OpenCMISS Programmer Documentation

OpenCMISS Programmer Documentation

Table of Contents

Chapter 1. Introduction

OpenCMISS is a project for the re-engineering of the CMISS computational engine.

1. CMISS

CMISS is an interactive computer program for Continuum Mechanics, Image analysis, Signal processing and System Identification. It provides a mathematical modelling environment that allows the application of finite element analysis, boundary element and collocation techniques to a variety of complex bioengineering problems.

CMISS consists of two main components, cmgui and cm. cmgui is a graphical front end with advanced 3D display and modelling capabilities. It is the open source project written in C/C++. cm is a computational backend that may be run remotely on powerful workstations or supercomputers. cm is developed under Fortran 77 and has some licensing limitations.

2. Shared Memory Computing vs Distributed Memory Computing

cm uses OpenMP for shared memory computing. A shared memory system is relatively easy to program since all processors share a single view of data and the communication between processors can be as fast as memory accesses to a shared location. However, the CPU-to-memory connection will eventually become a bottleneck.

Distributed memory systems have increased CPU and memory scalibility compared to shared memory systems, but are much harder to program. The MPI standard is used for message passing.

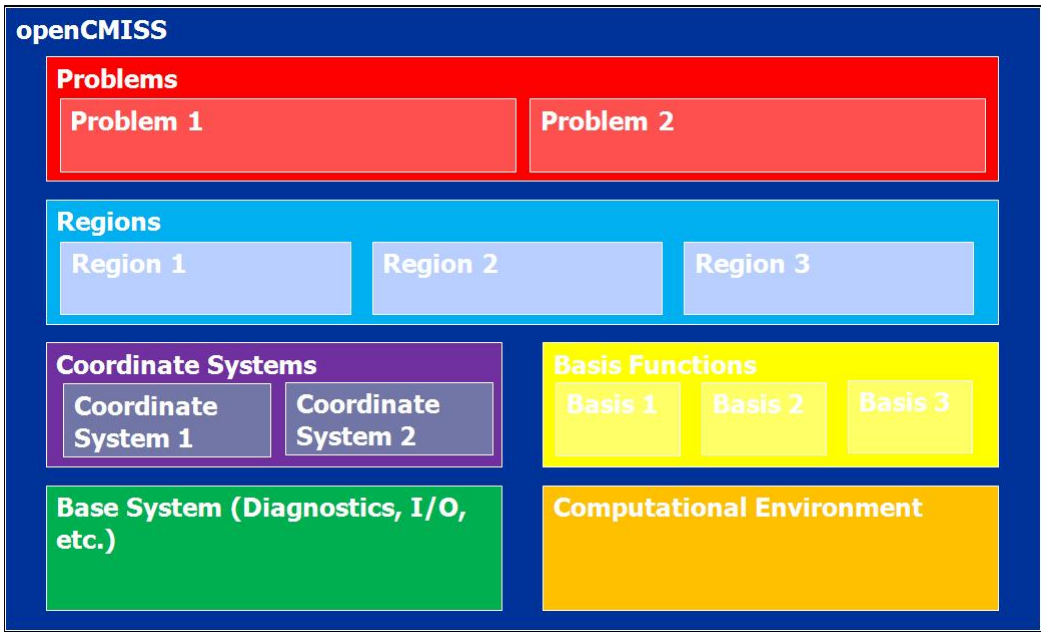
3. Objectives of OpenCMISS

OpenCMISS is re-engineering of CMISS cm component. It is intended to give a library based approach to enable use in multiple applications. It is developed under Fortan 95 and Modular, easily extendable and programmable. It will be integrated with MPI to provide modular, easily extendable and programmable code which is applicable for either MPI based distributed system, OpenMP based shared memory system and/or serial code. It will be open sourced.

Chapter 2. OpenCMISS Concepts

openCMISS has the following top level objects.

- Basis functions
- Coordinate systems
- Nodes
- Regions
- Problems
- Computational environments
- Base system(Diagnostics, I?O etc.)



1. Basis Functions

Basis functions are the key item to specify the field approximation/interpolation and the linking of nodes and elements to form a mesh. Currently, it has two types: Lagrange-Hermite tensor product and Simplex. Lagrange-Hermite tensor product can be further divided into linear to cubic lagrange, cubic and quadratic hermite. It can be arbitrarily collapsed (two or more nodes in the same location) in any one direction or in any two directions to give a degenerate basis. Simplex basic functions could contain line, triangular and tetrahedral elements. It could be linear, quadratic or cubic. Arbitrary Gaussian quadrature can integrate from 1st to 5th order (3rd order for lines at the moment). Can only have the same order in each direction at the moment. Specifying a basis function automatically generates all necessary line and face basis functions as sub-bases of the basis function.

Basis function has the following attributes:

- User number
- Global number
- Family number

- Finished tag
- Type
- Is Hermite
- Number of XI
- Number of XI coordinates
- ...

2. Coordinate Systems

Coordinate system is a system for assigning an n-tuple of numbers or scalars to each point in an n-dimensional space. It anchors the regions within the real world. Coordinate system can have different types such as:

- Rectangular cartesian
- Cylindrical polar
- Spherical polar
- Prolate spheroidal
- Oblate spheroidal

