README

This document goes along with the C++ implementation of the methods in the paper "A parallel Heap-Cell Method for Eikonal equations," [1]. We release the source code of our algorithm under the GNU GPL license. The compilation and execution have been tested only on Linux systems using either the gcc compiler or the Intel icpc compiler. All instructions that follow are for Linux systems. There is no makefile; code can be compiled by running "compileScript.sh".

1. Usage. Most of the code's configuration can be handled by modifying GlobalVars3D.h and the `main()` function in main.cpp. In `main()`, each algorithm has its own subroutine:

1. `FMM()`, Fast Marching Method
2. `FSM()`, Fast Sweeping Method
3. `LSM(&numSweeps)`, Locking Sweeping Method
4. `DetrixheFSM()`, Detrixhe Fast Sweeping Method (parallel)
5. `HCM(false, cellTimes, cellExclTimes, AvS)`, Heap-Cell Method
6. `pHCM(false, cellTimes, cellExclTimes, AvS)`, parallel Heap-Cell Method (parallel)
7. `Run_Record_Scaling(params)`, runs all methods for a fixed M, including parallel methods up to `numPTrials` threads.
8. `Run_RecordTiming(params)`, runs all serial methods for several different M, with these parameters set in GlobalVars3D.h. See item 1 of the next section.

When using HCM and pHCM, the number of gridpoints must be divisible by the number of cells. The program parameters are:

1. m, the number of gridpoints along each dimension of the uniform Cartesian grid. Default 96.
2. s, the speed function. The set of available functions are "sinusoid1," "sinusoid2," "checkerboard," "shellMaze," and "constantf." Default constantf.
3. c, the number of checkers per dimension used in the checkerboard example. Default 11.
4. n, the number of trials to run the method. Default 1.
5. e, the exit set. The set of available ones are "center," "lowerleftcorner," and "zaxis." Default center.

An example run would be:

./parallelEik.exe -m 160 -c checkerboard -c 14 -e center

2. Reproducing the results from [1].

1. Figure 4: In main.cpp: `main()`, use `RunRecordTiming()`. Make sure in `HCM()` on the line with `SetCellGlobals()` that the power of 2 is (`l`) (as in "el"). GlobarlVars3D.h: `numMtrials = 4`, `mStart = 128`, `mStep = 64`, `numCellDivs = 5`. After compiling (sh compileScript.sh), run ./parallelEik.exe -s constantf, ./parallelEik.exe -s sinusoid1, or ./parallelEik.exe -s sinusoid2. The output files are EikonalTimingData.m and numSweeps_M.txt.
2. Figures 5 and 6: In main.cpp: `main()`, use `RunRecordScaling()`. Make sure in both HCM and pHCM on the line `SetCellGlobals()` that the power of 2 is (`l+1`) (as in "el plus one"). GlobarlVars3D.h: `numMtrials = 1`, `numCellDivs = 4`, `numPTrials = 32`. After compiling (sh compileScript.sh), run ./parallelEik.exe -s constantf -m 320, ./parallelEik.exe -s sinusoid1 -m 320, or ./parallelEik.exe -s sinusoid2 -m 320. The output file is EikonalTimingData.m.
3. Figure 7: same settings as item 2, but run ./parallelEik.exe -s constantf -m 128, ./parallelEik.exe -s sinusoid1 -m 128, or ./parallelEik.exe -s sinusoid2 -m 128.
4. Figure 8: same settings as items 1 and 2, respectively, but in GlobalVars3D.h uncomment `#define EARLY_SWEEP 1`
5. Figure 9: same settings as items 1 and 2, respectively, but in GlobalConfiguration.h change line 54 from `#define DOUBLE double` to `#define DOUBLE float`.
6. The additional examples can be tested by running with ./parallelEik.exe -s shellMaze, ./parallelEik.exe -s checkerboard -c 11, and ./parallelEik.exe -s checkerboard -c 41.

If you have any other questions, please email Adam Chacon at adamdante@gmail.com.

REFERENCES

[1] A. Chacon and A. Vladimirsky, *A parallel Heap-Cell Method for Eikonal equations*, Technical Report; available from `http://www.math.cornell.edu/~vlad/papers/pHCM`