MIN3P-THCm User Manual for Parallel Configuration – Draft

Danyang Su[a], Mingliang Xie[a], K. Ulrich Mayer[a], Kerry T.B. MacQuarrie[b]

[a]Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada

[b]Department of Civil Engineering, University of New Brunswick, P.O. Box 4400, Fredericton, NB, E3B 5A3, Canada

December 2018

Contents

This manual is aimed at users with basic parallel computing knowledge. It focuses on using the parallel configuration for ParMIN3P-THCm code, a parallelized version of theMIN3P-THCm code (Mayer 1999, Mayer et al., 2002). ParMIN3P-THCm supports desktop PCs, workstations and supercomputers, with consideration of Windows, Linux/Unix and OS X operating systems. Please note that installation of the required library is system- and compiler-dependent. Installation of the required library is beyond the scope of this manual. Users should refer to the system and compiler they use for the detailed library installation.

In this manual, a detail parallel configuration is provided for users to configure the parameters needed to run the parallel code. The configuration focuses on the parallel solver, parallel matrix assembly, and parallel performance analysis and testing. If this file does not exist or is not specified in the input file when running the parallelized code, the default configuration will be used instead.

Three parallelized MIN3P-THCm versions are available, including the shared-memory OpenMP version, the distributed-memory MPI version, and the hybrid MPI-OpenMP version. The OpenMP version uses threading technology for acceleration while the MPI parallel version uses the domain decomposition method, building on the PETSc (Balay et al., 2014a, 2014b) library. The OpenMP parallel version is suitable for small-scale acceleration and is easy to use while the MPI and hybrid MPI-OpenMP version is for large-scale acceleration and is much more challenging to use.

1 SYSTEM REQUIREMENTS

Redistributable libraries are needed for the OS without developing an environment. Currently, for Windows OS, the program is compiled with the Intel Visual Fortran (IVF) compiler. For Linux/Unix OS, the program is compiled and tested with both IVF and GFortran. Before running the parallel version of MIN3P-THCm, please install the IVF redistributables library if the code is compiled using IVF or a corresponding library.

1.1 SYSTEM REQUIREMENTS FOR WINDOWS

OpenMP parallel version

For a 32-bit OS, please install Intel(R) Visual Fortran Redistributables on IA-32 if the IVF compiler is used. For a 64-bit OS, please install Intel(R) Visual Fortran Redistributables on 64 if the IVF compiler is used.

The redistributable libraries corresponding to the developing environment can be download via: http://software.intel.com/en-us/articles/redistributable-libraries-for-intel-c-and-visual-fortran-composer-xe-2013-for-windows.

MPI parallel version

The MPI parallel version is compiled together with the PETSc library. For instructions on how to install and configure PETSc on your local machine, please visit http://www.mcs.anl.gov/petsc/.

Hybrid MPI-OpenMP parallel version

Please include both OpenMP and OpenMP system requirements, as abovementioned.

1.2 SYSTEM REQUIREMENTS FOR LINUX/UNIX/OS X

Installation of OpenMP, MPI and PETSc libraries is system-dependent. Please refer to the library installation guide for the system you are using.

2 PARALLEL CONFIGURATION

There are three ways to set the parallel input parameters: 1) use the default solver configuration; 2) use a specified parallel configuration file, and 3) use parallel configuration commands. Please note that all the input parameters from parallel configuration commands are available from the parallel configuration file. However, not all parallel input parameters from the configuration file are available from the commands.

2.1 DEFAULT PARALLEL SOLVER CONFIGURATION

When the solver configuration file or command is not specified, the parallel MIN3P-THCm version uses default solver configuration.

Default configuration for OpenMP version

Solver Type

The default solver type for OpenMP parallel version is WatSolv, which is partially parallelized.

Convergence parameters

Default WatSolv solver convergence parameters are read from the input file prefix.dat.

Number of threads

Default number of threads are read from system environment variable

(OMP_NUM_THREADS).

For Windows version, this value can be set by

set OMP_NUM_THREADS[=num]

and for Linux/Unix version, this value can be set by

export OMP_NUM_THREADS[=num]

Default configuration for MPI version

Solver Type

The default solver type for MPI parallel version is PETSc linear solver.

Convergence parameters

Default PETSc solver convergence parameters:

```
KSPType          = "kspgmres"
PCType           = "pcbjacobi"
pc_factor_shift = "none"
rtol             = 1.0E-5
abstol           = 1.0E-30
dtol             = 1.0E5
maxits           = 50000
```

Number of processors

Default number of processors are set from mpirun/mpiexec or other similar command to run MPI job.

Default configuration for hybrid MPI-OpenMP version

Same as default configuration for MPI version except that every processor has OMP_NUM_THREADS threads.

2.2 PARALLEL CONFIGURATION FROM FILE

Configuration file entry

To use a parallel configuration file, please add the following command and data in the "Data Block 1: global control parameters" block in the MIN3P input file.

```
'parallel solver configuration file'
'solver.cfg'
```

where `'solver.cfg'` is the path of the configuration file. A relative file path is supported.

Configuration file format

The entire configuration is saved in an ASCII text file (e.g., parallel.cfg). The line beginning with an exclamation mark (!) is a comment line. The characters are NOT case-sensitive. Lines beginning with an alphabetic character are command lines or a data lines. Lines beginning with a numerical character are data lines. Not all the command lines or data lines are required. The latter command will overwrite the former command if they have the same target.

Global solver setting

Global solver setting includes the solver type for flow and reactive transport, as well as the number of threads for global use.

