Integrating Discrete Exterior Calculus into the ParaFEM library

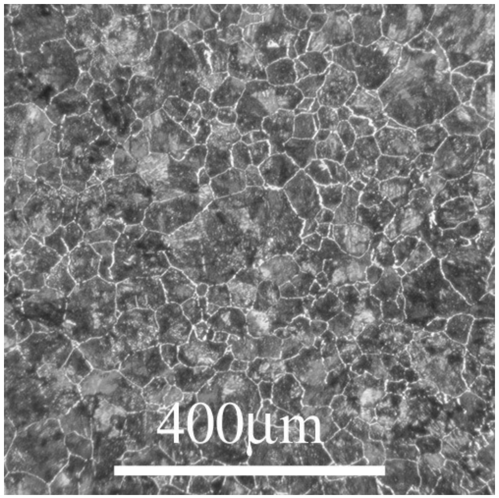
Leveraging 30+ years of development to accelerate new research

Pieter D. Boom; Andrey P. Jivkov; and Lee Margetts
Department of MACE, University of Manchester

January 13, 2022

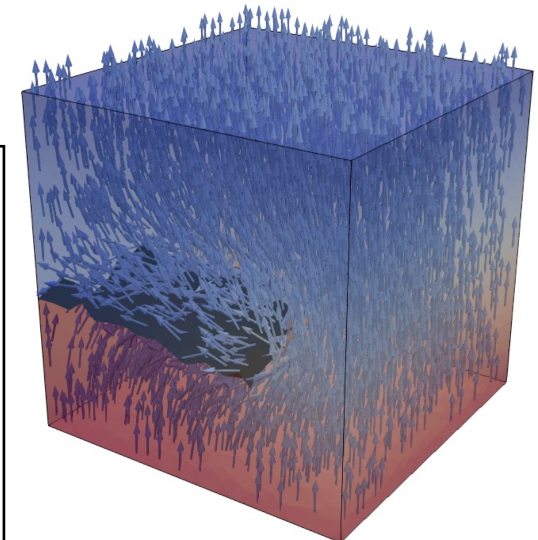
ARCHER2-eCSE01-12:ParaGEMS: Integrating discrete exterior calculus (DEC) into ParaFEM for geometric analysis of solid mechanics

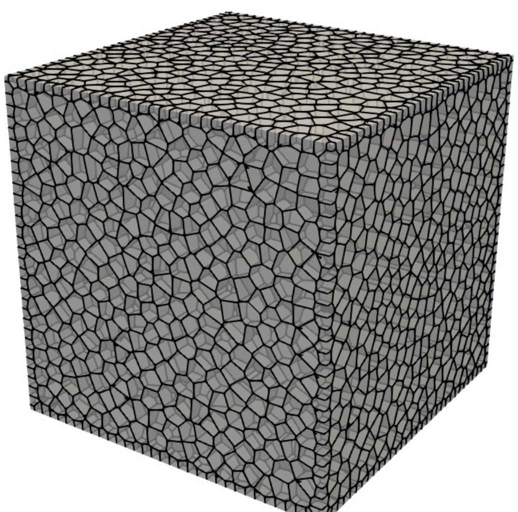
Scientific Aim

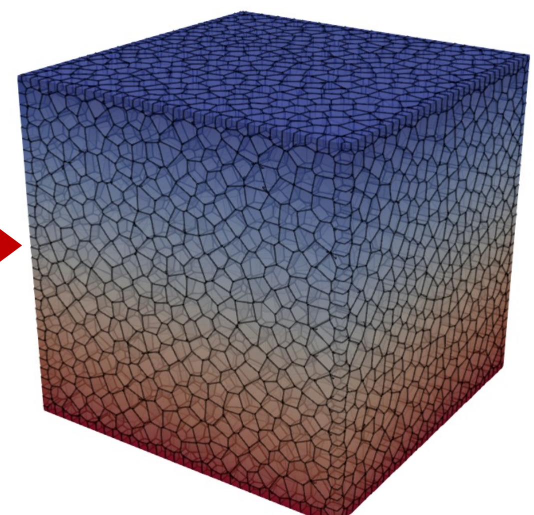


Numerical modelling of processes evolving on discrete mesoscale structures of materials :

- Heterogeneous or discontinuous
- Multi-dimensional
- Multiphysics



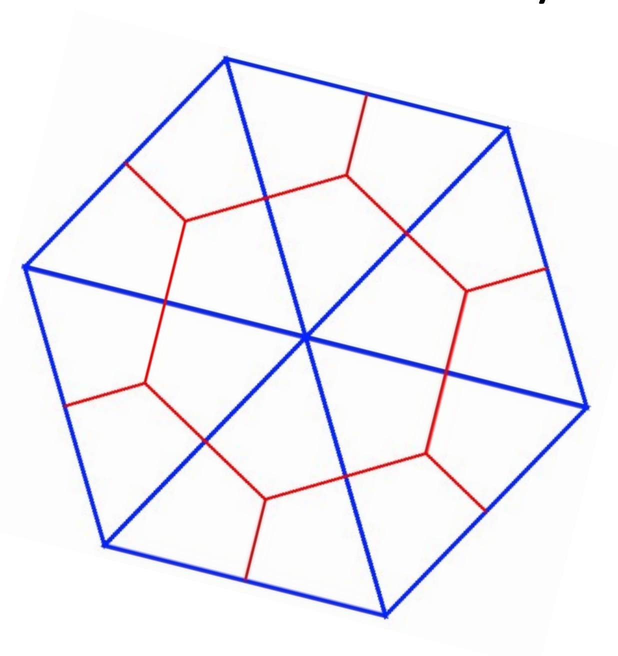


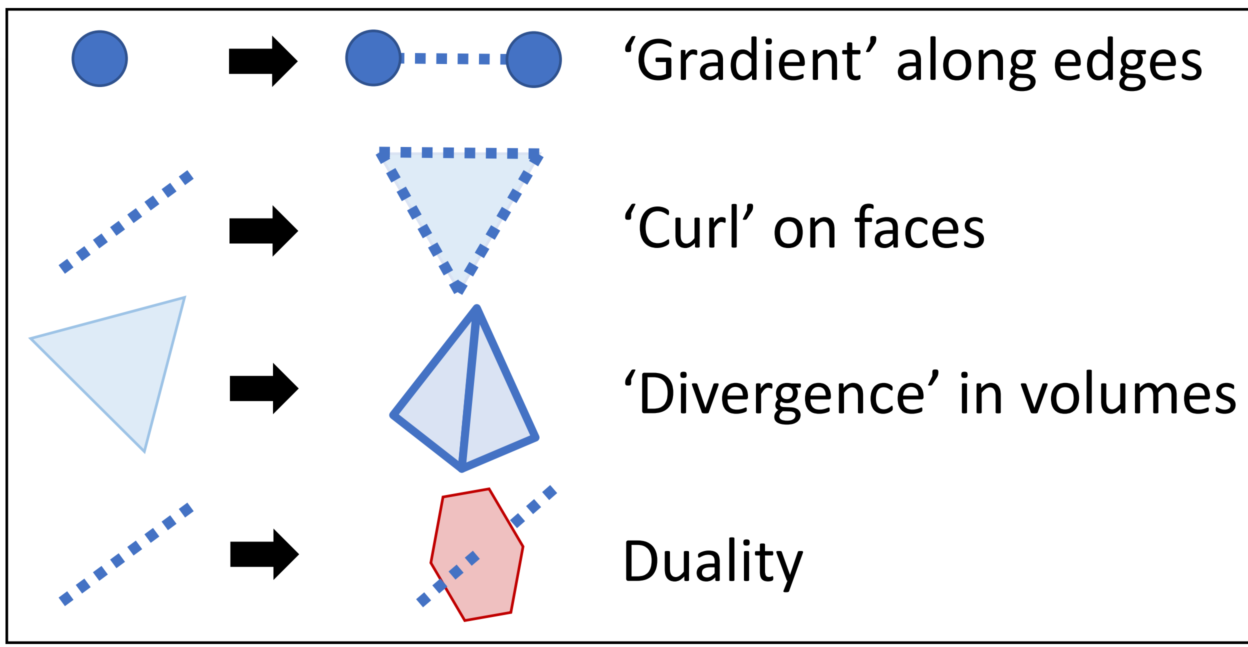


Discrete Exterior Calculus (DEC)

More generally: discrete forms of Exterior Calculus

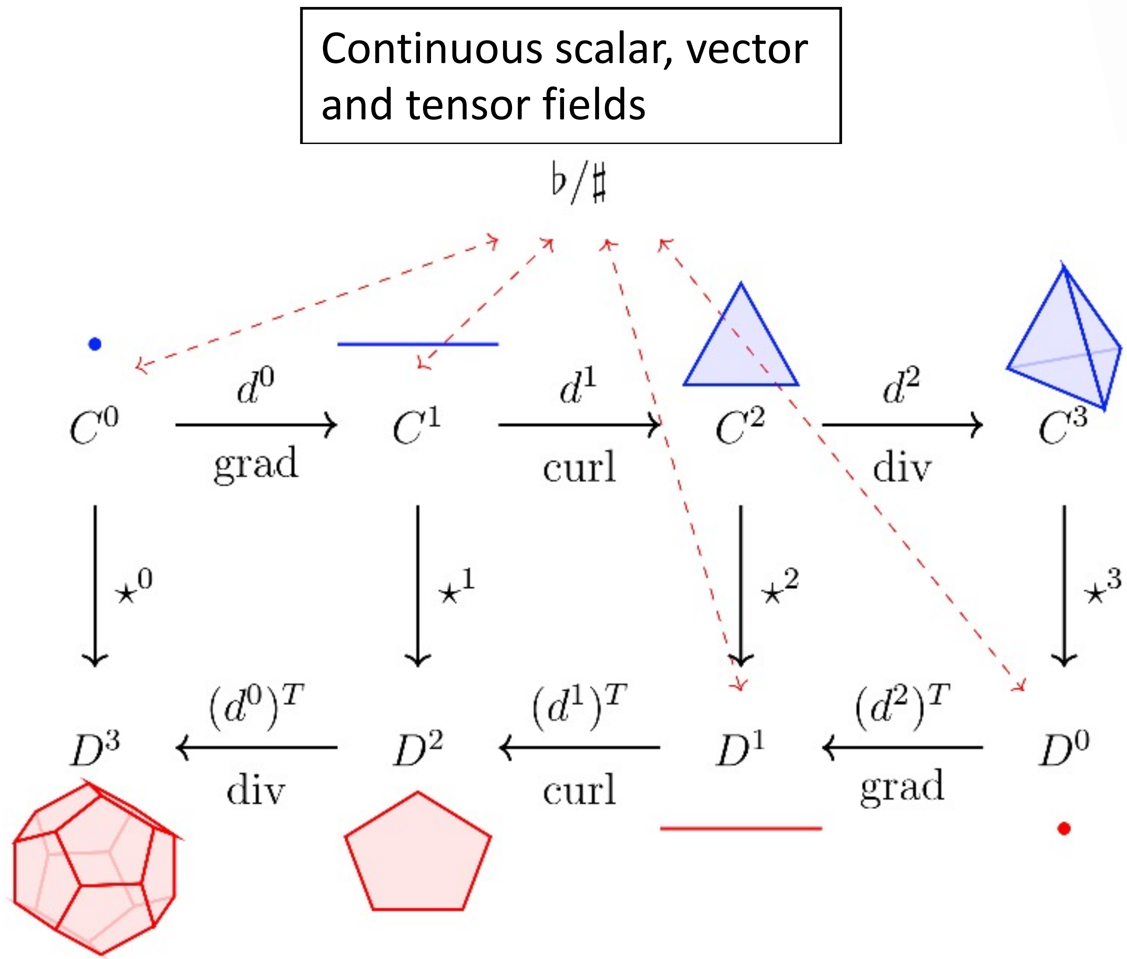
- Rather than discretising continuous fields, we want to develop fundamentally discrete representations of physical processes that are defined by the geometry and topology of the mesh