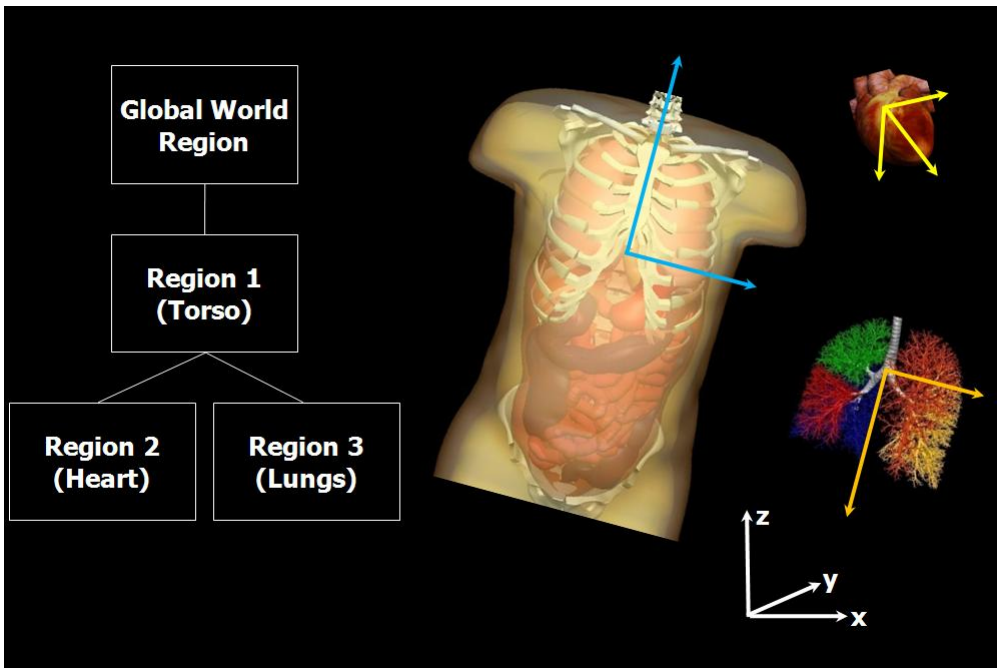
There is a global (world) coordinate system aligned with 3D rectangular cartesian space.

Coordinate system has the following attributes:

- User number
- Finished tag
- Type
- Number of dimensions
- Focus (for prolate-spheriodal system only)
- Origins
- Orientation

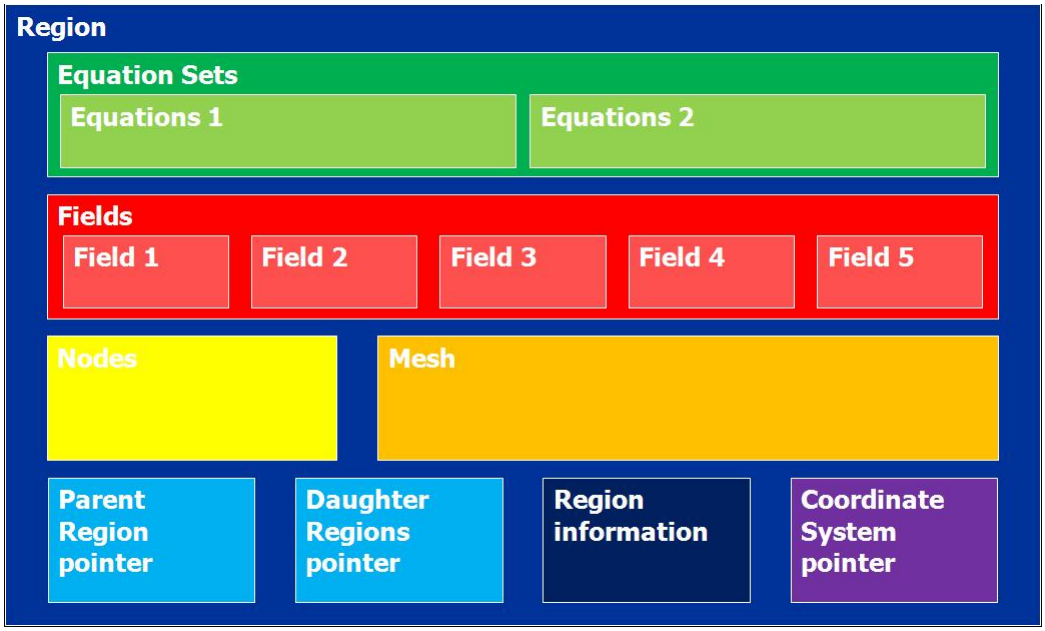
3. Regions

Regions are one of the primary objects in openCMISS. Regions are hierarchical in nature in that a region can have one parent region and a number of daughter sub-regions. Daughter regions are related in space to parent regions by an origin and an orientation of the regions coordinate system. Daughter regions may only have the same or fewer dimensions as the parent region. There is a global (world) region (number 0) that has the global (world) coordinate system.



Region has the following attributes

- User number
- Finished tag
- Label
- Number of sub(daughter) regions
- Coordinate system pointer
- Nodes
- Meshes
- Fields
- Equations
- Parent region pointer
- Daughter regions pointers



3.1. Nodes

There are three places storing nodal information. Nodes associated with region defines the nodes identification and the nodes geometric (initial) position.