There are three solvers in the parallel version of MIN3P-THCm. The first solver, WatSolv (or WS209), is the default solver of MIN3P-THCm for the sequential version. WatSolv is partially parallelized. The second solver, PARDISO, is a shared-memory directly solver. The third solver is a solver package included in PETSc.

Global command and data

Global MPI commands

```
GLOBAL: USE MIN3P INPUT PARAMETERS FIRST
```

This command indicates that the input parameters in the MIN3P input file have priority over the input parameters in the parallel solver configuration file. By default, the parameters in this file will overwrite the same parameters in the MIN3P input file.
For example, if there are solver convergence parameters in both files, you can use the command if you want to use MIN3P input parameters first.

```
GLOBAL: USE NUMBER OF THREADS FROM MPI
```

If this command is enabled, the number of threads in the OpenMP calling will be replaced by the number of processors in the MPI calling.

Global OpenMP commands

Set the number of threads for global use, including the number of threads for matrix assembly, parallel solver, as well as other parallel regions.

```
GLOBAL: NUMBER OF THREADS
```
n

Set the number of threads *n* for global use. The number of threads for global use includes matrix assembly, matrix solver and other parallel regions. If this command is not specified, the default number of threads for global use is 1.

```
GLOBAL: NUMBER OF LOOPS THRESHOLD
```
n

Set the loop threshold for parallel running. Loops like vector initialization will be parallelized if the loop number is larger than this threshold.

Global solver type

Set the solver type for flow and reactive transport equations. The solver type is given by a command line and a data line.

```
SOLVER TYPE
```
i

Set the solver type for both flow equations and reactive transport equations. If *i* is 0, use WatSolv as the solver, if *i* is 1, use PARDISO as the solver. If *i* is 2, use the solver from the PETSc package. For the OpenMP version, if this command is not specified, the default solver, WatSolv, will be used. For MPI and hybrid MPI-OpenMP version, the default solver from PETSc will be used if the command is not specified.

```
SOLVER TYPE FLOW
```
i

Set the solver type for flow equations only. If *i* is 0, use WatSolv as the solver. If *i* is 1, use PARDISO as the solver. If *i* is 2, use the solver from the PETSc package. For the OpenMP version, if this command is not specified, the default solver WatSolv will be used. For MPI and thee hybrid MPI-OpenMP version, the default solver from PETSc will be used if the command is not specified.

```
SOLVER TYPE REACTIVE TRANSPORT
```
i

Set the solver type for reactive transport equations only. If *i* is 0, use WatSolv as the solver; however, if *i* is 1, use PARDISO as the solver. If *i* is 2, use the solver from the PETSc package. For the OpenMP version, if this command is not specified, the default solver, WatSolv, will be used. For MPI and the hybrid MPI-OpenMP version, the default solver from PETSc will be used if the command is not specified.

Sample configuration

```
!> ***********************************************************
!>             Block:  Golobal solver setting
!> ***********************************************************
!SOLVER TYPE
!1

SOLVER TYPE FLOW
0

SOLVER TYPE REACTIVE TRANSPORT
1

GLOBAL: NUMBER OF THREADS
4
```

Domain decomposition setting

Global domain decomposition for MPI parallel version and hybrid MPI-OpenMP parallel version. The parameter user needs to specify is the stencil width. By default, the stencil width is 1, which can meet the requirements for most of the cases, but for the case with "vanleer" spatial weighting, as it will consider the second upstream point for flux limiter, the stencil width should be 2 or more. Domain decomposition is based on PETSc DMDA for the structured grid.

Sample configuration

```
!> ***********************************************************
!>             Block:  Domain decomposition setting
!> ***********************************************************
PETSC: STENCIL WIDTH
2
```

Matrix assembly setting

This section is only used for the OpenMP version or the hybrid MPI-OpenMP version. The matrix assembly setting includes matrix assembly type, number of threads used for matrix assembly and chunk size factor. The matrix assembly setting is separated for flow and reactive transport. If this part is not specified, and the number of threads for global use is 1, then the system will run in sequential mode for matrix assembly.

Matrix assembly type

Set the matrix assembly type for flow and reactive transport equations. The matrix assembly type is given by a command line and a data line.

```
matrix assembly: type in flow
i
```

Set the matrix assembly type for flow equations only. If *i* is 0, use the sequential mode (non-parallel) for the matrix assembly. If *i* is 1, use OpenMP parallel mode for the matrix assembly. If this command is not specified, sequential mode is the default.

```
matrix assembly: type in reactive transport
i
```

Number of threads

Set the matrix assembly type for reactive transport equations only. If *i* is 0, use sequential mode (non-parallel) for the matrix assembly. If *i* is 1, use the OpenMP parallel mode for the matrix assembly. If this command is not specified, sequential mode is the default.

```
matrix assembly: number of threads in flow
n
```

Set the number of threads *n* in the matrix assembly for flow equations only. If not specified, use ‘global: number of threads’ instead.

```
matrix assembly: number of threads in reactive transport
n
```

Set the number of threads *n* in the matrix assembly for reactive transport equations only. If not specified, use ‘global: number of threads’ instead.

Schedule type

matrix assembly: schedule type in flow
i

Set the schedule type in the matrix assembly for flow equations only. If *i* is 0, use the static schedule method. If *i* is 1, use the dynamic schedule method. The default value is 0. Currently, schedule type is pre-defined in the compiling option. DO NOT use the setting here.

```
matrix assembly: schedule type in reactive transport
i
```

Set the schedule type in the matrix assembly for reactive transport equations only. If *i* is 0, use the static schedule method. If *i* is 1, use the dynamic schedule method. The default value is 0. Currently, schedule type is pre-defined in the compiling option. DO NOT use the setting here.

