NEPTUNE :
FINITE ELEMENT MODELLING @ UKAEA

Owen Parry, UKAEA

Will Saunders, UKAEA

Ed Threlfall, UKAEA

NEPTUNE Workshop

5-6 September 2022

UK Atomic Energy Authority

FEM in NEPTUNE

Summary of UKAEA activities

1. 1D scrape-off layer modelling with finite elements
   (Will Saunders, Owen Parry)

2. Nektar++ changes for NEPTUNE integration
   (Owen Parry)

3. Finite element exterior calculus
   (Ed Threlfall)

1D SOL Proxyapp

Model description

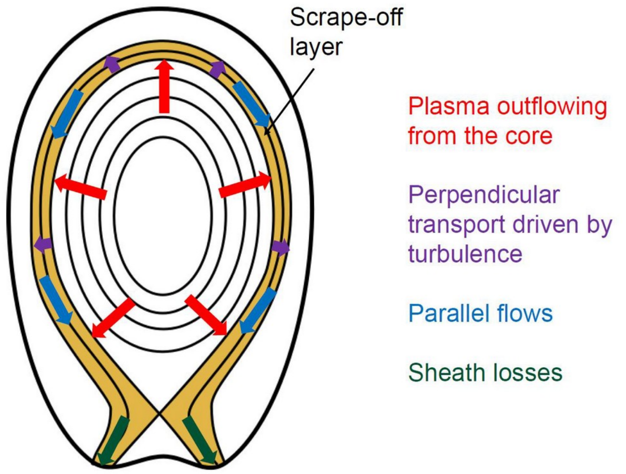
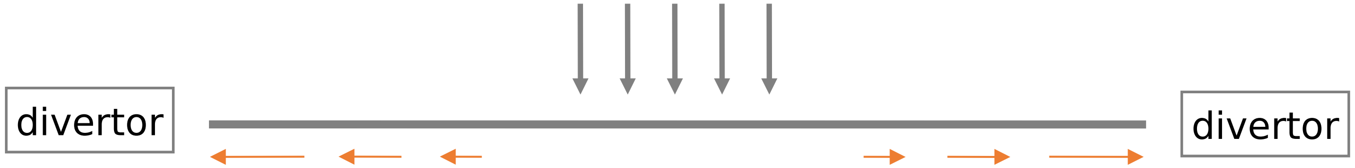


Image from Riva, 2019



$$U_r \frac{\partial n}{\partial t} = -\frac{\partial}{\partial s}(nu) + S^n,$$

$$U_r \frac{\partial}{\partial t}(nu) = -\frac{\partial}{\partial s}(nu^2) - \frac{\partial}{\partial s}(nT) + S^u,$$

$$U_r \frac{\partial}{\partial t}\left((g-2)nT + nu^2\right) = -\frac{\partial}{\partial s}(gnuT + nu^3) + \kappa_d \frac{\partial^2 T}{\partial s^2} + S^E,$$

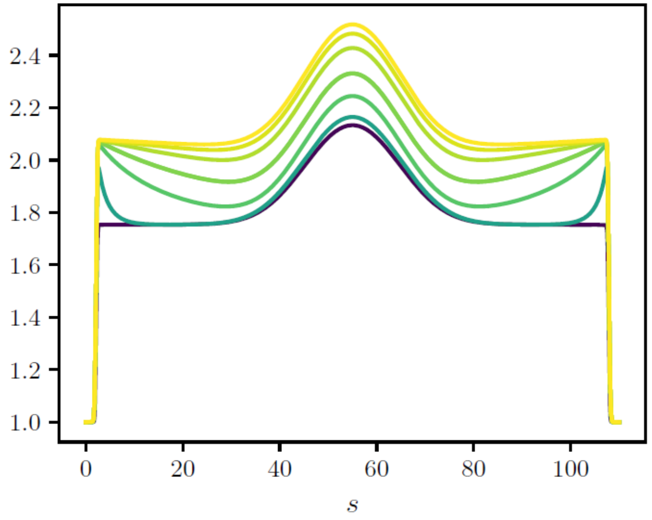
g: Specific heat capacity
κ_d: Conduction coefficient

- Simple, 1D model of the tokamak scrape-off layer
- Flux tube aligned with a magnetic field line
- Mass deposited into the domain by perpendicular transport
- Single fluid of electrons and ions
- Sonic outflow at either end to approximate sheath BCs
- No neutrals