Node has the following attributes

- User number
- Global number
- Label
- Initial Position

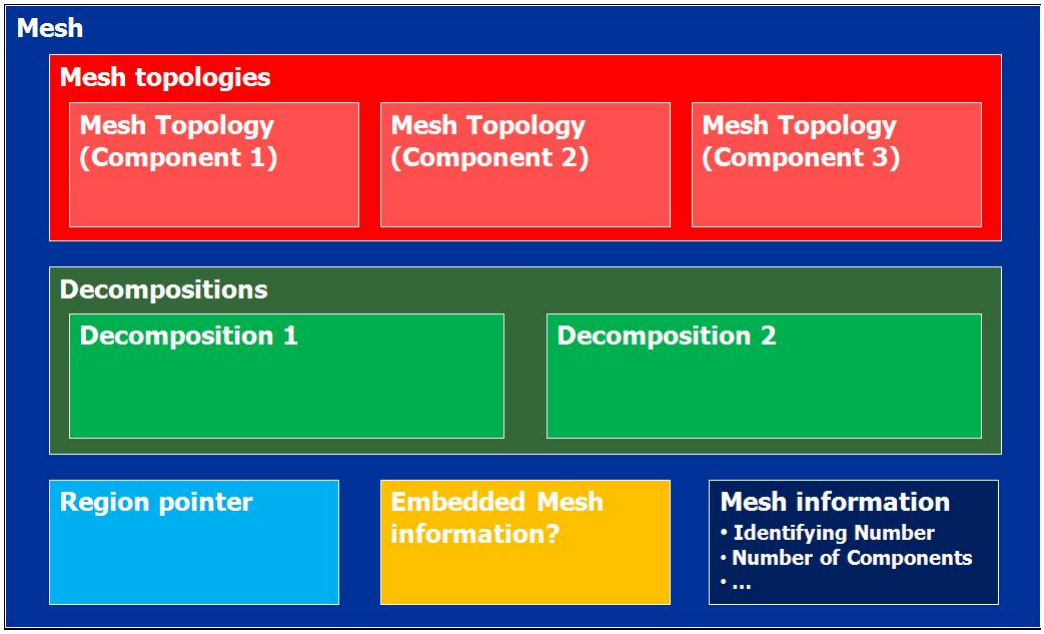
3.2. Meshes

Meshes are topological constructs within a region which fields use to define themselves. Meshes are made up of a number of mesh components. All mesh components in a mesh must “conform”, that is have the same number of elements, Xi directions etc.

Mesh has the following attributes:

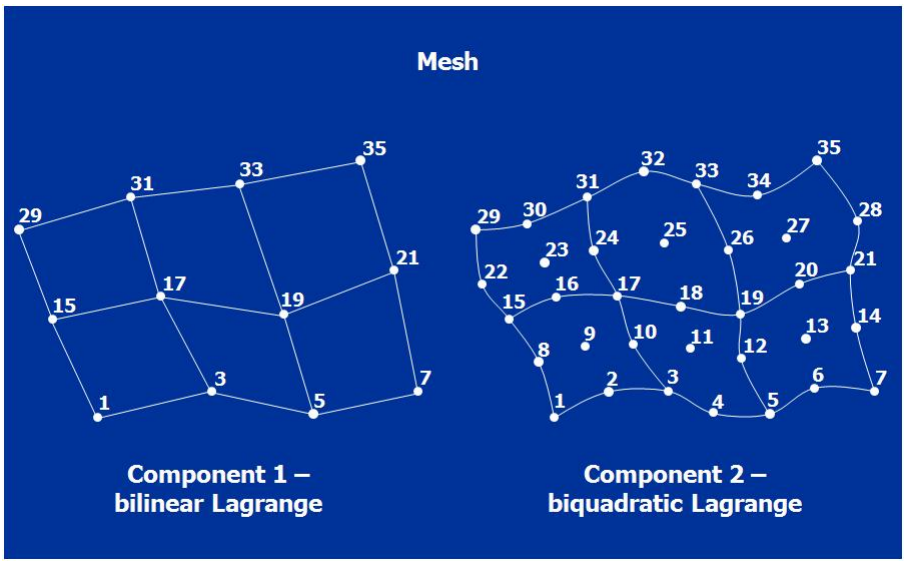
- User number
- Global number
- Finished tag
- Region pointer
- Number of dimensions
- Number of components
- Embedded flag
- Embedding mesh pointer

- Embedded meshes pointers
- Number of elements
- Number of faces
- Number of lines
- Mesh topology pointers
- Decomposition pointers



3.2.1. Mesh Topology

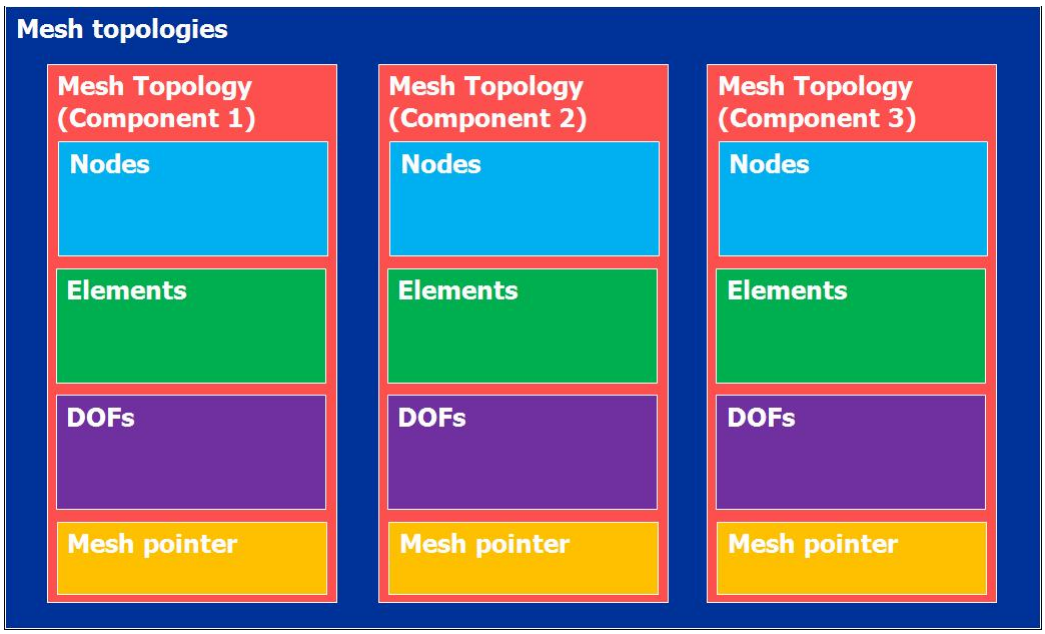
Mesh components (Topology) are made up from nodes, elements and basis functions. A new mesh component is required for each different form of interpolation e.g., one mesh component is bilinear Lagrange and another is biquadratic Lagrange.



