Readme

Parts 1 and 2 of this assignment are located in separate folders. Within each folder is a shell script, _compile_and_run.sh, that will compile all relevant executables and run them with the inputs required for generating results as used in the report. If this shell script fails, this document contains more explicit instructions on compiling and running the files.

To generate the plots in the report, python scripts have been added to the pre-made results folders. If results have been generated correctly, these files should generate all plots of interest.

Part 1

Part 1 contains only a single cpp runtime script, `simulation.cpp, along with its dependencies:`

- `forces_and_integrators.cpp`
  Contains all functions relevant to the physical properties of the system and time-series integration (e.g. RK-4 Integration, planetary gravity etc)
- `sysvec_utils.cpp`
  Utility functions and overloads for system state vectors
- `vector_utils.cpp`
  Utility functions and overloads for vectors
- `manybody_example.hpp`
  Contains parameters for an example run (mode 4) of the simulation with many bodies

To compile, call:

```
g++ -o simulation simulation.cpp forces_and_integrators.cpp
sysvec_utils.cpp vector_utils.cpp
```

Once compiled, the executable will generate and save results for questions 1-3 if called with the relevant command line flag. To produce results for all questions and save with the filename required for plotting, use:

```
./simulation 1 ./results/pt_1a.dat 100

./simulation 2 ./results/pt_1b.dat 100

./simulation 3 ./results/pt_1c.dat 100

./simulation 4 ./results/pt_1d.dat 2600 1 1 0.01 5
```

All of these files can be compiled automatically by using the shell script "_compile.sh".

Part 2

Part 2 contains 3 compile-able cpp scripts:

- `singlegrid.cpp`
  Runs a single MCMC chain for a grid of fixed size at fixed temperature and describes / prints the resultant end state
- `temp_sweep.cpp`
  Calculates performs progressive MCMC chains over a range of temperatures and saves the mean and variance of energy and spin for each temperature and each chain.
- `singlegrid_parallel.cpp`
  Runs several MCMC chains and prints their running times. For use in multi-thread benchmarking

Alongside the dependencies:

- `vector_utils.cpp`
  Utility functions and overloads for vectors. Re-used from assignment 1. In this instance only really used to generate temperature grids more easily
- `monte_carlo.cpp`
  Contains all functions related to the MCMC runs, including functions for calculating the mean and variance of vectors
- `grid.hpp`
  Contains an object class for the grid along with all physically relevant functions, like energy calculations

As `monte_carlo.cpp` contains `openMP` functions, all files must be compiled using the `-fopenmp` flag:

```
g++ -o singlegrid singlegrid.cpp monte_carlo.cpp
rand_utils.cpp vector_utils.cpp -fopenmp

g++ -o temp_sweep temp_sweep.cpp monte_carlo.cpp
rand_utils.cpp vector_utils.cpp -fopenmp

g++ -o singlegrid_parallel singlegrid_parallel.cpp
monte_carlo.cpp rand_utils.cpp vector_utils.cpp -fopenmp
```

All results for plotting / answering questions are generated by temp_sweep.cpp. To produce the results as used in the report, call with command line arguments like:

```
./temp_sweep  8 0.0 5.0 32 ./results/temp_sweep_8.dat

./temp_sweep 16 0.0 5.0 32  ./results/temp_sweep_16.dat

./temp_sweep  32 0.0 5.0 32 ./results/temp_sweep_32.dat

./temp_sweep  64 0.0 5.0 32 ./results/temp_sweep_64.dat
```

For parts a and b, and for part c:

```
./temp_sweep  128 2.0 2.26918531421 32
./results/temp_sweep_128_nearcrit.dat 100000 1000 1
```

Running / Inputs

All compiled executables are set to take arguments from command line. Arguments must be given in order, and will defer to default values if not provided. E.g.

```
./simulation 1 ./results/pt_1a.dat 100
```

Will run simulation with $mode = 1$, save results to `./results/pt_1a.dat`, use a maximum simulation time of $tmax = 100$, and will use default values for v_{launch}, $thet, dt$ and sparseness.

Part 1: simulation

Input	Type	Constraints	Default	Desc
mode	int	1-4	1	Determines the type of simulation to
output_dir	str		results/sim_results.dat	Location to save outputs to
tmax	double	>0	100	Max sim time
v_launch	double	>0	1.1376	Launch velocity for modes 1-3
thet	double	>0	-1.08	Starting angle of moon in orbit, rad
dt	double	>0, <tmax	0.01	Timestep for RK4 integration
sparseness	int	>0, <maxits	1	Number of int steps per output line

Part 2: singlegrid

Input	Type	Constraints	Default	Desc
N	int	>0	8	Dimension of NxN Grid
T	double	>=0	$0.75T_{crit}$	Temperature for Monte carlo simulation
Nits	int	>0	10,000	Number of full sweeps of the grid / samples to draw for the chain
Nburn	int	>=-0	1,000	Number of burn-in itterations in monte carlo
flips_per	int	>0	0(NxN)	Number of bit flips per 'sample' If set to zero, will default to $N \times N$ flips.
seed	int		0	Seed for random number generation

Part 2: singlegrid_parallel

Input	Type	Constraints	Default	Desc
N	int	>0	64	Dimension of NxN Grid
T	double	>=0	$0.75T_{crit}$	Temperature for Monte carlo simulation
Nits	int	>0	10,000	Number of full sweeps of the grid / samples to draw for the chain
Nburn	int	>=-0	1,000	Number of burn-in itterations in monte carlo

Part 2: temp_sweep

Input	Type	Constraints	Default	Desc
N	int	>0	8	Dimension of NxN Grid
T_min	double	>=0	0.0	Minimum temp for sweep
T_max	double	>=0	5.0	Maximum temp for sweep
Ntemps	int	>0	32	Number of temperatures in sweep
Nits	int	>0	10,000	Number of full sweeps of the grid / samples to draw for the chain
Nburn	int	>=-0	1,000	Number of burn-in itterations in monte carlo
flips_per	int	>0	0(NxN)	Number of bit flips per 'sample' If set to zero, will default to $N \times N$ flips.
seed	int		0	Seed for random number generation
printgrid	bool		True	If true, will print the final state of the grid at each temperature for the last chain as it completes its sweep