, ib, c, jc, ic, nzmax, iw, ierr)

Parametri di input:

nrow – intero, numero di righe della matrice A e della matrice C

ncol – intero, numero di colonne della matrice B e della matrice C

job – intero. Indicatore del lavoro. Se job=0 vengono creati solo i vettori jc e ic, ovvero solo la struttura della matrice C viene creata ma non i suoi valori

a, ja, ia – sono i 3 array per la memorizzazione della matrice A nel formato CSR

b, jb, ib – sono i 3 array per la memorizzazione della matrice B nel formato CSR

nzmax – intero, rappresenta la lunghezza dei vettori c e jc

iw – array di interi, area di lavoro di lunghezza uguale al numero di colonne della matrice A

Parametri di output:

c, jc, ic – sono i 3 array per la memorizzazione della matrice prodotto C nel formato CSR

ierr – intero, indicatore di errore. ierr=0 indica esecuzione corretta; ierr=k>0 indica che l'esecuzione è terminata nel calcolo della k-esima riga della matrice C

Esempio 1 : amub (C = A*B)

subroutine amub (nrow, ncol, job, a, ja, ia, b, jb, ib, c, jc, ic, nzmax, iw, ierr)

a =	1.	4.	9.	2.	8.	5.	7.
ja =	1	2	2	4	2	3	5
ia =	1	3	5	6	8	8	
b =	1.	2.	4.	2.	1.	3.	1.
jb =	2	3	1	5	5	4	1
ib =	1	3	5	6	7	8	

In INPUT

nrow = 5
ncol = 5
job = 1
nzmax = 15
iw[ncol]

In OUTPUT

ierr = 0

c =	16.	1.	2.	8.	36.	6.	18.	12.	32.	21.	7.
jc =	1	2	3	5	1	4	5	4	1	5	1
ic =	1	5	8	9	11	12					

Esempio 2 : aplb (C=A+ B)

subroutine aplb (nrow, ncol, job, a, ja, ia, b, jb, ib, c, jc, ic, nzmax, iw, ierr)

Parametri di input:

nrow – intero, numero di righe della matrice A e della matrice B

ncol – intero, numero di colonne della matrice A e della matrice B

job – intero. Indicatore del lavoro. Se job=0 vengono creati solo i vettori jc e ic, ovvero solo la struttura della matrice C viene creata ma non i suoi valori

a, ja, ia – sono i 3 array per la memorizzazione della matrice A nel formato CSR

b, jb, ib – sono i 3 array per la memorizzazione della matrice B nel formato CSR

nzmax – intero, rappresenta la lunghezza dei vettori c e jc

iw – array di interi, area di lavoro di lunghezza uguale al numero di colonne della matrice A

Parametri di output:

c, jc, ic – sono i 3 array per la memorizzazione della matrice prodotto C nel formato CSR

ierr – intero, indicatore di errore. ierr=0 indica esecuzione corretta; ierr=k>0 indica che l'esecuzione è terminata nel calcolo della k_esima riga della matrice C

Esempio 2 : aplb (C=A+ B)

subroutine aplb (nrow, ncol, job, a, ja, ia, b, jb, ib, c, jc, ic, nzmax, iw, ierr)

a =	1.	4.	9.	2.	8.	5.	7.
ja =	1	2	2	4	2	3	5
ia =	1	3	5	6	8	8	
b =	1.	2.	4.	2.	1.	3.	1.
jb =	2	3	1	5	5	4	1
ib =	1	3	5	6	7	8	

In INPUT

nrow = 5
ncol = 5
job = 1
nzmax = 10
iw[ncol]

In OUTPUT

ierr = 0

c =	1.	5.	2.	4.	9.	2.	2.	4.	1.	8.	5.	3.	1.	7.
jc =	1	2	3	1	2	4	5	4	5	2	3	4	1	5
ic =	1	4	8	10	13	15								

BLASSM : matvec.f

Il modulo matvec.f esegue operazioni di base che coinvolgono una matrice ed un vettore, ad es. il prodotto matrice per vettore e la risoluzione di sistemi triangolari

In dettaglio analizziamo alcune delle 15 routine contenute in matvec.f:

- AMUX : esegue il prodotto di una matrice per un vettore ($y = A\,x$). La matrice sparsa A è memorizzata nel formato CSR.
- ATMUX : esegue il prodotto della trasposta di una matrice per un vettore ($y = A^T \cdot x$). La matrice sparsa A è essere memorizzata nel formato CSR. Si noti come questa routine può eseguire anche il prodotto di una matrice sparsa A per un vettore, con A memorizzata nel formato CSC.
- LSOL : risolve un sistema la cui matrice è triangolare inferiore unitaria (elementi diagonali uguali a 1).
  La matrice è memorizzata nel formato CSR

BLASSM : matvec.f

- LDSOL : risolve un sistema la cui matrice è triangolare inferiore.
  La matrice è memorizzata nel formato MSR (Modified Sparse Row). Gli elementi della diagonale sono memorizzati in ordine inverso.

- LSOLC : risolve un sistema la cui matrice è triangolare inferiore ed unitaria. La matrice è memorizzata nel formato CSC.

- USOL : risolve un sistema la cui matrice è triangolare superiore ed unitaria. La matrice è memorizzata nel formato CSR

- USOLC : risolve un sistema la cui matrice è triangolare superiore ed unitaria. La matrice è memorizzata nel formato CSC

⋮

Le altre routine contenute in matvec.f eseguono le stesse operazioni delle routine illustrate ma con la matrice coinvolta nell'operazione memorizzata in altri formati.

Esempio 1 : amux (y=A*x)

subroutine amux (n, x, y, a , ja, ia)

Parametri di input:

n – intero, numero di righe della matrice A

x – array di reali di lunghezza pari alle colonne della matrice A

a, ja, ia – sono i 3 array per la memorizzazione della matrice A nel formato CSR

Parametri di output:

y – array di reali di lunghezza n che contiene il prodotto A*x

Esempio 1 : amux (y=A*x)

subroutine amux (n, x, y, a , ja, ia)

x =	1.	2.	3.	4.	5.		
a =	1.	4.	9.	2.	8.	5.	7.
ja =	1	2	2	4	2	3	5
ia =	1	3	5	6	8	8	

In INPUT

n = 5

In OUTPUT

y =	9.	26.	16.	31.	35.