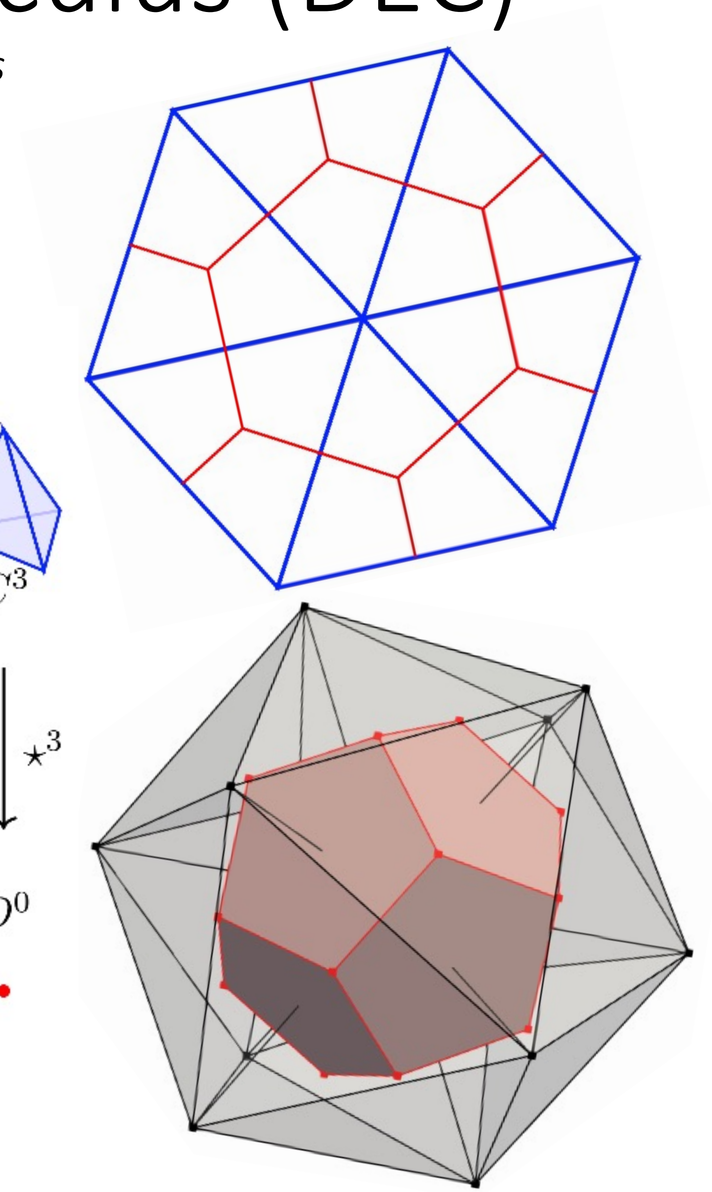




Discrete Exterior Calculus (DEC)

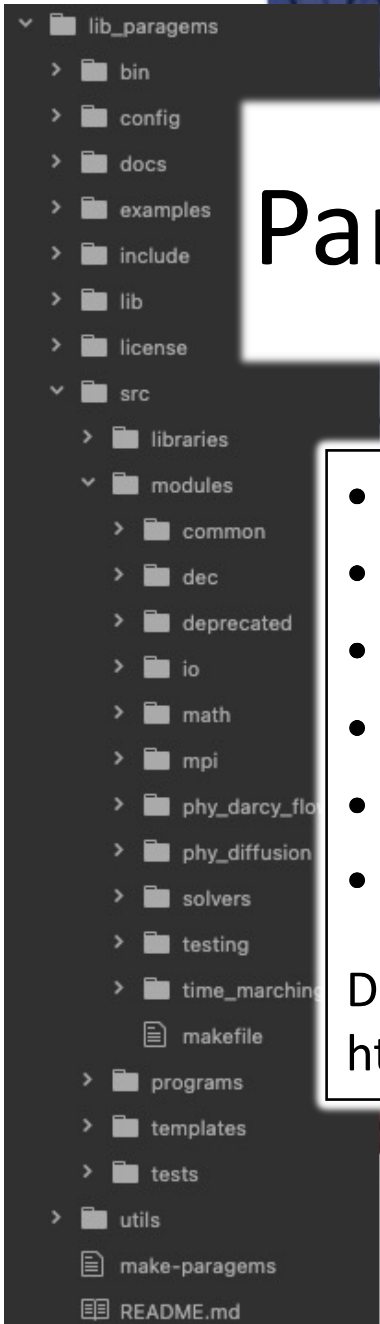
More generally: discrete forms of Exterior Calculus

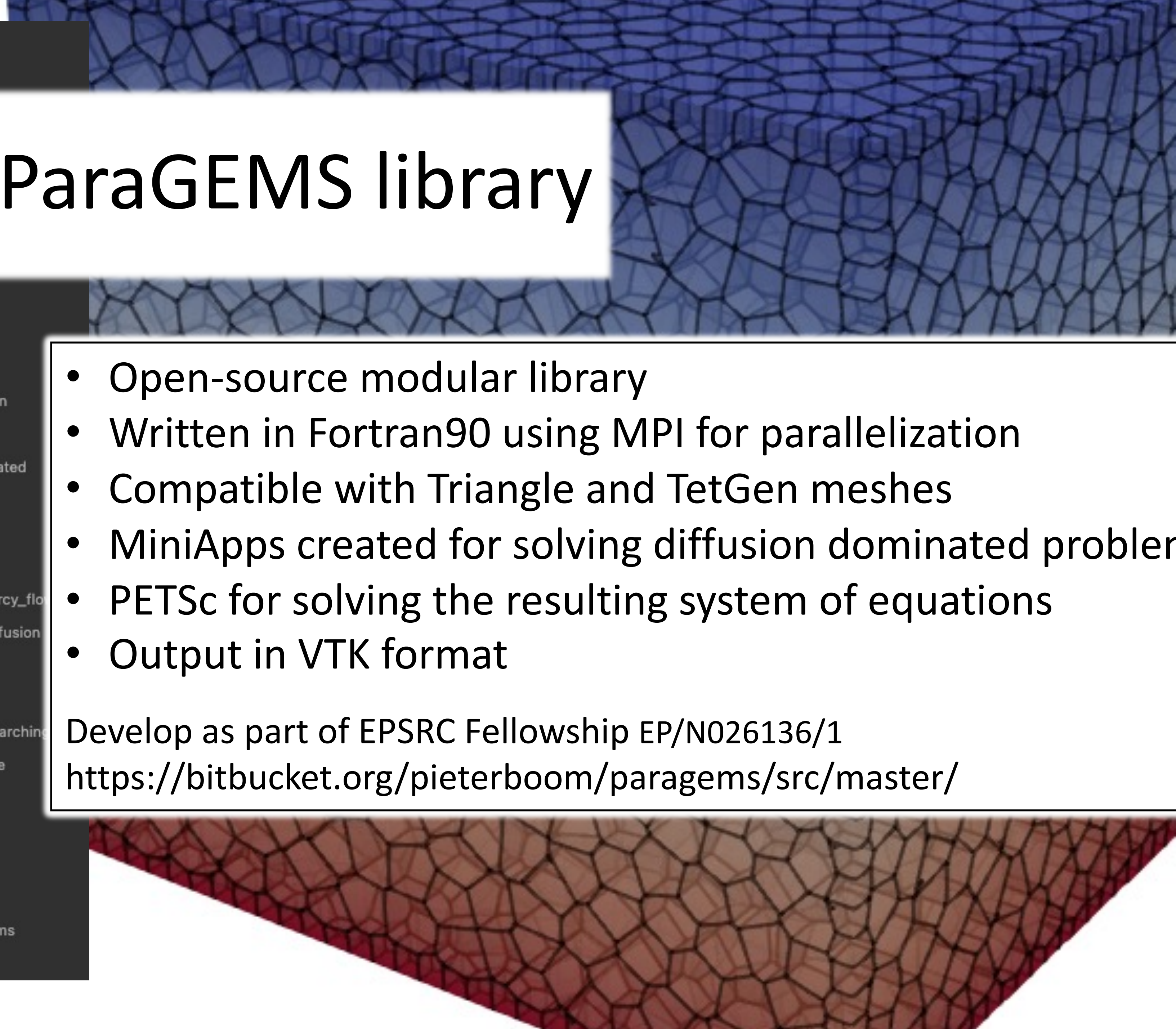




ARCHER2-eCSE Project Aims

1. **Integrate the parallelised DEC library (ParaGEMS) into the Finite-Element library ParaFEM to solve:**
   - Heat diffusion in pristine media (linear scalar);
   - Heat diffusion in cracked media (scalar discontinuous); and
   - Nonlinear elasticity (nonlinear/vector-valued)
2. **Improve the parallel scaling of the integrated libraries, targeting:**
   - 80-90% parallel efficiency on 64k; and
   - 70-80% parallel efficiency on 128k cores
3. **Implement hierarchical testing for software sustainability**
4. **Promoting the user and developer community on ARCHER2**
   - Public repositories, Software releases, Documentation, Tutorials, Seminars





