MSPAI 1.2 – Modified SParse Approximate Inverse Preconditioner

Thomas Huckle
Alexander Kallischko
Matous Sedlacek

2009 – 08 – 25

Technische Universität München
Forschungs- und Lehreinheit Informatik V
Scientific Computing in Computer Science

Contents

1 Abstract

Given a sparse matrix A the SPAI preconditioner computes a sparse approximate inverse matrix M by minimizing $\|AM - I\|_F$. The MSPAI algorithm is a generalization of SPAI, an extension of SPAI with target form to

$$\|CM - B\|_F. \tag{1}$$

This allows to compute explicit matrix approximations in either a factorized or unfactorized form. Furthermore, this enables to add some possibly dense rows to the underlying matrices, which are then also taken into account during the computation. These additional constraints for the Frobenius norm minimization generalize the idea of classical probing techniques, which are restricted to explicit approximations and very simple probing constraints. By a weighting factor, the resulting preconditioner can be forced to be optimal on certain probing subspaces represented by the additional rows.

The algorithm proceeds until $\|CM - B\|_2 < \epsilon$. By varying ϵ the user may control the quality and the cost of the preconditioner. A very sparse preconditioner is cheap to compute, but may not lead to significant improvements, whereas M becomes too expensive to compute if it becomes too dense.

The MSPAI preconditioner has various options to start several different algorithms such as a caching algorithm or QR updates for a faster computation of M.

2 MSPAI 1.2 Usage

2.1 Installation

Unfortunately, there is no common configure script and no common installation package at the moment. As soon as there will be any user-friendly installation process this will be changed immediately. There is a stand-alone serial as well as parallel version of the MSPAI 1.2 . For now, one has to decide which version to use, adapt the specific `Makefile` and just type `make` in the `src` directory to start compiling the sourcecode.

2.2 How to run MSPAI 1.2

- **Serial version:**

  ```
  ./mspai-1.2 <matrix.mtx> <pattern.mtx |-1| -2> <targetmatrix.mtx | -1> [options]
  ```

- **Parallel version:**

  ```
  mpirun -np <n> ./mspai-1.2 <matrix.mtx> <pattern.mtx |-1| -2> <targetmatrix.mtx
  | -1> [options]
  ```

The **mandatory parameters** are:

`matrix.mtx`	The system matrix in *Matrix Market* [5] format.
`pattern.mtx |-1| -2`	Path to start pattern file in *Matrix Market* format or `-1` for generating a diagonal start pattern automatically without any file or `-2` for generating a start pattern which prescribes the pattern of C itself for the preconditioner. No file required in these two cases.
`targetmatrix.mtx |-1`	Path to target matrix file in *Matrix Market* format or `-1` for generating an identity matrix automatically without file. This approximates the inverse C^{-1}.
`n`	Number of processors to use in MPI environment for parallel computation of M.

For getting help on shell use the `-h` option (e.g. `./mspai-1.2 -h`).

2.3 Options

Optional parameters can be used after the mandatory input in any order, always
`-optional_parameter <number|path>`.

Optional parameters are:

`-h(elp)`	**Parameter descriptions to shell** *Usage:* `./mspai-1.2 -h(elp)` *Default:* Don't display help Print the MSPAI usage and optional paramaters to shell.
`-ep`	**ϵ tolerance** *Default:* 0.4 *Choice:* `ep` ≥ 0.0 `ep` controls the quality of the approximation of M to the inverse of C. Higher values of `ep` lead to more computational work, more fill-in and usually to better preconditioners.

`-ns` **Max. number of improvement steps per column**
Default: 5
Choice: `ns` ≥ 0
Every column of the inverse C^{-1} is approximated in several improvement steps (pattern updates). The quality of the approximation is determined by ϵ. If the approximation could not reach the accuracy of ϵ after `ns` steps, MSPAI will use the best approximation so far.