Mesh topology has the following attributes:

- Mesh component number

- Mesh pointer
- Nodes pointers
- Element pointers
- DOFs pointers

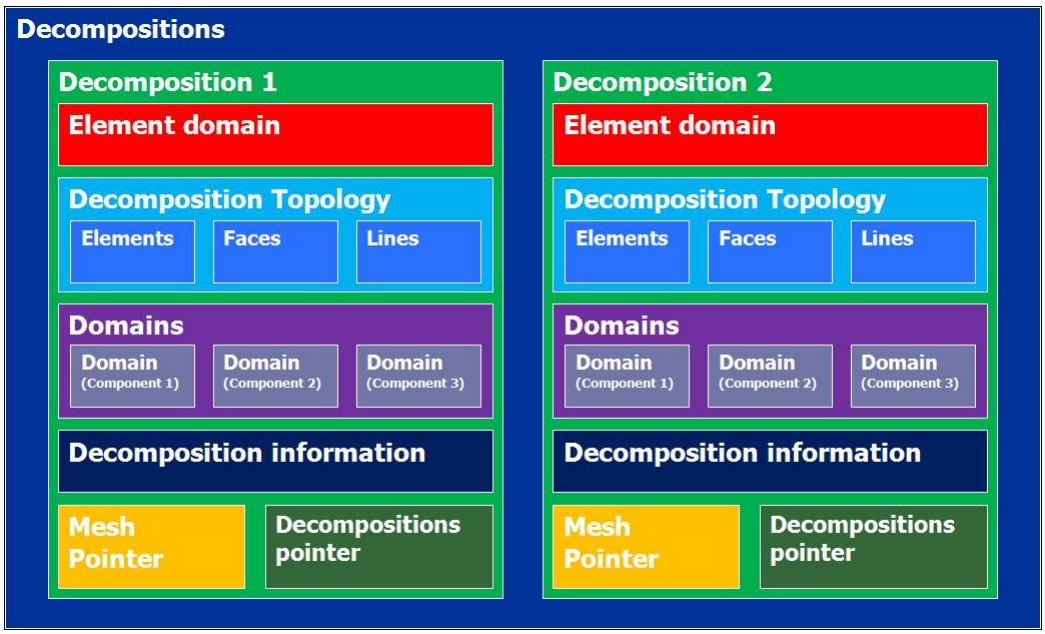


3.2.2. Decompositions

Mesh decomposition (partitioning) is used to split a computationally expensive mesh into smaller subdomains (parts) for parallel computing.

Decomposition has the following attributes

- User number
- Global number
- Finished tag
- Mesh pointer
- Mesh component number
- Decomposition type
- Number of domains
- Number of edge cut
- Element domain numbers
- Decomposition topology pointer
- Domains pointers(list of domain which has the same size as the number of components in the mesh)

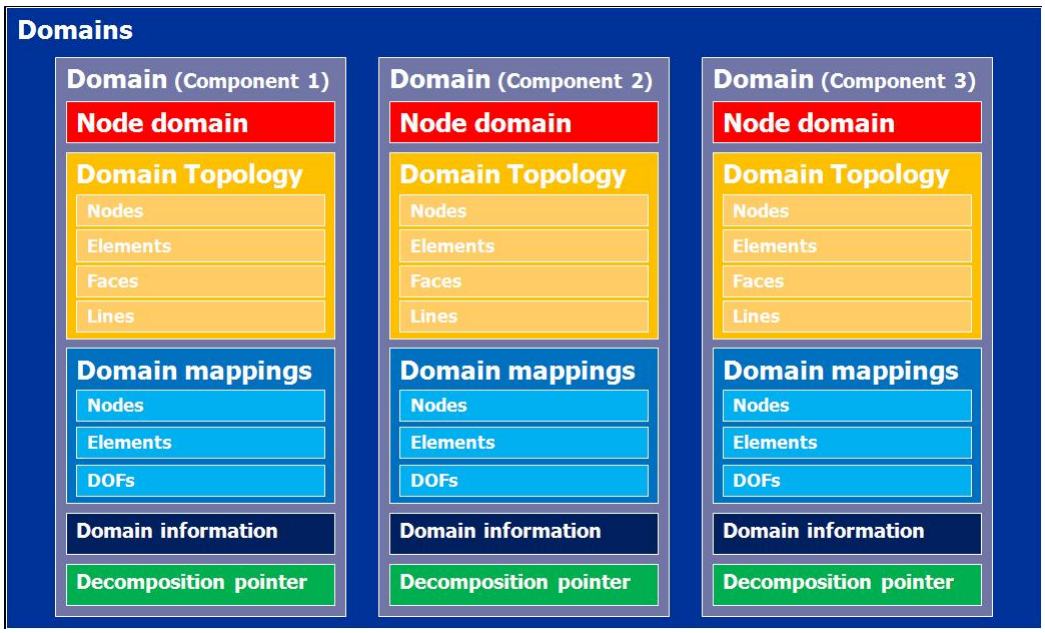


3.2.2.1. Domain

Each domain stores domain information for relevant mesh component.

The domain object contains the following attributes:

- Decomposition pointer
- Mesh pointer
- Mesh component number
- Region pointer
- Number of dimensions
- Node domain(The domain number that the np'th global node is in for the domain decomposition. Note: the domain numbers start at 0 and go up to the NUMBER_OF_DOMAINS-1)
- Domain mappings(for each mapped object e.g. nodes, elements, etc)
- Domain topology pointer(elements, nodes, DOFs)