ParaGEMS library

- Open-source modular library
- Written in Fortran90 using MPI for parallelization
- Compatible with Triangle and TetGen meshes
- MiniApps created for solving diffusion dominated probler
- PETSc for solving the resulting system of equations
- Output in VTK format

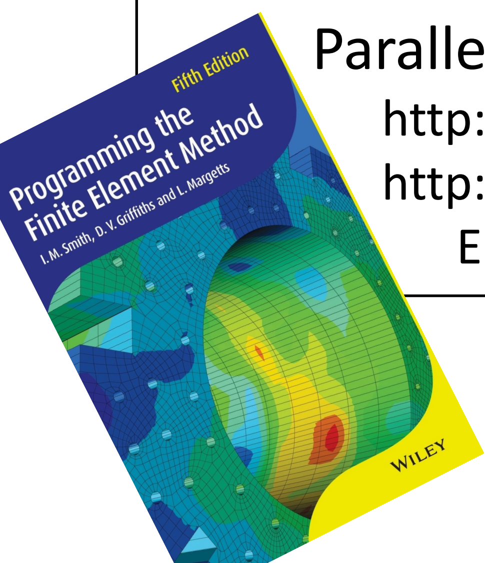
Develop as part of EPSRC Fellowship EP/N026136/1
https://bitbucket.org/pieterboom/paragems/src/master/

ParaFEM library

- Open-source modular library
- Written in Fortran90 using MPI for parallelization
- ~70 mini Apps for solving diverse physical problems:
  - Problems with >1 billion degrees of freedom
  - Using ~64,000 cores
- Many existing software and IO interfaces
- Textbook - Chapter 12:
  - Parallel computing, GPUs, Cloud computing

    http://parafem.org.uk

    http://www.amazon.com/Programming-Finite-Element-Method-Smith/dp/1119973341



Goose Femur

300,000,000 dof

Source: Zartasha Mustansar

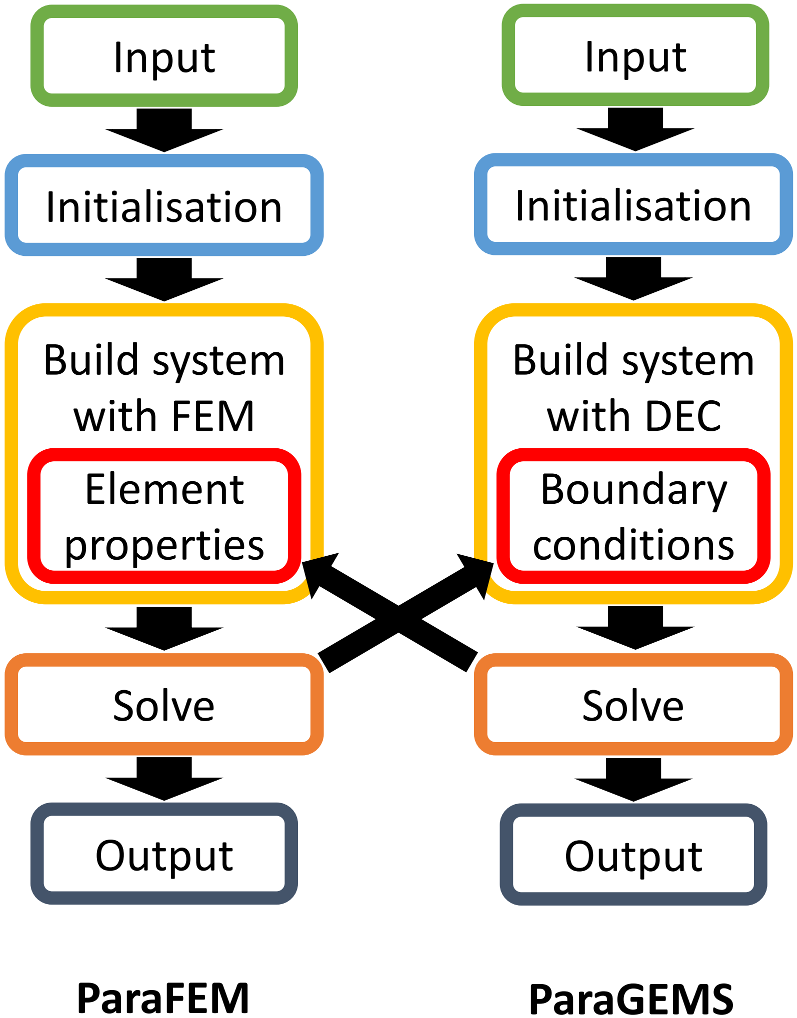
simpleware

http://www.simpleware.com

ARCHER2-eCSE Project Aims

1. **Integrate the parallelised DEC library (ParaGEMS) into the Finite-Element library ParaFEM to solve:**
   - Heat diffusion in pristine media (linear scalar);
   - Heat diffusion in cracked media (scalar discontinuous); and
   - Nonlinear elasticity (nonlinear/vector-valued)
2. **Improve the parallel scaling of the integrated libraries, targeting:**
   - 80-90% parallel efficiency on 64k; and
   - 70-80% parallel efficiency on 128k cores
3. **Implement hierarchical testing for software sustainability**
4. **Promoting the user and developer community on ARCHER2**
   - Public repositories, Software releases, Documentation, Tutorials, Seminars

Preliminary work on ParaGEMS



Preliminary work on ParaGEMS

- Porting to ARCHER2 with make files and compiling with PETSc+Hypre (also done for ParaFEM)

```
pboom@ln03:~/paragems/config> ls
archer2_amd.inc        csf.inc
archer2_cray.inc       mac.inc
archer2_gnu.inc        mapos1.inc
archer_old.inc         mk_defs.inc
archer_old_petsc.inc   template.inc
```

```
pboom@ln03:~/paragems/src/libraries/PETSC> ls
arch-linux-c-debug  gmakefile        make.log
buildsystem.log     gmakefile.test   petsc-3.14.0.tar.gz
CODE_OF_CONDUCT.md  GNUmakefile      petscdir.mk
config              include          RDict.log
configure           index.html       RDict.log.bkp
configure.log       interfaces       setup.py
configure.log.bkp   lib              share
CONTRIBUTING        LICENSE          src
CTAGS               makefile         systems
docs                makefile.html    TAGS
```

Preliminary work on ParaGEMS

- Porting to ARCHER2 with make files (also done for ParaFEM) and compiling with PETSc + Hypre

```
pboom@ln03:~/paragems/config> ls
archer2_amd.inc        csf.inc
archer2_cray.inc       mac.inc
archer2_gnu.inc        mapos1.inc
archer_old.inc         mk_defs.inc
archer_old_petsc.inc   template.inc
```

```
pboom@ln03:~/paragems/src/libraries/PETSC> ls
arch-linux-c-debug  gmakefile       make.log
buildsystem.log     gmakefile.test  petsc-3.14.0.tar.gz
CODE_OF_CONDUCT.md  GNUmakefile     petscdir.mk
config              include         RDict.log
configure           index.html      RDict.log.bkp
configure.log       interfaces      setup.py
configure.log.bkp   lib             share
CONTRIBUTING        LICENSE         src
CTAGS               makefile        systems
docs                makefile.html   TAGS
```

cse-develop · pe-scripts / sh / petsc.sh

kevinstratford Restore 3.13.3; repair fortran config

2 contributors

Executable File | 240 lines (227 sloc) | 6.95 KB

```sh
#!/bin/sh
#
# Build and install the PETSc library.
#
# Copyright 2019, 2020, 2021 Hewlett Packard Enterprise Development LP.
####

PACKAGE=petsc
VERSIONS='
  3.10.3:f03650ea5592313dd2b8be7ae9cc498369da660185b58f9e98689a9bc355e982
  3.10.5:6fa0574aebc6e6cb4eea206ef9a3a6037e20e8b54ca91346628a37f79af1407f
  3.11.4:006177b4059cd40310a3e9a4bf475f3a8c276b62d8cca4df272ef88bdfc2f83a
  3.12.5:b4e9aae06b1a343bc5a7fee975f391e7dbc7086fccc684068b5e0204ffa3ecad
  3.13.3:dc744895ee6b9c4491ff817bef0d3abd680c5e3c25e601be44240ce65ab4f337
  3.13.6:67ca2cf3040d08fdc51d27f660ea3157732b24c2f47aae1b19d63f62a39842c2
  3.14.2:87a04fd05cac20a2ec47094b7d18b96e0651257d8c768ced2ef7db270ecfb9cb
'

_pwd(){ CDPATH= cd -- $1 && pwd; }
_dirname(){ _d=`dirname -- "$1"`;  _pwd $_d; }
top_dir=`_dirname "$0"`

. $top_dir/_preamble.sh
```

Preliminary work on ParaGEMS

- Code profiling and optimisation
  - Split large routines and generalize functions
  - Simplify data structures and removing 'dead weight'
  - IO improvements (inspired by ParaFEM routines)

```
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!/****/s* dec_mod/calc_bndry_cobndry
!* SYNOPSIS
  SUBROUTINE calc_bndry_cobndry(k)
                                            undary operator] --
                                            oundaries --

  !-- build working array with element/boundary data --
  ALLOCATE(bndry(num_elm(k-1),k+2)); CALL build_bndry_work_array(bndry,k)

  !-- sort working array --
  ALLOCATE(work(num_elm(k-1),k+2))
  CALL int_merge_sort_rows(bndry,num_elm(k-1),1,k-1,work); DEALLOCATE(work)

  !-- count number of unique (local/external) [co-]boundaries --
  ALLOCATE(bndry_cnt(num_elm(k)),cobndry_cnt(num_elm(k-1)),&
    lwork(num_elm(k)*k))
  CALL count_bndry_cobndry(bndry,k,cnt,bndry_cnt,cobndry_cnt,ext_cnt,lwork)

  !-- allocate [co-]boundary structures and variables --
  CALL allocate_bndry_cobndry(k,cnt,bndry_cnt,cobndry_cnt,ext_cnt)

  !-- setup boundaries external to the current process (not local),
  !   setup parallel mapping, setup node indices for (k-1)^th order element
  !   structure, and setup co-boundaries for (k-1)^th order element
```

