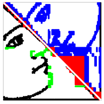
Technische Universität Braunschweig

Institute for Computational Mathematics



JANUS BLOCK ILU Guide

http://bilu.tu-bs.de

Matthias Bollhöfer, May 28, 2021

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Preconditioning Systems

Objective

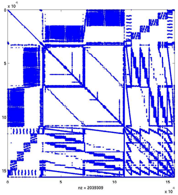
Given a large sparse nonsingular matrix A and a linear system

$$Ax = b,$$

1. construct approximate factorization $A \approx \tilde{A} = LDU$
2. solve $Ax = b$ using a preconditioned Krylov subspace iteration method

How large, how sparse, and why using an approximate factorization?

- system size $n = 10^5 \to 10^9$, number of nonzeros typically less $100n$
- memory requirements, computation time



Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Getting started with C

- downlowad JANUS at `https://github.com/mbollhoefer/JANUS`
- before you can use JANUS BLOCK ILU, you are asked to download and install external packages
  - Install the approximate mininum degree (AMD) [Amestoy, Davis and Duff 1996] ordering from the SuiteSparse and create a library `libamd.a`
  - Install the MT-METIS package [LaSalle and Karypis 2013]
  - Install MC64 from the HSL Mathematical Software Library, e.g. adding the object files to `libjanus.a` or somewhere else
  - make sure to have efficient versions of BLAS and LAPACK installed, e.g., the MKL library would be helpful but it is not mandatory
- After that you are ready to use the JANUS library. You will find in the subdirectory `janus/samples` a `makefile` along with a sample C-code `janusdriver.c` which explains the usage of the JANUS BLOCK ILU preconditioning package.

Getting started with C — a sample code

```
#include <janus.h>
...
// major JANUS variables
SparseMatrix A;
JanusOptions options;
JanusPrec    PREC;
...
// tag your matrix, e.g.,  as real, nonsymmetric
A.isreal=1;A.isdefinite=0;A.issymmetric=0;A.ishermitian=0;
JanusDefaultOptions(&options); // use default options
// compute BLOCK ILU
ierr=JanusFactor(&A, &PREC, options);
// iterative solver
ierr=JanusSolver(&A,&PREC,rhs,sol,30,1e-6,1000,&iter);
JanusDelete(&PREC); // release preconditioner
```

Technische Universität Braunschweig

Parameter setting

JANUS offers several parameters

Some Default Parameters

```
options
 .matching:                  1 | 0
 .ordering:                  PERM_MTMETIS | PERM_NONE|PERM_AMD|PERM_RCM
 .droptol:                   1.0e-03 | ...
 .cosine:                    0 | 1
 .blocking_strategy:         BLOCK_ILUPT | BLOCK_NONE|BLOCK_SUPERNODES
 .progressive_aggregation:   1 | 0
 .symmetric_structure:       0 | 1
 .invert_blocks:             1 | 0
```

Parameter Setting

Most of the parameters you will find familiar

`matching`	improve diagonal dominance using maximum weighted matchings
`ordering`	preprocess the system by a symbolic reordering (e.g. PERM_AMD for 'Approximate Minimum Degree')
`droptol`	threshold to drop small entries during the factorization
`cosine`	cosine-based strategy to build blocks in the initial matrix
`blocking_strategy`	heuristic approach to improve the given scalar or block structure preparing the BLOCK ILU
`progressive_aggregation`	aggregate blocks during the factorization to obtain even larger blocks
`symmetric_structure`	only use symmetric permutations
`invert_blocks`	compute $LD^{-1}U$ or LDU

How to set up your own parameters

- Your iterative solver does not converge
  $\Rightarrow$ reduce `options.droptol`

How to set up your own parameters

- Your iterative solver does not converge
  ⇒ reduce `options.droptol`

- You prefer to use JANUS inside your own Krylov subspace method
  ⇒ use `JanusSol(&PREC, rhs,sol, buff,m)`, `JanusSolT(&PREC, rhs,sol, buff,m)`, `JanusSolH(&PREC, rhs,sol, buff,m)` instead for solving `m` right hand sides with JANUS, its transpose or its conjugate transpose. `buff` could either be set to `NULL` or you provide `n*m` spaces.

