ARTIFACT DESCRIPTION

A. Getting and Compiling DENDRO

The DENDRO simulation code is freely available at GitHub (https://github.com/paralab/Dendro-GR) under the MIT License. The latest version of the code can be obtained by cloning the repository

```
$ git clone git@github.com:paralab/Dendro-GR.git
```

The following dependencies are required to compile DENDRO

- C/C++ compilers with C++11 standards and OpenMP support
- MPI implementation (e.g. openmpi, mvapich2)
- ZLib compression library (used to write `.vtu` files in binary format with compression enabled)
- BLAS and LAPACK are optional and not needed for current version of DENDRO
- CMake 2.8 or higher version

Note: We have tested the compilation and execution of DENDRO with `intel`, `gcc` 4.8 or higher, `openmpi`, `mpich2` and `intelmpi` and `craympi` (in Titan) using the linux operating systems.

To compile the code, execute these commands

```
$ cd <path to DENDRO directory>
$ mkdir build
$ cd build
$ ccmake ../
```

The following options for DENDRO can then be set in cmake:

- `DENDRO_COMPUTE_CONSTRAINTS` : Enables the computation of Hamiltonian and momentum constraints
- `DENDRO_CONSEC_COMM_SELECT` : If `ON` sub-communicators are selected from consecrative global ranks, otherwise sub-communicators are selected complete binary tree of global ranks (note that in this case global communicator size need to a power of 2).
- `DENDRO_ENABLE_VTU_CONSTRAINT_OUT` : Enables constraint variable output while time-stepping
- `DENDRO_ENABLE_VTU_OUTPUT` : Enables evolution variable output while time-stepping
- `DENDRO_VTK_BINARY` : If `ON` vtu files are written in binary format, else ASCII format (binary format recommended).
- `DENDRO_VTK_ZLIB_COMPRES` : If `ON` binary format is compressed (only effective if `DENDRO_VTK_BINARY` is `ON`)
- `HILBERT_ORDERING` : Hilbert SFC used if `ON`, otherwise Morton curve is used. (Hilbert curve is recommended to reduce the communication cost.)
- `NUM_NPES_THRESHOLD` : When running in large scale set this to $\sqrt{p}$ where p number of mpi tasks for better performance.
- `RK_SOLVER_OVERLAP_COMM_AND_COM` : If `ON` non blocking communication is used and enable overlapping of communication and computation *unzip* (recommended option), otherwise blocking synchronized *unzip* is used.

After configuring DENDRO, generate the Makefile (use `c` to configure and `g` to generate). Then execute `make all` to build all the targets. On completion, `bssnSolver` will be the main executable as related to this paper.

B. Getting Started: Running `bssnSolver`

`bssnSolver` can be run as follows.

```
$ mpirun -np <number of mpi tasks>\
  ./bssnSolver \
<parameter file name>.par
```