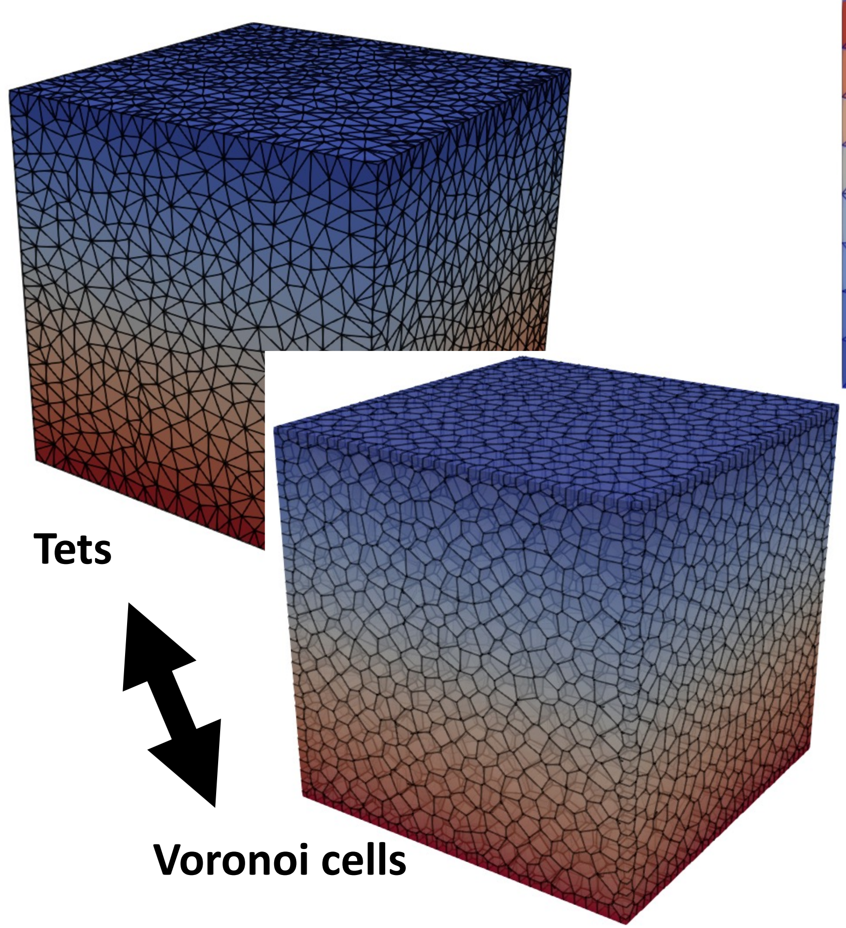
```
!-- get some basic information for reuse --
k=dim_cmplx
read_size=max_read_size
stride=2; buffer_size=read_size*stride
r_buffer_size=read_size
num_read_iters=glb_num_elm(k)/read_size
resid_read_size=mod(glb_num_elm(k),read_size)
ptr=0; fib=1
ALLOCATE(int_buffer(buffer_size),real_buffer(r_buffer_size))

IF (rank == root) THEN      !-- on root process --
  !-- allocate integer communication buffer --
  ALLOCATE(node_indices(dim_cmplx))
  !-- read data in chunks
  DO i=0,num_read_iters
    IF (i == num_read_iters) THEN
      read_size=resid_read_size;  buffer_size=read_size*stride
      r_buffer_size=read_size
    END IF
    !-- Read
```

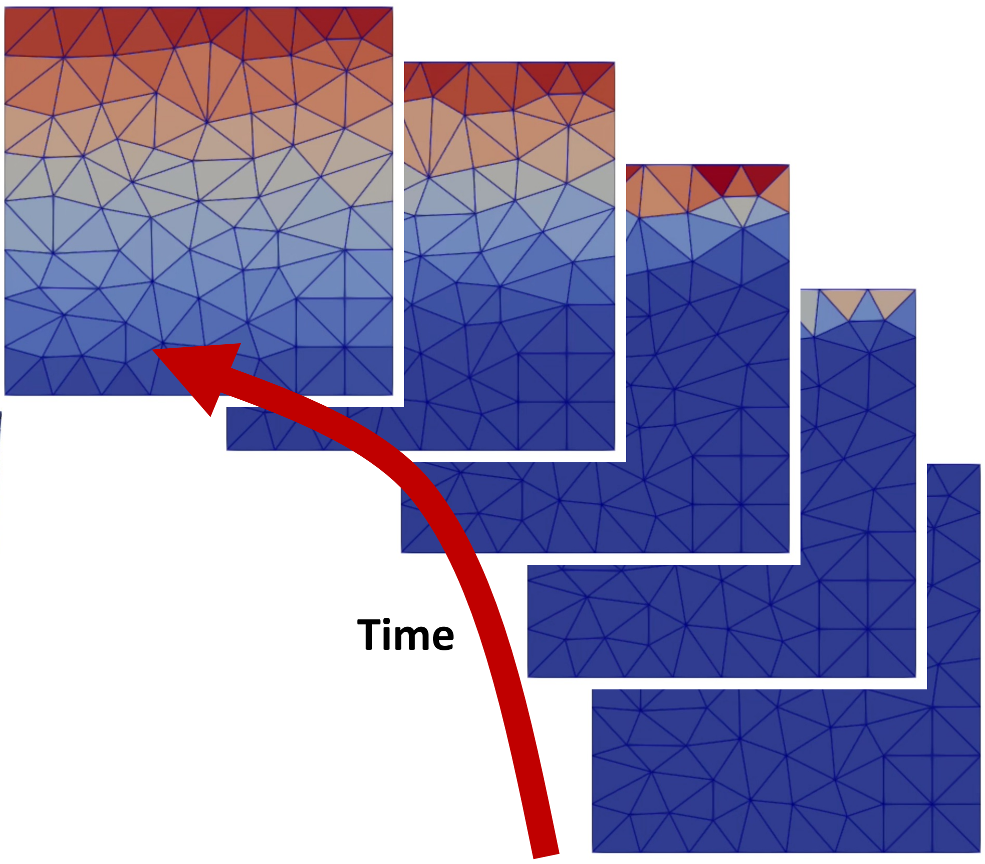
```
CALL MPI_BCAST(int_buffer,buffer_size,MPI_INTEGER,0,MPI_COMM_WORLD,ier)
```

Combined improvements: 2/3 reduction in CPU time for ~6 million cell mesh on 128 processes

Preliminary work on ParaGEMS

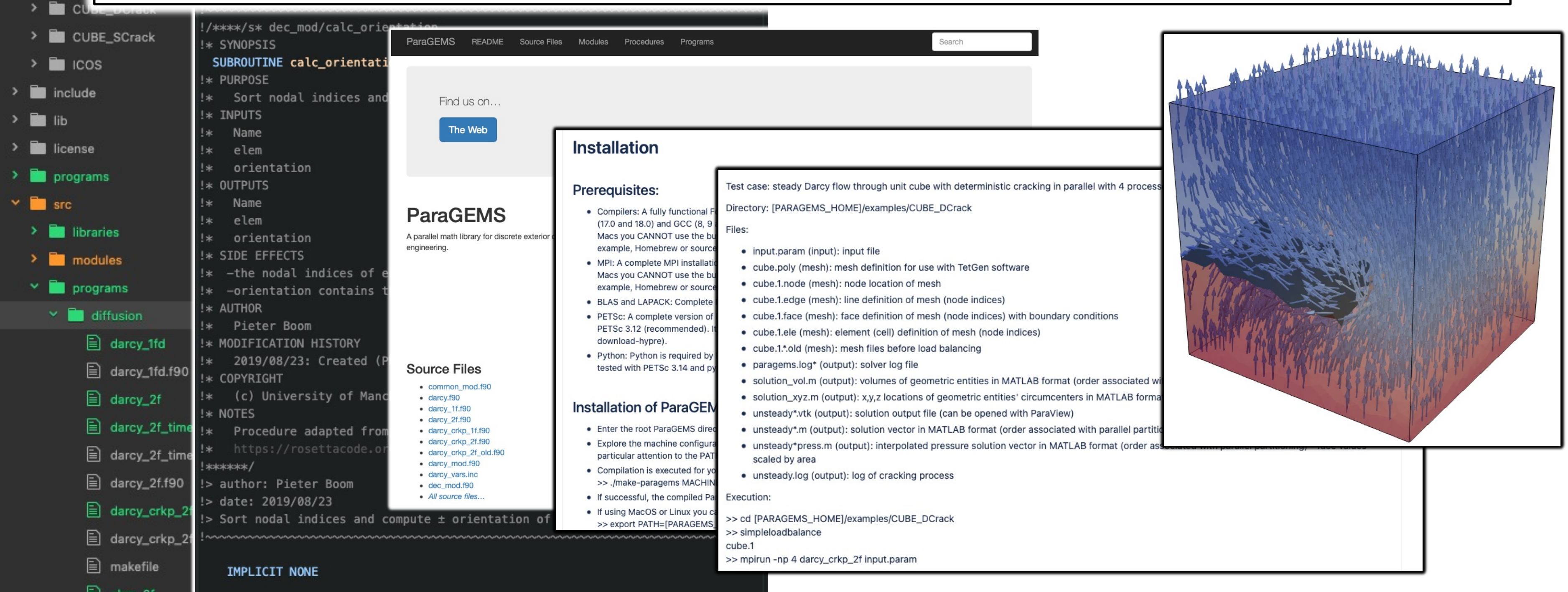
- Extended the library to support new MiniApps: formulations (dual, 1-field, 2D), physics, implicit multistep Runge-Kutta