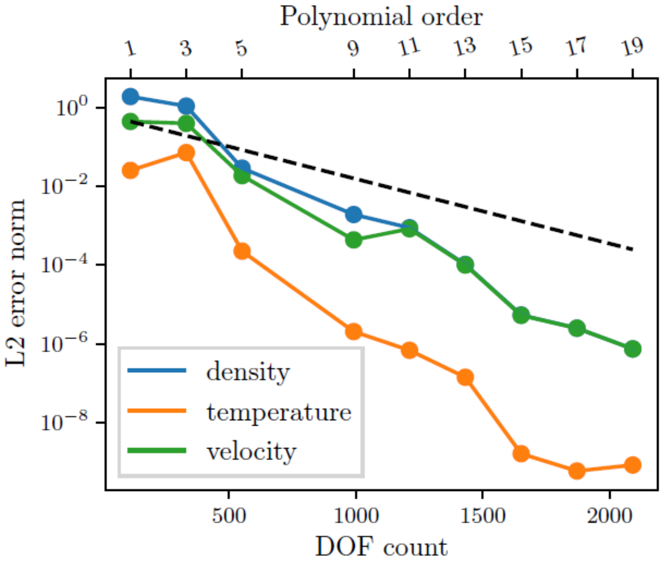
1D SOL Proxyapp

Soldrake (Will Saunders)

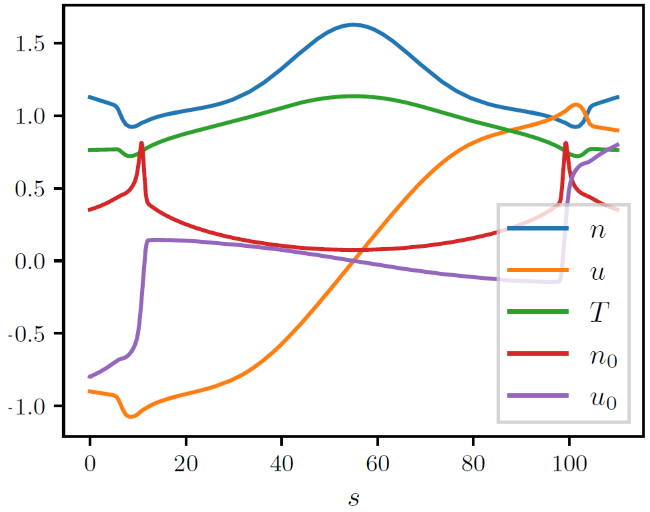
- *Firedrake* used to implement equations in weak form with Unified Form Language (UFL)
- Continuous Lagrange basis functions
- *Irksome* used for (Runge-Kutta) time integration
- Evolve to near equilibrium from ICs, add small amount of diffusion to stabilise.
- Use result as initial guess fc … ed



Varying conduction



Error as a function of N_{DOFs}



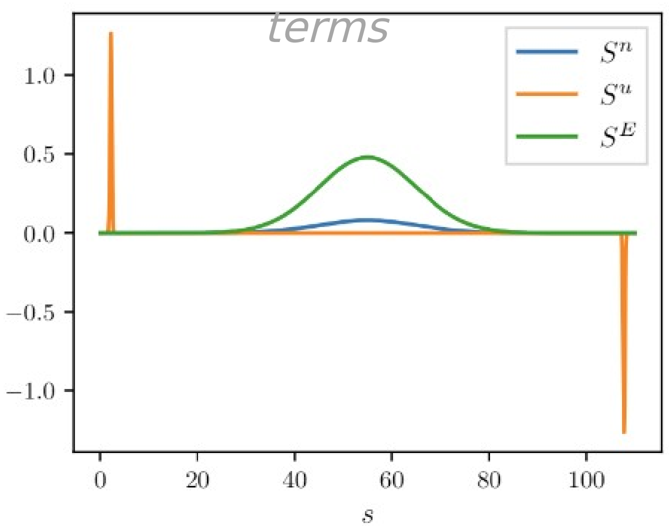
Including neutrals

[75] W. Saunders and E. Threlfall. *Finite Element Models: Performance*. Tech. rep. CD/EXCALIBUR-FMS/0047-M2.2.2. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0047-M2.2.2.pdf. UKAEA, 2021.

