Science and Technology Facilities Council

Scientific Computing

Towards automatic code generation for high-order compact schemes

Dr Jianping Meng
Computational Engineering Group
Computational Science and Engineering
Scientific Computing Department STFC Daresbury Laboratory

ExCALIBUR Turbulence Meeting – 29 Sep 2022 - Imperial College London

Motivation

Science and Technology Facilities Council
Scientific Computing

Separation of concerns (and domain specific language approach)
The algorithms that encapsulate the mathematics and physics of the problem are separated from the computational science of their implementation.
Exascale computer - The UK plans to have such a supercomputer by 2025

9K IBM POWER9 and 27K NVIDIA GPU

9K AMD EPYC and 37K AMD (Instinct) GPU

Motivation

Science and Technology Facilities Council
Scientific Computing

Separation of concerns (and domain specific language approach)
The algorithms that encapsulate the mathematics and physics of the problem are separated from the computational science of their implementation.
Complex mathematics?

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \tfrac{1}{3}\mu \nabla(\nabla \cdot \mathbf{u}) + \rho \mathbf{g}.$$

OpenSBLI: a domain-specific langauge type modelling framework that is capable of expanding a set of differential equations written in Einstein notation, and automatically generating C/C++ code that performs the finite difference approximation to obtain a solution**. Currently with the OP-DSL backend.**

Compact scheme

Science and Technology Facilities Council
Scientific Computing

From computational point of view, it is a type of relatively complicated finite-difference scheme, which requires:

- Prepare matrix (a,b,c,d)
- Linear solver
- Extra treatments for (periodic) boundary condition

$$\frac{1}{4}f'_{i-1} + f'_i + \frac{1}{4}f'_{i+1} = \frac{3}{2}\frac{f_{i+1} - f_{i-1}}{2\Delta}$$

Using the OP-DSL, we need following lines

```
ops_par_loop(preprocessX, "preprocessX", compact3d, 3, iterRange,
             ops_arg_dat(u, 1, S3D_7PT, "double", OPS_READ),
             ops_arg_dat(a, 1, S3D_000, "double", OPS_WRITE),
             ops_arg_dat(b, 1, S3D_000, "double", OPS_WRITE),
             ops_arg_dat(c, 1, S3D_000, "double", OPS_WRITE),
             ops_arg_dat(d, 1, S3D_000, "double", OPS_WRITE),
ops_arg_idx());
ops_tridMultiDimBatch_Inc(3, 0, size, a, b, c, d, ux, trid_ctx);
```

- It is better not to repeat all these sentence and arguments for each terms

Design and implementation

Science and Technology Facilities Council
Scientific Computing

- Combine the power of Python and a little more abstraction from C/C++ and OPS

In this case, we propose a general C/C++ function template to make codes more concise. Otherswise, it will leads to much longer codes than typical explicit schemes

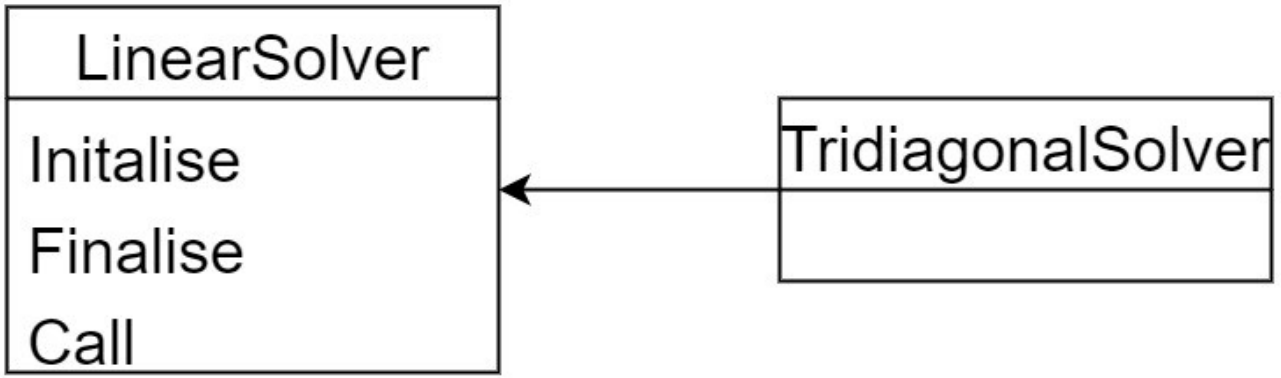
- No interference on existing codes following OOP principle