Chunk size in parallel loop

```
matrix assembly: chunk size factor in flow
n
```

Set the chunk size factor for the matrix assembly for flow equations only. This value should be from 0 to '(loop count)/(number of processors)'. If the value is not specified or 0, use the system default value. The default value depends on the schedule type.

```
matrix assembly: chunk size factor in reactive transport
n
```

Set the chunk size factor for the matrix assembly for reactive equations only. This value should be from 0 to '(loop count)/(number of processors)'. If the value is not specified or 0, use the system default value. The default value depends on the schedule type.

Sample configuration

```
!> **********************************************************
!>              Block:  Matrix assembly setting
!> **********************************************************


MATRIX ASSEMBLY: TYPE IN FLOW
1

MATRIX ASSEMBLY: NUMBER OF THREADS IN FLOW
4

MATRIX ASSEMBLY: SCHEDULE TYPE IN FLOW
0

MATRIX ASSEMBLY: CHUNK SIZE FACTOR IN FLOW
0


MATRIX ASSEMBLY: TYPE IN REACTIVE TRANSPORT
1

MATRIX ASSEMBLY: NUMBER OF THREADS IN REACTIVE TRANSPORT
4

MATRIX ASSEMBLY: SCHEDULE TYPE IN REACTIVE TRANSPORT
0

MATRIX ASSEMBLY: CHUNK SIZE FACTOR IN REACTIVE TRANSPORT
0
```

WatSolv solver setting

For the present version, WatSolv (WS209) is not fully parallelized. Only part of the loops in the solver is parallelized. The only parameter that can be controlled is the number of threads. It is only valid for the OpenMP parallel version.

Number of Threads

```
ws209: number of threads
n
```

Set the number of threads *n* for WatSolv (WS209). This is valid only if the solver type is WatSolv. If not specified, use the default value 1 as the number of threads.

Sample configuration

```
!> ****************************************************
!>             block:  ws209 solver setting
!> ****************************************************
ws209: number of threads
4
```

PARDISO solver setting

This section is only used for the OpenMP version when the PARDISO solver is used. PARDISO is a thread-safe, high-performance, robust, memory-efficient and easy-to-use software for solving large sparse symmetric and asymmetric linear systems of equations on shared memory. The following settings are valid only if the solver type is PARDISO.

Solver test

```
pardiso: solver test with ws209
```

Set this command if you want to check the result of the matrix solver with WatSolv (WS209). This command is valid only if the solver type is WatSolv. Do not include this command in normal use. This is only used for testing and will significantly deteriorate performance.

Number of Threads

```
pardiso: number of threads
n
```

Set the number of threads *n* for the PARDISO solver. If the number of threads is 0, use the number of threads determined by PARDISO, which is usually the number of physical cores.

Solver refinement

```
pardiso: max iterative refinement steps in flow
n

pardiso: max iterative refinement steps in reactive transport
n
```

Set the maximum number of iterative refinement steps *n* that the solver will perform for flow equations and reactive transport equations, respectively. The solver will perform no more than the absolute value of this parameter for iterative refinement and will stop the process if a satisfactory level of accuracy of the solution in terms of backward error is achieved. If this parameter is negative, the accumulation of the residuum is using extended precision real types. The default value is 9.

Pivoting perturbation

```
pardiso: pivoting perturbation in flow
n

pardiso: pivoting perturbation in reactive transport
n
```

This parameter controls how to handle small pivots or zero pivots for asymmetric matrices for flow equations and reactive transport equations, respectively. It indicates the iterative refinement contraction rate. The default value is 13, which means eps = 10^{-13} is used in handling small pivots.

Preconditioning

```
pardiso: cgs criterion in flow
n

pardiso: cgs criterion in reactive transport
n
```

This parameter controls preconditioned CGS [sonn89] for asymmetric matrices for flow equations and reactive transport equations, respectively. The parameter *n* has the form 10*l+k.If k=0; the factorization is always computed as required by the phase. If k=1, CGS iteration replaces the LU computation. The preconditioner is LU, which was computed in a previous step (the first step or last step with a failure) in a sequence of solutions needed for identical sparsity patterns. For example, if n is 31, the solver will use LU-preconditioned CGS iteration with a stopping criterion of 1.0e^{-3} for asymmetric matrices. If n is 61, the solver will use LU-preconditioned CGS iteration with a stopping criterion of 1.0^{-6} for asymmetric matrices. The default value is 0.

Maximum solver iteration

```
pardiso: maximum solver iteration in flow
n

pardiso: maximum solver iteration in reactive transport
n
```

This parameter controls the maximum solver iteration *n* for flow equations and reactive transport equations, to see if symbolic factorization should be refreshed before the next iteration. In the PARDISO solver, when the previous reordering is not good enough to get the correct results, reorder the matrix (symbolic factorization). For asymmetric cases, it's better to call the reorder step for each matrix, but this can result in wasted time. When a preconditioned CGS is used, this value will be compared to the number of completed iterations. Otherwise, this value will be compared to the number of iterative refinement steps performed. When the iteration number is larger than the specified value, symbolic factorization should be redone. The default value is 9.

