NESO: NEPTUNE EXPLORATORY SOFTWARE

James Cook

NEPTUNE Workshop

Cosener's House, Abingdon, 5-6 Sept. 2022

UK Atomic Energy Authority

NESO: High level goals

Neptune Exploratory SOftware

Bring together key technologies as foundational building blocks for Neptune in the future

- Nektar++ for its high-p DG/CG capabilities to solve generic PDEs
- NESO-Particles for SYCL-enabled charged and neutral particles
- NESO brings together Nektar++ and NESO-Particles as a mini-app framework for solving the 2D3V electrostatic particle-in-cell (PIC) problem, as a first step.

- This is a first non-trivial physics example with this set of capabilities.

NESO: Status

Neptune Exploratory SOftware

http://en.wikipedia.org/wiki/Neso_(moon)

http://github.com/ExCALIBUR-NEPTUNE/NESO

- Prototype NEPTUNE code
  - Fluid and/or particles (currently solves 1+1D Vlasov-Poisson using a PIC approach)
  - Implemented in SYCL (both Intel oneAPI and hipSYCL)
  - Can be used as a library of routines for treating particles, and can be built against Nektar++
- Plan: solve plasma system with particles by Q1 2023
  - Spectral/hp on unstructured 2D grids, plus particles in 3D
  - Nektar++ to provide Spectral/hp solvers, grids, challenge is to interface particles with finite elements
- Lessons learnt so far:
  - SYCL is usable, if incomplete (e.g. no `sum_allreduce` onto a vector, needed for depositing particles on grid)
  - “Performance portable” paradigms aren’t portable yet…
    - Toolchains are fragile, implementations are immature, standards are a moving target…
    - Can run on CPUs, AMD GPUs

Particle-in-cell codes

- On the fastest timescales of MCF plasmas, the particles are considered collisionless.
- We must resolve these fast plasma oscillations and gyro-motion to accurately resolve the region close to the divertor.
- 3D or 2D axisymmetric fluid models do not capture these effects so we must represent that velocity distributions of various species at each location in configuration space, which at the most complete description contains the 3 components of the velocity vector, meaning we end up with a 3D3V or axisymmetric 2D3V problem.
- Particles can be used as a computationally inexpensive way of representing the velocity space at the expense of particle noise.

- If one can write a PIC code then the machinery is there for a neutrals code*

*Subject to caveats from Will Saunders' talk.

PIC-code as a proxy-app

What does it gives us?

Experience with:

1. High order hybridised-DG & CG Poisson solve
2. GPU enabled particles
3. Interactions between particles on the GPU and fields on the CPU
4. Semi-implicit schemes.