  The existing applications should be run without problem

- Consider future requirements while implementing using simple problems

Linear solver class

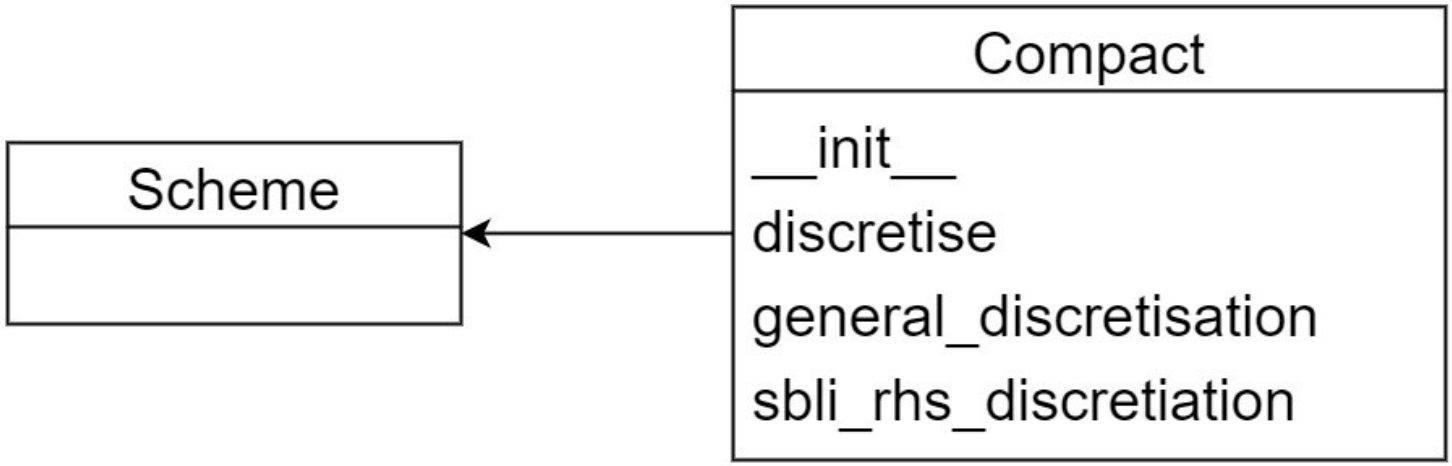
Science and Technology Facilities Council
Scientific Computing



For generating code to initialise, finalise and call linear algebra solver

Compact class

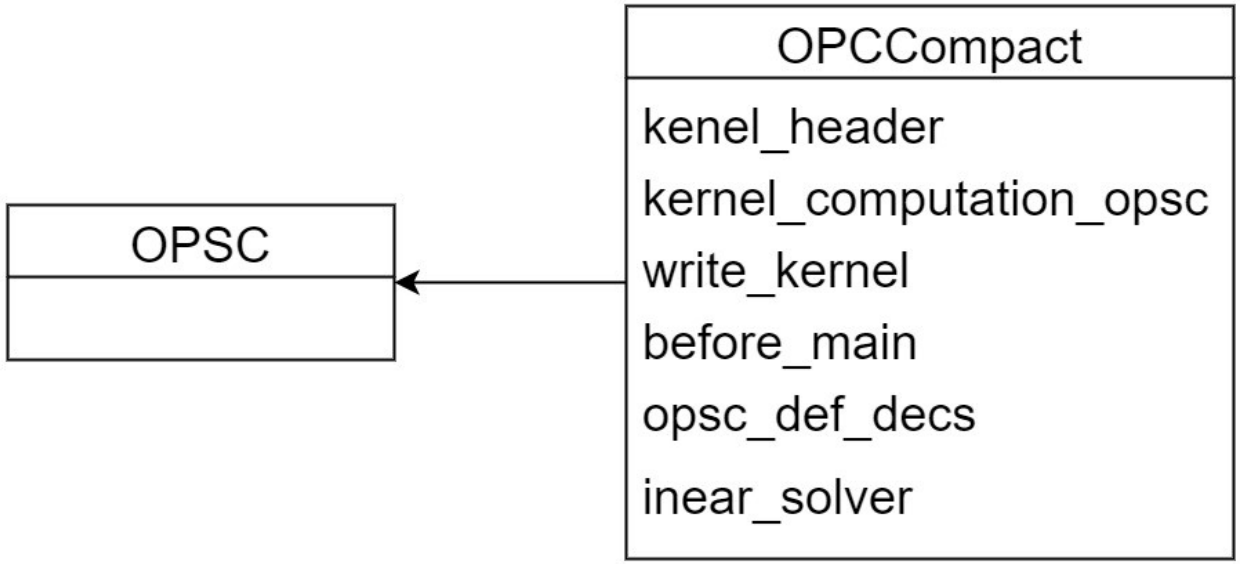
Science and Technology Facilities Council
Scientific Computing



To use the compact scheme to discrete derivatives.