Esempio 2 : atmux ($y=A^T *x$)

subroutine atmux (n, x, y, a , ja, ia)

Parametri di input:

n – intero, numero di righe della matrice A

x – array di reali di lunghezza pari alle colonne della matrice A

a, ja, ia – sono i 3 array per la memorizzazione della matrice A nel formato CSR

Parametri di output:

y – array di reali di lunghezza n che contiene il prodotto $A^T *x$

Esempio 2 : atmux ($y=A^T *x$)

subroutine atmux (n, x, y, a , ja, ia)

x =	1.	2.	3.	4.	5.		
a =	1.	4.	9.	2.	8.	5.	7.
ja =	1	2	2	4	2	3	5
ia =	1	3	5	6	8	8	

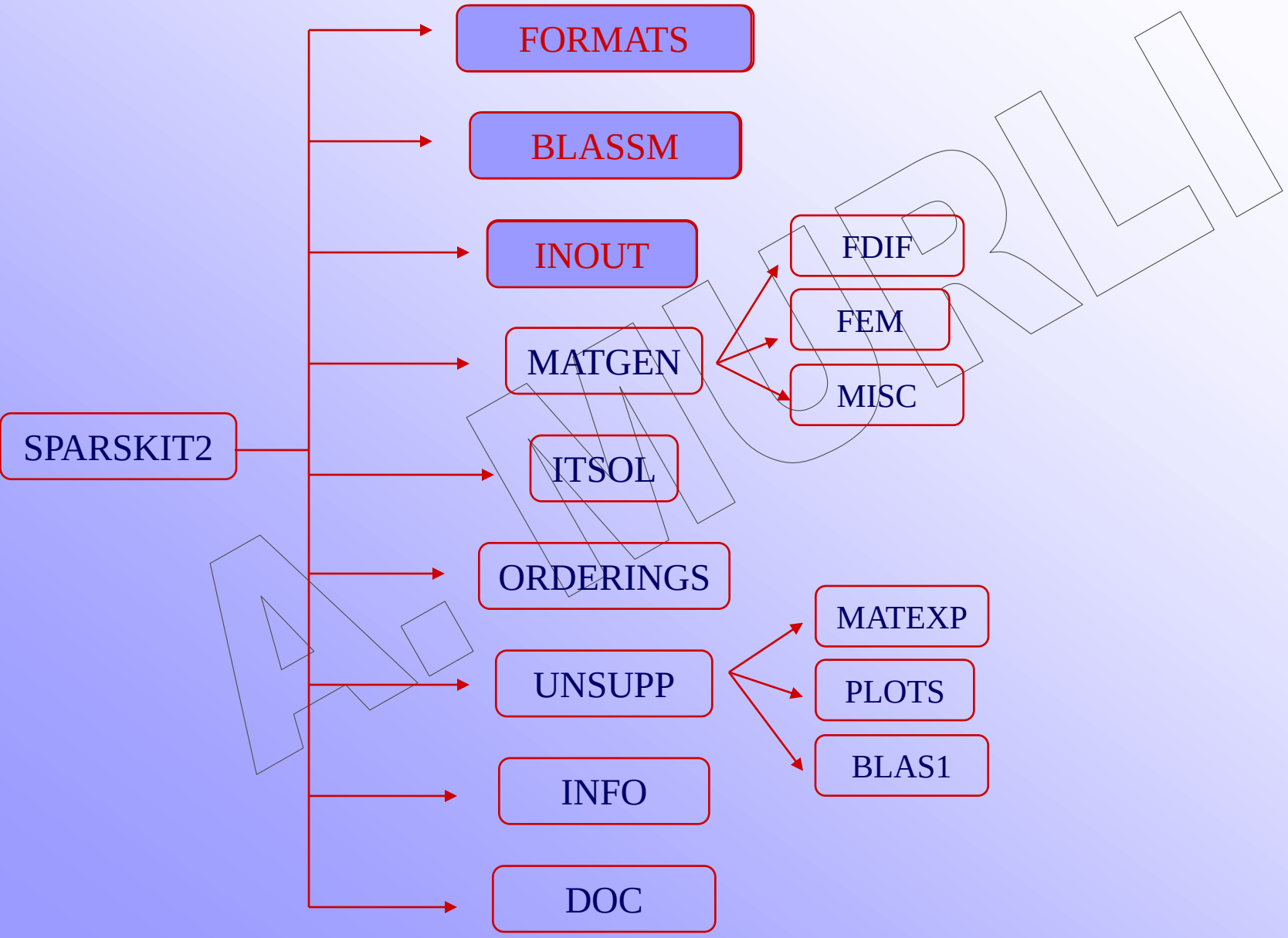
In INPUT

n = 5

In OUTPUT

y =	1.	54.	20.	16.	35.

Dettagli su SPARSKIT



INOUT

INOUT comprende routine per la ***lettura, scrittura, plot e visualizzazione*** delle strutture delle matrici sparse.

Analizziamo in dettaglio alcune delle 11 routine contenute in inout.f:

HB, o **Harwell Boeing Sparse Matrix File Format**, memorizza una matrice sparsa in un file. Lo spazio richiesto è ridotto usando un *compressed column storage format*. Se la matrice è letta da file si utilizza lo stesso formato per rappresentarla in memoria

- READMT : legge una matrice nel formato HB (Harwell Boeing Sparse Matrix File Format) [Compressed Column Storage (CCS)= CSC o CSR]
- PRTMT : crea un file HB a partire da un'arbitraria matrice nei formati CSR o CSC.
- PSPLTM : genera in un file ps il plot della struttura della matrice A.
- PLTMT : genera un file *pic* per il plot della struttura della matrice A (*pic files contain images in an uncompressed format*).
- READSK : legge una matrice nel formato CSR.
- PRTUNF : scrive matrici in formato CSR in un file non formattato, cioè scritto senza un formato particolare.
- READUNF : legge file non formattati contenenti matrici in formato CSR.

⋮

Esempio 1 : readmt

Il puntatore al file in cui è rappresentata la matrice in formato H/B

In INPUT

```
title                                                                   key
             5         1         1         3         0
RUA                    5         5        12         0
(6I3)          (12I3)          (5E15.8)          (5E15.8)
 1  4  6  8 11 13
 1  3  5  1  2  2  3  1  4  5  3  5
 2.02765219E-01 6.03792479E-01 1.98814268E-01 1.52739270E-02 7.46785677E-01
 8.46221418E-01 5.25152496E-01 8.38118445E-01 3.79481018E-01 8.31796018E-01
 4.28892365E-01 1.89653748E-01
```

Che descrive la matrice in formato denso:

$$A = \begin{pmatrix} 2.02765219E-01 & 0. & 6.03792479E-01 & 0. & 1.98814268E-01 \\ 1.52739270E-02 & 7.46785677E-01 & 0. & 0. & 0. \\ 0. & 8.46221418E-01 & 5.25152496E-01 & 0. & 0. \\ 8.38118445E-01 & 0. & 0. & 3.79481018E-01 & 8.31796018E-01 \\ 0. & 0. & 4.28892365E-01 & 0. & 1.89653748E-01 \end{pmatrix}$$