Maximum residual

```
pardiso: maximum residual in flow
e

pardiso: maximum residual in reactive transport
e
```

This parameter controls the maximum residual *e* for flow equations and reactive transport equations, to see if symbolic factorization should be refreshed before the next iteration. In the PARDISO solver, when the previous reordering is not good enough to get the correct results, the matrix (symbolic factorization) should be reordered. For asymmetric cases, it's better to call the reorder step for each matrix, but this can result in lost time. When the maximum residual is larger than the specified value, do symbolic factorization again. The default value is 1.0^{-5}.

Symbolic factorization type

```
pardiso: symbolic factorization type in flow
i

pardiso: symbolic factorization type in reactive transport
```

i

This parameter controls the symbolic factorization type *i* for flow equations and reactive transport equations. If the symbolic factorization type is 0, do symbolic factorization only once before the iteration. If the symbolic factorization type is 1, do symbolic factorization at every newton iteration step. When the symbolic factorization type is 0, the "matrix solver iteration" and "maximum residual" parameters are valid.

Tips on the PARDISO solver

PARDISO is a direct solver. If the system is well-conditioned, the speed of PARDISO is usually not as fast as that of an iterative solver. If the system is not well-conditioned, it may converge better than the iterative solver. For an ill-conditioned matrix, it may fail. You can also output the condition number to see if the matrix is ill-conditioned.

Sample configuration

```
!> *****************************************************
!>             block:  pardiso solver setting
!> *****************************************************

! pardiso: solver test with ws209

pardiso: number of threads
0

pardiso: max iterative refinement steps in flow
9

pardiso: max iterative refinement steps in reactive transport
9

pardiso: pivoting perturbation in flow
13

pardiso: pivoting perturbation in reactive transport
13

pardiso: cgs criterion in flow
0

pardiso: cgs criterion in reactive transport
0

pardiso: maximum solver iteration in flow
5
pardiso: maximum solver iteration in reactive transport
5

pardiso: maximum residual in flow
1.0e-8
pardiso: maximum residual in reactive transport
1.0e-8

pardiso: symbolic factorization type in flow
```

```
0

pardiso: symbolic factorization type in reactive transport
0
```

PETSc solver setting

This section is only used when the MPI version or hybrid MPI-OpenMP version is used. PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modelled by partial differential equations.

Please note the following setting is based on the PETSc 3.5.x version. The code can support PETSc 3.4.x to 3.6.x version. It may support the most updated version, later than 3.6.x, but this is not guaranteed.

Solver test

```
petsc: solver test with ws209
```

Set this command if you want to check the result of the matrix solver with WatSolv (WS209). This command is valid only if the solver type is WatSolv. Do not include this command in normal use. This is only used for testing and will significantly deteriorate performance.

Default configuration

```
petsc: use default configuration in flow
```

Set this command to use the default configuration for the flow problem by PETSc. If you want to use the default solver configuration or if you want to read the configuration from the command line, remove this command being a comment. Otherwise, comment this:

```
petsc: use default configuration in reactive transport
```

Set this command to use the default configuration for the reactive transport problem by PETSc. If you want to use the default solver configuration or if you want to read the configuration from the command line, remove this command from being a comment. Otherwise, comment this.

PETSc Krylov methods

```
petsc: ksp type in flow
kspgmres                !or gmres or kspbcgs or bcgs
```

```
petsc: ksp type in reactive transport
kspgmres                !or gmres or kspbcgs or bcgs
```

Set the PETSc Krylov method for the flow problem and the reactive transport problem, respectively. The following methods are supported in PETSc: “gmres” and “bcgs”. Please note that this command can add the prefix ksp.

PETSc preconditioner type

```
petsc: preconditioner type in flow
pcbjacobi                  !same as bjacobi

petsc: preconditioner type in reactive transport
pcbjacobi                  !same as bjacobi
```

Set the PETSc preconditioner type for the flow problem and the reactive transport problem. The following methods are supported in PETSc: “none”, “jacobi”, “lu”, “bjacobi”, “ilu”, “asm” and “ksp”. Please note that some of the preconditioning methods are not parallelized. Look at the PETSc manual for more detail.

PETSc norm type

```
petsc: ksp norm type in flow
ksp_norm_preconditioned

petsc: ksp norm type in reactive transport
ksp_norm_preconditioned
```

Set the norm type for the flow and reactive transport problem. The following norm type is supported:

- KSP_NORM_NONE - skips computing the norm; this should only be used if you are using the Krylov method as a smoother with a fixed small number of iterations. It implicitly sets KSPSkipConverged as the KSP convergence test and is supported only by CG, Richardson, Bi-CG-stab, CR, and CGS methods.
- KSP_NORM_PRECONDITIONED - the default for left preconditioned solves; it uses the l2 norm of the preconditioned residual
- KSP_NORM_UNPRECONDITIONED - uses the l2 norm of the true b - Ax residual, supported only by CG, CHEBYSHEV, and RICHARDSON and is automatically true for right (see KSPSetPCSide()) preconditioning.
- KSP_NORM_NATURAL - supported by KSPCG, KSPCR, KSPCGNE, KSPCGS

PETSc KSP solver convergence criteria

```
petsc: ksp convergence criteria type in flow
kspdefault

petsc: ksp convergence criteria type in reactive transport
kspuserdefined
```

Set the KSP solver convergence criteria for flow and reactive transport, respectively. By default, the KSP solver will reach convergence when rnorm < MAX (rtol * rnorm_0, abstol) or divergence if rnorm > dtol * rnorm_0. The rnorm here is the preconditioned residual norm. If the '-ksp_norm_type unpreconditioned' is used, rnorm is the true residual norm. If the solver convergence criteria is set to user-defined criteria, then the true residual will be calculated every iteration. The solver will reach convergence when rnorm < MAX (rtol * rnorm_0, abstol).

