

---

# **SyLVER Documentation**

***Release v2019-01-31***

**Florent Lopez**

**Jan 07, 2019**



**CONTENTS:**

<b>1 Purpose</b>	<b>3</b>
<b>2 Derived types</b>	<b>5</b>
<b>3 Indices and tables</b>	<b>9</b>
<b>Index</b>	<b>11</b>



SyLVER



## PURPOSE

SyLVER is a sparse direct solver for computing the solution of **large sparse symmetrically-structured linear systems** of equations. This includes both positive-definite and indefinite sparse symmetric systems as well as unsymmetric system whose sparsity pattern is symmetric.

The solution of the system of equations:

$$AX = B$$

is achieved by computing a factorization of the input matrix. the following cases are covered:

1.  $A$  is **symmetric positive-definite**, we compute the **sparse Cholesky factorization**:

$$PAP^T = LL^T$$

where the factor  $L$  is a lower triangular matrix and the matrix  $P$  is a permutation matrix used to reduce the **fill-in** generated during the factorization. Following the matrix factorization the solution can be retrieved by successively solving the system  $LY = PB$  (forward substitution) and  $L^T PX = Y$  (backward substitutions).

2.  $A$  is **symmetric indefinite**, then we compute the sparse  $LDL^T$  decomposition:

$$A = PLD(PL)^T$$

where  $P$  is a permutation matrix,  $L$  is unit lower triangular, and  $D$  is block diagonal with blocks of size  $1 \times 1$  and  $2 \times 2$ .

3.  $A$  is **unsymmetric**, then we compute the sparse  $LU$  decomposition:

$$P_s A P_s^T = P_n L U Q_n$$

where  $P_s$  is a permutation matrix corresponding to the fill-reducing permutation whereas  $P_n$  and  $Q_n$  are meant to improve the numerical property of the factorization algorithm.  $L$  is lower triangular, and  $U$  is unit upper triangular.

The code optionally supports hybrid computation using one or more NVIDIA GPUs.





## DERIVED TYPES

### **type sylver\_options**

The derived data type `sylver_options` is used to specify the options used within `SYLVER`. The components, that are automatically given default values in the definition of the type, are:

#### **Type fields**

- % **print\_level** [*integer,default=0*] :: the level of printing. The different levels are:

< 0	No printing.
= 0	Error and warning messages only.
= 1	As 0, plus basic diagnostic printing.
> 1	As 1, plus some additional diagnostic printing.

- % **unit\_diagnostics** [*integer,default=6*] :: Fortran unit number for diagnostics printing. Printing is suppressed if <0.
- % **unit\_error** [*integer,default=6*] :: Fortran unit number for printing of error messages. Printing is suppressed if <0.
- % **unit\_warning** [*integer,default=6*] :: Fortran unit number for printing of warning messages. Printing is suppressed if <0.
- % **ordering** [*integer,default=1*] :: Ordering method to use in analyse phase:

0	User-supplied ordering is used ( <i>order</i> argument to <code>spldlt_analyse()</code> or <code>splu_analyse()</code> ).
1 (de- fault)	METIS ordering with default settings.
2	Matching-based elimination ordering is computed (the Hungarian algorithm is used to identify large off-diagonal entries. A restricted METIS ordering is then used that forces these on to the subdiagonal). <b>Note:</b> This option should only be chosen for indefinite systems. A scaling is also computed that may be used in <code>spldlt_factor()</code> or <code>splu_factor()</code> (see %scaling below).

- % **nemin** [*integer,default=32*] :: supernode amalgamation threshold. Two neighbours in the elimination tree are merged if they both involve fewer than `nemin` eliminations. The default is used if `nemin`<1.
- % **use\_gpu** [*logical,default=true*] :: Use an NVIDIA GPU if present.

- % **scaling** *[integer,default=0]* :: scaling algorithm to use:

<=0 (de- fault)	No scaling (if <code>scale(:)</code> is not present on call to <code>spldlt_factor()</code> or <code>splu_factor()</code> , or user-supplied scaling (if <code>scale(:)</code> is present).
=1	Compute using weighted bipartite matching via the Hungarian Algorithm (MC64 algorithm).
=2	Compute using a weighted bipartite matching via the Auction Algorithm (may be lower quality than that computed using the Hungarian Algorithm, but can be considerably faster).
=3	Use matching-based ordering generated during the analyse phase using <code>options%ordering=2</code> . The scaling will be the same as that generated with <code>options%scaling=1</code> if the matrix values have not changed. This option will generate an error if a matching-based ordering was not used during analysis.
>=4	Compute using the norm-equilibration algorithm of Ruiz.

- % **nb** *[integer,default=256]* :: Block size to use for parallelization of large nodes on CPU resources.
- % **pivot\_method** *[integer,default=1]* :: Pivot method to be used on CPU, one of:

0	Aggressive a posteori pivoting. Cholesky-like communication pattern is used, but a single failed pivot requires restart of node factorization and potential recalculation of all uneliminated entries.
1 (de- fault)	Block a posteori pivoting. A failed pivot only requires recalculation of entries within its own block column.
2	Threshold partial pivoting. Not parallel.