Esempio 1 : readmt

$$A = \begin{pmatrix} 2.02765219E-01 & 0. & 6.03792479E-01 & 0. & 1.98814268E-01 \\ 1.52739270E-02 & 7.46785677E-01 & 0. & 0. & 0. \\ 0. & 8.46221418E-01 & 5.25152496E-01 & 0. & 0. \\ 8.38118445E-01 & 0. & 0. & 3.79481018E-01 & 8.31796018E-01 \\ 0. & 0. & 4.28892365E-01 & 0. & 1.89653748E-01 \end{pmatrix}$$

In OUTPUT

a =

2.02765 219E-01	6.03792 479E-01	1.98814 268E-01	1.52739 270E-02	7.46785 677E-01	8.46221 418E-01	5.25152 496E-01	8.38118 445E-01	3.79481 018E-01	8.31796 018E-01	4.28892 365E-01	1.89653 748E-01

ja =

1	3	5	1	2	2	3	1	4	5	3	5

ia =

1	4	6	8	11	13

Esempio 2 : prtmt

a =

2.02765 219E-01	6.03792 479E-01	1.98814 268E-01	1.52739 270E-02	7.46785 677E-01	8.46221 418E-01	5.25152 496E-01	8.38118 445E-01	3.79481 018E-01	8.31796 018E-01	4.28892 365E-01	1.89653 748E-01

ja =

1	3	5	1	2	2	3	1	4	5	3	5

In INPUT

ia =

1	4	6	8	11	13

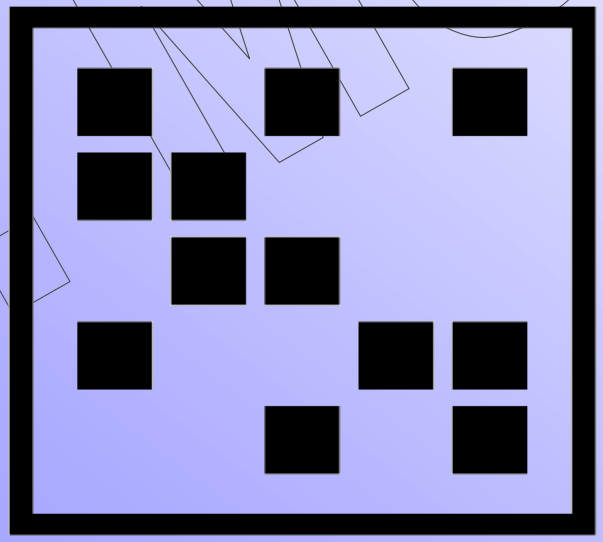
In OUTPUT


```
title                                                   key
          5         1         1         3         0
RUA                 5         5        12         0
(6I3)       (12I3)      (5E15.8)          (5E15.8)
 1  4  6  8 11 13
 1  3  5  1  2  2  3  1  4  5  3  5
 2.02765219E-01 6.03792479E-01 1.98814268E-01 1.52739270E-02 7.46785677E-01
 8.46221418E-01 5.25152496E-01 8.38118445E-01 3.79481018E-01 8.31796018E-01
 4.28892365E-01 1.89653748E-01
```

Esempio 3 : Hb2ps.ex

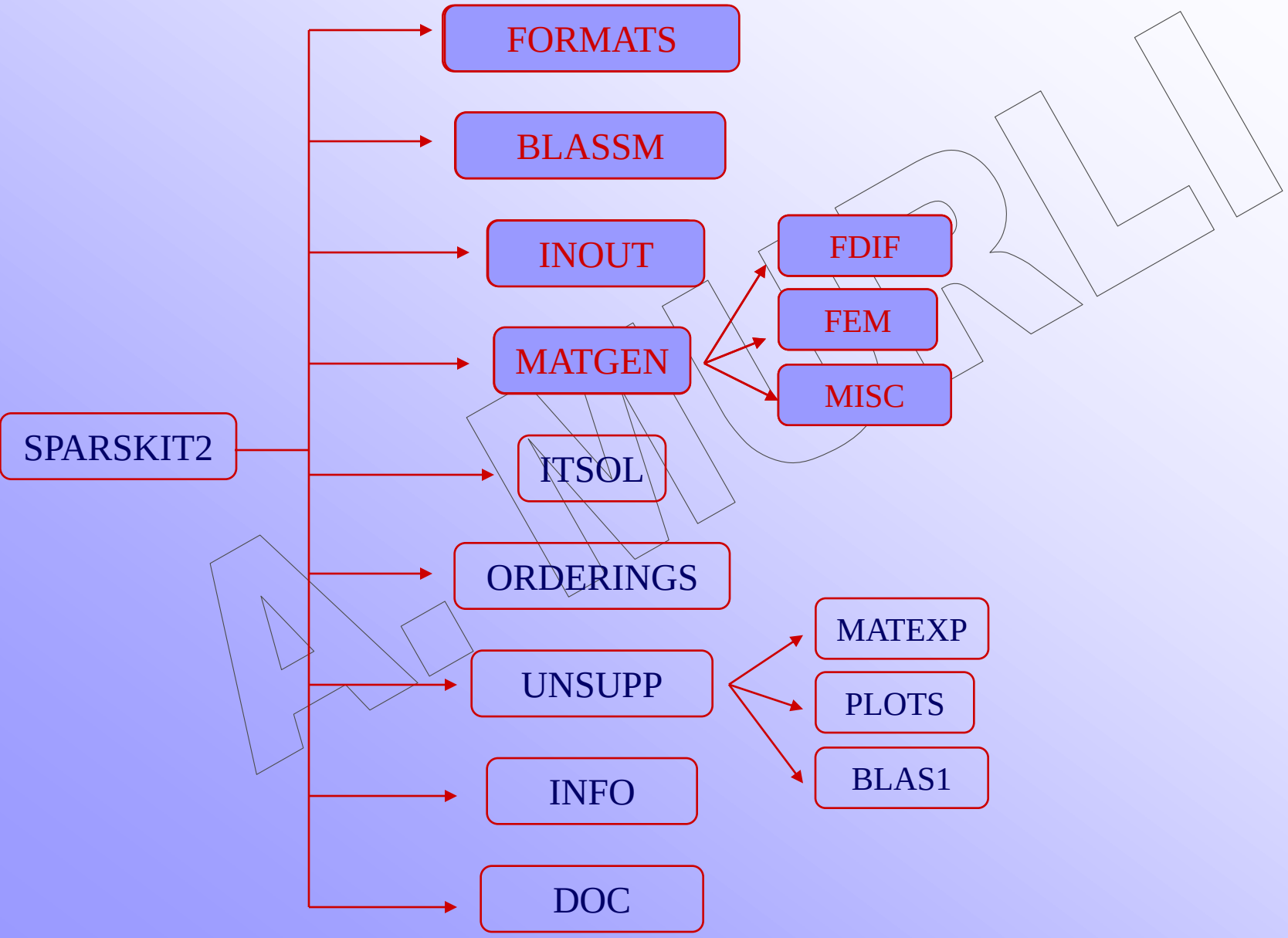
```
title                                                    key
             5         1         1         3         0
RUA                    5         5        12         0
(6I3)        (12I3)        (5E15.8)         (5E15.8)
 1  4  6  8 11 13
 1  3  5  1  2  2  3  1  4  5  3  5
 2.02765219E-01 6.03792479E-01 1.98814268E-01 1.52739270E-02 7.46785677E-01
 8.46221418E-01 5.25152496E-01 8.38118445E-01 3.79481018E-01 8.31796018E-01
 4.28892365E-01 1.89653748E-01
```