`-mn` **Max. number of new nz candidates per step**
Default: 5
Choice: `mn` ≥ 0
For each improvement step there will be chosen `mn` new indices for the current pattern to compute a better approximation in the next step. If no new candidates can be found, the iteration aborts and the approximation will finish.

`-hs` **Hashtable size**
Default: 6
Choice: `hs` $\in \{0, 1, 2, 3, 4, 5, 6\}$
MSPAI uses a hashtable to cache remote messages and avoid redundant process communication. It is recommended to use larger hashtables to avoid the linear rehashing mechanism. `-hs 0` forces to switch the hashtable off – no communication will be cached locally.
`-hs 0` Don't use any hashtable.
`-hs 1` Hashtable has size 101.
`-hs 2` Hashtable has size 503.
`-hs 3` Hashtable has size 2503.
`-hs 4` Hashtable has size 12503.
`-hs 5` Hashtable has size 62501.
`-hs 6` Hashtable has size 104743.
This parameter is not available in serial version.

`-wp` **Write preconditioner to file**
Default: 1 (true)
Choice: `wp` $\in \{0, 1\}$
Whether to write the computed preconditioner into a file or not. The preconditioner will be written in *Matrix Market* format into the specified output file within the current working directory. Default output file is `precond.mtx`.

`-up` **Use upper pattern (maximum sparsity pattern)**
Default: Don't use upper pattern
Choice: `-up <path> | 1 | 2`

This way it is possible to predefine where the preconditioner will have its nz entries. Using this option in parallel MSPAI it is possible to prerequest all remote data once at the beginning. See `-pm` and `-pk`.
`-up 1` will generate a maximum identity pattern automatically without any file.
`-up 2` will generate a sparsity pattern for the preconditioner, which prescribes the pattern of C itself. No file required.

`-cs` **Cache size**
Default: 0 (don't use cache)
Choice: $0 \leq$ `cs` $\leq 1e7$
MSPAI can be invoked with a caching algorithm to cache QR decompositions and whole LS problems. Using the cache may reduce the computation time significantly when C is highly structured. It is recommended to use this option with a larger cache size (e.g. 500) if C is structured. For highly structured matrices smaller cache sizes (e.g. 50) may be sufficient. `-cs 0` will switch the caching algorithm off.

`-um` **Use mean value**
Default: 1 (true)
Choice: `um` $\in \{0, 1\}$
Whether to use a mean bound value for augmenting indices during an improvement step or not. During each improvement step new nz candidates will be computed for a better approximation. With this mean value the number of indices will be reduced to get only few candidates with best improvement. Using this recommended option will reduce the computation time.

`-qr` **Using QR levels**
Default: 0 (Don't use QR updates)
Choice: `qr` $\in \{0, 1, 2, 3, 4, 5\}$
It is possible to use a QR update algorithm for updating previously computed QR decompositions. This is much faster than computing a new decomposition for each LS problem. Optionally there are several QR levels to use:
`-qr 0` will switch the QR levels off. This will run a "normal" MSPAI mode.
`-qr 1` will cause an automatic switch between dense and sparse decompositions by means of the density of the submatrices $\hat{C}$. See option `-fg` for details.
`-qr 2` will cause dense QR updates using LAPACK [4].
`-qr 3` will cause sparse QR updates using CSparse [2] for sparse computation.
`-qr 4` will cause hybrid QR updates using both the CSparse and LAPACK library for updating the decompositions.
`-qr 5` will run without updates, but with sparse QR decompositions using the CSparse library. This option is useful for comparison without QR updates using the LAPACK library (`-qr 0`).

`-fg` **Density of submatrices $\hat{C}$**
Default: 0.3
Choice: $0.0 \leq$ `fg` ≤ 1.0
Using density for QR level option `-qr 1`. Due to the density of each submatrix $\hat{C}$ of a LS problem, it will be decided whether to use the LAPACK (density $\geq$ `fg`) or CSparse library (density $<$ `fg`) for the current decomposition.