1D SOL Proxyapp

Nektar++ solver

- Nektar++ version of the same solver, implemented by Dave Moxey with support from OP
- Choose constants such that the problem reduces to solving incompressible Euler equations
- Discretisation: Discontinuous Galerkin
- Time evolution: 4th order Runge-Kutta
- to match Soldrake implementation



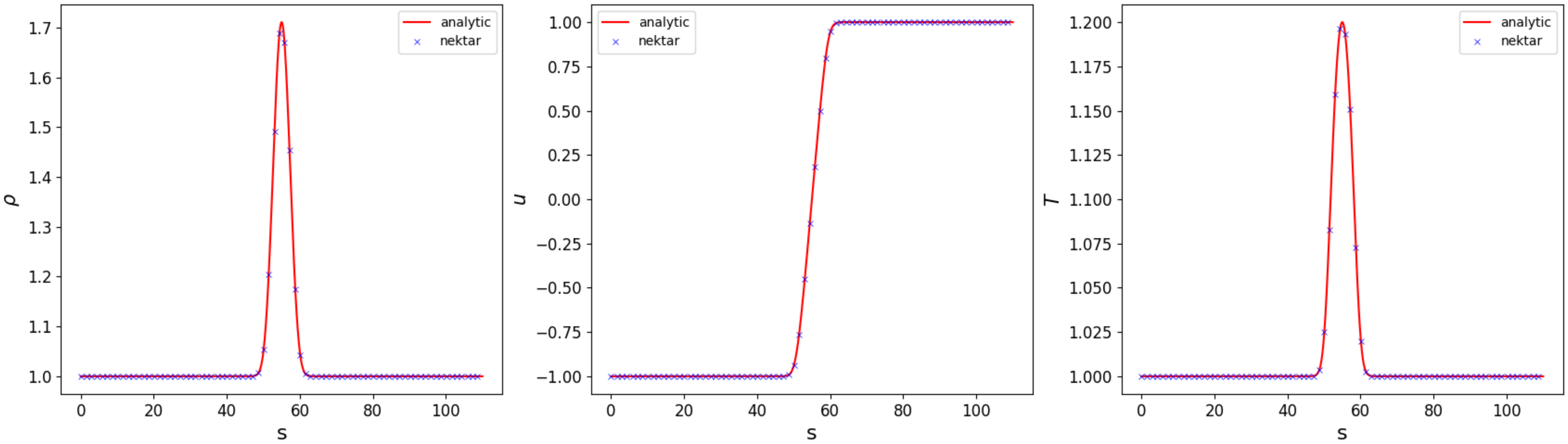
New Nektar++ solver

- Define equation system, based on existing advection solver
- Specify desired source terms via Forcing function framework
- Configure and run solver on simple 1D domain

Code at https://github.com/ExCALIBUR-NEPTUNE/nektar-1d-sol

1D SOL Proxyapp

Nektar++ results



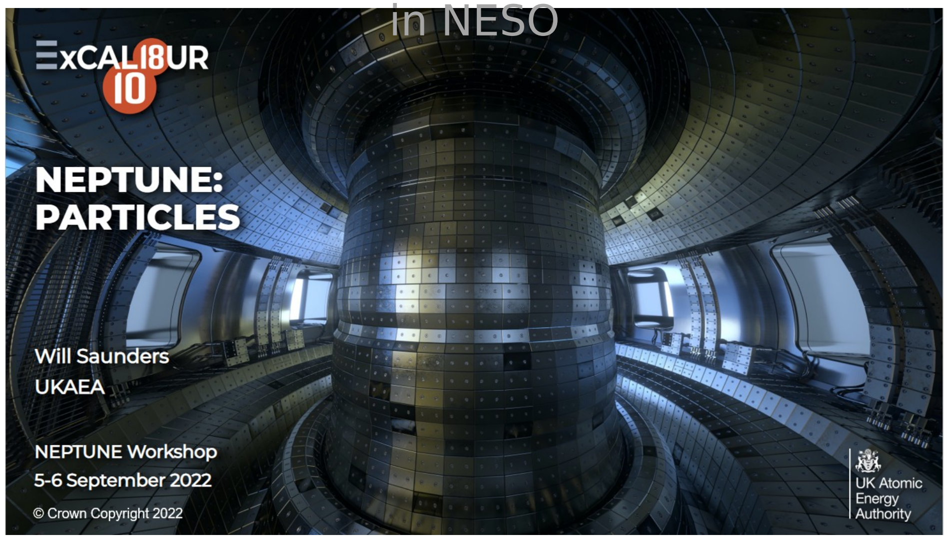
- Good agreement with analytical solution presented by Arter (and with Soldrake)

Wayne Arter. Study of source terms in the SOLF1D edge code. Technical report, Eurofusion/CCFE, 2015. CCFE-DETACHMENT-RP2-Draft.