OPSCCompact class

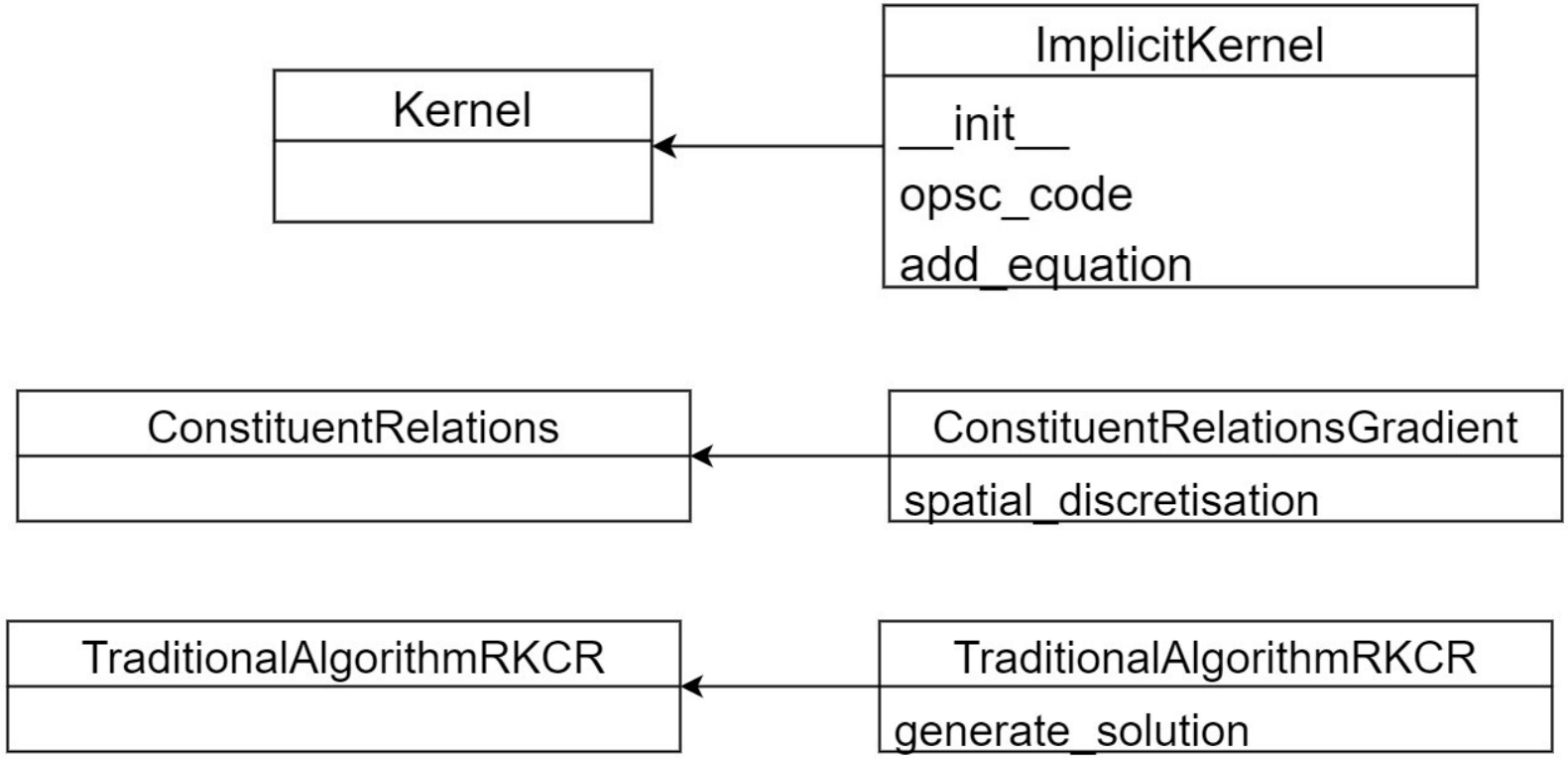
Science and Technology Facilities Council
Scientific Computing