`-pk` **Prerequesting columns**
Default: 0
Choice: `pk` ≥ 0
When using a maximum sparsity pattern (see `-up`), it is possible to prerequest `pk` columns additionally to a requested column. This option makes only sense when using the hashtable (`-hs`). The prerequested columns will be cached and are locally available for further computations. It is recommended to use small values (e.g. 5) when deciding for this option. Significant time improvements usually occur only when all remote columns are prerequested once at the beginning (`-pm 1`). This option is only usable in the parallel version.

`-pm` **Prerequesting columns only once at the beginning**
Default: 0 (false)
Choice: `pm` $\in \{0, 1\}$
When using a maximum sparsity pattern, it is possible to prerequest `pk` columns once at the beginning. This may speedup the calculation time because each process will prerequest all its columns at once at the beginning and may work on local data afterwards. No remote communication will be done after this single MPI traffic. When deciding for this option, it is recommended to set `pk` to the number of columns of the system matrix C. This option is only usable in the parallel version.

`-Ce` **Use probing vectors (probing matrix) for C**
Default: Don't use `-Ce`
Choice: `-Ce <path>`
If probing vectors are to be appended to the system matrix and thus generate the generalized form C, a file in *Matrix Market* format containing the **transposed** probing matrix has to be used. Ensure to have two probing files defined (`-Be` and `-Ce`), containing **transposed** probing matrices, and that in case of Schur probing the number of columns has to be smaller than of the system matrix. See parameter dependencies Section 2.4 for details.

`-Be` **Use probing vectors (probing matrix) for target matrix B**
Default: Don't use `-Be`
Choice: `-Be <path>`

If probing vectors are to be appended to the target matrix B and thus generate the generalized form B, a file in *Matrix Market* format containing the **transposed** probing matrix has to be used. Ensure to have two probing files defined (`-Ce` and `-Be`), containing **transposed** probing matrices, and that the number of columns is equivalent to that of `-Ce`. See parameter dependencies Section 2.4 for details.

`-rho` **Weight ρ for probing conditions**
Default: 1.0
Choice: `rho` ≥ 0.0
Weight for the probing conditions. ρ will be multiplied to each probing matrix component internally before computing the preconditioner.

`-schur` **Use Schur probing**
Default: 0 (false)
Choice: `schur` $\in \{0, 1\}$
Whether to use Schur probing or not. Ensure to use probing matrices `-Ce` and `-Be` when Schur probing requested. See parameter dependencies Section 2.4 for details.

`-ch` **Use hashtable**
Default: 0 (false)
Choice: `ch` $\in \{0, 1\}$
Whether to use a hashtable to cache QR decompositions and whole LS problems or not. Using the hashtable may reduce the computation time significantly when C is highly structured. Note that there is no upper bound for the size of the hashtable. This option is useful for comparison with a fixed cache size `-cs` and uses another hashtable than option `-hs`.

`-out` **Specify output file**
Default: `precond.mtx`
Choice: `out` be any string containing alphanumeric characters
Where to write the computed preconditioner. The preconditioner will be written to a file named by the specified output string in the current working directory.

Usage examples:

- Starting serial MSPAI to compute the preconditioner for the matrix *orsirr_2* [5]. Based on a diagonal start pattern the system matrix will be approximated inversely. With an ϵ tolerance of 10^{-3} there will be a maximum of 15 improvement steps, within each a maximum of 8 new candidates will be added to the current pattern without using a mean value bound. With switched off Caching and QR levels this is the unrestrained MSPAI algorithm without any runtime optimizations. A maximum sparsity pattern is used to predefine where entries are allowed within the preconditioner.

  ```
  ./mspai-1.2 /Matrices/orsirr_2.mtx -1 -1 -ep 0.001 -ns 15 -mn 8 -cs 0 -qr 0
  -up /Matrices/upperpattern.mtx -um 0
  ```