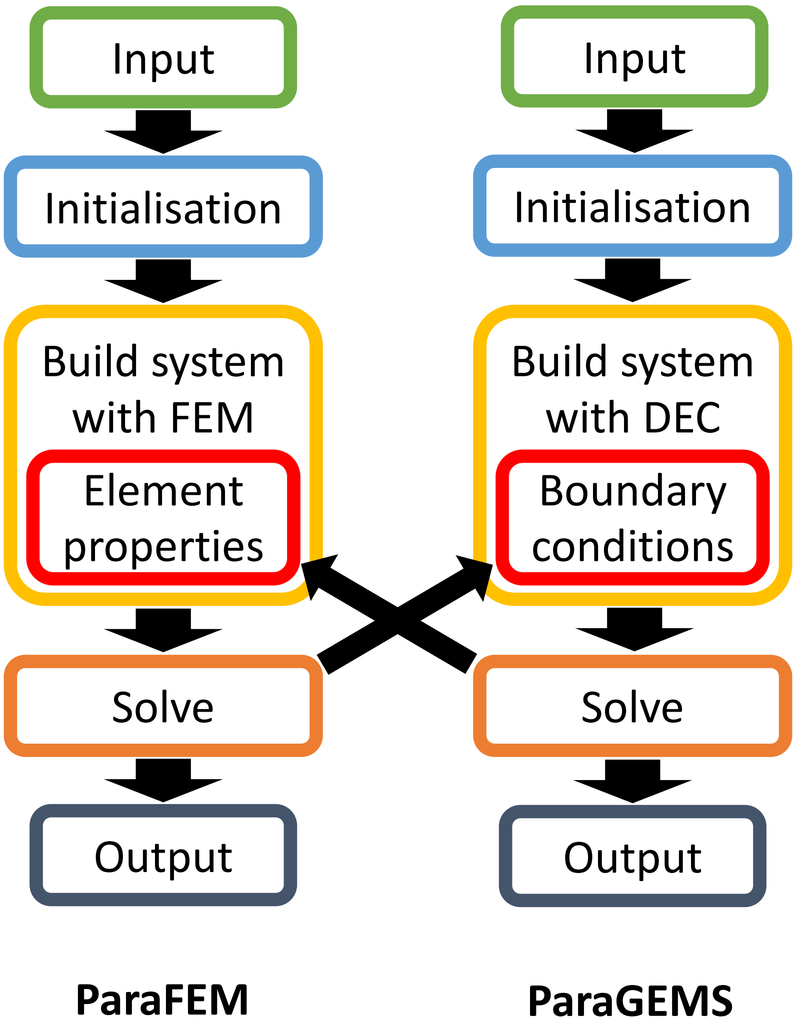


Preliminary work on ParaGEMS

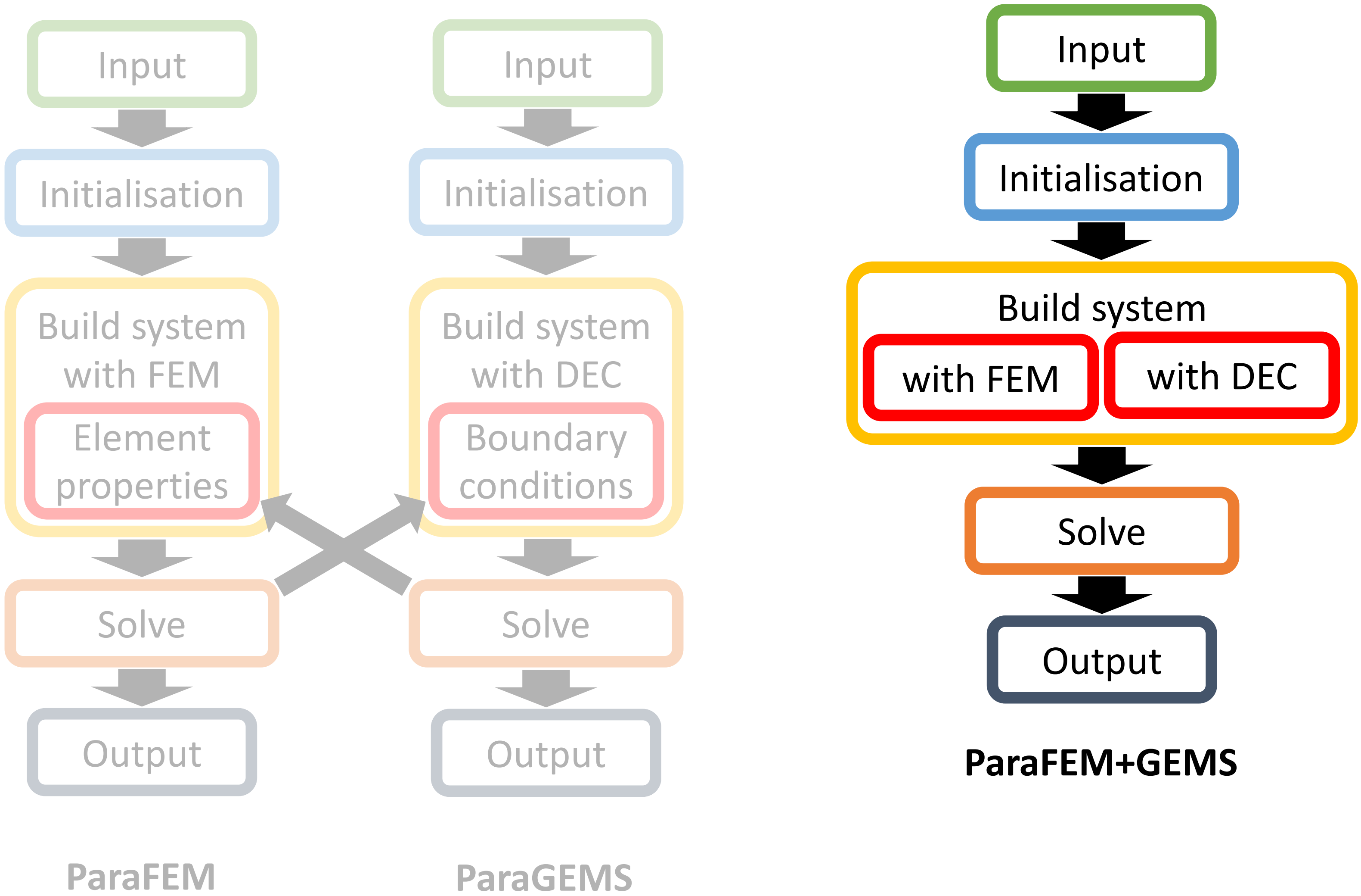
- Code sustainability:
  - Automatic documentation using RoboDocs and FORD
  - Improve readme and add installation instructions
  - Testing routines and programs
  - Create examples with tutorials for the MiniApps



Integration of ParaGEMS and ParaFEM

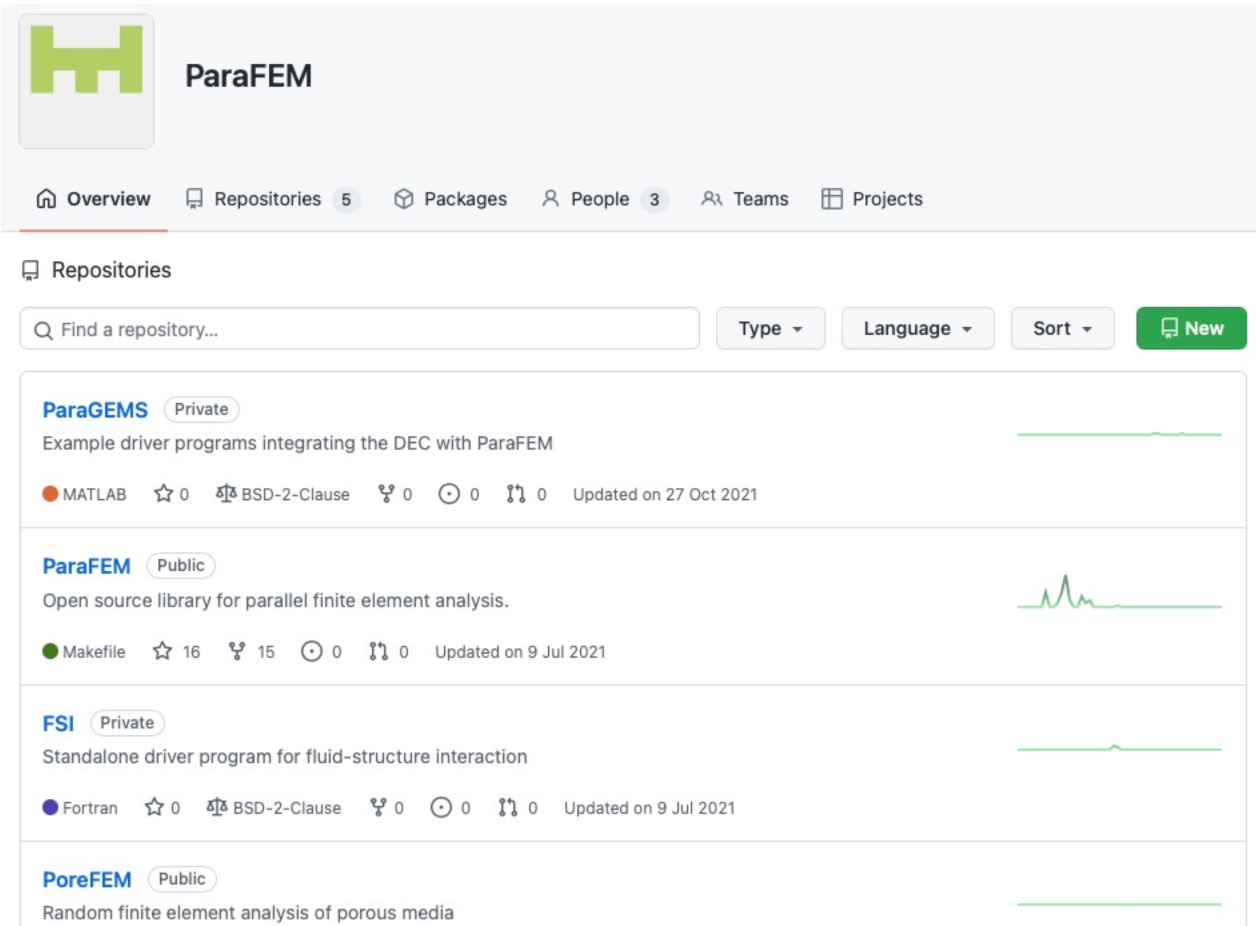


Integration of ParaGEMS and ParaFEM



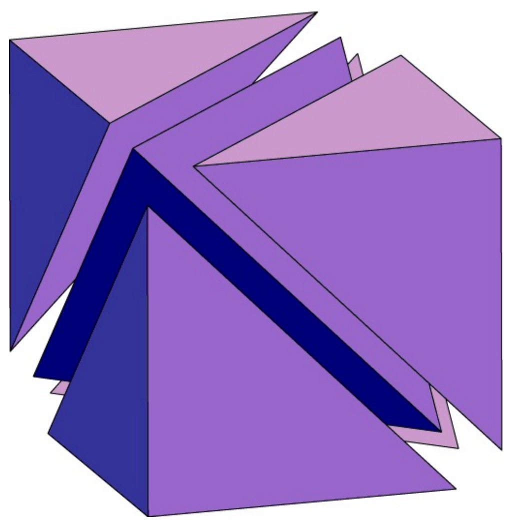
Integration of ParaGEMS and ParaFEM

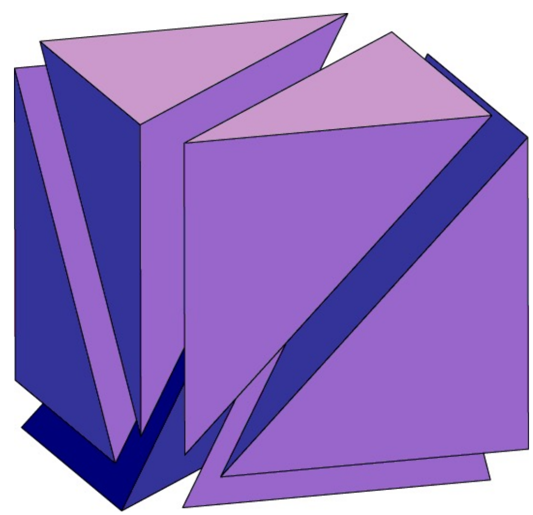
- Move from Bitbucket to Github under the ParaFEM umbrella