PETSc KSP solver convergence tolerance

The following commands only take effect when the KSP solver convergence criteria are set to user-defined.

```
petsc: relative convergence tolerance in flow
1.0e-5

petsc: relative convergence tolerance in reactive transport
1.0e-6
```

Set the relative convergence tolerance for flow and reactive transport problems, respectively. The default value is 1.0^{-5}.

```
petsc: absolute convergence tolerance in flow
1.0e-16

petsc: absolute convergence tolerance in reactive transport
1.0e-20
```

Set the absolute convergence tolerance for flow and reactive transport problems, respectively. The default value is 1.0^{-20}.

```
petsc: divergence tolerance in flow
1.0e5

petsc: divergence tolerance in reactive transport
1.0e5
```

Set the divergence tolerance for flow and reactive transport problems, respectively. The default value is 1.0^{5}.

```
petsc: maximum number of iterations in flow
```

```
200

petsc: maximum number of iterations in reactive transport
500
```

Set the maximum number of iterations for flow and reactive transport problems, respectively. The default value is 1000.

Sample configuration

```
!> *********************************************************
!>             block:  petsc solver setting
!> *********************************************************
!petsc: solver test with ws209

petsc: stencil width
2

petsc: use default configuration in flow

petsc: ksp type in flow
kspgmres

petsc: preconditioner type in flow
pcbjacobi

!petsc: ksp norm type in flow
!ksp_norm_preconditioned

petsc: ksp convergence criteria type in flow
kspdefault

petsc: relative convergence tolerance in flow
1.0e-5

petsc: absolute convergence tolerance in flow
1.0e-20

petsc: divergence tolerance in flow
1.0e5

petsc: maximum number of iterations in flow
100

petsc: use default configuration in reactive transport

petsc: ksp type in reactive transport
kspgmres

petsc: preconditioner type in reactive transport
pcbjacobi

!petsc: ksp norm type in reactive transport
!ksp_norm_preconditioned

petsc: ksp convergence criteria type in reactive transport
userdefined
```

```
petsc: relative convergence tolerance in reactive transport
1.0e-5

petsc: absolute convergence tolerance in reactive transport
1.0e-20

petsc: divergence tolerance in reactive transport
1.0e5

petsc: maximum number of iterations in reactive transport
100
```

LIS solver setting

LIS is a **L**ibrary of **I**terative **S**olvers for linear systems. It is a scalable parallel software library for solving linear equations and eigenvalue problems that arise in the numerical solution of partial differential equations using iterative methods (www.ssisc.org/lis/). All LIS solver related setting should go to block: lis solver setting in the solver configuration file. Please note that LIS for MPI parallel and hybrid MPI-OpenMP version uses domain decomposition from PETSc. It should be built with PETSc. The LIS version used for the current MIN3P-THCm version is 1.7.36.

Solver test

```
lis: solver test with ws209
```

Set this command if you want to check the results of linear equations with results solved by WatSolv (WS209). This command is valid only if the solver type is WatSolv. Do not include this command in the normal use. This is only used for testing and debugging. The command will significant deteriorate the performance.

Default configuration

```
lis: use default configuration in flow

lis: use default configuration in reactive transport
```

To use default LIS solver configuration, please specify the above two commands for flow problem and reactive transport problem, respectively.

LIS Krylov subspace methods

```
lis: ksp type in flow
bicg                     !or gmres, bicgstab
```

```
lis: ksp type in reactive transport
bicg                     !or gmres, bicgstab
```

Set Krylov subspace method for linear solver for flow and reactive transport problem.

LIS preconditioner type

```
lis: preconditioner type in flow
ilu

lis: preconditioner type in reactive transport
ilu
```

```
Set LIS preconditioner type for flow problem and reactive transport
problem, respectively. The following methods are supported in this file:
"none", "jacobi" and "ilu"
```

LIS solver precision

```
lis: solver precision in flow
double

lis: solver precision in reactive transport
quad
```

```
Set LIS solver precision for flow problem and reactive transport problem,
respectively. It support both double precision and quad precision.
Double precision operations sometimes require large number of iterations
because of the rounding error. Quadruple precision operations can
improve this operation. Both matrix and vectors are still double
precision.
```

LIS solver convergence parameters

```
lis: convergence tolerance in flow
1.0e-10

lis: maximum number of iterations in flow
500

lis: convergence tolerance in reactive transport
1.0e-10

lis: maximum number of iterations in reactive transport
1000
```

Set the relative convergence tolerance and maximum number of iterations for flow problem and reactive transport problem, respectively.

LIS solver other parameters

```
lis: solver options in flow
```

```
-ilu_fill 2 -print mem

lis: solver options in reactive transport
-ilu_fill 1
```

Set other options for lis solver. The above mentioned parameters can also be specified in this part. For the detail commands, please look into LIS user manual.