Nektar++ integration in NEPTUNE
NESO

- Nektar functionality already being used in NEPTUNE prototype code
- See *NESO* git repo at https://github.com/ExCALIBUR-NEPTUNE/NESO + Will and James' talks:

Coupling between PIC code, Nektar++ in NESO



NESO overview

Nektar++ integration in NEPTUNE

Changes to Nektar core libraries

tatus quo: define session in one or more XML files

XML session file

```xml
<?xml version="1.0" encoding="utf-8" ?>
<NEKTAR>
    <GEOMETRY DIM="1" SPACE="1">
        <VERTEX>
        <ELEMENT>
        <COMPOSITE>
        <DOMAIN>
    </GEOMETRY>

    <EXPANSIONS>

    <FORCING>

    <CONDITIONS>
        <PARAMETERS>
        <SOLVERINFO>
        <VARIABLES>
        <BOUNDARYREGIONS>
        <BOUNDARYCONDITIONS>
        <FUNCTION NAME="InitialConditions">
        <FUNCTION NAME="ExactSolution">
    </CONDITIONS>

</NEKTAR>
```

Solver code

Session, including mesh, read here

```cpp
SessionReaderSharedPtr session = SessionReader::CreateInstance(argc, argv);
MeshGraphSharedPtr mesh = MeshGraph::Read(session);
string driverName("Standard");
DriverSharedPtr driver = GetDriverFactory().CreateInstance(driverName, session, mesh);
driver->Execute();
session->Finalise();
```

Pass session filepath(s) to the solver at runtime:

```
my_solver.exe [path_to_session.xml]
```

Nektar++ integration in NEPTUNE

Changes to Nektar core libraries

```xml
<CONDITIONS>
    <PARAMETERS>
        <P> TimeStep       = 5e-4              </P>
        <P> NumSteps       = 1000000            </P>
        <P> IO_CheckSteps  = 10000              </P>
        <P> IO_InfoSteps   = 1000               </P>
        <P> Gamma          = 5.0/3.0              </P>
        <P> GasConstant    = 1.0               </P>
        <P> pInf           = 1.0               </P>
        <P> rhoInf         = 1.0               </P>
        <P> uInf           = 1.0               </P>
    </PARAMETERS>
    <SOLVERINFO>
        <I PROPERTY="EQTYPE"                VALUE="SOL"                  />
        <I PROPERTY="Projection"            VALUE="DisContinuous"        />
        <I PROPERTY="AdvectionType"         VALUE="WeakDG"               />
        <I PROPERTY="TimeIntegrationMethod" VALUE="ClassicalRungeKutta4" />
        <I PROPERTY="UpwindType"            VALUE="ExactToro"            />
    </SOLVERINFO>
    <VARIABLES>
        <V ID="0"> rho </V>
        <V ID="1"> rhou </V>
        <V ID="2"> E </V>
    </VARIABLES>
```

XML session configuration

```cpp
// Variable names
std::vector<std::string > varNames = { "rho", "rhou", "E" };

// Set params of type double
std::map<std::string, NekDouble> dblParams;
dblParams["Timestep"] = 5e-4;
dblParams["Gamma"] = 5.0/3;
dblParams["GasConstant"] = 1.0;
dblParams["pInf"] = 1.0;
dblParams["rhoInf"] = 1.0;
dblParams["uInf"] = 1.0;

// Set params of type int
std::map<std::string, int> intParams;
intParams["IO_CheckSteps"] = 10000;
intParams["IO_InfoSteps"] = 1000;
intParams["NumSteps"] = 1000000;

// Set solver properties
std::map<std::string, std::string> solverInfo;
solverInfo["EQTYPE"] = "SOL";
solverInfo["Projection"] = "DisContinuous";
solverInfo["AdvectionType"] = "WeakDG";
solverInfo["TimeIntegrationMethod"] = "ClassicalRungeKutta4";
solverInfo["UpwindType"] = "ExactToro";

// Set up session
LibUtilities::SessionConfigurableSharedPtr session;
session = LibUtilities::SessionConfigurable::CreateInstance(argc, argv,
                                        "sol1D", varNames, dblParams, intParams, solverInfo);
```