Integration of ParaGEMS and ParaFEM

- Element decomposition - specifically Hexs to 5 or 6 Tets
  - 64 possible cases once faces are decomposed
  - Select decomposition that minimises negative areas/volumes in the dual Voronoi mesh
  - Clever algorithm vs caseselect()

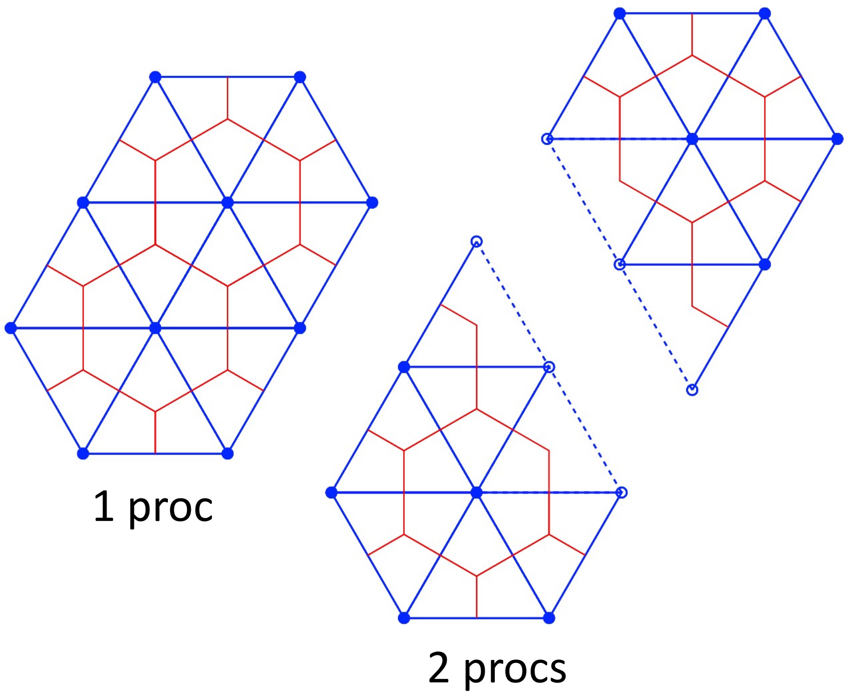




Images from: https://www.ics.uci.edu/~eppstein/projects/tetra/

Integration of ParaGEMS and ParaFEM

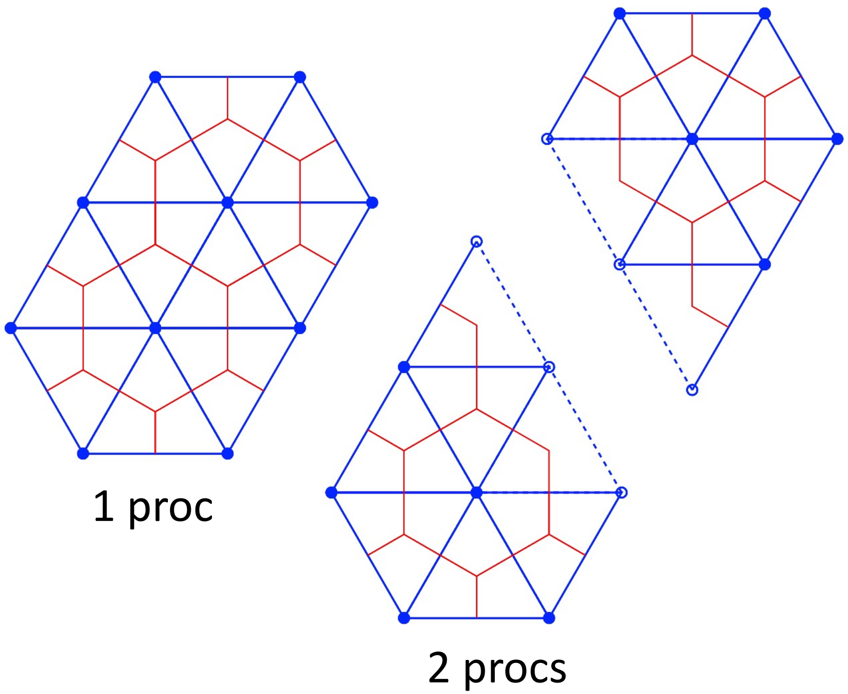
- Change in structure and parallelization mentality: elementwise



