Integration of PETSc-DMPlex with Firedrake

Firedrake Firedrake is a Python-based framework for the performance portable solution of PDEs using the finite element method (FEM). It employs the Unified Form Language (UFL) and FEniCS Form Compiler (FFC) from the FEniCS Project and fields and meshes from Fluidity. The parallel execution of the FEM solver is accomplished by the PyOP2 system.

PETSc-DMPlex DMPlex is a set of data management objects and routines provided by the PETSc library that encapsulate the topology of unstructured meshes in an acyclic graph format. It provides a high-level API to create algebraic structures (Mat and Vec objects) from the given mesh topology and manages communication between them in PDE-based applications.

Project objectives The key objective of this work is the integration of PETSc's data management API for unstructured meshes (DMPlex) with the Firedrake framework for performance portable finite element computation. This integration will improve the interoperability and portability of Firedrake by allowing the framework to utilise PETSc's interfaces for various common mesh formats and removing legacy dependencies. It will also improve performance due the use of DMPlex's runtime mesh decomposition capabilities, which removes I/O bottlenecks due to reading mesh decompositions from disk.

- **Interoperability** - DMPlex supports multiple common meshing formats, such as CGNS and ExodusII, as well as providing an interface layer for the creation of custom mesh format readers. Utilising DMPlex as an abstraction layer for mesh topology information therefore significantly increases Firedrake's interoperability with other mesh-based applications and provides an efficient way of adding support for further meshing formats to Firedrake. Fluidity or other numerical codes covered by the Platform grant may also be integrated with the DMPlex interface to make them compatible with Firedrake.
- **Portability** - Firedrake currently utilises mesh handling capabilities of the CFD code Fluidity via legacy interfaces to Fortran code. Replacing this legacy dependency with DMPlex functionality significantly increases Firedrake's portability.
- **Scalable I/O** - DMPlex has been integrated with two mesh decomposition libraries (Metis and Chaco) allowing it to perform mesh decomposition at runtime. Utilising this capability voids the need for explicit domain decomposition during pre-processing and removes a significant scalability bottleneck due to reading halo information from disk. Furthermore, DMPlex objects are planned to support parallel reads and writes for mesh mesh topology and field data, providing scalable I/O in the near future.

Alignment with PRISM's strategy
Retention of key staff: This provides bridge funding for Dr Michael Lange. He has a strong computational science background and is an experienced PETSc developer. These skills will be required in many of the simulation tools being developed within PRISM.
Feasibility study: This is the first time outside the PETSc development team that anyone is attempting to integrate DMPlex into an application code. The success of this project could pave the way to achieving interoperability of data between all simulation software within PRISM.
Longer-term research: Following from the previous point, this is expected to greatly enhance software sustainability and productivity.
International networking: This work will be done in collaboration with PETSc developers at Argonne National Laboratory and University of Chicago. Dr Matt Knepleyi's visit to Imperial College circa March 2014 will support this collaboration.
Long term strategy: The aim is to identify world leading open source software which we can incorporate into our own research and technology development. This also presents exciting collaboration opportunities. In the case of PETSc we have formed a productive two way international collaboration with many adjunct possibilities.

Work plan The key objective of this work is the integration of DMPlex functionalities with Firedrake. Since both projects are under ongoing development, this task requires collaboration with the PETSc development team to communicate software engineering requirements as well as bug reports and patches for problems found during development.

- Creating `FunctionSpace` objects in Firedrake that derive topology information from DMPlex objects for simple meshes.
- Identifying the required DMPlex API calls and making them available to Firedrake's Python core via *petsc4py* wrappers (`https://bitbucket.org/petsc/petsc4py`).
- Utilise PETSc's API to perform the parallel halo exchange for fields derived from DMPlex-based `FunctionSpace` objects.
- Use DMPlex sub-meshing to represent boundary conditions and apply them to DMPlex-based fields.
- Integrate Firedrake's mesh extrusion algorithms with the DMPlex API.