To geneate the framework of the main source file

Implicit kernel, CR, and algorithm

Science and Technology Facilities Council
Scientific Computing



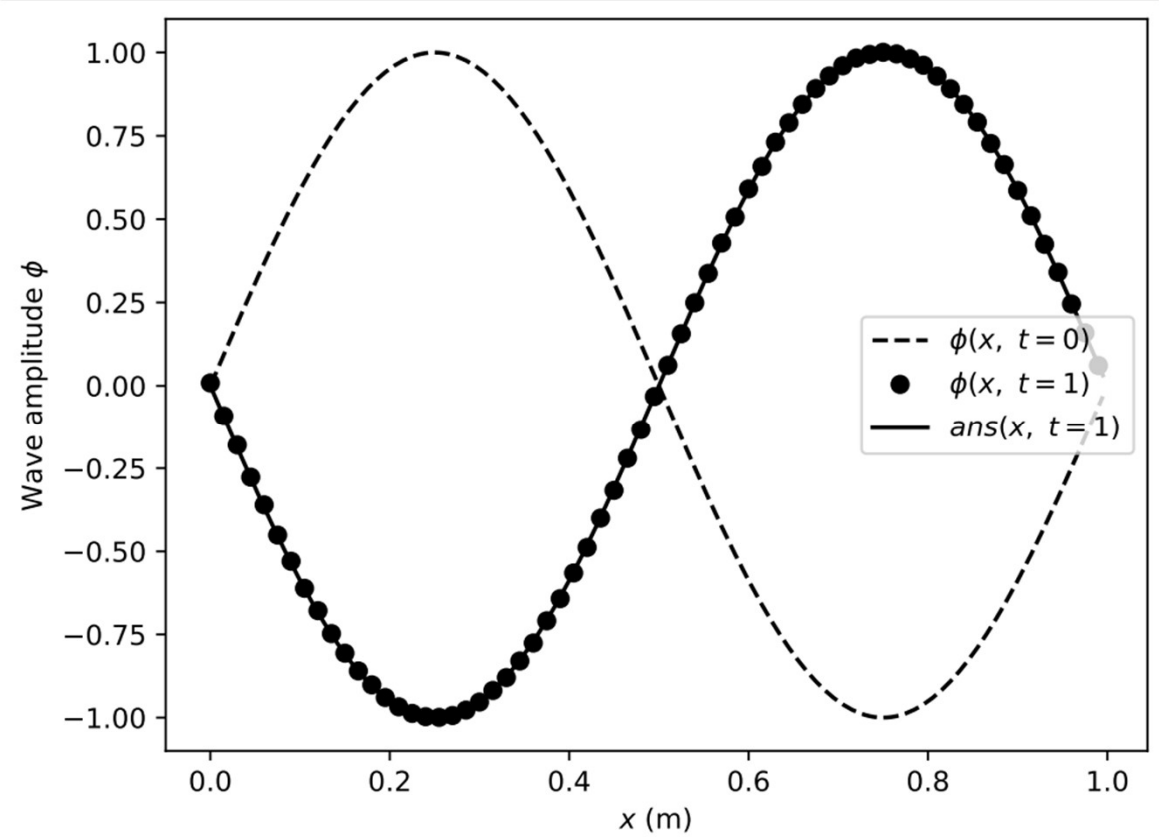
One-dimensional wave equation

Science and Technology Facilities Council
Scientific Computing

$$\frac{\partial \phi}{\partial t} = -c\frac{\partial \phi}{\partial x}$$

$$\phi|_{t=0} = \sin(2\pi x)$$

Generated code successfully run on both GPU and CPU



Taylor - Green vortex

Science and Technology Facilities Council
Scientific Computing

- Lead to shorter source code
- The source code looks correct by comparing with those of the central differencing schemes
- The codes compiles without problem.
- Further numerical tests are under way (after we investigate the proper periodic boundary or other alternatives.)