How to set up your own parameters

- Your iterative solver does not converge
  ⇒ reduce `options.droptol`

- You prefer to use JANUS inside your own Krylov subspace method
  ⇒ use `JanusSol(&PREC, rhs,sol, buff,m)`, `JanusSolT(&PREC, rhs,sol, buff,m)`, `JanusSolH(&PREC, rhs,sol, buff,m)` instead for solving `m` right hand sides with JANUS, its transpose or its conjugate transpose. `buff` could either be set to `NULL` or you provide `n*m` spaces.

- You are interested in some statistics concerning JANUS
  ⇒ `JanusNnz(&PREC, &nnz, &mxblock, &avblock, &stddev)` tells you about the nonzeros (`nnz`), the size of the largest diagonal block (`mxblock`), the average block size (`avblock`) and their standard deviation (`stddev`)

How to set up your own parameters

- Your iterative solver does not converge
  $\Rightarrow$ reduce `options.droptol`

- You prefer to use JANUS inside your own Krylov subspace method
  $\Rightarrow$ use `JanusSol(&PREC, rhs,sol, buff,m)`, `JanusSolT(&PREC, rhs,sol, buff,m)`, `JanusSolH(&PREC, rhs,sol, buff,m)` instead for solving `m` right hand sides with JANUS, its transpose or its conjugate transpose. `buff` could either be set to `NULL` or you provide `n*m` spaces.

- You are interested in some statistics concerning JANUS
  $\Rightarrow$ `JanusNnz(&PREC, &nnz, &mxblock, &avblock, &stddev)` tells you about the nonzeros (`nnz`), the size of the largest diagonal block (`mxblock`), the average block size (`avblock`) and their standard deviation (`stddev`)

- You would like to handle complex–valued or symmetric matrices
  $\Rightarrow$ pass your data and tag ther matrix suitably, (e.g. `A.isreal=0` or `A.issymmetric=1`)

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Getting started with MATLAB

After adding *JANUS* system path (e.g. $\gg$ `addpath 'janus'`) a large sparse system $Ax = b$ could be solved as follows:

Approximate Factorization

```
>> [PREC,options]=janus(A);
```

Getting started with MATLAB

After adding *JANUS* system path (e.g. $\gg$ `addpath 'janus'`) a large sparse system Ax = b could be solved as follows:

Approximate Factorization

$\gg$ `[PREC,options]=janus(A);`

$\Rightarrow$ preconditioner is built using the default options.

Getting started with MATLAB

After adding *JANUS* system path (e.g. $\gg$ `addpath 'janus'`) a large sparse system Ax = b could be solved as follows:

Approximate Factorization

```
>> [PREC,options]=janus(A);
```

$\Rightarrow$ preconditioner is built using the default options.

Iterative Solution

```
>> x=janussolver(A,b,30,1e-6,1000,PREC);
```

Getting started with MATLAB

After adding *JANUS* system path (e.g. $\gg$ `addpath 'janus'`) a large sparse system $Ax = b$ could be solved as follows:

Approximate Factorization

```
>> [PREC,options]=janus(A);
```

$\Rightarrow$ preconditioner is built using the default options.

Iterative Solution

```
>> x=janussolver(A,b,30,1e-6,1000,PREC);
```

$\Rightarrow$ system is solved.