Hb2ps.ex



Questo eseguibile utilizza una routine contenuta in PLOTS

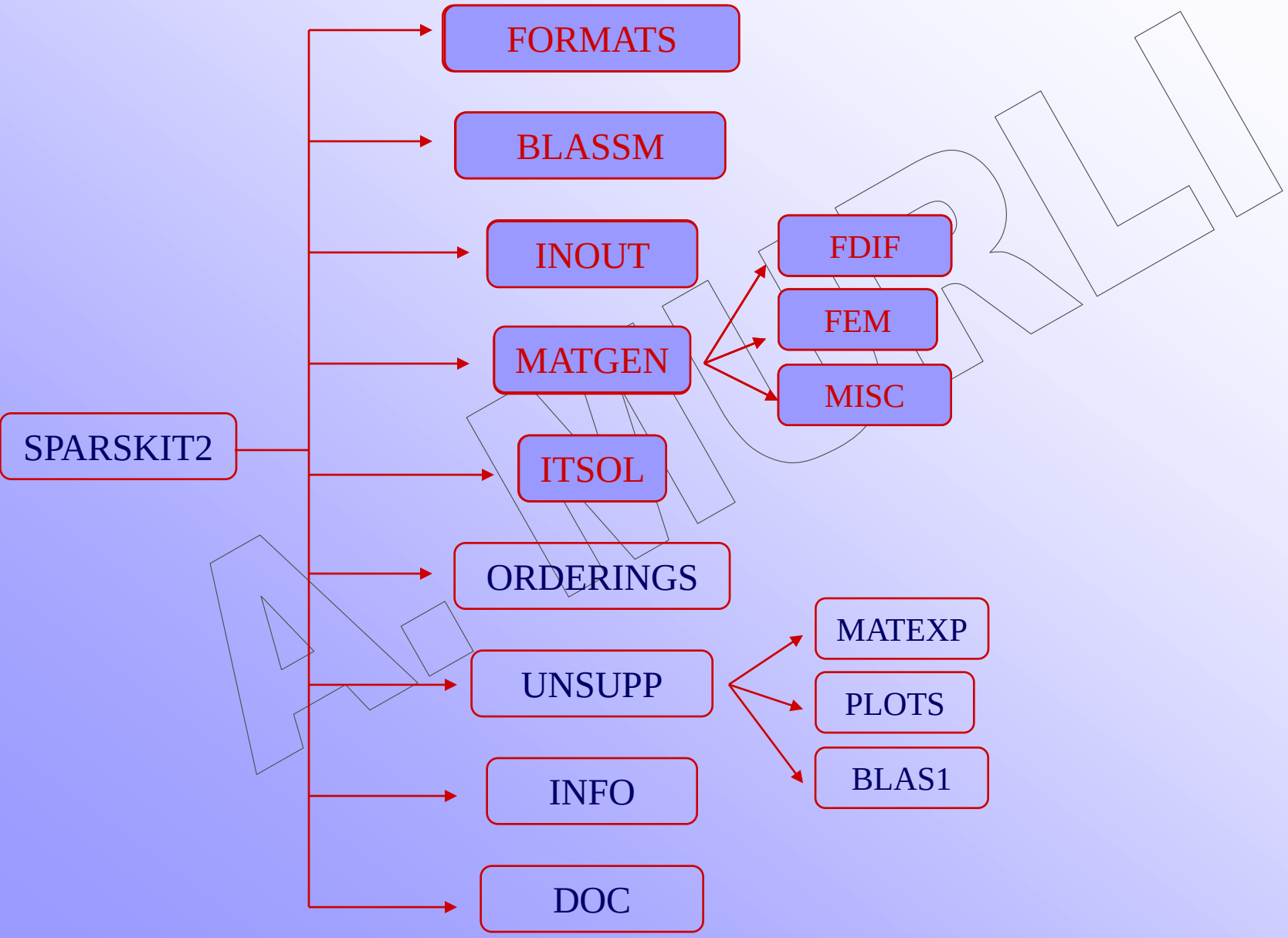
Dettagli su SPARKIT



MATGEN

Nella directory MATGEN sono presenti alcune subroutine e driver per la generazione di test case
La directory è organizzata in tre sotto directory :
FDIF, FEM e MISC.