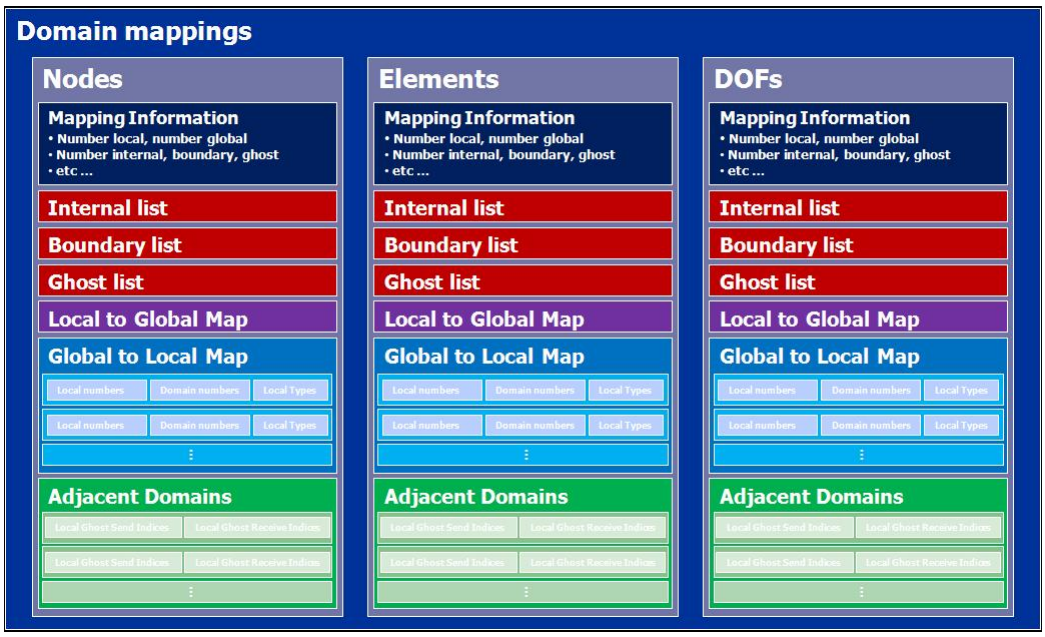


3.2.2.1.1. Domain Mappings

Stores information for each mapped object e.g. nodes, elements, etc.

The domain mapping contains the following attributes:

- Number of local
- Total number of local
- Numbers of domain local
- Number of global
- Number of domains
- Number of internal
- Internal list
- Number of boundary
- Boundary list
- Number of ghost
- Ghost list
- Local to global map
- Global to local map
- Number of adjacent domains
- Pointer to list of adjacent domains by domain number
- List of adjacent domains

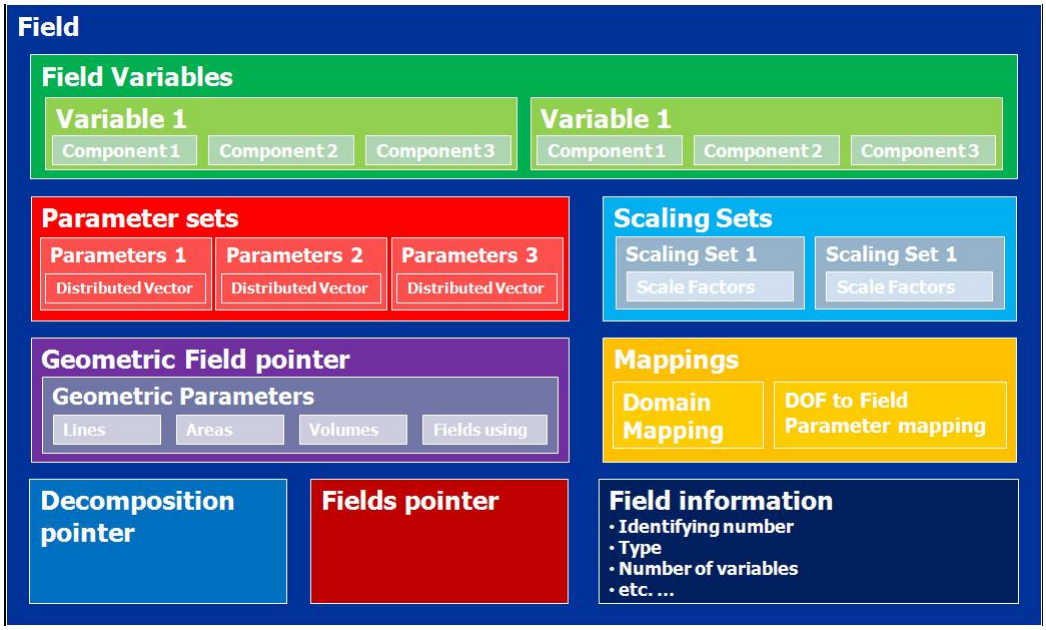


3.3. Fields

Fields are the central object for storing information and framing the problem. Fields have a number of field variables i.e., u, #u/#n, #u/#t, #2u/#t2. Each field variable has a number of components. A field is defined on a decomposed mesh. Each field variable component is defined on a decomposed mesh component.

Field can contains the following attributes:

- User number
- Global number
- Finished tag
- Region pointer
- Type(Geometric, Fibre, General, Material, Source)
- Dependent type(Independent, Dependent)
- Dimension
- Decomposition pointer
- Number of variables
- Variables
- Scalings sets
- Mappings(DOF->Field parameters)
- Parameter sets(distributed vectors)
- Geometric field pointer
- Geomatric field parameters
- Create values cache

