1 Instructions for compiling and running DDalpha

In order to compile the software MPI has to be installed. 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 using a two-grid method. 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 (periodic in the samples) and a configuration, the corresponding 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, `NtxNzxNyxNx_random`, with a SU(3) gauge configuration. DDalpha does not assemble the Dirac matrix due to its large size, but in case it is necessary to have the matrix information, one can compile `conf/random/dirac_matrix.c` 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`

and the configuration has to be in the same directory. For the moment this program only assembles the matrix with periodic boundary conditions. Essentially, it implements the following formula

$$\begin{aligned} D(\vec{n},\vec{m}) = (m_0+4)\,\frac{1}{a} I_{12}\delta_{\vec{n},\vec{m}} \\ -\frac{1}{2a}\sum_{\mu=0}^{3}\left[(I_4-\gamma_\mu)\otimes U_\mu(\vec{n})\delta_{\vec{n}+\hat{\mu},\vec{m}} + (I_4-\gamma_\mu)\otimes U_\mu^\dagger(\vec{n}-\hat{\mu})\delta_{\vec{n}-\hat{\mu},\vec{m}}\right], \end{aligned}$$

where m_0 is a free parameter (I fixed it to -0.5), a is the lattice spacing (I fixed it to one), I_{12} and I_4 are the twelve and four dimensional identity matrices respectively, γ_μ are the Dirac matrices

$$\gamma_0 = \gamma_x = \begin{pmatrix} & & & i \\ & & i & \\ & -i & & \\ -i & & & \end{pmatrix}, \quad \gamma_1 = \gamma_y = \begin{pmatrix} & & & -1 \\ & & 1 & \\ & 1 & & \\ -1 & & & \end{pmatrix},$$

$$\gamma_2 = \gamma_z = \begin{pmatrix} & & i & \\ & & & -i \\ -i & & & \\ & i & & \end{pmatrix}, \quad \gamma_3 = \gamma_t = \begin{pmatrix} & & 1 & \\ & & & 1 \\ 1 & & & \\ & 1 & & \end{pmatrix}.$$

The vectors $\vec{n}$ refer to the lattice sites which live in the volume

$$V = \{\vec{n} = (n_0, n_1, n_2, n_3) | n_\mu = 0, 1, \ldots, N_\mu - 1; \mu = 0, 1, 2, 3\},$$

$$N_0 = N_x, \quad N_1 = N_y, \quad N_2 = N_z, \quad N_3 = N_t.$$

and $\hat{\mu}$ is a unit vector in the direction indexed by μ. The *gauge links* $U_\mu(\vec{n})$ are random SU(3) matrices stored in the configuration file. To read the binary with $D(\vec{n}, \vec{m})$ I wrote `conf/random/read_matrix.c`. It can be modified for handling the non-zero entries of the Dirac matrix as you need.

The path to the configuration file can be changed in the sample files. To simplify the execution you can use the `run` script. In case it does not work, perhaps you have to modify lines 190 and 200, which depend on how you excecute programs with MPI on your machine. The sample files have to be passed as input parameters at the execution call, for instance

`./run -i` sample8.ini.

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.

2 Printing the test vectors

The interpolator is built by arranging the test (complex) vectors in columns over the aggregates, as explained in Ref. [1]. DDalpha computes these vectors in the setup phase. The number of test vectors is a free parameter of the code and can be changed in the sample files. Through numerical experimentation it was observed that between 20 and 30 vectors works fine. The test vectors at level l are printed in a file with name `testvector_level`l`.txt`. They are arranged in a single column separated by dashed lines. For the finest level (highest value of l) each vector has $12 \times N_x \times N_y \times N_z \times N_t$ complex entries, which corresponds to the Dirac matrix dimensions.

References

[1] A. Frommer, K. Kahl, S. Krieg, B. Leder and M. Rottmann. Adaptive aggregation based domain decomposition multigrid for the lattice Wilson Dirac operator. *SIAM J. Sci. Comp.*, **36**(4):A1581–A1608, (2014).