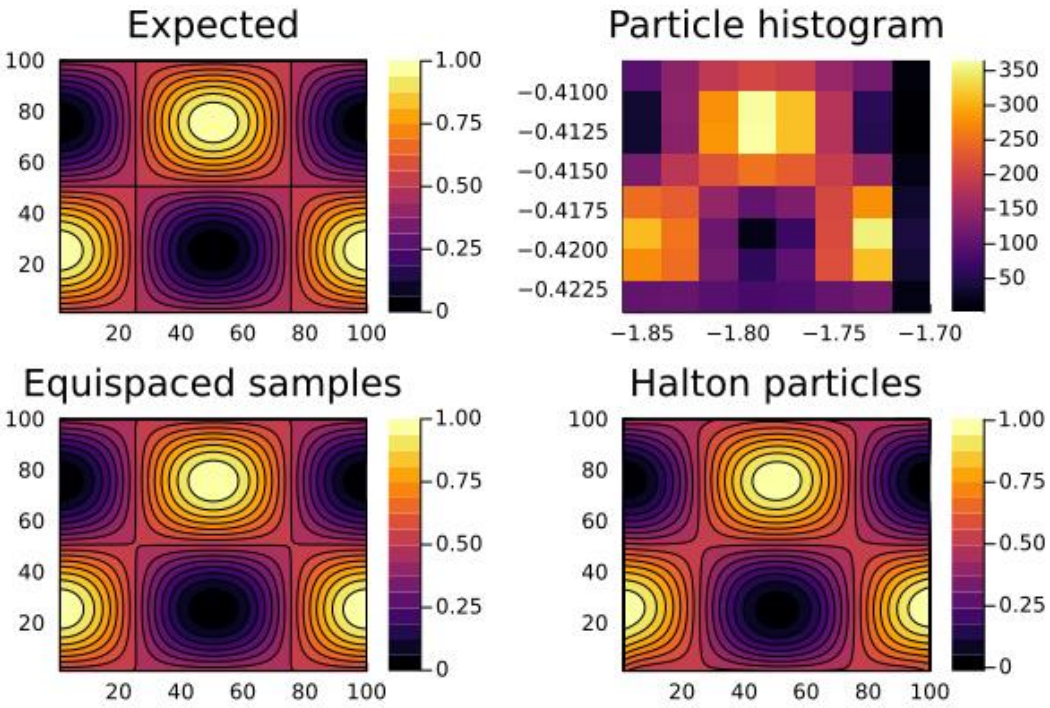
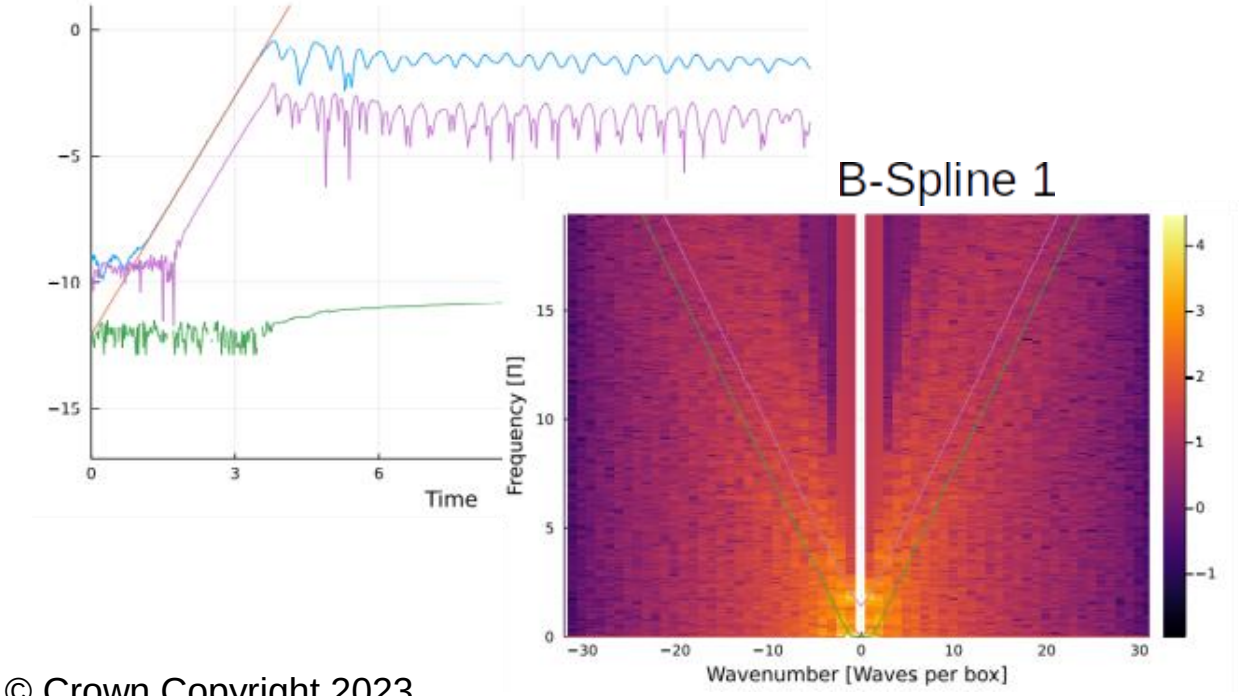