Dettagli su SPARKIT



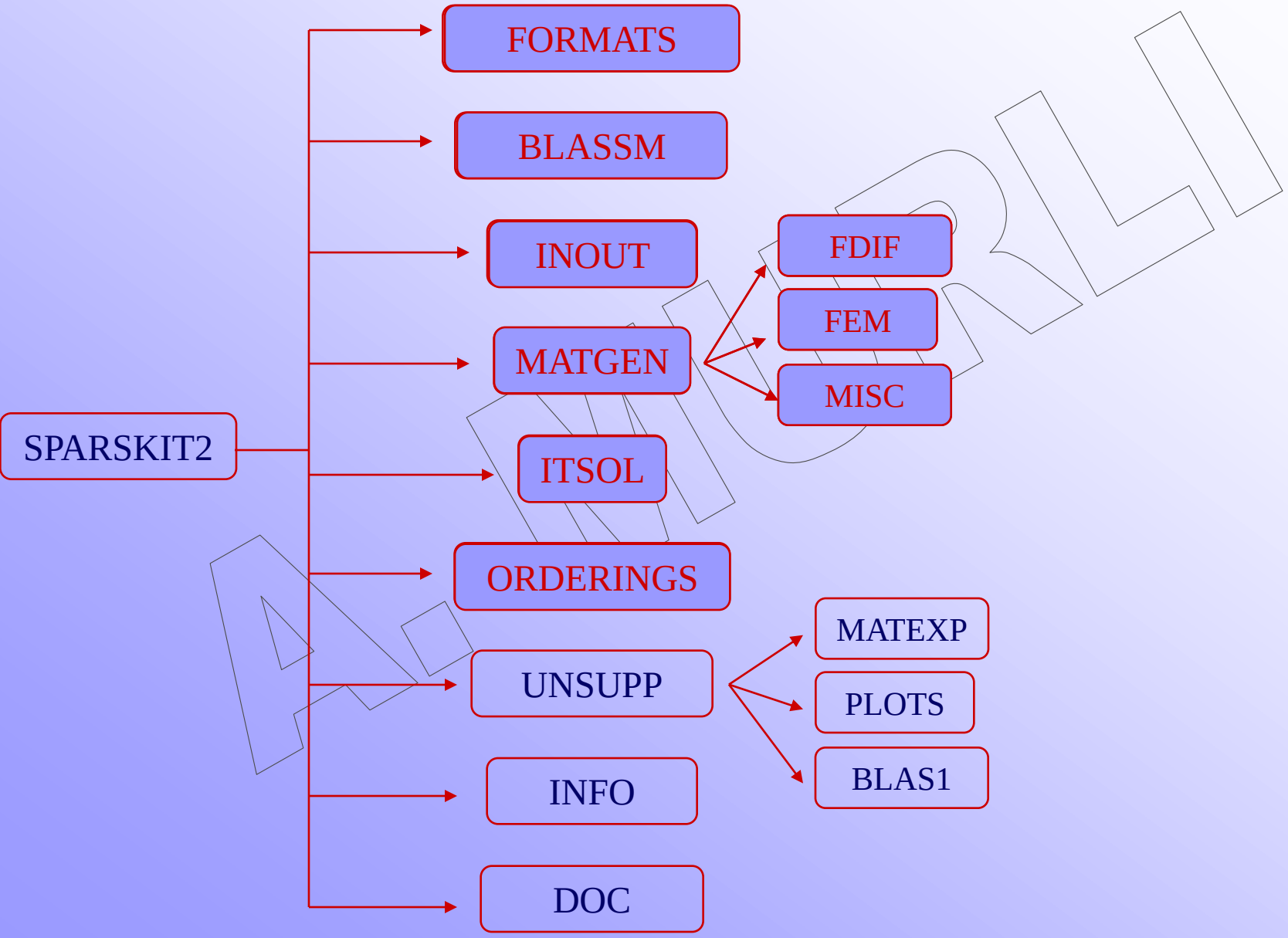
ITSOL

ITSOL comprende due driver principali: ILUT e ITERS:

ILUT implementa l'algoritmo del GMRES (metodo iterativo per la risoluzione di un sistema lineare) precondizionato.

ITERS implementa nove dei più utilizzati metodi iterativi per la risoluzione di sistemi lineari

Dettagli su SPARKIT



ORDERINGS

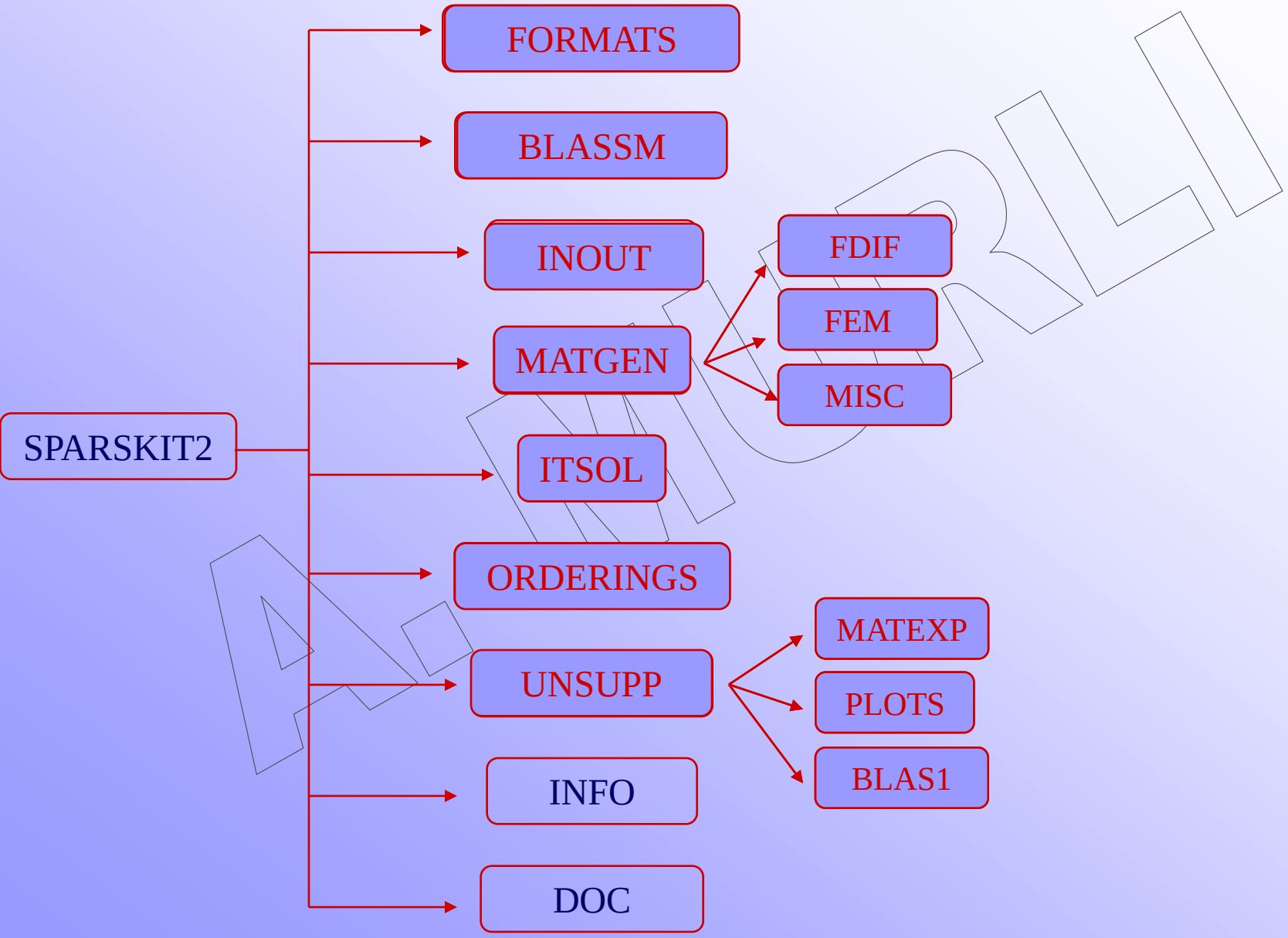
Contiene tre routine che permettono di riordinare gli elementi di una matrice utilizzando vari metodi standard. I metodi sono classificati in relazione:

1. Alle linee di livello (levset.f),
2. All'intensità dei colori di ciascun elemento (color.f)
3. Alle *componenti fortemente connesse* (ccn.f)

Perché riordinare una matrice?

