SyLVER Documentation

Release v2019-01-31

Florent Lopez

Jan 07, 2019

CONTENTS:

1 Purpose **3**

2 Usage overview **5**

3 Basic Subroutines **7**
- 3.1 SpLDLT 7
- 3.2 SpLU 7

4 Derived types **9**

5 Indices and tables **13**

Index **15**

SyLVER

CHAPTER

ONE

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 achived 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 1×1 and 2×2.

3. A is **unsymmetric**, then we compute the sparse :math *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.

CHAPTER

TWO

USAGE OVERVIEW

Solving $AX = B$ using SyLVER is a four stage process.

- If A is *symmetric*:
  1. Call `spldlt_analyse()` to perform a symbolic factorization, stored in *spldlt_akeep*.
  2. Call `spldlt_factor()` to perform a numeric factorization, stored in *spldlt_fkeep*. More than one numeric factorization can refer to the same *spldlt_akeep*.
  3. Call `spldlt__solve()` to perform a solve with the factors. More than one solve can be performed with the same *spldlt_fkeep*.
  4. Once all desired solutions have been performed, free memory with `spldlt_free()`.
- If A is *unsymmetric*:
  1. Call `splu_analyse()` to perform a symbolic factorization, stored in *splu_akeep*.
  2. Call `splu_factor()` to perform a numeric factorization, stored in *splu_fkeep*. More than one numeric factorization can refer to the same *splu_akeep*.
  3. Call `splu_solve()` to perform a solve with the factors. More than one solve can be performed with the same *splu_fkeep*.
  4. Once all desired solutions have been performed, free memory with `splu_free()`.

CHAPTER

THREE

BASIC SUBROUTINES

In the below, all reals are double precision unless otherwise indicated.

3.1 SpLDLT

subroutine spldlt_analyse (*akeep*, *n*, *ptr*, *row*, *options*, *inform*, *ncpu*[, *order*, *val*])

Perform the analyse (symbolic) phase of the factorization for a matrix supplied in Compressed Sparse Column (CSC) format. The resulting symbolic factors stored in *spldlt_akeep* should be passed unaltered in the subsequent calls to ssids_factor().

Parameters

- **akeep** *[spldlt_akeep,out]* :: returns symbolic factorization, to be passed unchanged to subsequent routines.
- **n** *[integer,in]* :: number of columns in A.
- **integer** (long) ptr(n+1) *[in]* :: column pointers for A (see CSC format).
- **row** (ptr(n+1)-1) *[integer,in]* :: row indices for A (see CSC format).
- **options** *[sylver_options,in]* :: specifies algorithm options to be used (see `sylver_options`).
- **inform** *[sylver_inform,out]* :: returns information about the execution of the routine (see `sylver_inform`).
- **ncpu** *[integer,in]* :: Number of CPU available for the execution.

Options

- **order** (n) *[integer,inout]* :: on entry a user-supplied ordering (options%ordering=0). On return, the actual ordering used (if present).
- **val** (ptr(n+1)-1) *[real,in]* :: non-zero values for A (see CSC format). Only used if a matching-based ordering is requested.

3.2 SpLU

subroutine splu_analyse (*akeep*, *n*, *ptr*, *row*, *options*, *inform*, *ncpu*[, *order*, *val*])

Perform the analyse (symbolic) phase of the factorization for a matrix supplied in Compressed Sparse Column (CSC) format. The resulting symbolic factors stored in *splu_akeep* should be passed unaltered in the subsequent calls to ssids_factor().

Parameters

- **akeep** *[splu_akeep,out]* :: returns symbolic factorization, to be passed unchanged to subsequent routines.
- **n** *[integer,in]* :: number of columns in A.
- **integer** (long) ptr(n+1) *[in]* :: column pointers for A (see CSC format).
- **row** (ptr(n+1)-1) *[integer,in]* :: row indices for A (see CSC format).
- **options** *[*sylver_options*,in]* :: specifies algorithm options to be used (see `sylver_options`).
- **inform** *[*sylver_inform*,out]* :: returns information about the execution of the routine (see `sylver_inform`).
- **ncpu** *[integer,in]* :: Number of CPU available for the execution.

Options

- **order** (n) *[integer,inout]* :: on entry a user-supplied ordering (options%ordering=0). On return, the actual ordering used (if present).
- **val** (ptr(n+1)-1) *[real,in]* :: non-zero values for A (see CSC format). Only used if a matching-based ordering is requested.

CHAPTER

FOUR

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 (*order* argument to *spldlt_analyse()* or *splu_analyse()*).
1 (default)	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). **Note:** This option should only be chosen for indefinite systems. A scaling is also computed that may be used in `spldlt_factor()` or `splu_factor()` (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 (default)	No scaling (if `scale(:)` is not present on call to `spldlt_factor()` or `splu_factor()`, or user-supplied scaling (if `scale(:)` 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 options%ordering=2. The scaling will be the same as that generated with options%scaling= 1 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 (default)	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 (indefinte 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 2×2 pivots used by the factorization (i.e. in the matrix D).
- **% stat** *[integer]* :: Fortran allocation status parameter in event of allocation error (0 otherwise).

inform%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 ptr(:).
-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 options%action=.false.
-6	Matrix found not to be positive definite but posdef=true.
-7	ptr(:) and/or row(:) not present, but required as `ssids_analyse()` was called with check=.false,.
-8	options%ordering out of range, or options%ordering=0 and order parameter not provided or not a valid permutation.
-9	options%ordering=-2 but val(:) was not supplied.
-10	ldx<n or nrhs<1.
-11	job is out-of-range.
-13	Called `ssids_enquire_posdef()` on indefinite factorization.
-14	Called `ssids_enquire_indef()` on positive-definite factorization.
-15	options%scaling=3 but a matching-based ordering was not performed during analyse phase.
-50	Allocation error. If available, the stat parameter is returned in inform%stat.
-51	CUDA error. The CUDA error return value is returned in inform%cuda_error.
-52	CUBLAS error. The CUBLAS error return value is returned in inform%cublas_error.
+1	Out-of-range variable indices found and ignored in input data. inform%matrix_outrange is set to the number of such entries.
+2	Duplicate entries found and summed in input data. inform%matrix_dup 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 options%scaling=3).

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

S

spldlt_analyse() (fortran subroutine), **7**
splu_analyse() (fortran subroutine), **7**
sylver_inform (fortran type), **10**
sylver_options (fortran type), **9**