- Starting parallel MSPAI on three processors to compute the preconditioner for the matrix *orsirr_2*. Start pattern is the sparsity pattern of the system matrix. With an ϵ tolerance of 10^{-2} and a maximum of 12 improvment steps, within each a maximum of 5 new candidates using the mean value bound will be added to the current pattern, the system matrix will be approximated inversely. Using dense QR updates will significantly improve the computation time.

  ```
  mpirun -np 3 opt01 opt02 opt03 ./mspai-1.2 /Matrices/orsirr_2.mtx -2 -1 -ep
  0.01 -ns 12 -mn 5 -cs 0 -qr 2
  ```

- Starting parallel MSPAI on five processors to compute the preconditioner for the matrix *orsirr_1* [5]. Based on a specific start pattern from file, the system matrix will be approximated inversely. In a static MSPAI without any improvment steps Schur probing is requested. There are two probing matrices passed, one for C (`-Ce`) and one for B (`-Be`). The caching algorithm is requested with cache size 80.

  ```
  mpirun -np 5 opt01 opt02 opt03 opt04 opt05 ./mspai-1.2
  /Matrices/orsirr_1.mtx /Matrices/startpattern.mtx -1 -ns 0 -cs 80 -qr 0
  -Ce /Matrices/probingmatrixCe.mtx -Be /Matrices/probingmatrixBe.mtx -schur
  1
  ```

2.4 Parameter dependencies

Several parameters are only usable when others are or are not used as well. In case of incorrect use, an error message will be printed to shell. The following subsection should give a short survey. Parameters which are not listed do not have any dependencies and can be used arbitrary.

`targetmatrix.mtx`	**Only usable with:** `-ns 0`, `-qr 0` The target and system matrix must be square and must have the same number of columns, if probing is not used.
`-up`	**Only usable with:** `-schur 0` Maximum sparsity pattern and start pattern must be square and must have the same number of columns.
`-cs`	**Only usable with:** `-qr 0`, `-ch 0`
`-qr`	**Only usable with:** `-cs 0`, `-ch 0`, `-ns` ≥ 1, `-schur 0`, target matrix option `-1` **Not usable with:** `-Ce`, `-Be`
`-pk`	**Only usable with:** `-up <path> | 1 | 2`, `-schur 0`
`-pm`	**Only usable with:** `-up <path> | 1 | 2`, `-schur 0`
`-Ce`	**Only usable with:** `-Be`, `-ns 0`, `-qr 0` The dimension of the probing matrices must be the same. If Schur probing is not used, the number of columns of each probing matrix must be equal to that of C.
`-Be`	**Only usable with:** `-Ce`, `-ns 0`, `-qr 0`, The dimension of the probing matrices must be the same. If Schur probing is not used, the number of columns of each probing matrix must be equal to that of C.
`-schur`	**Only usable with:** `-Ce` and `-Be`, target matrix option `-1` In case of Schur probing the probing matrix `-Ce` must have less columns than the system matrix C. The zero block will be filled automatically in front of `-Ce`. Furthermore a valid start pattern file must be passed. The options `-1` and `-2` for start patterns are not usable with Schur probing. The start pattern must have as many rows as the system matrix and as many columns as `-Ce`. The probing matrix `-Be` must have the same dimensions as `-Ce`. The identity and zero block for the target matrix will be generated automatically.
`-ch`	**Only usable with:** `-qr 0`, `-cs 0`

3 FAQ

3.1 What does MSPAI stand for?

MSPAI stands for Modified SParse Approximate Inverse. The name Modified SPAI is an extension to the original name SPAI. "The name was invented some time in the Spring of 1994 during a typical lunch outside on the beautiful lawn of the Main Quad at Stanford University: an improvised "dejeuner sur l'herbe" amongst Rodin sculptures below sunny Californian skies. Present were T. Huckle (TU-Munich), M. Grote and A.-J. van der Veen (TU-Delft)." [3]

3.2 Why does MSPAI taking forever to compute the preconditioner?

Make sure your ϵ value is not too small. You could start with a larger value (e.g. 0.3) and reduce it progressively until total execution time starts to increase again.