Per ottimizzare l'applicazione di alcuni metodi numerici,
ad esempio la fattorizzazione LU, applicati a matrici sparse
assegnate in uno specifico formato,
risulta conveniente riordinare prima la matrice
secondo particolari metodi; Sparskit offre una scelta
tra i metodi di ordinamento più standard ed utilizzati.

Dettagli su SPARKIT



UNSUPP

Nella directory UNSUPP sono presenti routine (*di supporto*) che non sono di proprietà di SPARSIKIT.

Ci sono 3 sotto directory:

- BLAS1 contiene alcune delle routines del primo livello della libreria BLAS: dcopy, ddot, cswap, etc…;

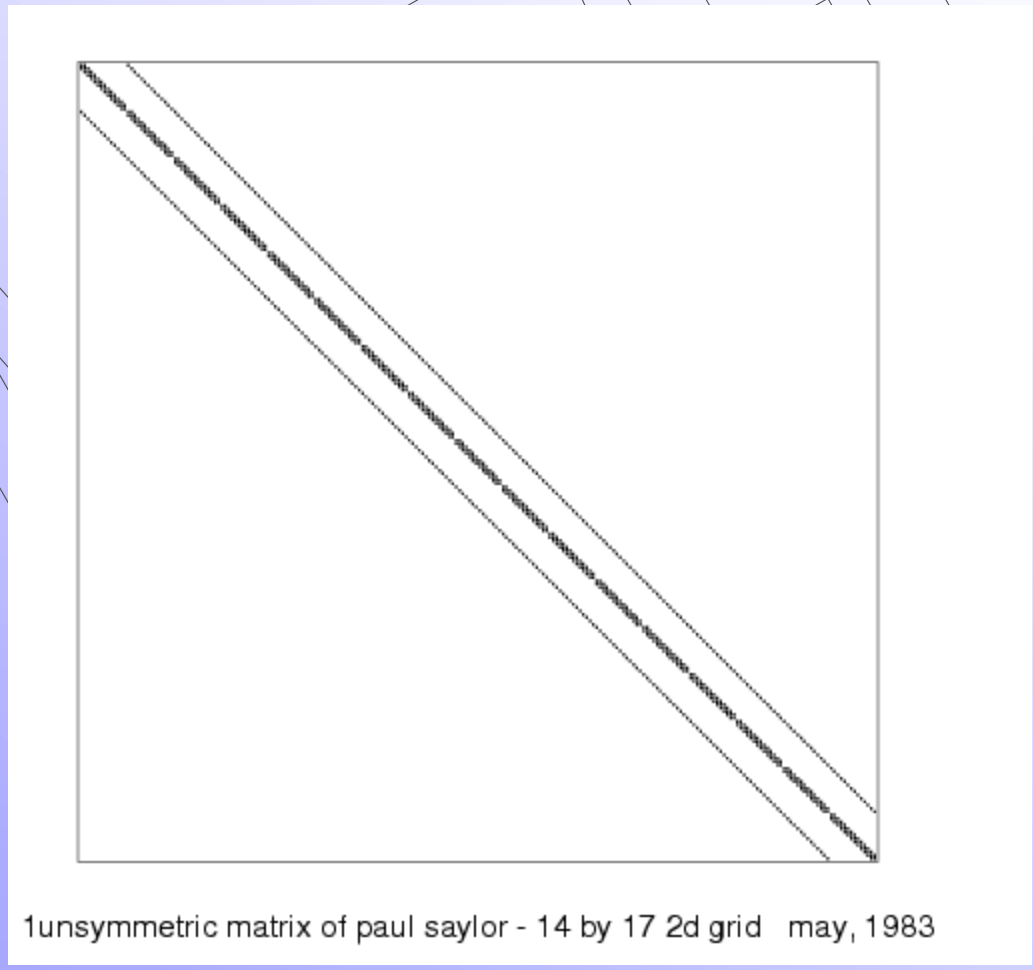
- PLOTS contiene routine per la stampa della struttura sparsa delle matrici;

- MATEXP contiene routine per eseguire operazioni con matrici esponenziali, ad esempio il prodotto di una matrice esponenziale per un vettore: o la risoluzione di equazioni differenziali a derivate parziali