Sample configuration

```
!> *********************************************************
!>              Block: LIS solver setting
!> *********************************************************
lis: solver test with ws209

lis: use default configuration in flow

lis: ksp type in flow
bicg

lis: preconditioner type in flow
none

lis: solver precision in flow
double

lis: convergence tolerance in flow
1.0e-12

lis: maximum number of iterations in flow
1000

!lis: solver options in flow
!-i cg -p ssor -print mem

lis: use default configuration in reactive transport

lis: ksp type in reactive transport
bicg

lis: preconditioner type in reactive transport
none

lis: solver precision in reactive transport
double

lis: convergence tolerance in reactive transport
1.0e-12

lis: maximum number of iterations in reactive transport
1000

!lis: solver options in reactive transport
!-i cg -p ssor -print mem
```

Output setting

Runtime statistics analysis

```
output runtime statistics analysis
```

This command outputs the runtime profiling, including runtime of the matrix assembly, factorization and substitution in each Newton iteration step, as well as the runtime of flow and reactive transport in each time step. The output file is in Tecplot data format.

Matrix data set and rhs

```
output sparse matrix data set and rhs
```

This command outputs the matrix data set for all the linear equations. This is a time-consuming process. Enabling this command will significantly increase the runtime. Use this only in testing, for example, when iteration fails.

Condition number

```
output condition number
```

This command outputs the estimated condition number of the matrix before solving the equations. Enabling this command will significantly increase the runtime. Use this only in testing, for example, when iteration fails.

Sample configuration

```
!> *********************************************************
!>             block:  output setting
!> *********************************************************

output runtime statistics analysis

output sparse matrix data set and rhs

output condition number
```

OpenMP Parallel Control for Debugging

This section is only used for code testing and debugging. Add/comment the following commands to enable/disable the parallel routine using OpenMP for the specified section. Please use this together with the source code.

```
!> *********************************************************
!>             Block:  OpenMP Parallel Controls (Optional)
!> *********************************************************
!> Format
!> Command for specified subroutine
!> Threshold for loop amount
!>
!> If the threshold is smaller than the loop amount, OpenMP
!> parallelization is enabled for this subroutine, otherwise,
!> if the threshold is larger than the loop amount, OpenMP
!> parallelization is disabled for this subroutine.
!> By default, the threshold is 1, that is, all the OpenMP parallelization
!> is enabled.
!> *********************************************************
mbalrt: number of threads 1
1
mbalrt: number of threads 2
1
mbalrt: number of threads 3
1
mbalrt: number of threads 4
1
mbalrt: number of threads 5
1
mbalrt: number of threads 6
1
mbalrt: number of threads 7
1
mbalrt: number of threads 8
1
mbalrt: number of threads 9
1
mbalrt: number of threads 10
1
mbalrt: number of threads 11
1
msysrt: number of threads 1
1
msysrt: number of threads 2
1
msysrt: number of threads 3
1
msysrt: number of threads 4
1
msysrt: number of threads 5
1
msysrt: number of threads 6
1
mbal_mcd: number of threads 1
1
mbal_mcd: number of threads 2
1
mbal_mcd: number of threads 3
1
infcrtdd: number of threads 1
1
```

```
infcrtdd: number of threads 2
1
infcrtdd: number of threads 3
1
infcrt_a: number of threads 1
1
infcrt_a: number of threads 2
1
infcrt_a: number of threads 3
1
infcrt_g: number of threads 1
1
infcrt_g: number of threads 2
1
infcrt_mcd: number of threads 1
1
infcrt_mcd: number of threads 2
1
diffcoff_mcd: number of threads 1
1
i2upfind: number of threads 1
1
i2upfind_heat: number of threads 1
1
ddtds: number of threads 1
1
ddtds_energybal: number of threads 1
1
ddtds_energybal: number of threads 2
1
comp_bc_ice: number of threads 1
1
comp_bc_ice: number of threads 2
1
timeloop: number of threads 1
1
infheat_c: number of threads 1
1
infheat_d: number of threads 1
1
infevap: number of threads 1
1
updatedd: number of threads 1
1
updatedd: number of threads 2
1
updatedd_ener: number of threads 1
1
updatedd_ener: number of threads 2
1
updatedd_ener: number of threads 3
1
ddvsflow: number of threads 1
1
seepfdd: number of threads 1
1
tstepvs: number of threads 1
1
updatevs: number of threads 1
```

```
1
updatevs: number of threads 2
2
seepface: number of threads 1
1
soilparm: number of threads 1
1
msysdd: number of threads 1
1
msysdd: number of threads 2
1
msysdd: number of threads 3
1
msysvs: number of threads 1
1
mbalvs: number of threads 1
1
mbalvs: number of threads 2
1
mbalvs: number of threads 3
1
energysys: number of threads 1
1
energy_bal: number of threads 1
1
energy_bal: number of threads 2
1
velodd: number of threads 1
1
nexttime: number of threads 1
1
nexttime: number of threads 2
1
infcvs: number of threads 1
1
xyzcoord: number of threads 1
1
cvolume: number of threads 1
1
iajavs: number of threads 1
1
iajavs: number of threads 2
1
iajavs_dp: number of threads 1
1
iajavs_ener: number of threads 1
1
iajavs_ener: number of threads 2
1
matrix_uti: number of threads 1
1
iajart: number of threads 1
1
iajart: number of threads 2
1
initpppm: number of threads 1
1
initppdd: number of threads 1
1
```

```
initppvs: number of threads 1
1
initppvs: number of threads 2
1
initppeb: number of threads 1
1
initppeb: number of threads 2
1
initicvs: number of threads 1
1
initsatw: number of threads 1
1
initsatw: number of threads 2
1
initsatw: number of threads 3
1
initicener: number of threads 1
1
initprob: number of threads 1
1
initprob: number of threads 2
1
initicrt: number of threads 1
1
initicdd: number of threads 1
1
restart_r: number of threads 1
1
batreac: number of threads 1
1
batreac: number of threads 2
1
```