Parameter setting

JANUS offers several parameters

Some Default Parameters

```
≫ [PREC,options]=janus(A); options
options =
                 matching:1
                 ordering:'mtmetis'
                  droptol:1.0000e-03
             perturbation:1
       symmetric_structure:0
                   cosine:0
        blocking_strategy:'ilupt'
 progressive_aggregation:1
            invert_blocks:1
```

Parameter Setting

`isdefinite`	indicate that your matrix is positive definite (only used if the matrix is Hermitian)
`matching`	improve diagonal dominance using maximum weighted matchings
`ordering`	symbolic reordering (e.g. 'amd' for 'Approximate Minimum Degree')
`droptol`	threshold to drop small entries
`cosine`	cosine-based strategy to build blocks
`blocking_strategy`	heuristic approach to improve the given scalar or block structure preparing the BLOCK ILU (e.g. 'ilupt' or 'supernodes')
`progressive_aggregation`	aggregate blocks during the factorization to obtain even larger blocks
`symmetric_structure`	only use symmetric permutations
`invert_blocks`	compute $LD^{-1}U$ or LDU

How to set up your own parameters

- Your iterative solver does not converge
  $\Rightarrow$ reduce `options.droptol`, call `[PREC,options]=janus(A,options);`

How to set up your own parameters

- Your iterative solver does not converge
  ⇒ reduce `options.droptol`, call `[PREC,options]=janus(A,options);`

- You prefer to use JANUS inside your own Krylov subspace method
  ⇒ use `sol=janussol(PREC,rhs);` or `sol=janussolt(PREC,rhs);` or `sol=janussolh(PREC,rhs)` instead for solving with JANUS, its transpose or its conjugate transpose.

How to set up your own parameters

- Your iterative solver does not converge
  $\Rightarrow$ reduce `options.droptol`, call `[PREC,options]=janus(A,options);`

- You prefer to use JANUS inside your own Krylov subspace method
  $\Rightarrow$ use `sol=janussol(PREC,rhs);` or `sol=janussolt(PREC,rhs);` or `sol=janussolh(PREC,rhs)` instead for solving with JANUS, its transpose or its conjugate transpose.

- You are interested in some statistics concerning JANUS
  $\Rightarrow$ `nz=janusnnz(PREC)` tells you about the nonzeros. Other information can be directly accessed from `PREC`.

How to set up your own parameters

- Your iterative solver does not converge
  $\Rightarrow$ reduce `options.droptol`, call `[PREC,options]=janus(A,options);`

- You prefer to use JANUS inside your own Krylov subspace method
  $\Rightarrow$ use `sol=janussol(PREC,rhs);` or `sol=janussolt(PREC,rhs);` or `sol=janussolh(PREC,rhs)` instead for solving with JANUS, its transpose or its conjugate transpose.

- You are interested in some statistics concerning JANUS
  $\Rightarrow$ `nz=janusnnz(PREC)` tells you about the nonzeros. Other information can be directly accessed from `PREC`.

- You would like to handle complex–valued or symmetric matrices
  $\Rightarrow$ `janus` will select the correct driver on its own!

How to set up your own parameters

- Your iterative solver does not converge
  $\Rightarrow$ reduce `options.droptol`, call `[PREC,options]=janus(A,options);`

- You prefer to use JANUS inside your own Krylov subspace method
  $\Rightarrow$ use `sol=janussol(PREC,rhs);` or `sol=janussolt(PREC,rhs);` or `sol=janussolh(PREC,rhs)` instead for solving with JANUS, its transpose or its conjugate transpose.

- You are interested in some statistics concerning JANUS
  $\Rightarrow$ `nz=janusnnz(PREC)` tells you about the nonzeros. Other information can be directly accessed from `PREC`.

- You would like to handle complex–valued or symmetric matrices
  $\Rightarrow$ `janus` will select the correct driver on its own!