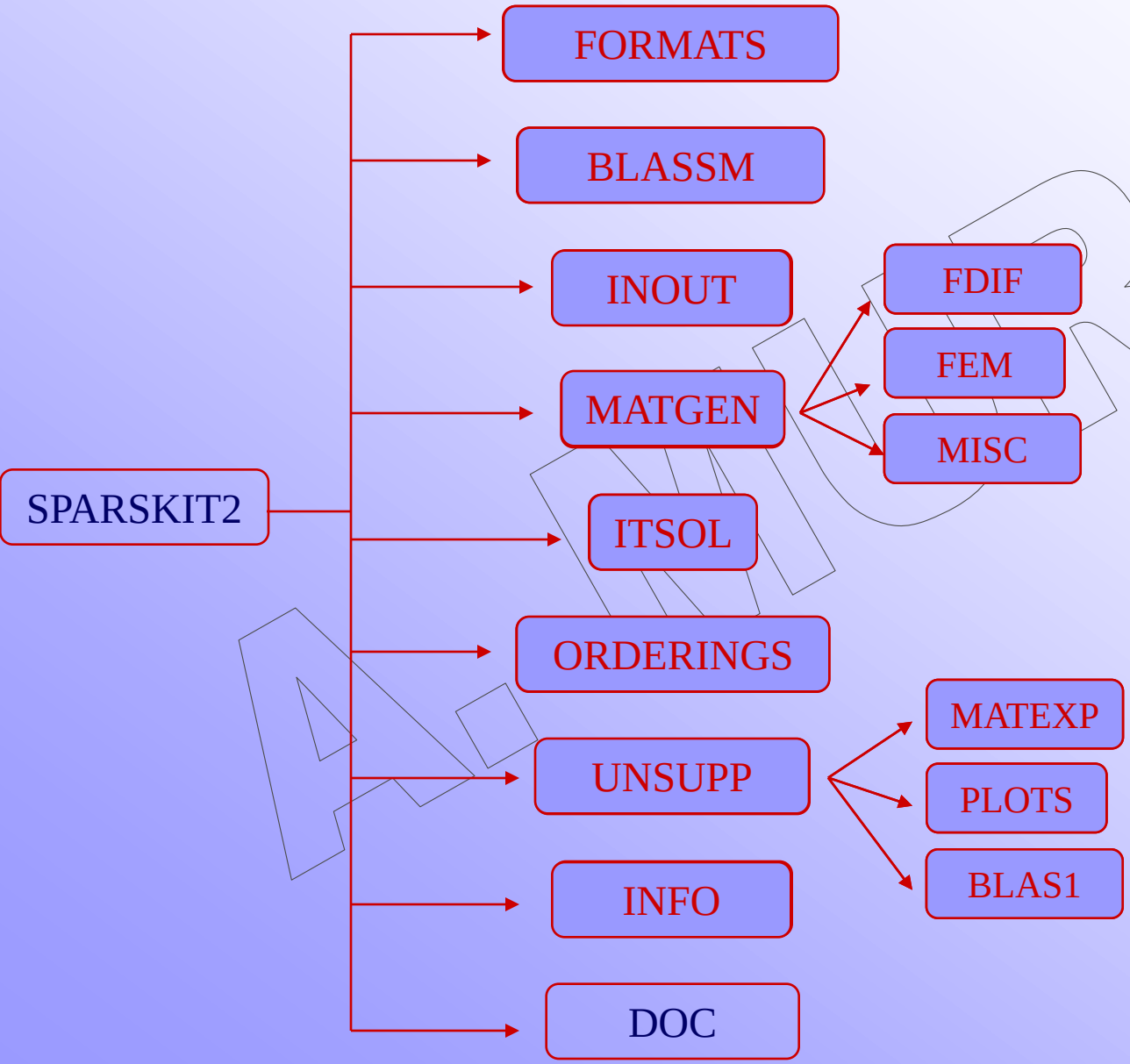
Esempio 1 : psgrd in PLOTS

La routine PSGRID permette di avere in input una matrice nel formato CSR e di creare un file in formato PS che ne visualizzi la struttura sparsa.

Esempio della visualizzazione del file PS di una matrice assegnata



Dettagli su SPARSKIT



INFO

Nella directory INFO sono presenti routines che permettono di avere informazioni sulla matrice sparsa assegnata, ad esempio:

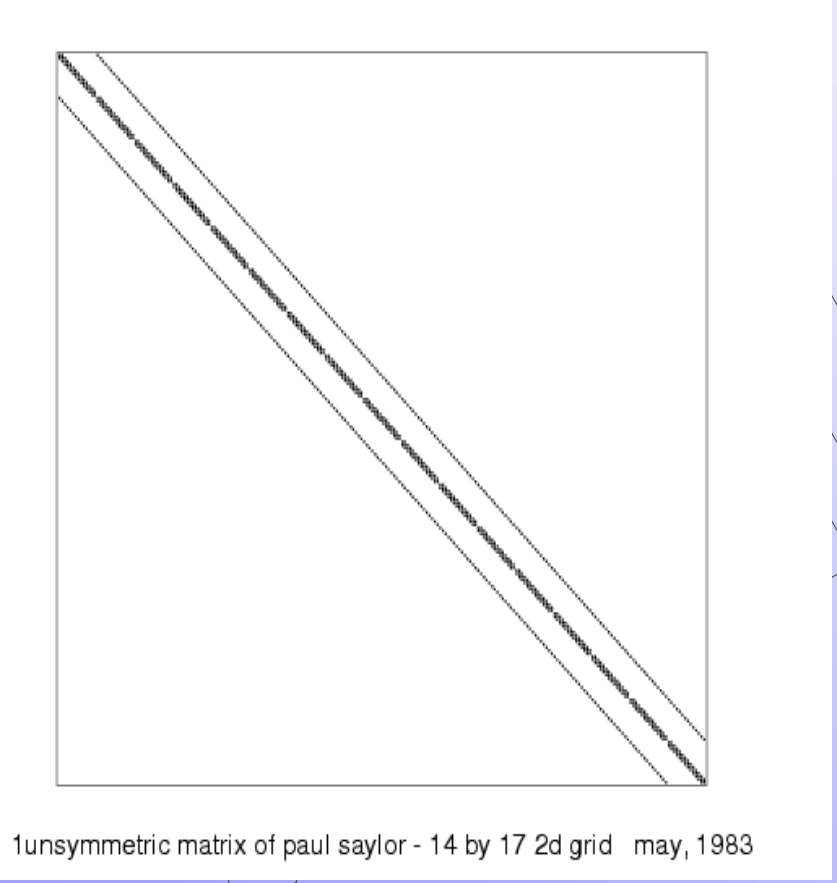
-DINF01 permette di ottenere informazioni elementari, anche di tipo statistico, sulla matrice sparsa assegnata;

-VBRINFO stampa informazioni sulla matrice assegnata nel formato VBR (Variable Block Row)

Variable Block Row=generalizzazione del CSR format per matrici a blocchi

Esempio 1 : INFO

Assegnata in input il file in formato HB della matrice A il cui grafo è :