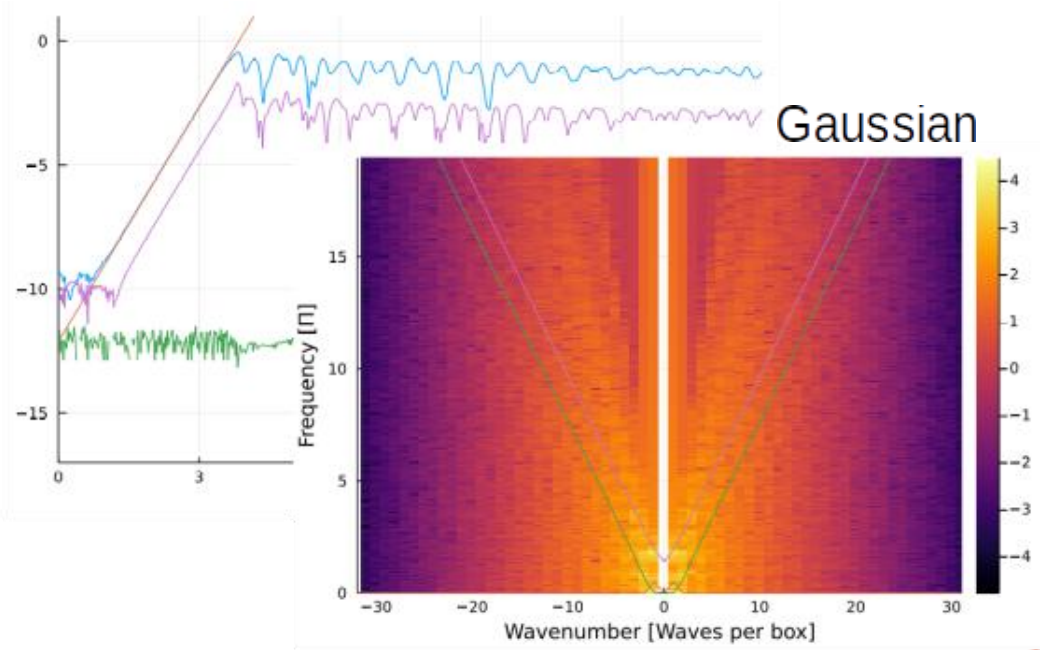


