1 Instructions for compiling and running DDalpha

In order to compile the software MPI has to be loaded. With this requirement it is enough to run the `Makefile` to build DDalpha

```
make -f Makefile -j numberofthreads wilson.
```

This will generate two executables: `dd_alpha_amg` and `dd_alpha_amg_db`. The latter is the developer version, which is not relevant here. The program requires many input parameters to run properly. These are explained in detail in the documentation `user_doc.pdf`). However, I provide two sample scripts, `sample4.ini` and `sample8.ini`, for facilitating the usage. They correspond to an inversion of the Dirac matrix on 4^4 and 8^4 lattices respectively. The right-hand side of the matrix problem can be chosen to be random or a vector with all its entries equal to one. The sample files choose a vector with ones. DDalpha also requires a *gauge configuration* to run. Given the boundary conditions and a configuration, the correspondent Dirac matrix is inverted by the program. Random configurations can be created by compiling the source file `conf/random/random_conf.c`

```
gcc random_conf.c -o rand.x -lm
```

When executing this program the lattice dimensions have to be passed through the terminal

```
./rand.x Nt Nz Ny Nx
```

This generates a binary file with name `NtxNzxNyxNx_random` with a gauge configuration. DDalpha does not assemble the Dirac matrix due to its large size. However, in case that it is necessary to have the matrix information, file `conf/random/dirac_matrix.c` can be compiled in a similar manner as `random_conf.c`,

```
gcc dirac_matrix.c -o dirac_matrix.x -lm,
```

to build the matrix for a given configuration and dump its non-zero entries into a binary file. Once again, the lattice dimensions have to be passed from the terminal

```
./dirac_matrix.x Nt Nz Ny Nx.
```

For the moment this program only assembles the matrix with periodic boundary conditions. To read the binary I provide the file `conf/random/read_matrix.c`. It can be modified to your convenience for handling the non-zero entries of the Dirac matrix.

In the future it will be convenient to use configurations from real physical simulations I will provide this data, but for now random configurations should be enough for testing.

The path to the configuration file `dd_alpha_amg` can be changed in the sample files.

To simplify the execution you can use the `run` script. You only have to modify the sample file path.

2 Printing the test vectors

The interpolator is built by arranging the test vectors in columns. DDalpha computes this vectors in the setup phase, starting with random vectors. The number of test vectors is a free parameter of the code, it can be changed in the sample files. In order to print the test vector it is necessary to pass a flag to the compiler

```
make CFLAGS=" -DTESTVECTOR_ANALYSIS "
```

When running the program with this compiler flag the program will print the test vectors into a .txt file. It is important to note that each time the program is run, the test vectors will be different, because they start as random guesses. However, in every execution the set of test vectors should approximate the *near kernel* well.