	Main	kernel
Central	651	727
Compact	798	461

```
void CompactDifference4thX1st(ops_block& block, ops_dat& u, ops_dat& a,
                              ops_dat& b, ops_dat& c, ops_dat& d, ops_dat& ux,
                              ops_tridsolver_params* trid, double delta,
                              int layer = 0) {
  int* size{u->size};
  int* dm{u->d_m};
  int* dp{u->d_p};
  int spaceDim{block->dims};
  int* tridSize{new int[spaceDim]};
  for (int i = 0; i < spaceDim; i++) {
    tridSize[i] = size[i] - dp[i] + dm[i];
  };
  int* iterRange{new int[2 * spaceDim]};
  for (int i = 0; i < spaceDim; i++) {
    iterRange[2 * i] = -layer;
    iterRange[2 * i + 1] = size[0] + layer;
  };
  ops_par_loop(PreprocessX4thCompact1st, "preprocessX", block, 1, iterRange,
               ops_arg_dat(u, 1, neighbor_stencil, "double", OPS_READ),
               ops_arg_dat(a, 1, local_stencil, "double", OPS_WRITE),
               ops_arg_dat(b, 1, local_stencil, "double", OPS_WRITE),
               ops_arg_dat(c, 1, local_stencil, "double", OPS_WRITE),
               ops_arg_dat(d, 1, local_stencil, "double", OPS_WRITE),
               ops_arg_dat(ux, 1, local_stencil, "double", OPS_WRITE),
               ops_arg_idx(), ops_arg_gbl(&size[0], 1, "int", OPS_READ),
               ops_arg_gbl(&layer, 1, "int", OPS_READ),
               ops_arg_gbl(&delta, 1, "double", OPS_READ));
  ops_tridMultiDimBatch_Inc(spaceDim, 0, tridSize, a, b, c, d, ux, trid);
  delete[] iterRange;
  delete[] tridSize;
}
```

```
for (int stage = 0; stage <= 2; stage++) {
    CompactDifference4thX1st(opensbliblock00, phi_B0, a, b, c, d, wk0_B0,
                             trid_at_opensbliblock000, Delta0block0, 0);
    int iteration_range_4_block0[] = {0, block0np0};
    ops_par_loop(opensbliblock00Kernel004, "Convective residual ",
                 opensbliblock00, 1, iteration_range_4_block0,
                 ops_arg_dat(wk0_B0, 1, stencil_0_00, "double", OPS_READ),
                 ops_arg_dat(Residual_TimeDer_phi_B0_t_B0, 1, stencil_0_00,
                             "double", OPS_WRITE));

    int iteration_range_9_block0[] = {0, block0np0};
    ops_par_loop(
        opensbliblock00Kernel009, "Sub stage advancement", opensbliblock00, 1,
        iteration_range_9_block0,
        ops_arg_dat(Residual_TimeDer_phi_B0_t_B0, 1, stencil_0_00, "double",
                    OPS_READ),
        ops_arg_dat(phi_RKold_B0, 1, stencil_0_00, "double", OPS_READ),
        ops_arg_dat(phi_B0, 1, stencil_0_00, "double", OPS_WRITE),
        ops_arg_gbl(&rknew[stage], 1, "double", OPS_READ));

    int iteration_range_8_block0[] = {0, block0np0};
    ops_par_loop(opensbliblock00Kernel008, "Temporal solution advancement",
                 opensbliblock00, 1, iteration_range_8_block0,
                 ops_arg_dat(Residual_TimeDer_phi_B0_t_B0, 1, stencil_0_00,
                             "double", OPS_READ),
                 ops_arg_dat(phi_RKold_B0, 1, stencil_0_00, "double", OPS_RW),
                 ops_arg_gbl(&rkold[stage], 1, "double", OPS_READ));

    ops_halo_transfer(exchange5_block0);
    ops_halo_transfer(exchange6_block0);
  }
```