- % **small** *[real,default=1d-20]* :: threshold below which an entry is treated as equivalent to 0.0.
- % **u** *[real,default=0.01]* :: relative pivot threshold used in symmetric indefinite case. Values outside of the range  $[0, 0.5]$  are treated as the closest value in that range.

## type sylver\_inform

Used to return information about the progress and needs of the algorithm.

### Type fields

- % **cpu\_flops** *[integer]* :: number of flops performed on CPU
- % **cublas\_error** *[integer]* :: CUBLAS error code in the event of a CUBLAS error (0 otherwise).
- % **cuda\_error** *[integer]* :: CUDA error code in the event of a CUDA error (0 otherwise). Note that due to asynchronous execution, CUDA errors may not be reported by the call that caused them.
- % **flag** *[integer]* :: exit status of the algorithm (see table below).
- % **integer** (long) :: number of flops performed on GPU
- % **matrix\_dup** *[integer]* :: number of duplicate entries encountered (if `ssids_analyse()` called with `check=true`, or any call to `ssids_analyse_coord()`).

- % **matrix\_missing\_diag** *[integer]* :: number of diagonal entries without an explicit value (if `ssids_analyse()` called with `check=true`, or any call to `ssids_analyse_coord()`).
- % **matrix\_outrange** *[integer]* :: number of out-of-range entries encountered (if `ssids_analyse()` called with `check=true`, or any call to `ssids_analyse_coord()`).
- % **matrix\_rank** *[integer]* :: (estimated) rank (structural after analyse phase, numerical after factorize phase).
- % **maxdepth** *[integer]* :: maximum depth of the assembly tree.
- % **maxfront** *[integer]* :: maximum front size (without pivoting after analyse phase, with pivoting after factorize phase).
- % **num\_delay** *[integer]* :: number of delayed pivots. That is, the total number of fully-summed variables that were passed to the father node because of stability considerations. If a variable is passed further up the tree, it will be counted again.
- % **integer** :: number of entries in  $L$  (without pivoting after analyse phase, with pivoting after factorize phase).
- % **integer** :: number of floating-point operations for Cholesky factorization (indefinite needs slightly more). Without pivoting after analyse phase, with pivoting after factorize phase.
- % **num\_neg** *[integer]* :: number of negative eigenvalues of the matrix  $D$  after factorize phase.
- % **num\_sup** *[integer]* :: number of supernodes in assembly tree.
- % **num\_two** *[integer]* :: number of  $2 \times 2$  pivots used by the factorization (i.e. in the matrix  $D$ ).
- % **stat** *[integer]* :: Fortran allocation status parameter in event of allocation error (0 otherwise).

in- form%flag	Return status
0	Success.
-1	Error in sequence of calls (may be caused by failure of a preceding call).
-2	$n < 0$ or $ne < 1$ .
-3	Error in <code>ptr(:)</code> .
-4	CSC format: All variable indices in one or more columns are out-of-range. Coordinate format: All entries are out-of-range.
-5	Matrix is singular and <code>options%action=.false.</code>
-6	Matrix found not to be positive definite but <code>posdef=true</code> .
-7	<code>ptr(:)</code> and/or <code>row(:)</code> not present, but required as <code>ssids_analyse()</code> was called with <code>check=.false.,</code>
-8	<code>options%ordering</code> out of range, or <code>options%ordering=0</code> and order parameter not provided or not a valid permutation.
-9	<code>options%ordering=-2</code> but <code>val(:)</code> was not supplied.
-10	$ldx < n$ or $nrhs < 1$ .
-11	job is out-of-range.
-13	Called <code>ssids_enquire_posdef()</code> on indefinite factorization.
-14	Called <code>ssids_enquire_indef()</code> on positive-definite factorization.
-15	<code>options%scaling=3</code> but a matching-based ordering was not performed during analyse phase.
-50	Allocation error. If available, the stat parameter is returned in <code>inform%stat</code> .
-51	CUDA error. The CUDA error return value is returned in <code>inform%cuda_error</code> .
-52	CUBLAS error. The CUBLAS error return value is returned in <code>inform%cublas_error</code> .
+1	Out-of-range variable indices found and ignored in input data. <code>inform%matrix_outrange</code> is set to the number of such entries.
+2	Duplicate entries found and summed in input data. <code>inform%matrix_dup</code> is set to the number of such entries.
+3	Combination of +1 and +2.
+4	One or more diagonal entries of $A$ are missing.
+5	Combination of +4 and +1 or +2.
+6	Matrix is found be (structurally) singular during analyse phase. This will overwrite any of the above warning flags.
+7	Matrix is found to be singular during factorize phase.
+8	Matching-based scaling found as side-effect of matching-based ordering ignored (consider setting <code>options%scaling=3</code> ).

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## INDEX

### S

sylver\_inform (fortran type), [6](#)

sylver\_options (fortran type), [5](#)