Example parameter files can be found in `BSSN_GR/pars/`. The following is an example parameter file for equal mass ratio binary inspirals.

```
{
"DENDRO_VERSION": 5.0,
"BSSN_RESTORE_SOLVER":0,
"BSSN_IO_OUTPUT_FREQ": 10,
"BSSN_REMESH_TEST_FREQ": 5,
"BSSN_CHECKPT_FREQ": 50,
"BSSN_VTU_FILE_PREFIX": "bssn_gr",
"BSSN_CHKPT_FILE_PREFIX": "bssn_cp",
"BSSN_PROFILE_FILE_PREFIX": "bssn_rl",
"BSSN_DENDRO_GRAIN_SZ": 100,
"BSSN_ASYNC_COMM_K": 4,
"BSSN_DENDRO_AMR_FAC": 1e0,
"BSSN_WAVELET_TOL": 1e-4,
"BSSN_LOAD_IMB_TOL": 1e-1,
"BSSN_RK_TIME_BEGIN": 0,
"BSSN_RK_TIME_END": 1000,
"BSSN_RK_TIME_STEP_SIZE": 0.01,
"BSSN_DIM": 3,
"BSSN_MAXDEPTH": 12,
"ETA_CONST": 2.0,
"ETA_R0": 30.0,
"ETA_DAMPING": 1.0,
"ETA_DAMPING_EXP": 1.0,
"BSSN_LAMBDA": {
"BSSN_LAMBDA_1": 1,
"BSSN_LAMBDA_2": 1,
"BSSN_LAMBDA_3": 1,
"BSSN_LAMBDA_4": 1
},
"BSSN_LAMBDA_F": {
"BSSN_LAMBDA_F0": 1.0,
"BSSN_LAMBDA_F1": 0.0
},
"CHI_FLOOR": 1e-4,
"BSSN_TRK0": 0.0,
"KO_DISS_SIGMA": 1e-1,
"BSSN_BH1": {
"MASS":0.48528137423856954,
"X": 4.00000000e+00,
"Y":0.0,
"Z": 1.41421356e-05,
"V_X": -0.00132697,
"V_Y": 0.1123844,
"V_Z": 0,
"SPIN": 0,
"SPIN_THETA":0,
"SPIN_PHI": 0
},
"BSSN_BH2": {
"MASS":0.48528137423856954,
"X":-4.00000000e+00,
"Y":0.0,
"Z":1.41421356e-05,
"V_X": 0.00132697,
"V_Y": -0.1123844,
"V_Z": 0,
"SPIN": 0,
"SPIN_THETA":0,
"SPIN_PHI": 0
}
}
```

Here we list the key options for `bssnSolver` with a short description.

- `BSSN_RESTORE_SOLVER` : Set 1 to restore RK solver from latest checkpoint.
- `BSSN_IO_OUTPUT_FREQ` : IO (i.e. `vtu` files) output frequency
- `BSSN_CHECKPT_FREQ` : Checkpoint file output frequency

- BSSN_REMESH_TEST_FREQ : Remesh test frequency (i.e. frequency in time steps that is being tested for remeshing)
- BSSN_DENDRO_GRAIN_SZ : Number of octants per core
- BSSN_ASYNC_COMM_K : Number of variables that are being processed during an asynchronous *unzip* (< 24)
- BSSN_DENDRO_AMR_FAC : Safety factor for coarsening i.e. coarsen if and only if $W_c \leq AMR\_FAC \times WAVELET\_TOL$ where W_c is the computed wavelet coefficient.
- BSSN_WAVELET_TOL : Wavelet tolerance for WAMR.
- BSSN_MAXDEPTH : Maximum level of refinement allowed (≤ 30)
- KO_DISS_SIGMA : Kreiss-Oliger dissipation factor for BSSNKO formulation
- MASS : Mass of the black hole
- X : x coordinate of the black hole
- Y : y coordinate of the black hole
- Z : z coordinate of the black hole
- V_X : momentum of the black hole in x direction
- V_Y : momentum of the black hole in y direction
- V_Z : momentum of the black hole in z direction
- SPIN : magnitude of the spin of the black hole
- SPIN_THETA : magnitude of the spin of the black hole along θ
- SPIN_PHI : magnitude of the spin of the black hole along ϕ

1) Generating your own parameters: The intial data parameters for a black hole binary depend on the total mass ($M = m1 + m2$), the mass ratio q and the separation distance d. These parameters are calculated using the python script BSSN_GR/scripts/id.py. The command to generate parameters for $q = 10$, total mass $M = 5$ and separation $d = 16$ is

```
$ python3 id.py -M 5 -r 10 16
```

```
PUNCTURE PARAMETERS (par file foramt)
"BSSN_BH1": {
"MASS":4.489529,
"X":1.454545,
"Y":0.000000,
"Z": 0.000014,
"V_X": -0.020297,
"V_Y": 0.423380,
"V_Z": 0.000000,
"SPIN": 0.000000,
"SPIN_THETA":0.000000,
"SPIN_PHI": 0.000000
},
"BSSN_BH2": {
"MASS":0.398620,
"X":-14.545455,
"Y":0.000000,
"Z":0.000014,
"V_X": 0.020297,
"V_Y": -0.423380,
"V_Z": 0.000000,
"SPIN": 0.000000,
"SPIN_THETA":0.000000,
"SPIN_PHI": 0.000000
}
The tangential momentum is just an estimate, and the value for a
for a circular orbit is likely between (0.5472794147860968, 0.29947988193805547)
```