Figure:
Top left: expected function in a cell.
Lower left: reconstructed function using equi-spaced particles
Top right: particle histogram with big bins!
Lower right: reconstructed function using particles with Halton sequence positions

Particles & high order finite element basis functions.

- I implemented a 1D electrostatic PIC-(LS)FEM solver with 1st and 2nd degree b-splines and Gaussians as basis functions to check whether noise can be reduced adequately
- Good news! High wavenumber noise is significantly reduced with fields represented by high order basis functions.
- Expected news! The noise is terrible (destroys the physics) for lowest order field basis functions.





The NESO Repositories

Software development

NESO Public

C++ MIT 0 1 33 3 Updated 6 days ago

NESO-Particles Public

C++ MIT 1 0 0 0 Updated 9 days ago

NESO-Spack Public

Spack repository for installing NESO components and dependencies.

Python 0 0 0 0 Updated 2 hours ago

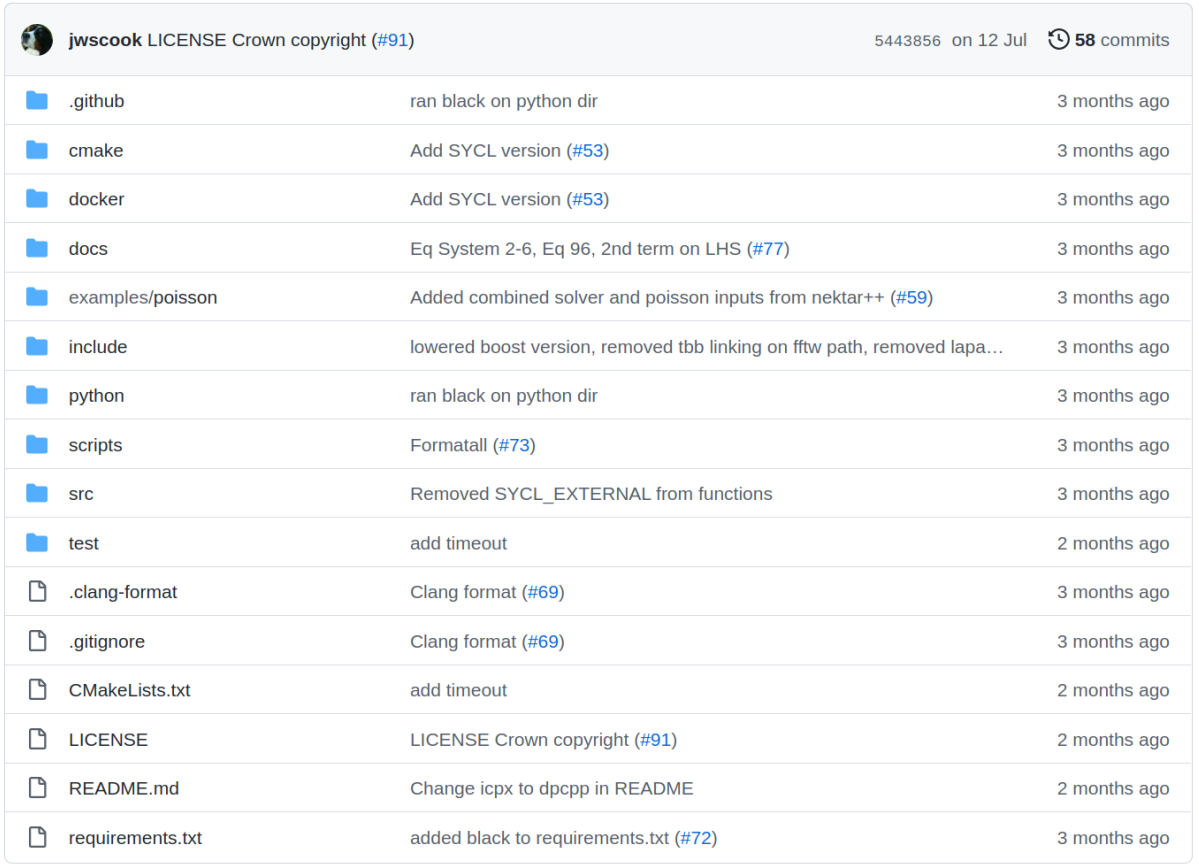
TODO: Use git submodules (or similar) to coordinate these repos

NESO

Repo contents

Plenty of standard stuff:

- cmake
- gtest
- github actions (TODO: nektar++ & SYCL Docker image)
- clang-format
- requirements.txt
- src with NESO source code and the nektar++ related code that we will maintain
- test code