Slice capability for OPS and OpenSBLI

Science and Technology Facilities Council
Scientific Computing

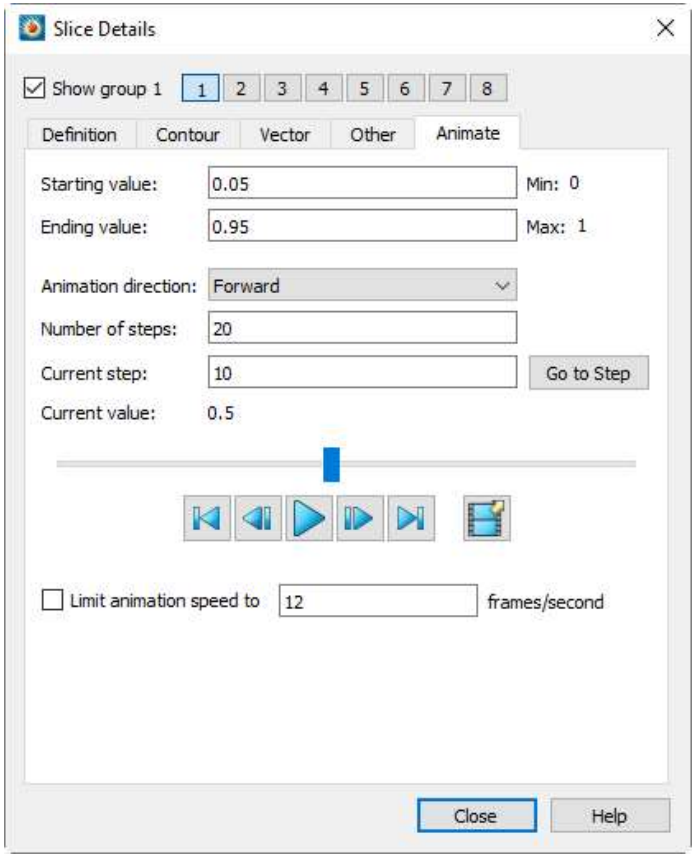
Motivation

- Always writing out 3D data is very expensive
- Often we would like to focus at a few planes
- Users from NASA, JAXA… requested

Desired capabilities

- Writing out a plane (say I=20)
- Writing out arbitrary selection of data
- Organised the slices in a specified structure

HDF5 files can be directly imported into Matlab, Python (h5py), Julia(HDF5), TecPlot, Visit and Paraview with a little more works (or convert to VTK)



Slice capability for OPS and OpenSBLI

Science and Technology Facilities Council
Scientific Computing

```
void ops_write_plane_hdf5(const ops_dat dat, const int cross_section_dir,
                          const int pos, char const *file_name,
                          const char *data_name);
```

Writ data of a plane specified by cross_section_dir, pos
data_name="block/time/rho" – HDF5 convention for making data structure where data are stored under block and time group
data_name="block_time_rho" – data stored at the root
Data dimension will be reduced by one

```
void ops_write_data_slab_hdf5(const ops_dat dat, const int
                              const char *file_name, const
```

Write data of an arbitrary slab specified by the range array

```
void ops_write_plane_group_hdf5(
    const std::vector<std::pair<int, int>> &planes, const s
    const std::vector<std::vector<ops_dat>> &data_list);

ops_write_plane_group_hdf5({{1, 16}, {0, 1}, {2, 16}}, "2",
                           {{u, v}, {u, v}, {u, v}});
```

```
1  I1=h5open("I1.h5","r")
[6]  0.3s
HDF5.File: (read-only) I1.h5
slice3Du
  1
    u
  2
    u
slice3Dv
  1
    v
  2
    v
```

Slice capability for OPS and OpenSBLI

Science and Technology Facilities Council
Scientific Computing

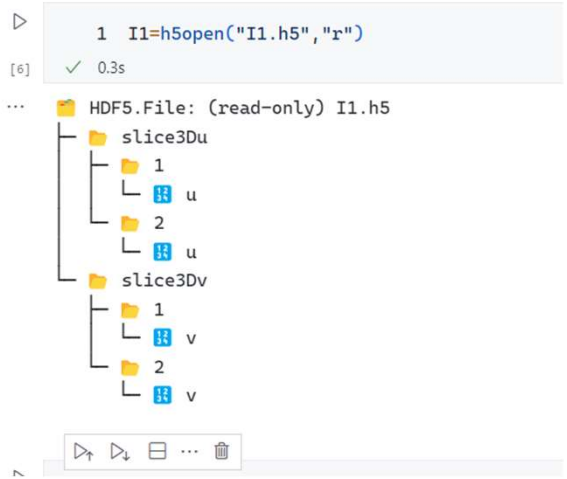
Archer2
Two-block mini-application
1024^3
Block 1: slice3Du
Block 2: slice3Dv

	128	256	512
1	8.94	14.49	21.87
2	9.6	17.97	20.56
3	12.06	14.74	21.29
avg	10.2	15.73	21.24

	128	256	512
1	2.7	5.42	3.63
2	2.9	3.23	4.17
3	2.68	3.78	7.78
avg	2.76	4.14	5.19

Bede2
Two-block mini-application
512^3
Block 1: slice3Du
Block 2: slice3Dv

4 nodes with 16 GPUS (NVIDIA V100)
First: 0.78
Second: 0.52



Concluding remarks

Science and Technology Facilities Council
Scientific Computing

- Lead to shorter source code for the TGV case
- The source code for the TGV case looks correct by comparing with the central differencing one.
- The implementation of slice HDF5 output
- Testing more cases and looking at performance including the ARM system

Thanks and Questions?