- You would like to display the BLOCK ILU
  $\Rightarrow$ `janusspy(PREC)`

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

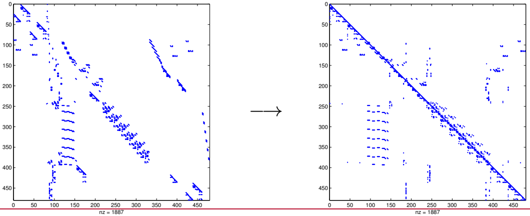
Matchings

- Maximum weighted matchings are a combinatorial graph theoretical approach to improve diagonal dominance

- Using matchings a general system A is
  1. rescaled
  2. permuted

  such that $A \rightarrow \Pi^T D_r A D_c$ satisfies $|a_{ij}^{new}| \leqslant 1$, $|a_{ii}^{new}| = 1$.

- Matchings can be symmetrized for systems satisfying $|A| = |A^T|$ with similar properties

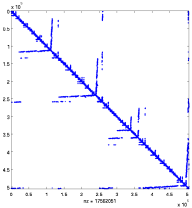


Outline

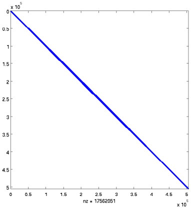
- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Symmetric reorderings

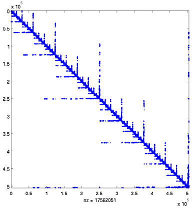
`'amd'` Approximate Minimum Degree



`'rcm'` Reverse Cuthill-McKee



`'mtmetis'` multithreaded Metis

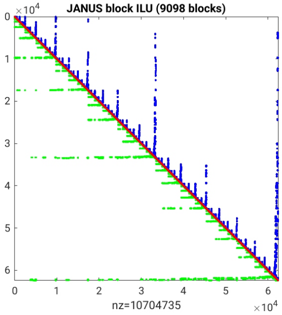


Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

BLOCK ILU

- Factorize $A \approx LDU$ or $LD^{-1}U$ using several blocking strategies to compute a BLOCK ILU with blocks of bigger size
- block structures allow to exploit dense matrix kernels



Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Automatic structure detection

JANUS offers many special purpose drivers for

- complex systems
  - general sparse (with GMRES)
  - complex symmetric (with SQMR)
  - complex Hermitian (with SQMR)
  - complex Hermitian positive definite (with CG)
- real systems
  - general sparse (with GMRES)
  - real symmetric (with SQMR)
  - real symmetric positive definite (with CG)

JANUS automatically detects

- real/complex systems
- symmetry structures

Automatic structure detection

JANUS asks YOU to specify if the system is positive definite

SPD Case — C-Code

```
A.isdefinite=1;
ierr=JanusFactor(&A,&PREC,options);
```

SPD Case — MATLAB-Code

```
>> options.isdefinite=1;
>> [PREC,options]=janus(A,options);
```

if no other options are set, the default options are used

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Additional features

- JANUS will always compute the main branch of $\log(\det A)$ (`PREC.logdet`)
- JANUS offers drivers for approximate matrix inversion via Neumann series based on the given symmetric block LDL^T (`DSYMbfspai`,`ZHERbfspai`,`ZSYMbfspai` for C code, `symbfspai` for the MATLAB interface)
- JANUS offers drivers for selected matrix inversion series based on the given symmetric block LDL^T (`DSYMselbinv`,`ZHERselbinv`,`ZSYMselbinv` for C code, `symselbinv` for the MATLAB interface)

Outline

- **Introduction — Using JANUS BLOCK ILU**
  - Preconditioning Systems
  - Getting started with C
  - Getting started with MATLAB
- **What's behind the toolbox**
  - Matchings
  - Symmetric reorderings
  - BLOCK ILU
- **Automatic structure detection**
- **Additional features**
- **Closing Remarks**

Closing Remarks

Watch the *JANUS* website at

`http://bilu.tu-bs.de`

You may download the sources at

`https://github.com/mbollhoefer/JANUS`

Matthias Bollhöfer, Olaf Schenk, Fabio Verbosio. A High Performance Level-Block Approximate LU Factorization Preconditioner Algorithm. *Applied Numerical Mathematics* 162:265-282, 2021. DOI:10.1016/j.apnum.2020.12.023