3.3 Does MSPAI always work?

In principle, yes. Unlike many other preconditioners, MSPAI cannot break down if the matrix C is non-singular. Moreover, if one keeps reducing ϵ, MSPAI will eventually compute the exact inverse C^{-1}. This may take a very long time. Of course there are several restrictions in using the MSPAI options. See Section 2.4 for details.

3.4 When I reduce ϵ the preconditioner does not really improve, why?

Try to increase the number of improvement steps `-ns` and/or the number of indices to be added during one step `-mn`.

3.5 Caching does not improve the computation time, why?

It is advised to use the caching algorithm only when the system matrix C is structured (band, block, etc...). Try to make the cache larger (e.g. 500) when C is supposed to have many different submatrices $\hat{C}$. Otherwise a small cache of size 50 should be sufficient.

3.6 Using probing always complains about dimension mismatch, why?

See Section 2.4 for details. Probing is only supported in static MSPAI yet (`-ns 0`). When using explicit or inverse probing, the dimensions of the probing matrices `-Ce` and `-Be` must have the same number of columns as C. In case of using Schur probing (`-schur 1`), these matrices must have less number of columns than C. Furthermore, a start pattern file has to be passed as well.

3.7 Which dimension do the system matrices may have?

There is no restriction to the number of columns and rows B and C must have. MSPAI 1.2 was tested with lots of matrices with different dimensions and densities. The largest matrices had approximately 10^6 columns and rows. Note that the system matrices must be square.

3.8 Which input matrices MSPAI was tested with?

Besides matrices (CFD, Laplace) generated on our own, MSPAI was tested for various matrices from Matrix Market [5] and from the University of Florida Sparse Matrix Collection [7].

4 Links

4.1 Contacts

- `Thomas Huckle`
  ```
  Department of Computer Science - Chair V
  Technische Universität München
  Boltzmannstr.  3
  85748 Garching bei München
  Germany
  ```
  ```
  ☎    +49-(0)89/289 18 609
  FAX  +49-(0)89/289 18 607
  ✉    huckle@informatik.tu-muenchen.de
  www  http://www5.in.tum.de/wiki/index.php/Univ.-Prof._Dr._Thomas_Huckle
  ```

- Matous Sedlacek
  ```
  Department of Computer Science - Chair V
  Technische Universität München
  Boltzmannstr.  3
  85748 Garching bei München
  Germany
  ```
  ```
  ☎    +49-(0)89/289 18 613
  FAX  +49-(0)89/289 18 607
  ✉    sedlacek@in.tum.de
  www  http://www5.in.tum.de/wiki/index.php/Matous_Sedlacek
  ```

4.2 Used tools

The MSPAI 1.2 implementation uses:

- **BLAS**: Basic Linear Algebra subroutines [1].
- **LAPACK**: Linear Algebra Package [4].
- **CSparse**: Concise Sparse Matrix package [2].
- **MPI**: Message Passing Interface for the parallel version [6].

4.3 References

References

[1] ATLAS - Automatically tuned linear algebra software. `http://math-atlas.sourceforge.net/`.

[2] CSparse - Concise Sparse Package. `http://www.cise.ufl.edu/research/sparse/CSparse/`.

[3] M. Grote and O. Broeker, *SPAI – SParse Approximate Inverse Preconditioner*, Spaidoc.pdf paper in the SPAI 3.2 package, 2005.

[4] LAPACK - Linear Algebra PACKage. `http://www.netlib.org/lapack/index.html`.

[5] Matrix Market. `http://math.nist.gov/MatrixMarket/`.

[6] MPI - Message Passing Interface. `http://www-unix.mcs.anl.gov/mpi/`.

[7] The University of Florida Sparse Matrix Collection. `http://www.cise.ufl.edu/research/sparse/matrices/`.