Example of running the code

OpenMP version

nprocs min3p-thcm/file/path input/file/name

MPI version

To run the parallel using nprocs processors, the following command is used to start the code:

mpiexec –n *nprocs min3p-thcm/file/path input/file/name*

Hybrid MPI-OpenMP version

To run the parallel using nprocs processors and *m* threads, the following command is used to start the code. Note: number of threads is set in the configuration file.

mpiexec –n *nprocs min3p-thcm/file/path input/file/name*

2.3 PARALLEL CONFIGURATION USING OMMAND LINES

Global command setting

```
-solver_configuration_file file/path
```

Set the solver configuration file to file/path.

```
-input_file input/name
```

Set the input file name.

```
-solver_type_flow i
```

Set the flow solver type to i (0, 1, 2).

```
-solver_type_react i
```

Set the reactive transport solver type to i (0, 1, 2).

```
-use_numofthreads_from_mpi
```

The global number of threads will be reset by mpi calling.

```
-numofthreads_global n
```

Set the global number of threads to n.

```
-numofloops_thred_global n
```

Set the global number of loop threshold for OpenMP parallelization. A loop count less than this threshold will not be parallelized.

Command lines for the OpenMP version

Command lines for the OpenMP version for flow problems

```
-matrix_assembly_type_flow i
```

Set matrix assembly type i in flow for OpenMP parallelization.

```
-numofthreads_matrix_flow n
```

Set the number of threads for matrix assembly in flow to n.

```
-schedule_type_flow i
```

Set the schedule type of the matrix assembly in flow to i.

```
-chunksize_factor_flow n
```

Set the chunk size factor for the matrix assembly in the flow problem to n.

Command lines for the OpenMP version for reactive transport problems

```
-matrix_assembly_type_react i
```

Set the matrix assembly type i in the reactive transport for OpenMP parallelization.

```
-numofthreads_matrix_react n
```

Set the number of threads for the matrix assembly in reactive transport to n.

```
-schedule_type_react i
```

Set the schedule type of the matrix assembly in reactive transport to i.

```
-chunksize_factor_react n
```

Set the chunk size factor for the matrix assembly in the reactive transport problem to n.

Command lines for the PARDISO solver setting

Global setting of the PARDISO solver

```
-solver_test_pardiso
```

Activate solver testing of PARDISO vs Watsolv.

```
-numofthreads_pardiso n
```

Set the number of threads for PARDISO to n.

Command lines for flow problems

```
-max_refine_itersteps_flow n
```

Set the maximum iterative refinement steps in flow to n.

`-npivotpertubation_flow n`

Set the pivoting perturbation in flow to n.

`-cgs_criterion_flow i`

Set the preconditioned CGS in flow to i.

`-max_iteration_flow n`

Set the maximum solver iteration in flow to n.

`-max_residual_flow r`
Set the maximum residual in flow to value r.

`-symfactor_type_flow i`

Set the symbolic factorization type in flow to i.

Command lines for reactive transport problems

`-max_refine_itersteps_react n`

Set the maximum iterative refinement steps in reactive transport to n.

`-npivotpertubation_react n`

Set the pivoting perturbation in reactive transport to n.

`-cgs_criterion_react i`

Set the preconditioned CGS in reactive transport to i.

`-max_iteration_react n`

Set maximum solver iteration in reactive transport to n.

```
-max_residual_react r
Set maximum residual in reactive transport to value r.
```

`-symfactor_type_react i`

Set the symbolic factorization type in reactive transport to i.

Command lines for the PETSc solver setting

Global setting of the PETSc solver

```
-solver_test_petsc
```

Set the PETSc solver test.

```
-stencil_width n
```

Set the stencil width to n.

```
-threadcomm_nthreads n
```

Set the number of threads per processor to n for the hybrid version when run with threadcomm in PETSc.

```
-mpicomm_nthreads_nthreads n
```

Set the number of threads per processors to n for the hybrid version when run without threadcomm in PETSc.

Command lines for flow problems

```
-use_petsc_default_flow
```

PETSc uses the default configuration for the flow problem.

```
-ksptype_flow strtype
```

Set the PETSc Krylov method for the flow problem to the specified value strtype.

```
-pctype_flow strtype
```

Set the PETSc preconditioner type for the flow problem to the specified value strtype.

```
-kspconvergencetype_flow strtype
```

Set the PETSc convergence criteria for the flow problem to type strtype.

```
-rtol_flow r
```

Set the relative convergence tolerance for the flow problem to r.

```
-abstol_flow                                                          r
```

Set the absolute convergence tolerance for the flow problem to r.

```
-dtol_flow r
```

Set the divergence-convergence tolerance for the flow problem to r.

```
-maxits_flow n
```

Set the maximum number of iterations for the flow problem to n.

Command lines for reactive transport problems

```
-use_petsc_default_react
```

PETSc uses the default configuration for the reactive transport problem.

```
-ksptype_ react strtype
```

Set the PETSc Krylov method for the reactive transport problem to the specified value strtype.

```
-pctype_react strtype
```

Set the PETSc preconditioner type for the reactive transport problem to the specified value strtype.

```
-kspconvergencetype_react strtype
```

Set the PETSc convergence criteria for the reactive transport problem to type strtype.

```
-rtol_react r
```

Set the relative convergence tolerance for the reactive transport problem to r.

```
-abstol_react                                                        r
```

Set the absolute convergence tolerance for the reactive transport problem to r.

```
-dtol_react r
```

Set the divergence-convergence tolerance for the reactive transport problem to r.

```
-maxits_react n
```

Set the maximum number of iterations for the reactive transport problem to n.