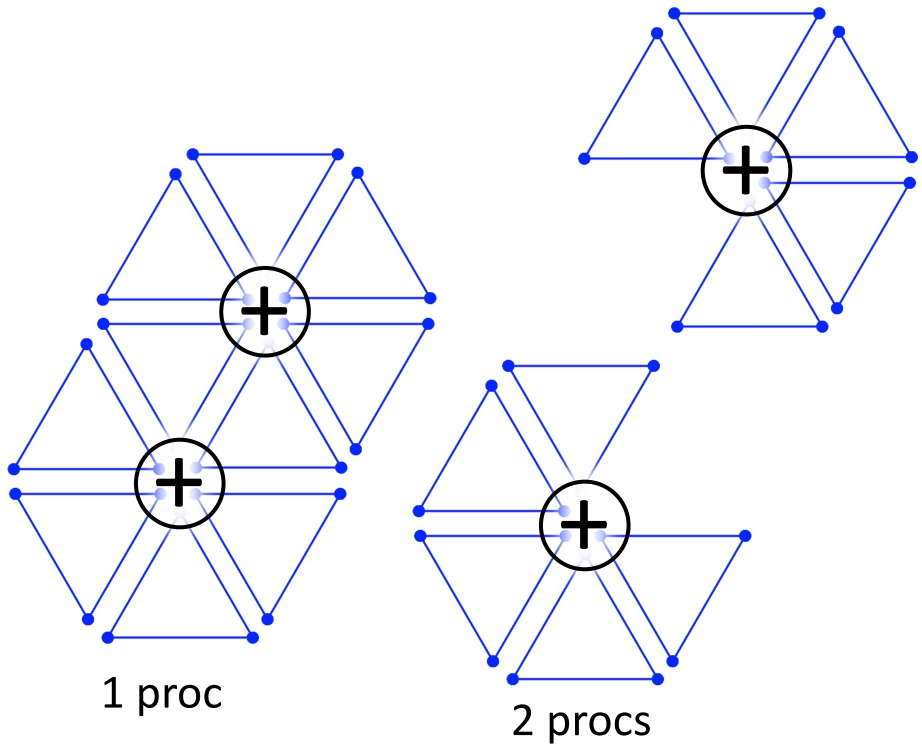
ParaGEMS

Integration of ParaGEMS and ParaFEM

- Change in structure and parallelization mentality: elementwise



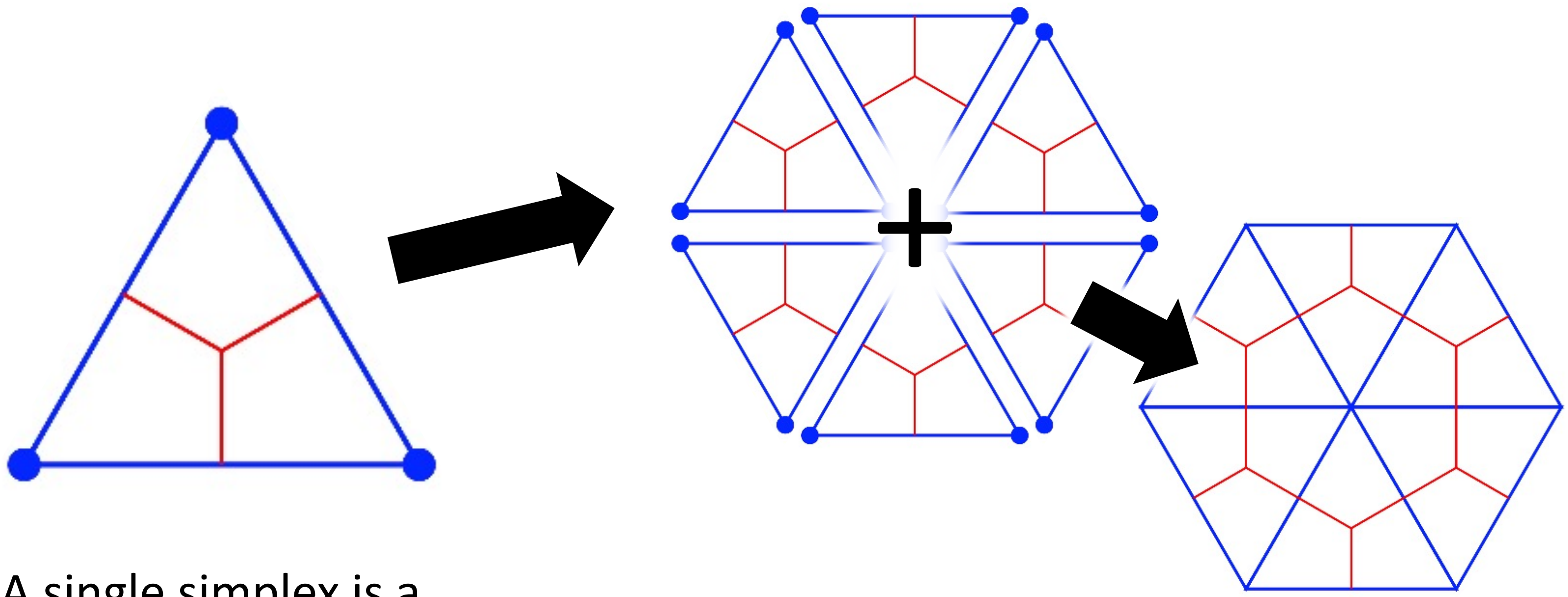
ParaGEMS



ParaFEM

Integration of ParaGEMS and ParaFEM

- Change in structure and parallelization mentality: elementwise

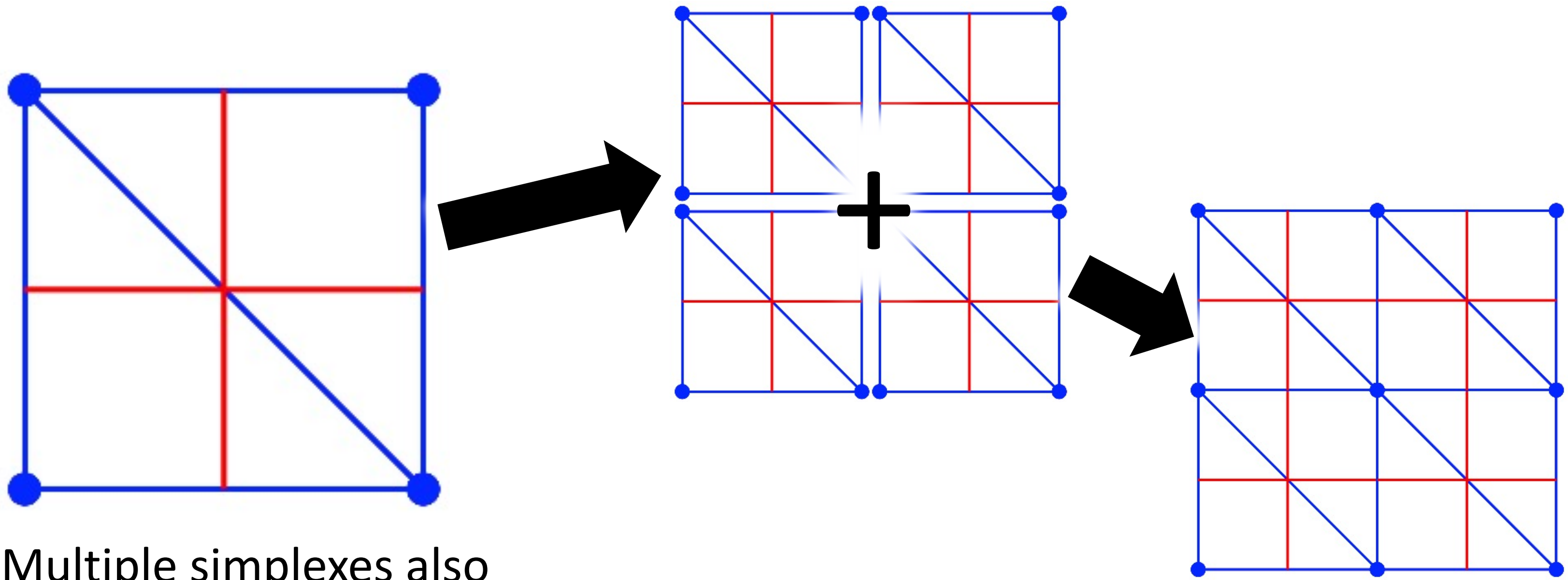


A single simplex is a valid complex

Composition of multiple elements

Integration of ParaGEMS and ParaFEM

- Change in structure and parallelization mentality: elementwise



Multiple simplexes also form a valid complex

Composition of multiple complexes

Integration of ParaGEMS and ParaFEM

- Constructing the modified system matrix

```
!-------------- element stiffness integration and storage ----------------
CALL sample(element,points,weights); storkc_pp=zero
elements_1: DO iel=1,nels_pp
  kcx=zero; kcy=zero; kcz=zero
  gauss_pts_1:  DO i=1,nip
    CALL shape_der (der,points,i); jac=MATMUL(der,g_coord_pp(:,:,iel))
    det=determinant(jac); CALL invert(jac); deriv=MATMUL(jac,der)
    row(1,:)=deriv(1,:); eld=deriv(1,:); col(:,1)=eld
    kcx=kcx+MATMUL(col,row)*det*weights(i); row(1,:)=deriv(2,:)
    eld=deriv(2,:); col(:,1)=eld
    kcy=kcy+MATMUL(col,row)*det*weights(i); row(1,:)=deriv(3,:)
    eld=deriv(3,:); col(:,1)=eld
    kcz=kcz+MATMUL(col,row)*det*weights(i)
  END DO gauss_pts_1
  storkc_pp(:,:,iel)=kcx*kx+kcy*ky+kcz*kz
END DO elements_1
```

ParaFEM p123 (from Chapter 12.3 in textbook)

Integration of ParaGEMS and ParaFEM

- Constructing the modified system matrix

```
!-------------- element stiffness integration and storage ----------------
CALL sample(element,points,weights); storkc_pp=zero
elements_1: DO iel=1,nels_pp
```

Convert hexs to tets

```
CALL elm2smplx(num_elm(dim_cmplx+1),lcl_complex(dim_cmplx+1)%node_indx,&
   g_coord_pp(:,:,iel),element,nod)
```

```
  gauss_pts_1:  DO i=1,nip
    CALL shape_der (der,points,i); jac=MATMUL(der,g_coord_pp(:,:,iel))
    det=determinant(jac); CALL invert(jac); deriv=MATMUL(jac,der)
    row(1,:)=deriv(1,:); eld=deriv(1,:); col(:,1)=eld
    kcx=kcx+MATMUL(col,row)*det*weights(i); row(1,:)=deriv(2,:)
    eld=deriv(2,:); col(:,1)=eld
    kcy=kcy+MATMUL(col,row)*det*weights(i); row(1,:)=deriv(3,:)
    eld=deriv(3,:); col(:,1)=eld
    kcz=kcz+MATMUL(col,row)*det*weights(i)
  END DO gauss_pts_1
  storkc_pp(:,:,iel)=kcx*kx+kcy*ky+kcz*kz
END DO elements_1
```

ParaFEM+GEMS (PF+G) pg123

Integration of ParaGEMS and ParaFEM

- Constructing the modified system matrix

```
!-------------- element stiffness integration and storage ---------------
CALL sample(element,points,weights); storkc_pp=zero
elements_1: DO iel=1,nels_pp
```

- Convert hexs to tets

```
  CALL elm2smplx(num_elm(dim_cmplx+1),lcl_complex(dim_cmplx+1)%node_indx,&
     g_coord_pp(:,:,iel),element,nod)
```

- Calculate topology

```
  DO k=dim_cmplx+1,2,-1;  CALL calc_bndry_cobndry(k);  END DO
```

- Calculate geometry

```
  DO k=2,dim_cmplx+1; CALL calc_circumcenters(k); END DO
  CALL calc_prml_unsgnd_vlm(2); CALL calc_dual_vlm(2);  CALL calc_prml_dir()
```

```
  gauss_pts_1:  DO i=1,nip
    CALL shape_der (der,points,i); jac=MATMUL(der,g_coord_pp(:,:,iel))
    det=determinant(jac); CALL invert(jac); deriv=MATMUL(jac,der)
    row(1,:)=deriv(1,:); eld=deriv(1,:); col(:,1)=eld
    kcx=kcx+MATMUL(col,row)*det*weights(i); row(1,:)=deriv(2,:)
    eld=deriv(2,:); col(:,1)=eld
    kcy=kcy+MATMUL(col,row)*det*weights(i); row(1,:)=deriv(3,:)
    eld=deriv(3,:); col(:,1)=eld
    kcz=kcz+MATMUL(col,row)*det*weights(i)
  END DO gauss_pts_1
  storkc_pp(:,:,iel)=kcx*kx+kcy*ky+kcz*kz
END DO elements_1
```

PF+G pg123

Integration of ParaGEMS and ParaFEM

- Constructing the modified system matrix

```
!------------- element stiffness integration and storage ---------------
 CALL sample(element,points,weights); storkc_pp=zero
 elements_1: DO iel=1,nels_pp
```

Convert hexs to tets •

```
CALL elm2smplx(num_elm(dim_cmplx+1),lcl_complex(dim_cmplx+1)%node_indx,&
   g_coord_pp(:,:,iel),element,nod)
```

Calculate topology •

```
DO k=dim_cmplx+1,2,-1;  CALL calc_bndry_cobndry(k);  END DO
```

Calculate geometry •

```
DO k=2,dim_cmplx+1; CALL calc_circumcenters(k); END DO
CALL calc_prml_unsgnd_vlm(2); CALL calc_dual_vlm(2);  CALL calc_prml_dir()
```

```
   gauss_pts_1: DO i=1,nod
                  DO j=1,lcl_complex(1)%num_cobndry(i)
```

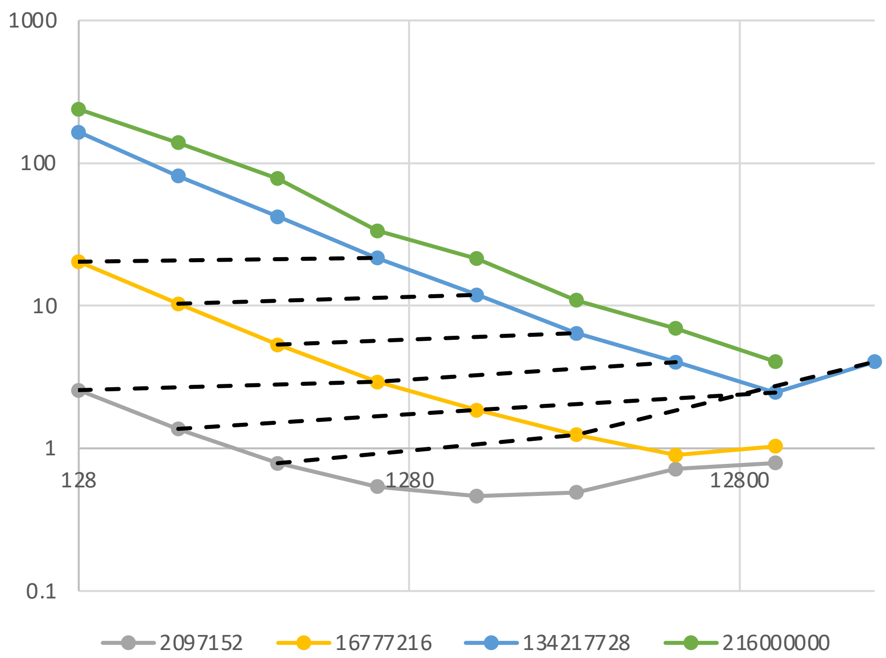
DEC formulation on the element •

```
ka = DOT_PRODUCT(kay,abs(lcl_complex(2)%prml_dir(k,:)))
kcx(i,m) = kcx(i,m) - ka*lcl_complex(2)%dual_volume(k) / &
   max(lcl_complex(2)%prml_volume(k),small)
kcx(i,i) = kcx(i,i) + ka*lcl_complex(2)%dual_volume(k) / &
   max(lcl_complex(2)%prml_volume(k),small)
```

