**General benchmarks (laptop)**

**Calculating 1E7 matrix products (2x2 real):**

Matlab (s): 4.99

Python + NumPy (s), Enthought: 9.69

Fortran + LAPACK (s): 13.19

Fortran + LAPACK + OpenMP (s): 3.08

**Calculating 1E6 matrix products (8x8 complex):**

Matlab (s): 1.33

Python + NumPy, Anaconda (s): 2.12

Python + NumPy (s), Enthought: 2.11

Python + Numpy, Intel (s): 2.15

Fortran + LAPACK (s): 1.90

Fortran + LAPACK + OpenMP (s): 0.46

**Calculating 1E5 Kronecker products:**

Matlab (s): 2.32

Python + NumPy, Anaconda (s): 4.45

Python + NumPy, Enthought (s): 4.37

Python + NumPy (s), Intel: 4.49

Fortran (s): 1.65

Fortran + OpenMP (s): 0.66 \*

\**The Fortran Kronecker function I adapted from an example online, so won’t be as optimised as a standard library function. Posting on Stackoverflow and asking for improvements may help significantly.*

**Calculating 1E4 matrix exponentials:**

Matlab (s): 1.37

Python + SciPy, Anaconda (s): 4.35

Python + SciPy, Enthought (s): 4.32

Python + SciPy, Intel (s): 4.23

Fortran + Expokit (s): 0.21

**Program benchmarks (laptop)**

**Solid effect:**

Matlab (s): 9.29

Python + NumPy (s), Anaconda: 31.06

Python + NumPy (s), Enthought: 29.22

Python + NumPy (s), Intel: 29.64

Python + F2PY (s): 4.18

Python + F2PY + OpenMP (s): 3.19 \*

**Calculating Liouville space propagator:**

Matlab (s): 8.46

Python + NumPy (s): 26.6

Python + F2PY (s): 1.78

Python + F2PY + OpenMP (s): 0.74 \*

*\*Currently there is a bug(?) associated with access violation when running multithreaded Kronecker product function, which may be limiting performance. A fix is to apply OpenMP statements to the function, however this needs to be investigated further.*

*\* Currently there is only a partial F2PY implementation, limited to Hamiltonian and propagator calculation.*

The significantly higher performance of Matlab for all tested matrix operations is puzzling, as both Python and Matlab are linked to LAPACK Fortran library for matrix multiplication. The Fortran benchmark supports this to a degree however it appears Matlab is somehow significantly more optimised. Need to test with ifort (Intel compiler) instead of gfortran, using Intel matrix multiplication DGEMM as it may be a multithreading issue.

The intel Python interpreter promises high performance utilising the MKL library, however this is not supported by benchmarks. It would be useful to post these online to see if I am doing something wrong, or if there are better interpreters/compilers to link to.

In conclusion Matlab is significantly faster than pure Python, however when combined with F2PY Python may perform up to an order of magnitude faster than Matlab. More work needs to be done to ensure Python speeds are representative, and to investigate other Fortran compilers and matrix multiplications.