Programmatic session configuration

Finite element exterior calculus (FEEC)
(Work by Ed Threlfall)

- Framework for formulating finite element methods with useful numerical properties
- Developed by Douglas Arnold, and others, in early 2000s
- Combines ideas from homological algebra, functional analysis, exterior calculus

Can be used to ensure that a numerical method
- is naturally stable without adding artificial dissipation
- explicitly conserves certain quantities (e.g. electric charge)

Used in several major PIC codes: GEMPIC, EMPIRE

Numerous examples in the literature that illustrate FEEC utility; for instance...

Finite element exterior calculus (FEEC)

An example with Firedrake

Example from Douglas Arnold's book, run with Firedrake

D.N. Arnold, *Finite Element Exterior Calculus*, CBMS-NSF regional conference series in applied mathematics **93**, SIAM.

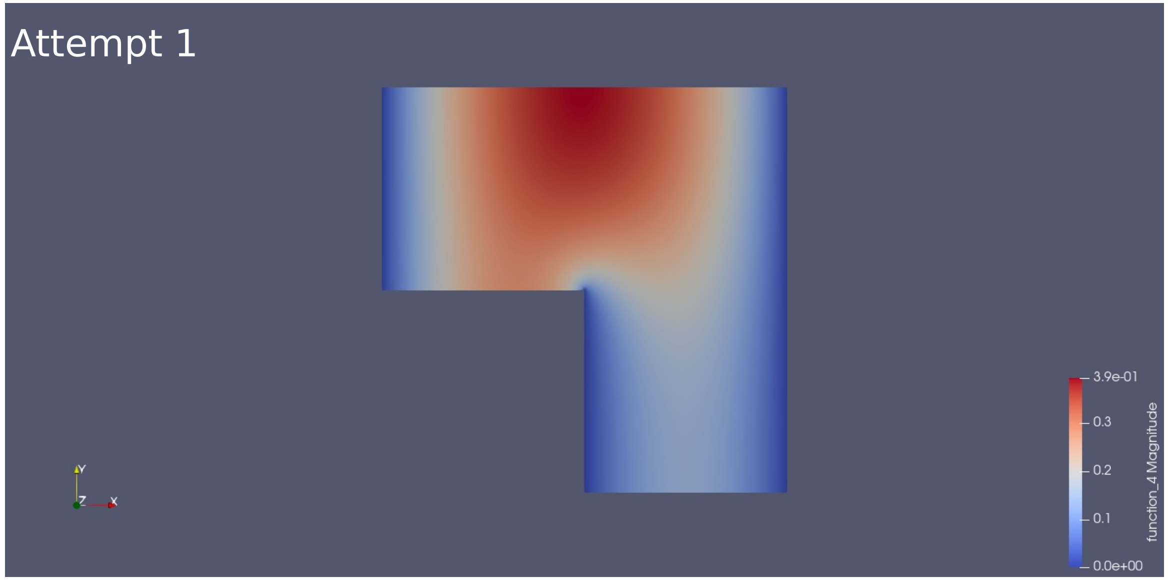
- "Re-entrant corner" problem:

  Solve $\nabla^2 \underline{u} = \underline{f}$ on an L-shaped domain with source on interior with $\underline{f} = (1,0)$ and $\underline{u} \cdot \underline{n} = \nabla \times \underline{u} = 0$ on the boundary.

Solution: singularity of the form

(r: distance from the corner)

6th order vector Lagrange FS for



Converged, smooth… wrong! Need *mixed* formulation

Finite element exterior calculus (FEEC)

An example with Firedrake

- "Re-entrant corner" problem:

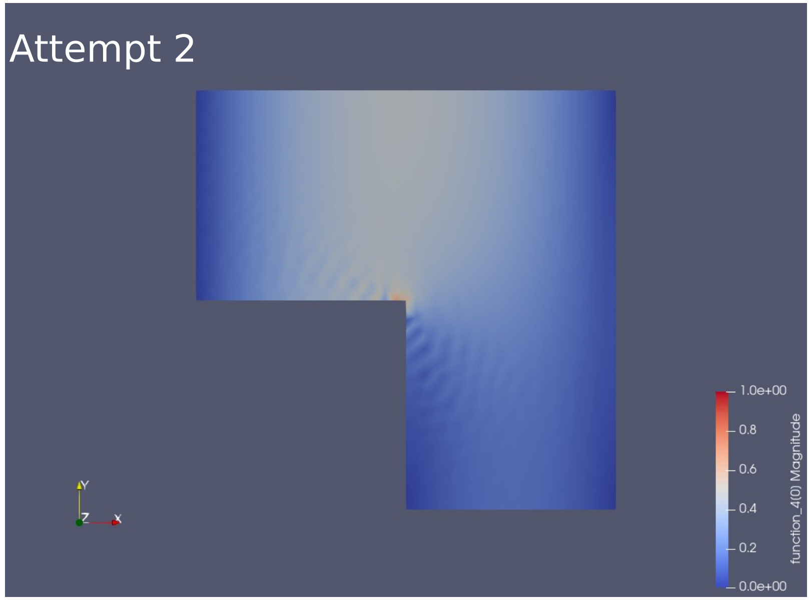
Solve $\nabla^2 \underline{u} = \underline{f}$ on an L-shaped domain with source on interior

with $\underline{f} = (1, 0)$ and $\underline{u} \cdot \underline{n} = \nabla \times \underline{u} = 0$ on the boundary.

Solution: singularity of the form

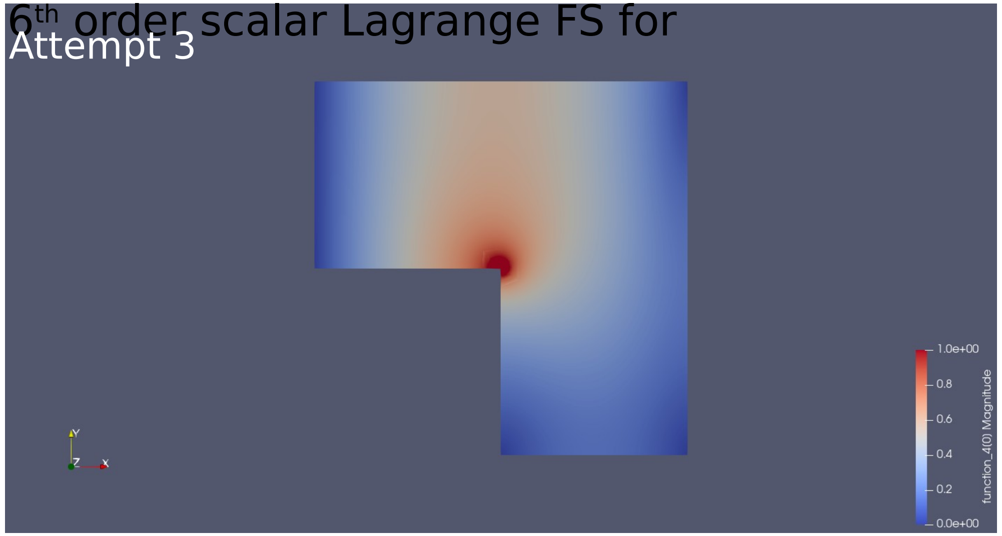
(r: distance from the corner)

6th order vector Lagrange FS for
6th order scalar Lagrange FS for



Spurious oscillations due to inconsistent choice of function spaces

6th order vector Raviart-Thomas "Edge" FS for
6th order scalar Lagrange FS for



Smooth, resolves singularity

ExCALIBUR

Finite element exterior calculus (FEEC)

Lessons

- FEEC framework can help guarantee stability and conservation in FEM approaches
- Principles are easily demonstrated in idealised examples, but apply equally to real-world modelling
- Wide range of element types available – need to study literature to make a sensible choice
- FEEC increasingly being applied in PIC codes – may well be an essential ingredient for NEPTUNE

Conclusion

Progress towards Nektar++/NEPTUNE integration

- Nektar++ already applied to several problems relevant to edge-plasma physics

- FEEC offers a promising avenue to improve solver stability, apply conservation laws in FEM

- FEM-related NEPTUNE development:

- PIC code / Nektar++ coupling: already started in NESO

- Nektar++ setup without XML: almost complete

- FEEC in Nektar++: future challenge