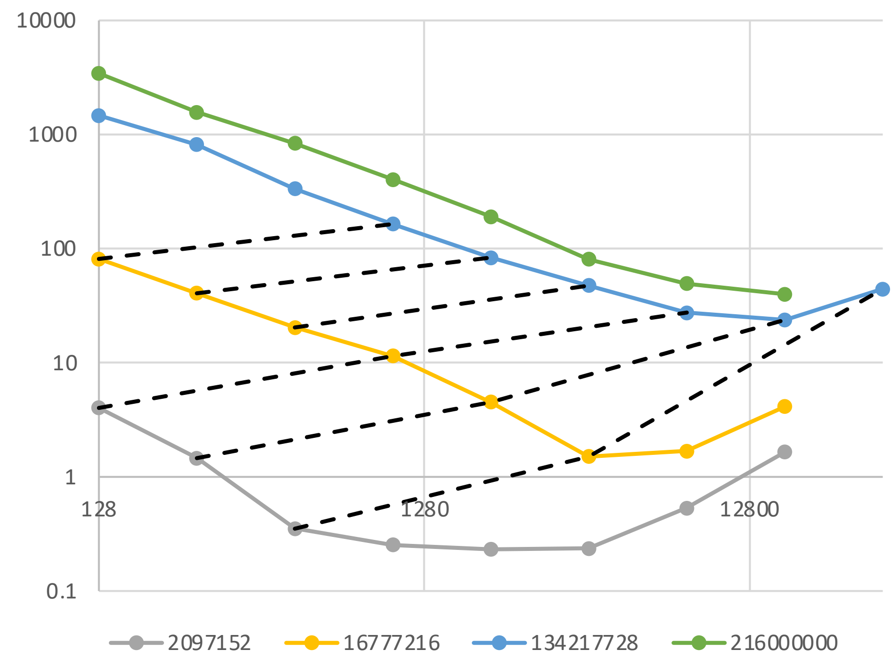
```
   END DO gauss_  END DO
   storkc_pp(:,:  END DO;  storkc_pp(:,:,iel)=kcx
END DO elements_1
```

PF+G pg123

Parallel scaling analysis

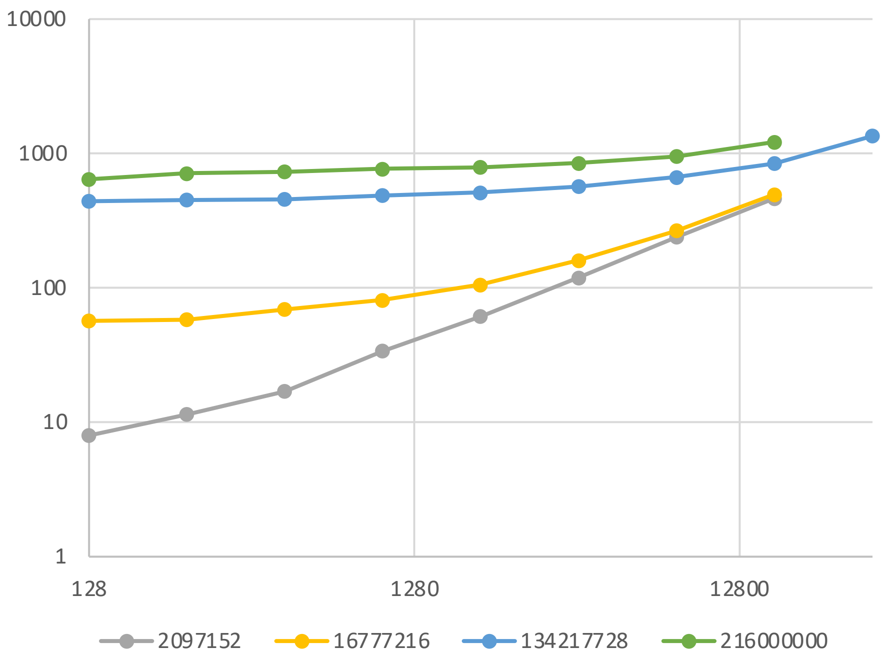


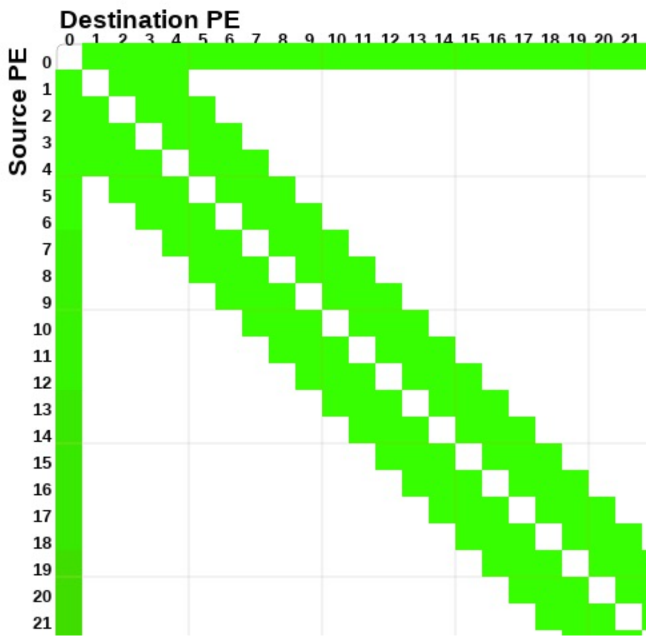
Construction of system matrix



Solution

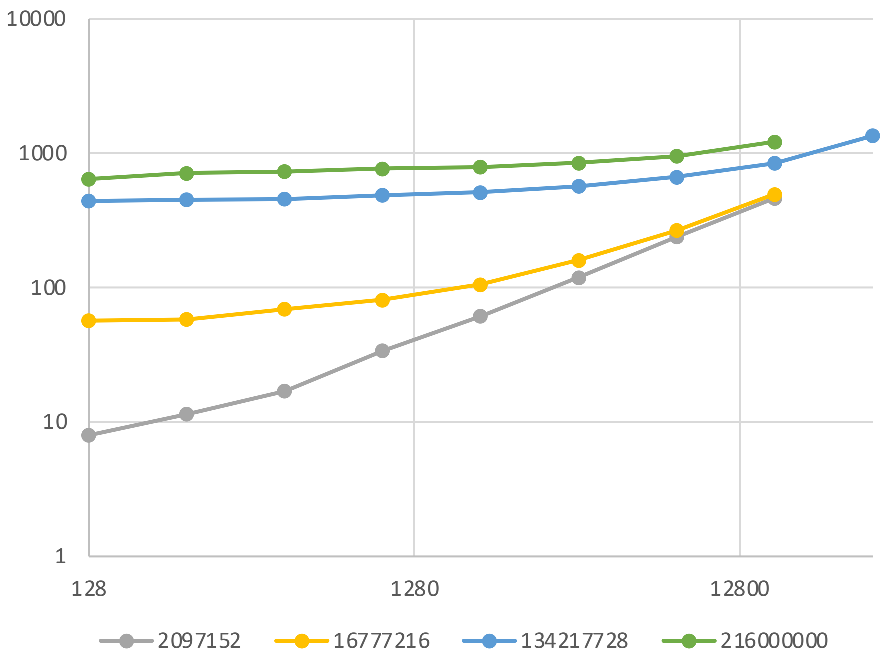
Parallel scaling analysis

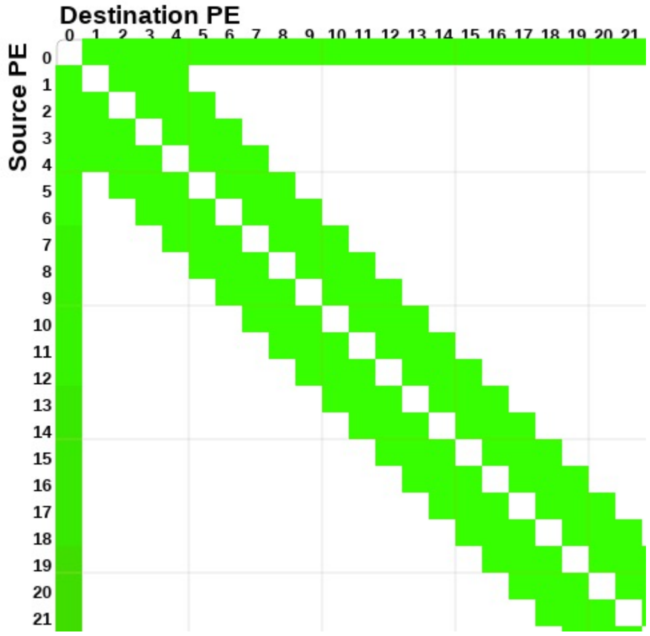




Initialisation: largely serial ASCII IO from root

Parallel scaling analysis





Initialisation: largely serial ASCII IO from root

(Binary IO routines already exist for ParaFEM)

Summary

- Recovered approximately O(100x) speedup for ~6 million cell mesh with 128 processes
  - Profiling and code optimization
    - Generalising/factoring code
    - Improving IO routines
  - Leveraging ParaFEM's optimized initialization and solution routines refined over 30 years of development
- Excellent weak and strong scaling of core routines on >16k cores
  - Problem (mesh) size dependent
- IO is a known an ongoing performance bottleneck for simple steady problems (but with a pathway forward for improvement)
- Now have an efficient tool to underpin new research using discrete forms of Exterior Calculus
- Extended the possible use of the ParaFEM library for engineering applications

Future work

- Profiling and optimization using much large meshes (>1 billion cells) and scaling beyond 16-32k cores
- Calculate critical core loading (# elements/core) for desired parallel efficiency
- Integrate binary IO (already exist) and implement MPI-IO
- Minimise unnecessary memory reallocation
- Broader discrete forms of Exterior Calculus (Forman, Berbatov, etc)
- Development of configuration scripts and containerisation
- Better testing using something like FRUIT

Thank you for your time!

Any Questions?

Code will be available at: https://github.com/ParaFEM