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, `main.cpp`, along with its dependencies:

- `forces_and_integrators.cpp / .hpp`
  Contains all functions relevant to the physical properties of the system and time-series integration (e.g. RK-4 Integration). Adapted from assignment 2.
- `complex_vector_utils.cpp / .hpp`
  Utility functions and overloads for vectors of complex doubles. Adapted from work similar work in previous assignments
- `_defs.hpp`
  Contains definitions and aliases common to other files

To compile, call:

```
g++ -o main main.cpp forces_and_integrators.cpp
complex_vector_utils.cpp
```

Once compiled, the executable will generate and save results for questions 1-4 if called with the relevant command line flags. To produce results for all questions as used in this report, use the following:

Question 2 – Generating different waves with various g values

```
./main 1 0.0 ./results/Q1-0-0
./main 1 0.1 ./results/Q1-0-1
./main 1 0.5 ./results/Q1-0-5
./main 1 2.0 ./results/Q1-2-0
./main 1 5.0 ./results/Q1-5-0
```

Question 3 – Wave packets with different momenta

```
./main 2 -1.0 ./results/Q2-0 0
./main 2 -1.0 ./results/Q2--1 -1
./main 2 -1.0 ./results/Q2-1 1
```

Question 4 – Two-Wave packets with varying phase shifts

```
./main 3 -1.0 ./results/Q3-0 0.1 0
./main 3 -1.0 ./results/Q3-0.10 0.1 0.1
./main 3 -1.0 ./results/Q3-0.25 0.1 0.25
./main 3 -1.0 ./results/Q3-0.5 0.1 0.5
./main 3 -1.0 ./results/Q3-1.0 0.1 1.0
```

All of these files can be compiled automatically by using the shell script "`_compile.sh`". The plot script for this section does not require specific file naming conventions, and instead prompts the user for a selection, so these exact commands are not strictly required.

Part 2

Part 2 contains 4 compile-able cpp scripts

- `_single_ring.cpp`
  A test-file for inspecting the properties of any 1D ising model ring, e.g. spin, Hamiltonian etc.
- `_Q1.cpp`
  Produces results for question 1, creating and diagonalizing systems of different atom counts. Does not save outputs
- `_Q2.cpp`
  Calculates the ground state energy of a system over a sweep of 'g' values. Will save outputs to nominated location.
- `_Q3.cpp`
  Produces results for question 4, calculating the ground state of a g=0 grid and time-evolving it. Will save outputs to nominated location.

Alongside the dependencies:

- `LP_solvers.cpp / .hpp`
  Contains wrapper functions for interacting with LAPACK's matrix diagonalization routines. Re-used from assignment 1
- `vector_utils.cpp / .hpp`
  Utility functions and overloads for vectors. Re-used from assignment 1 and 2 with minor modification
- `complex_vector_utils.cpp / .hpp`
  Utility functions and overloads for vectors of complex doubles. Re-used from part 1
- `matrix.cpp / .hpp`
  Contains the class for matrices with real double elements. Re-used from assignment 1 with some modification for matrix multiplication
- `matrix_complex.cpp / .hpp`
  As above but for complex matrices. Used in Q4 for time evolution
- `grid.hpp / .hpp`
  Contains an object class for the grid along with all physically relevant functions, like energy calculations
- `dot_and_convert.cpp / .hpp`
  Contains utilities for converting between real and complex vectors / matrices
- `physical_properties.cpp/ .hpp`
  Contains all physically meaningful functions, e.g. the hamiltonians
- `_defs.hpp`
  Contains definitions and aliases common to other files

As all scripts require use of the `LAPACK` matrix diagonalization routines, all files must be compiled using the `-llapack` and `-lblas` flags:

```
g++ -o _single_ring _single_ring.cpp LP_solvers.cpp
vector_utils.cpp matrix.cpp complex_vector_utils.cpp
matrix_complex.cpp dot_and_convert.cpp physical_properties.cpp
-llapack -lblas -O3
```

```
g++ -o _Q1 _Q1.cpp LP_solvers.cpp vector_utils.cpp matrix.cpp
complex_vector_utils.cpp matrix_complex.cpp
dot_and_convert.cpp physical_properties.cpp -llapack -lblas -
O3
```

```
g++ -o _Q2 _Q2.cpp LP_solvers.cpp vector_utils.cpp matrix.cpp
complex_vector_utils.cpp matrix_complex.cpp
dot_and_convert.cpp physical_properties.cpp -llapack -lblas -
O3
```

```
g++ -o _Q3 _Q3.cpp LP_solvers.cpp vector_utils.cpp matrix.cpp
complex_vector_utils.cpp matrix_complex.cpp
dot_and_convert.cpp physical_properties.cpp -llapack -lblas -
O3
```

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:

```
./_Q1
```

```
./_Q2
```

```
./_Q3
```

The plotting file for Q2 requires a specific naming convention. Generate results for different system sizes using:

```
./_Q2 129 0 8 4 ./results/energies-4
```

```
./_Q2 129 0 8 5 ./results/energies-5
```

```
./_Q2 129 0 8 6 ./results/energies-6
```

```
./_Q2 129 0 8 8 ./results/energies-8
```

```
./_Q2 129 0 8 10 ./results/energies-10
```

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.

```
./main 1 0.5 ./results/test.dat
```

Will run `main` with $mode = 1$ (cos-wave) and $g = 0.5$, saving results to `./results/test.dat` and using default values for all other parameters.

Part 1: main

Input	Type	Constraints	Default	Desc
mode	int	1-3	1	Determines the type of simulation to run.: 1. Question 2 – cos-wave initial start 2. Question 3 – sech-wave initial start 3. Question 4 – double sech wave initial start
g	double		0.0	Strength of particle self-interaction
output_dir	str		results/sim_results	Filename to save outputs to
u	double		0.0	Momentum for use in parts 2 and 3
phi	double		0.0	Phase shift for use in part 3 as a fraction of π. i.e. `phi=0.5` gives $\theta = \frac{1}{2}\pi$
Tmax	double	>0	40.0	Maximum simulation time
dt	double	>0, <tmax	0.01	Timestep for RK4 integration
sparse	int	>0, <maxits	1	Number of int steps per output line. Increase if output files are becoming too large.

Part 2: _single_ring

Input	Type	Constraints	Default	Desc
g	double		0.0	Strength of particle self-interaction
N	int	>0	8	Number of atoms in chain

Part 2: _Q1

Input	Type	Constraints	Default	Desc
g	double		0.0	Strength of particle self-interaction

Part 3: _Q2

Input	Type	Constraints	Default	Desc
g_step	int	>1	0.0	Nunber of g values to sweep over
g_min	double		0.0	Start of g value sweep
g_max	double		0.0	End of g value sweep
N	int	>0	8	Number of atoms in chain
out_url	str		./results/energies	Location / name to save outputs to

Part 3: _Q3

Input	Type	Constraints	Default	Desc
Tmax	double	>0	5.0	Maximum simulation time
dt	double	>0, <tmax	0.01	Timestep for time evolution calculations
g_1	double		0.0	Pre-quench g value
g_2	double		0.0	Post-Quench g value
N	int	>0	8	Number of atoms in chain
out_url	str		./results/quenchsim	Location / name to save outputs to