C. Symbolic interface and code generation

The BSSNKO formulation is a decomposition of the Einstein equations into 24 coupled hyperbolic PDEs. Writing the computation code for the BSSNKO formulation can be a tedious task. Hence we have written a symbolic python interface to generate optimized C code to compute the BSSNKO equations. All the symbolic utilities necessary to write the BSSNKO formulation in symbolic python can be found in GR/rhs_scripts/bssn/dendro.py and the symbolic BSSNKO code can be found in GR/rhs_scripts/bssn/bssn.py. This could be modified for more advanced uses of the code such as including new equations to describe additional physics or for introducing a different formulation of the Einstein equations.

D. Profiling the code

DENDRO contains built-in profiler code which enables one to profile the code extensively. On configuration, a user can enable/disable the internal profiling flags using ENABLE_DENDRO_PROFILE_COUNTERS and the profile output can be changed between a human readable version and a tab separated format using the flag BSSN_PROFILE_HUMAN_READABLE. Note that in order to profile communication, internal profile flags need to be enabled. The following is an example of profiling output for the first 10 time steps.

```
active npes : 16
global npes : 16
current step : 10
partition tol : 0.1
wavelet tol : 0.0001
maxdepth : 12
Elements : 4656
DOF(zip) : 279521
DOF(unzip) : 2078609
============ MESH =================
step                    min(#)      mean(#)     max(#)
ghost Elements          634         824.062     1065
local Elements          263         291         319
ghost Nodes             43781       55671.7     71693
local Nodes             14292       17470.1     20705
send Nodes              18760       24872.9     36861
recv Nodes              18113       24872.9     33777
========== RUNTIME ================
step                    min(s)      mean(s)     max(s)
++2:1 balance           0           0           0
++mesh                  1.9753      1.98299     1.98946
++rkstep                20.159      20.1856     20.1996
++ghostExchge.          1.81442     3.15703     4.49568
++unzip_sync            8.27839     9.67293     11.0991
++unzip_async           0           0           0
++isReMesh              0.04642     0.117357    0.207305
++gridTransfer          1.53709     1.54899     1.56531
++deriv                 1.98942     2.34851     2.76695
++compute_rhs           4.00119     4.61547     5.11566
--compute_rhs_a         0.0137962   0.0245449   0.0351532
--compute_rhs_b         0.0296426   0.0503471   0.069537
--compute_rhs_gt        0.111898    0.12846     0.15463
--compute_rhs_chi       0.0170642   0.0315392   0.044856
--compute_rhs_At        2.40738     2.72922     3.05622
--compute_rhs_K         0.358215    0.39879     0.457139
--compute_rhs_Gt        0.774211    0.933581    1.05702
--compute_rhs_B         0.0575426   0.071209    0.0855094
++boundary con          0           0.0421986   0.134712
++zip                   0.23529     0.260862    0.291513
++vtu                   0.0872362   0.101362    0.128246
++checkpoint            3.27e-06    3.85e-06    5.7469e-06
```

E. Visualizing the data

DENDRO can be configured to output parallel unstructured grid files in binary file format (.pvtu). These files can be visualized using any visualization tool which supports VTK file formats. All the images presented in this paper used Paraview due to its robustness and scalability. Paraview allows python based scripting to perform pvbatch visualization, an example pvpython script can be found in scripts/bssnVis.py