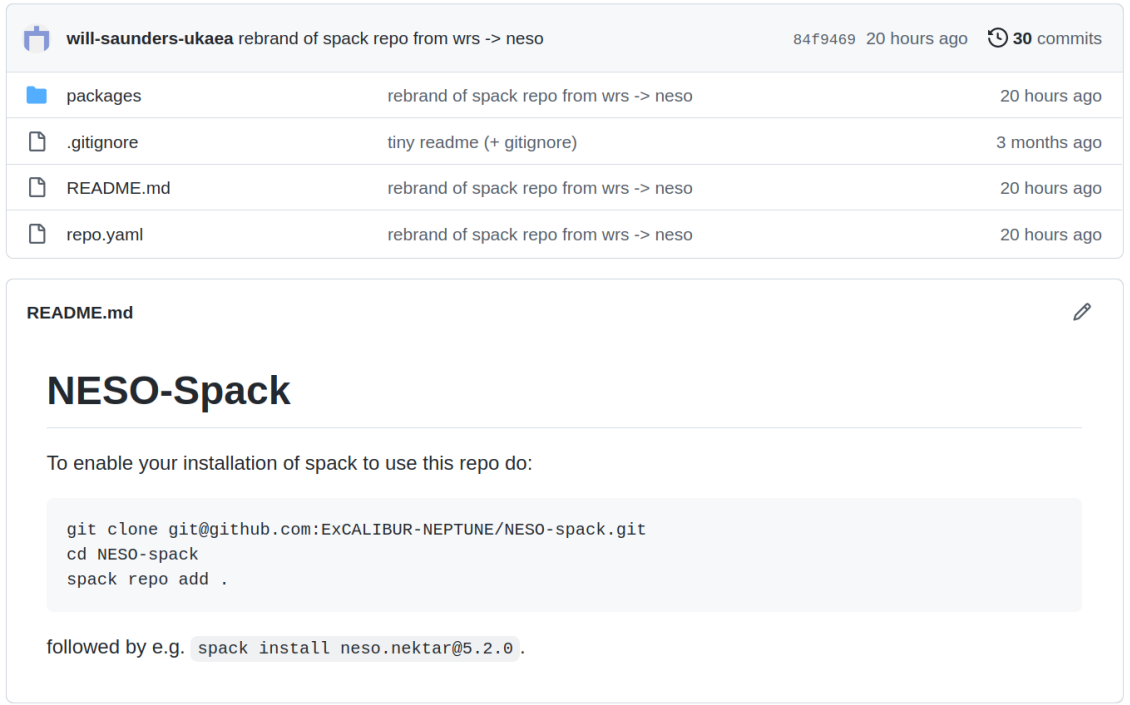
jwscook LICENSE Crown copyright (#91)	5443856 on 12 Jul	**58** commits
.github	ran black on python dir	3 months ago
cmake	Add SYCL version (#53)	3 months ago
docker	Add SYCL version (#53)	3 months ago
docs	Eq System 2-6, Eq 96, 2nd term on LHS (#77)	3 months ago
examples/poisson	Added combined solver and poisson inputs from nektar++ (#59)	3 months ago
include	lowered boost version, removed tbb linking on fftw path, removed lapa…	3 months ago
python	ran black on python dir	3 months ago
scripts	Formatall (#73)	3 months ago
src	Removed SYCL_EXTERNAL from functions	3 months ago
test	add timeout	2 months ago
.clang-format	Clang format (#69)	3 months ago
.gitignore	Clang format (#69)	3 months ago
CMakeLists.txt	add timeout	2 months ago
LICENSE	LICENSE Crown copyright (#91)	2 months ago
README.md	Change icpx to dpcpp in README	2 months ago
requirements.txt	added black to requirements.txt (#72)	3 months ago

NESO-Spack

Building dependencies from scratch

Users are able to build nektar++ against:

- Intel's oneapiSYCL implementation
- AMD's hipsycl SYCL implementation
- particular compilers, e.g.
- gcc@11.2.0
    - oneapi@2022.1.0 (i.e. dpcpp / icpx)
- python for NekPy
- your choice of MPI e.g. mpich



One can make a module file; `module load neso-hipsycl`

```
spack install --keep-stage -j 4 neso.nektar@5.2.0-f1598d ^python ^mpich %gcc@11.2.0
spack install --keep-stage -j 4 neso.hipsycl@0.9.2 %gcc@11.2.0
```

```
spack install --keep-stage -j 4 neso.nektar@5.2.0-f1598d +mkl %oneapi@2022.1.0 ^intel-
oneapi-mpi %oneapi@2022.1.0 ^intel-oneapi-mkl %oneapi@2022.1.0 ^python
```

NESO TODO

There's a lot to do

- Continue implement particle-field interactions
- Remove dependence on xml for configuration
- Introduce vector basis functions
- Marry up UI/UX of particle kernels with field configuration for users
- Build new operators in Nektar++ e.g. directional derivatives
- Retire our FFT solve in favour of Nektar++ solves
- Documentation
- Make the build process SYCL, Nektar++ and dependencies more robust and portable
- Performance testing of Particles and Nektar++
- Collision operators
- Incorporate neutral particles
- Particle diagnostics
- Visualisation
- A proper DSL for user interaction
- Python wrapping

FIN

UKAEA NEPTUNE:

Rob Akers
Wayne Arter
Matthew Barton
James Cook
Joseph Parker
Owen Parry
Will Saunders
Ed Threlfall

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.