How To Call Direct Solvers in FASP

Why direct solvers

In FASP, we use direct solvers in various ways; for example, as a coarsest level solver for AMG or a subspace solver for ASM.

External direct solvers

We currently support the following sparse direct solvers

```
- Pardiso in MKL
- MUMPS
- UMFPACK in SuiteSparse
- Strumpack
- SuperLU
```

There are all free for personal uses and some of them are open-source packages. In practice, these general-purpose sparse direct solvers are robust and usually do not require user tunned parameters. It is not clear which one is more efficient for a particular problem. But we recommend to try them in the order provided above.

You can refer to their official website for downloading and installation guide lines:

1. [SuiteSparse](#)
2. [SuperLU](#)
3. [MUMPS](#)
4. [Strumpack](#)
5. [Pardiso](#), Intel oneAPI base toolkit

Link with direct solvers

By default, the direct solvers will be disabled during build except you enable them explicitly.

(1) Pardiso in MKL

Pardiso has two versions. We now use the version in Intel oneMKL for simplicity and ease to use with Intel compiler tool chain. In order to link with Paridso, make sure that oneAPI base toolkit has been installed and the appropriate environment variables has been set.

```
mkdir Build; cd Build;
cmake -DUSE_PARDISO=ON ..
```

```
    make -j 8
    make install
```

(2) Strumpack

```
    mkdir Build; cd Build;
    cmake -DUSE_STRUMPACK=ON ..
    make -j 8
    make install
```

Use direct solvers

There is an example for using different direct solvers in `faspsolver/test/main/test.c`, which reads

```c
#if WITH_SuperLU // Call SuperLU if linked with -DUSE_SUPERLU=ON
    else if (solver_type == SOLVER_SUPERLU) {
        status = fasp_solver_superlu(&A, &b, &x, print_level);
    }
#endif

#if WITH_UMFPACK // Call UMFPACK if linked with -DUSE_UMFPACK=ON
    else if (solver_type == SOLVER_UMFPACK) {
        dCSRmat A_tran;
        fasp_dcsr_trans(&A, &A_tran);
        fasp_dcsr_sort(&A_tran);
        void* Numeric = fasp_umfpack_factorize(&A_tran, print_level);

        status = fasp_umfpack_solve(&A_tran, &b, &x, Numeric,
print_level);
        fasp_umfpack_free_numeric(Numeric);
        fasp_dcsr_free(&A_tran);
    }
#endif

#if WITH_MUMPS // Call MUMPS if linked with -DUSE_MUMPS=ON
    else if (solver_type == SOLVER_MUMPS) {
        status = fasp_solver_mumps(&A, &b, &x, print_level);
    }
#endif

#if WITH_PARDISO // Call PARDISO if linked with -DUSE_PARDISO=ON
    else if (solver_type == SOLVER_PARDISO) {
        fasp_dcsr_sort(&A);
        status = fasp_solver_pardiso(&A, &b, &x, print_level);
```

```
        }
    #endif

    #if WITH_STRUMPACK // Call STRUMPACK if linked with –DUSE_STRUMPACK=ON
        else if (solver_type == SOLVER_STRUMPACK) {
            fasp_dcsr_sort(&A);
            status = fasp_solver_strumpack(&A, &b, &x, print_level);
        }
    #endif
```

You can follow the example to call these solvers in your own code after you just like calling any other solvers provided in FASP. Note that these packages might different internal data structures for coefficient matrix, so we need to do a little preprocessing before calling them.