Command line for the Wstsolv (WS209) solver

```
-numofthreads_ws209 n
```

Set the number of threads for the WS209 solver to n.

Command lines for the output setting

```
-output_runtime
```

Output the runtime statistics analysis file.

```
-output_matrix
```

Output the sparse matrix data set and rhs file at every time step.

```
-output_matrix_timestep i
```

Output the sparse matrix data set and rhs file at time step i.

```
-type_matrix_format i
```

Set the output matrix format type to i.

```
-output_condition_number
```
Output the condition number.

Example of running the code

Parallel configuration from the command line has the same effect as that from the configuration file. For example, the following command in the configuration file and the command line has the same effect.

From the configuration file

```
petsc: stencil width
2

petsc: use default configuration in flow

petsc: ksp type in flow
gmres
```

From the command line

```
-stencil_width 2 -use_petsc_default_flow -ksptype_flow gmres
```

The running method is the same as that of the configuration file, except that the parameters are from the command line and value.

OpenMP version

nprocs min3p-thcm/file/path input/file/name

MPI version

To run the parallel using nprocs processors, the following command is used to start the code.

mpiexec –n *nprocs* min3p-thcm/file/path *input/file/name* -stencil_width *2* -use_petsc_default_flow -use_petsc_default_react

Hybrid MPI-OpenMP version

To run the parallel using nprocs processors and *m* threads, the following command is used to start the code. Note: the number of threads is set in the configuration file.

mpiexec –n *nprocs min3p-thcm/file/path input/file/name* -stencil_width *2* -use_petsc_default_flow -use_petsc_default_react -mpicomm_nthreads *m*

2.4 PARALLEL CONFIGURATION FROM PETSC OPTIONS

This part only works for PETSc parallel version. It inherits configuration from PETSc options, with prefix "flow_" for flow solver configuration and "react_" for reactive transport solver configuration. Besides solver options, other configuration is the same as PETSc default options.

To use PETSc options, please DO NOT specify solver configuration file in the input file.

Example of running the code

- Run with all default options from PETSc:

mpiexec –n *nprocs* min3p-thcm/file/path *input/file/name*

- Run with different PC type for flow and reactive transport with log summary:

mpiexec –n *nprocs* min3p-thcm/file/path *input/file/name* –flow_pc_type *ilu* –react_pc_type *bjacobi* –log_summary *log/file/name*

- Run with different PC type and different PC factor shift type for flow and reactive transprot:

mpiexec –n *nprocs* min3p-thcm/file/path *input/file/name* –flow_pc_type *ilu* –react_pc_type *bjacobi* *–flow_sub_pc_factor_shift_type* *nonzero* –*react_sub_pc_factor_shift_type positive_definite* –log_summary *log/file/name*

2.5 PROCESSOR AND THREAD BINDING

Processor and thread binding have a big effect on parallel performance. For either MPI, OpenMP or hybrid MPI-OpenMP, be sure to bind processor and thread to physical cores, not logical cores using hyperthreading. For most supercomputers, hyperthreading is turned off while for personal PC or workstation, by default, hyperthreading is turned on. Binding processor or thread is system dependent. Some of the job schedule system (e.g., slurm) on supercomputers are configured with this. Here are some example for processor and thread binding on personal PC or workstation using Linux/Unix system.

OpenMP

export GOMP_CPU_AFFINITY=”   ”
./program input

Binds threads to specific CPUs. The variable should contain a space-separated or comma-separated list of CPUs. This list may contain different kinds of entries: either single CPU numbers in any order, a range of CPUs (M-N) or a range with some stride (M-N:S). CPU numbers are zero based. For example, GOMP_CPU_AFFINITY="0 3 1-2 4-15:2" will bind the initial thread to CPU 0, the second to CPU 3, the third to CPU 1, the fourth to CPU 2, the fifth to CPU 4, the sixth through tenth to CPUs 6, 8, 10, 12, and 14 respectively and then start assigning back from the beginning of the list. GOMP_CPU_AFFINITY=0 binds all threads to CPU 0.

MPI

mpirun –np 4 –bind-to core ./min3p_thcm_omp input
Launch four processes and bind to 4 physical cores

mpirun -bind-to user:0,1,2,3 –np 4 *./min3p_thcm_mpi input*
Launch four processes and bind to physical cores from 0 to 3.

mpirun -bind-to user:0+1+2+3,4+5+6+7 -np 2 ./min3p_thcm_hybrid patchf
Launch two processes with four OpenMP threads each. Bind to physical cores from 0 to 7.

REFERENCES

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K. and Smith, B. F. and Zhang, H., 2014a. PETSc Web page. http://www.mcs.anl.gov/petsc.

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K. and Smith, B. F. and Zhang, H., 2014b. PETSc Users Manual. http://www.mcs.anl.gov/petsc.

Mayer, K.U., 1999. A numerical model for multicomponent reactive transport in variably saturated porous media. PhD Thesis. University of Waterloo.

Mayer, K.U., Frind, E.O., Blowes, D.W., 2002. Multicomponent reactive transport modeling in variably saturated porous media using a generalized formulation for kinetically controlled reactions. Water Resour. Res. 38, 1174, doi: 10:1029/2001WR000862.