In INPUT

Con il comando: info1.ex < HB_file

Si ottiene ……

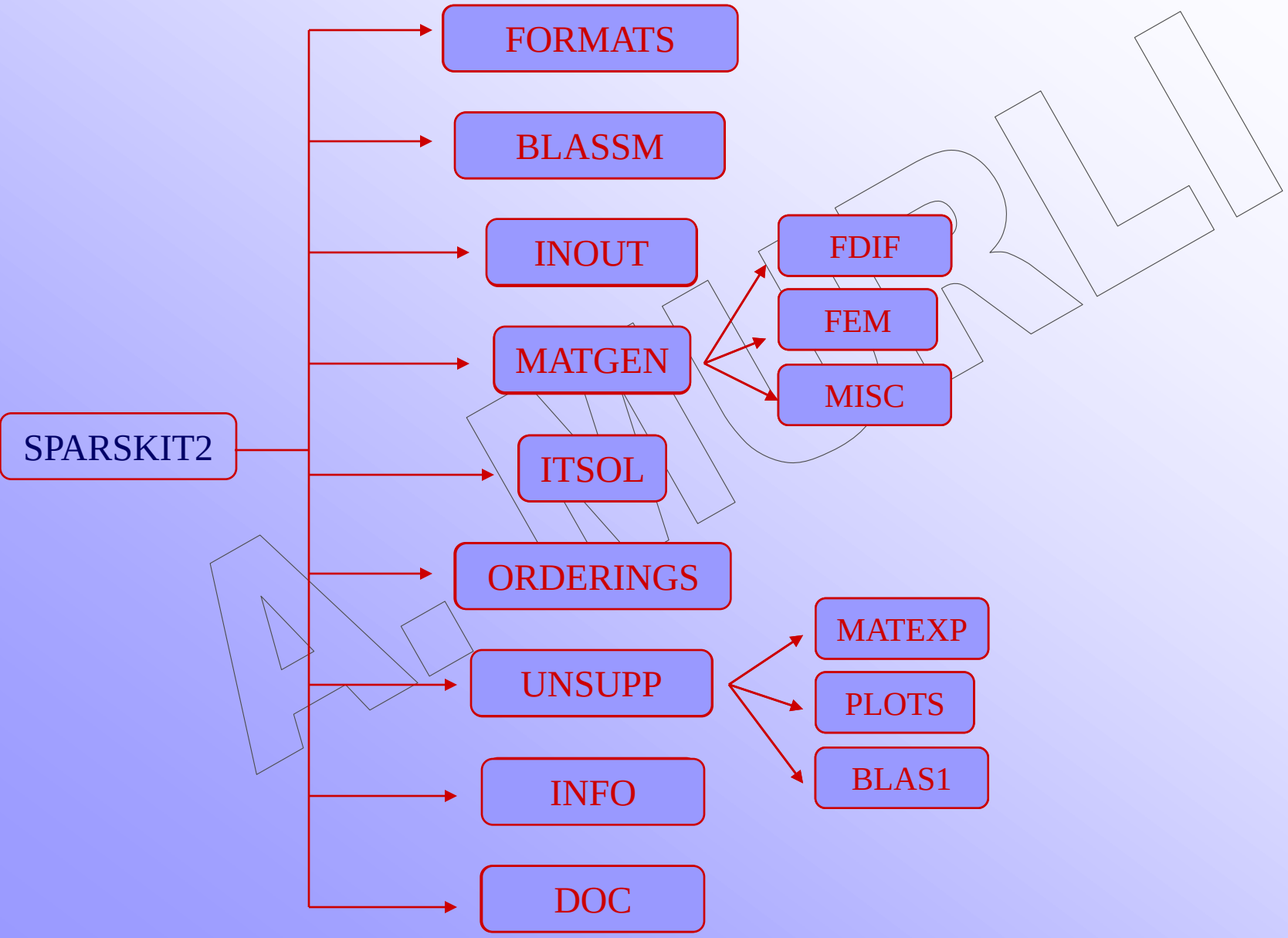
Esempio 1 : **INFO**

In OUTPUT

….. il file :

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* unsymmetric matrix of paul saylor - 14 by 17 2d grid    may, 1983          *
*                   Key = saylr1    , Type = rua                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    *  Dimension N                                         =          238  *
    *  Number of nonzero elements                          =         1128  *
    *  Average number of nonzero elements/Column           =       4.7395  *
    *  Standard deviation for above average                =       0.4757  *
    *  Nonzero elements in strict lower part               =          445  *
    *  Nonzero elements in strict upper part               =          445  *
    *  Nonzero elements in main diagonal                   =          238  *
    *  Weight of longest column                            =            5  *
    *  Weight of shortest column                           =            3  *
    *  Weight of longest row                               =            5  *
    *  Weight of shortest row                              =            3  *
    *  Matching elements in symmetry                       =         1128  *
    *  Relative Symmetry Match (symmetry=1)                =       1.0000  *
    *  Average distance of a(i,j)  from diag.              =    0.595E+01  *
    *  Standard deviation for above average                =    0.654E+01  *
    *-----------------------------------------------------------------------*
    *  Frobenius norm of A                                 =    0.886E+09  *
    *  Frobenius norm of symmetric part                    =    0.991E+09  *
    *  Frobenius norm of nonsymmetric part                 =    0.443E+09  *
    *  Maximum element in A                                =    0.306E+09  *
    *  Percentage of weakly diagonally dominant rows       =    0.761E+00  *
    *  Percentage of weakly diagonally dominant columns    =    0.744E+00  *
    *-----------------------------------------------------------------------*
    *  Lower bandwidth  (max: i-j, a(i,j) .ne. 0)          =           14  *
    *  Upper bandwidth  (max: j-i, a(i,j) .ne. 0)          =           14  *
    *  Maximum Bandwidth                                   =           29  *
    *  Average Bandwidth                                   =    0.275E+02  *
    *  Number of nonzeros in skyline storage               =         6536  *
    *  90% of matrix is in the band of width               =           27  *
    *  80% of matrix is in the band of width               =           27  *
    *  The total number of nonvoid diagonals is            =            5  *
    *  The  5 most important diagonals are (offsets)       :               *
    *    0    14   -14     1    -1                                         *
    *  The accumulated percentages they represent are      :               *
    *  21.1  41.0  60.8  80.4 100.0                                        *
    *-----------------------------------------------------------------------*
    *  The matrix does not have a block structure                          *
    *-----------------------------------------------------------------------*
```

Dettagli su SPARKIT



DOC

Nella directory DOC sono presenti i file
paper.ps e paper.tex,
che contengono la documentazione completa
del pacchetto SPARSKIT
rispettivamente nel formato PS e TEX,
ed il file QUICK_REF
che contiene le quick reference di SPARSKIT.

Dove è possibile reperire SPARSKIT ?

http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html

Autore : Yousef Saad

1. SOFTWARE AND DOCUMENTATION

- Documentation: "SPARSKIT: A basic tool-kit for sparse matrix computations" Post-script
- General information and copyright terms . Read this first before copying the software.
  See also the Lesser GNU License terms.
- Compressed tar file of the whole package (includes all documentation).

Informazioni generali sul pacchetto e sui diritti di licenza

Pacchetto e Documentazione in formato TAR

Manuale in formato PS

Software e Documentazione

Una volta scaricato il file SPARSKIT2.tar.gz si utilizza il comando:

```
tar xzvf  SPARSKIT2.tar.gz
```

Per generare la cartella SPARSKIT2 e le sue sottocartelle contenenti il software e la sua documentazione.

Algoritmi per la memorizzazione di matrici sparse

1. passaggio dalla matrice A ai tre vettori R C J

```
J(1):=1;
k:=1;
for i=1,n do
    for j=1,n do
        if a(i,j)≠0 then
            R(k):= a(i,j);
            C(k):=j;
             k=k+1;
        endif
        endfor
    J(i+1):=k;
endfor
```

2. passaggio dai tre vettori R C J alla matrice A

```
n=dim(J)-1;
for i=1,n do
    for k:=J(i),J(i+1)-1 do
            a(i,C(k)):=R(k);
endfor
endfor
```

3. passaggio dai tre vettori R C J alla matrice A (bis)

```
n=dim(J)-1;
l:=0;
for i=1,n do
    diff:= J(i+1)- J(i)
for k:=1,diff do
    l:=l+1;
            a(i,C(l)):=R(l);
endfor
endfor
```

A. MURLI

FINE ESERCITAZIONE !!!