FEMT Manual

CIMAT

Contents

Change log.......1
Introduction.......1
The solvers.......2
The tools.......3
MatSolver.......3
Environment variables.......3
Common variables.......4
Variables for iterative solvers.......4
FEMT License.......5

Change log

Fecha	Comentarios	Autor
2012.11.29	First draft	Miguel Vargas miguelvargas@cimat.mx

Introduction

FEMT is an open source muli-platform library and tools (Windows, Linux and Mac OS) for solving large sparse systems of equations in parallel. This software is specialy set to solve systems of equations resulting from finite element, finite volume and finite differences discretizations.

The library and tools are in continuos development, but they are quite stable, currently our research group use them in several other projects.

The FEMT source code is basically divided in three parts:

- The library, called FEMT, was developed in standard C++ using templates extensively. It includes several routines for solving sparse systems of equations, like conjugate gradient, biconjugate gradient, Cholesky and LU factorizations, these were implemented with OpenMP support. Also, the library includes an implementation of the Schur substructuring method, it was implemented with MPI to run in clusters of computers.
- A set of tools for using the FEMT library without pain. Learning how to use a library could take a lot of time, also not all users feel confortable programming in C++. With this in mind we developed several programs for accessing the library solvers through named pipes. From the user point of view, a named pipe is just a file where you write the system of equations using standard file functions, another file (named pipe) is used to read the result. See the tutorial below for an example. This makes possible to use the FEMT library from any programming languaje (as long it has support for accessing files), like C/C++, Fortran, Python, C#, Java, etc.
- Finite element simulation modules for GiD. GiD is a pre and postprocessor developed by CIMNE, with it you can design a geometry (2D and 3D), set materials and boundary conditions, mesh it, call a FEM solver module and visualize the results. The modules (problem types) implemented so far are: linear solid deformation (static and dynamic), heat difussion (static and dynamic) and electric potential (it calculates also capacitance matrices and sensitivity maps). These problem types use the FEMT library for solving the finite element problems. Several examples with different geometries are included.

Source code, building instructions, tutorials and extra documentation can be found at: http://www.cimat.mx/~miguelvargas/FEMT

The solvers

There are three kind of solvers: direct, iterative and domain decomposition. Direct and iterative solvers are designed to run in parallel in multi-core computers using OpenMP. The domain decomposition solver has been designed to run in clusters of computers using a combination of MPI (Message Passing Interface) and OpenMP.

We should remark that all the solvers can only be applied to sparse matrices (symmetric or not) that have symmetric structure. This is the case for matrices resulting from finite element, finite volume and finite differences problems.

The list of solvers is

- Cholesky LL’ (CH)
- Cholesky LDL’ (CH2)
- LU, the doolittle version (LU)
- Conjugate gradient (CG)
- Conjugate gradient + Jacobi preconditioner (CG-J)
- Conjugate gradient + incomplete Cholesky factorization preconditioner (CG-CH)
- Conjugate gradient + factorized sparse approximate inverse preconditioner (CG-INV)
- Biconjugate gradient (BiCG)

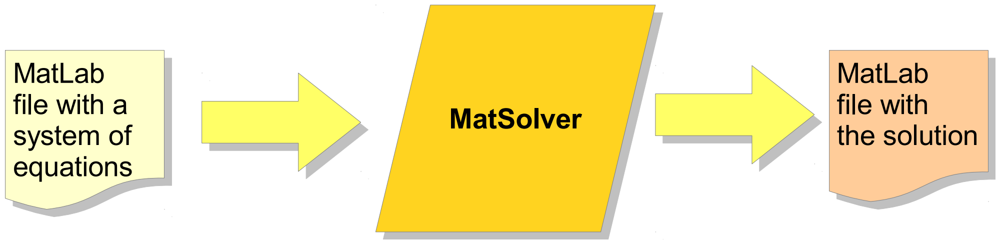
- Biconjugate gradient + Jacobi preconditioner (BiCG-J)
- Biconjugate gradient + incomplete LU factorization preconditioner (BiCG-LU)
- Biconjugate gradient + factorized sparse approximate inverse preconditioner (BiCG-INV)

The tools

Parameters to the tools are passed using environment variables. These variables are presented below.

MatSolver

A simple way to access the FEMT libary solvers is through systems of equations written in the MatLab file format, MatSolver reads this file, calls any of the solvers available and stores the result in a file with MatLab format.


MatLab file with a system of equations
MatSolver
MatLab file with the solution

Currently MatSolver only support the MAT-File 4 format. To save data with this format, using MatLab or Octave, you have to add the '-v4' parameter. This is an example:

```
save('-v4', 'data.mat', 'A', 'b');
```

The instruction to read a file with this format is simply:

```
load('data.mat');
```

Environment variables

Example for bash in Mac OS, GNU/Linux, BSD:

```
export SOLVER_THREADS=2
```

Example for Windows:

```
set SOLVER_THREADS=2
```

Common variables

SOLVER_TYPE

Chooses solver for all session.

Value	Description
1 (default)	Conjugate gradient
2	Cholesky decomposition LL'
3	Cholesky decomposition LDL'
4	Biconjugate gradient
5	LU decomposition

LOG_LEVEL

Determines the amount of output information provided by the program.

Value	Description
0	Only fatal error messages
1	Description of the current process
2 (default)	Solvers iterations are shown (default)

SOLVER_THREADS

Sets the maximum number of threads used by solvers. Performance will degrade if the number of threads exeeds tne number of cores in the system.

Value	Description
1 (default)	Solver is run in serial mode
> 1	Parallelization is aplied

Variables for iterative solvers

SOLVER_TOLERANCE

For iterative solvers, this defines the convergence criteria. The residual is defined as $\mathbf{r}=\mathbf{A}\,\mathbf{x}-\mathbf{b}$, iterations stops when $|\mathbf{r}=\mathbf{A}\,\mathbf{x}-\mathbf{b}|<t$, where t is the tolerance.

Value	Description
> 0	It can be any positive floating point value (double precision).

This parameter is used by the following solvers: CG, CG-J, CG-ICH, CG-INV, BiCG, Bi-CG-J, BiCG-ILU, BiCG-INV.

SOLVER_MAX_STEPS

For iterative solvers, it is the maximum number of iterations allowed.

Value	Description
1 to 2147483647	It can be any positive integer value, the default is 10000.

PRECONDITIONER_TYPE

The following values apply when using the conjugate gradient solver:

Value	Description
0	None
1 (default)	Jacobi preconditioner
2	Incomplete Cholesky preconditioner
5	Approximate inverse preconditioner

For the biconjugate gradient solver:

Value	Description
0	None
1 (default)	Jacobi preconditioner
4	Incomplete LU preconditioner
5	Approximate inverse preconditioner

PRECONDITIONER_LEVEL

This parameter controls how sparse is the preconditioner, a lower value means more sparse.

Value	Description
0 to 2147483647	It can be any positive integer value, the default is 1.

PRECONDITIONER_THRESHOLD

Sets the preconditioner threshold for approximate inverse

Value	Description
> 0	It can be any positive floating point value (double precision).

FEMT License