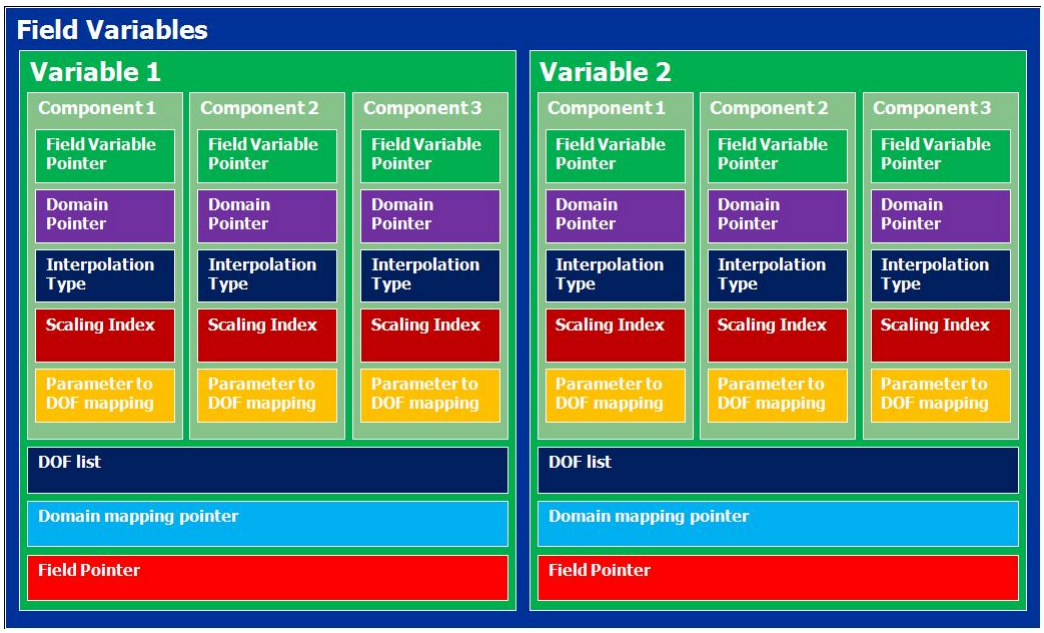


3.3.1. Field variable

Field variable stores variables for the field such as dependent variables. For example, in the Laplace's equation(FEM), it stores two variables: u and #u/#n. Each field variable has a number of components.

Field variable has the following attributes:

- Variable number
- Variable type
- Field pointer
- Region pointer
- Max number of interpolation parameters
- Number of DOFs
- Total number of DOFs
- Global DOF List
- Domain mapping pointer
- Number of components
- Components



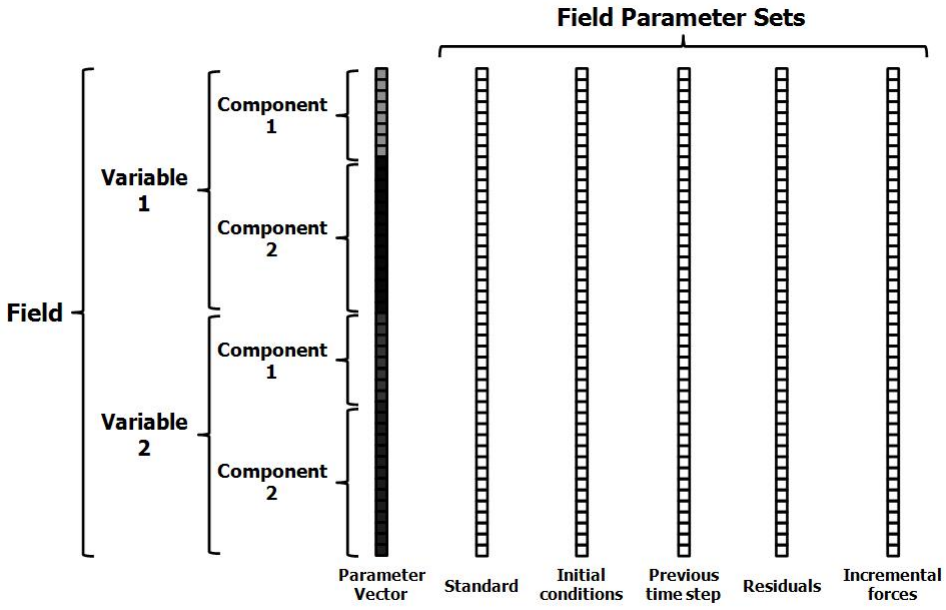
3.3.1.1. Field Variable Component

Field Variable Component has the following attributes:

- Component number
- Variable pointer
- Field pointer
- Interpolation type
- Mesh component number
- Scaling index
- Domain pointer
- Max number of interpolation parameters
- Mappings(Field paramters->DOF)

3.3.2. Parameter set

Parameter set stores values for each field variable component.



Parameter set has the following Attributes:

- Set index
- Set type
- Parameters pointer

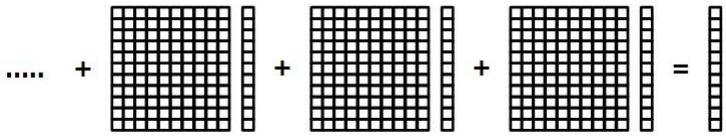
3.4. Equations Sets

Equations sets are aimed to have multiple classes, e.g. Elasticity, Fluid mechanics, Electromagnetics, General field problems, Fitting, Optimisation. Different equations are within each class, e.g. Bidomain, Navier-stokes etc. Each equation can use different solution techniques, e.g. FEM, BEM, FD, GFEM. The equation set is associated with a region and is built using the fields defined on the region.

The numerical methods are used which will result in a discretised matrix-vector form of the governing equations. openCMISS is designed to generate equations sets with a number of "equations" matrices.

e.g, damped mass spring system
$M\ddot{u} + C\dot{u} + Ku = f$

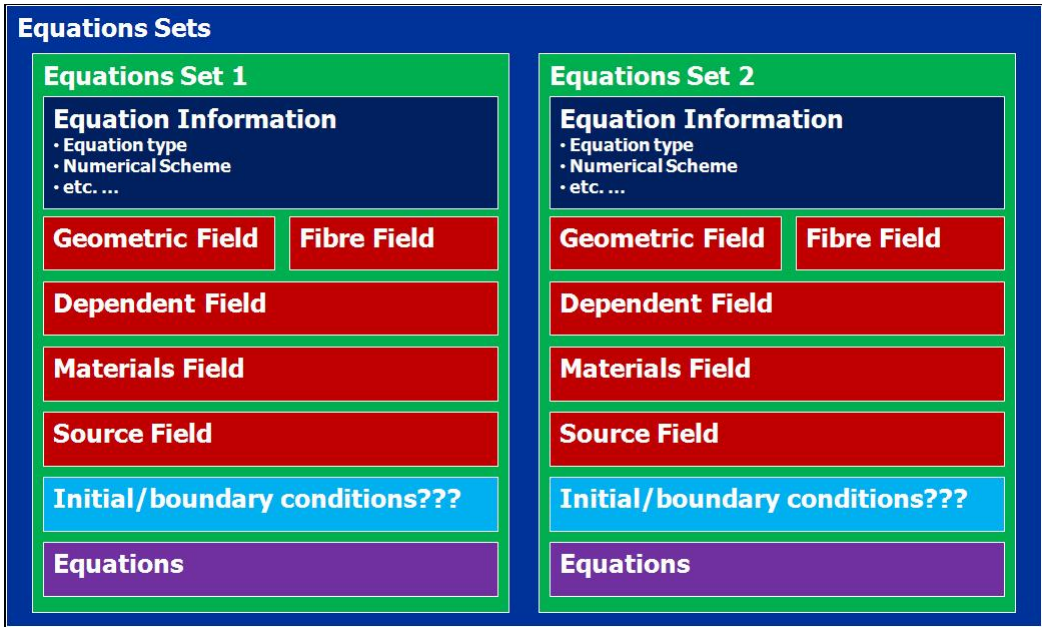
will be represented as:



Equations Set has the following attributes:

- User number
- Global number
- Finished tag
- Region pointer

- Class identifier
- Type identifier
- Sub type identifier
- Linearity type(?)
- Time dependence type(?)
- Solution method
- Geometry (fibre?) field pointer
- Materials field pointer
- Source field pointer
- Dependent field pointer
- Analytic info pointer(Analytic info stored in dependent field currently)
- Fixed conditions
- Equations pointer

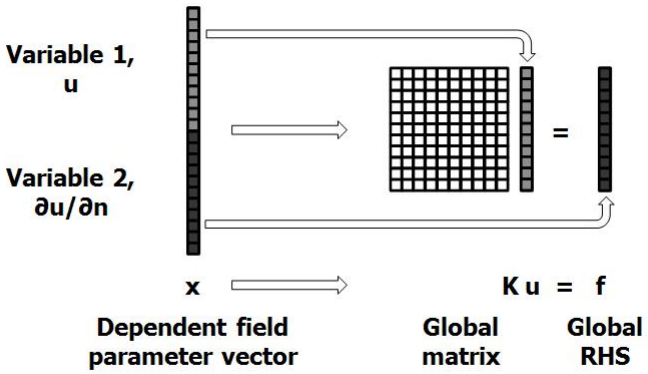


3.4.1. Equations

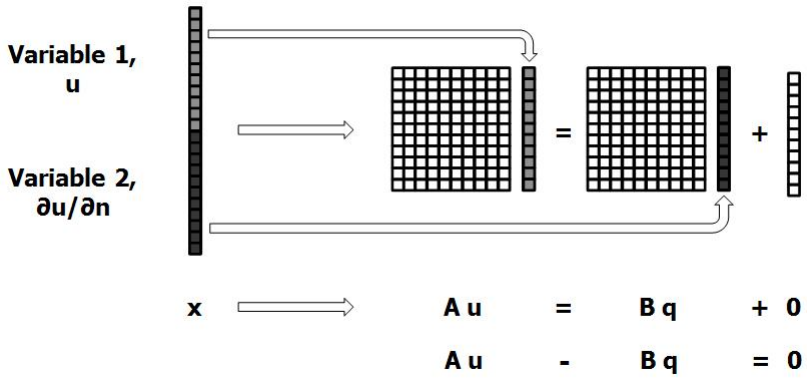
Equation holds the matrices and mapping information.

The Field variable to matrix mappings maps each field variable onto the equations matrices or RHS vector.

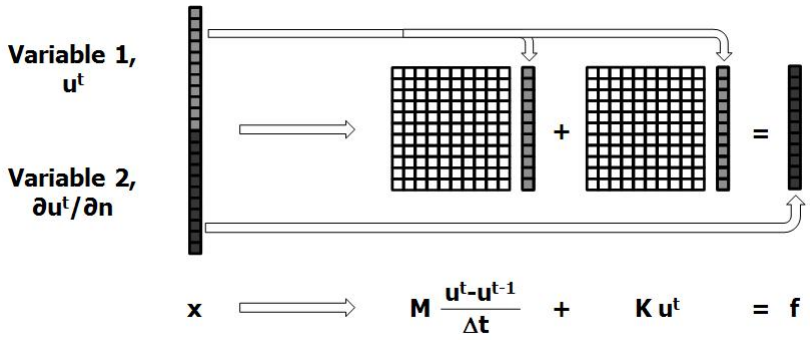
e.g. Laplace(FEM): 2 variables, 1 component



e.g. Laplace(BEM): 2 variables, 1 component



e.g. Heat equation(explicit time/FEM space): 2 variables, 1 component

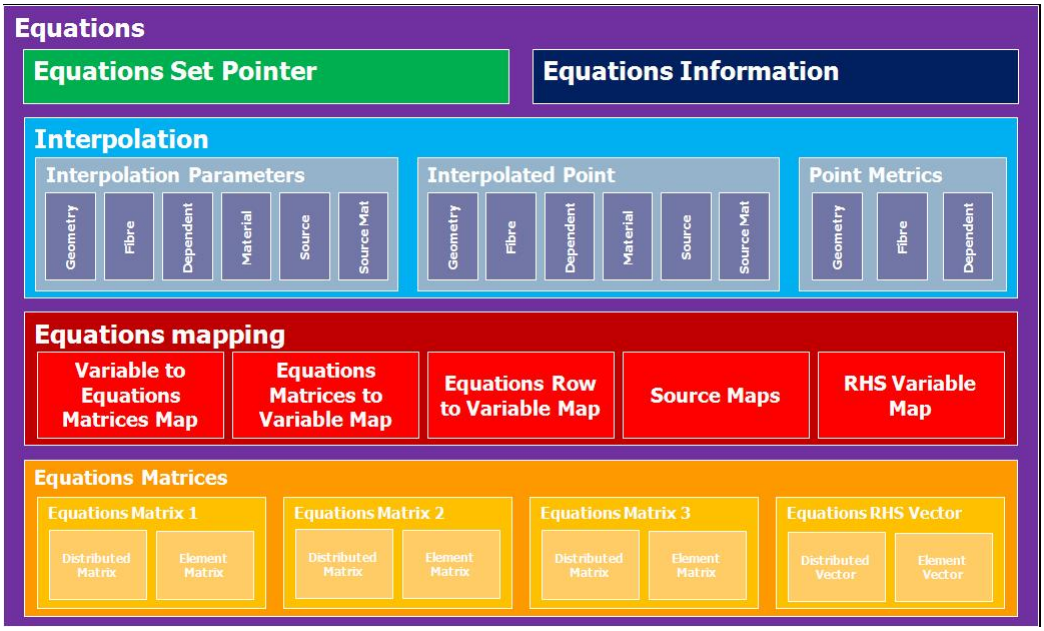


TODO matrix distribution

Equations has the following attributes:

- Equation set pointer
- Finished tag
- Output type
- Sparsity type
- Interpolation pointer
- Linear equation data pointer

- Nonlinear equation data pointer
- Time(non-static) data pointer
- Equations mapping pointer
- Equations Matrices

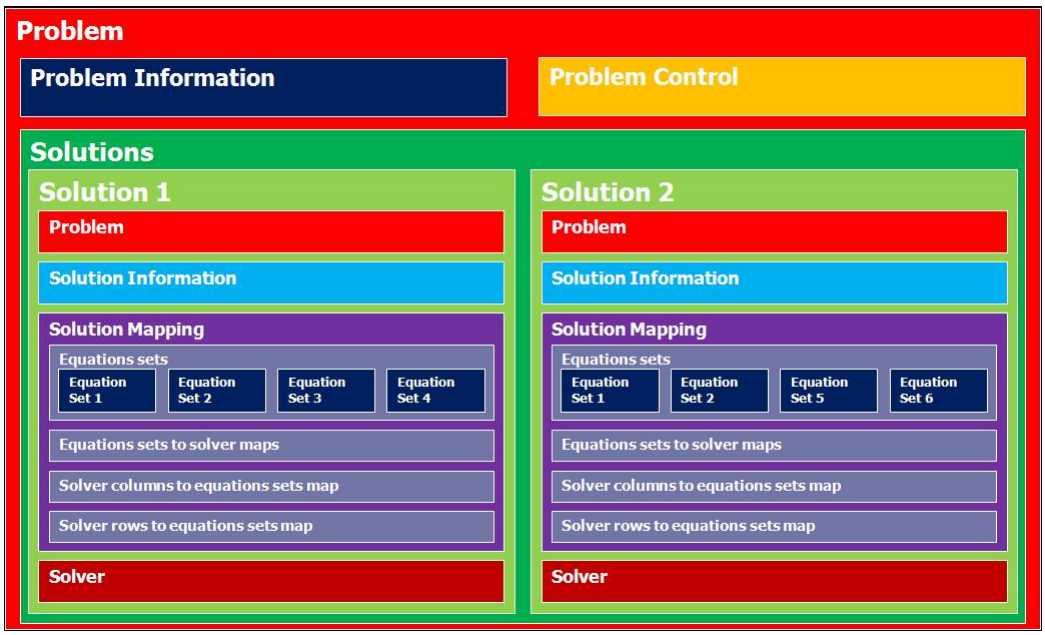


4. Problems

A problem has a number of solutions (each with their solver) inside a problem control loop. Problem associated with region via solution which maps to equations sets and hence links to region. Multiple problems can be in the same region, or multiple regions can woek to solve one problem.

Problem has the following attributes:

- User number
- Global number
- Finished tag
- Class
- Type
- Subtype
- Control pointer
- Number of solutions
- Soultions pointer



4.1. Solutions

Solution has the following attributes:

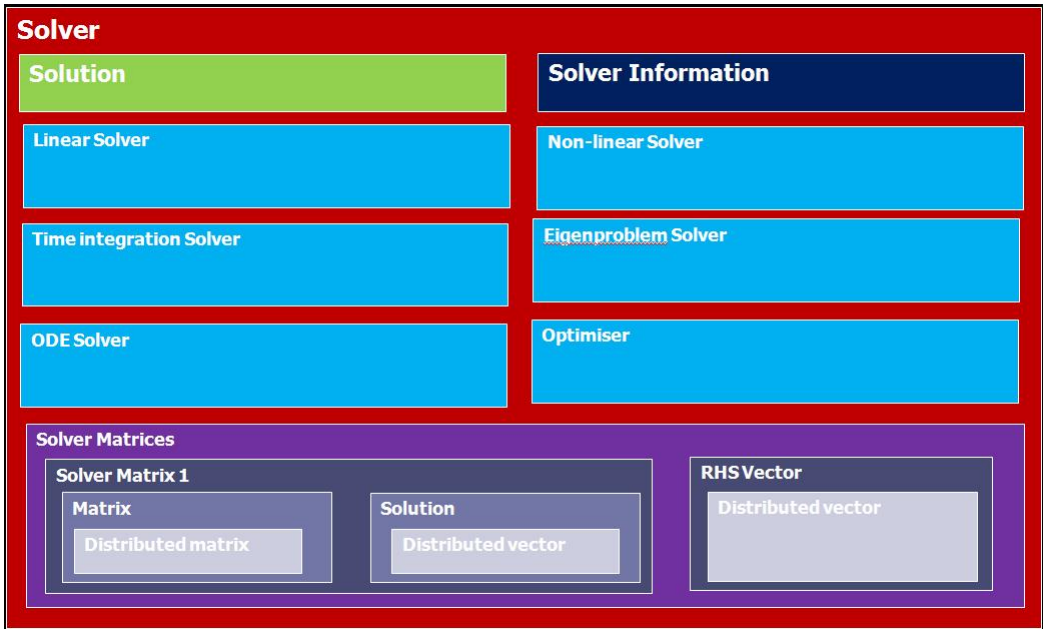
- Solution number
- Finished tag
- Linear solution data pointer
- Nonlinear solution data pointer
- Time (non-static) solution data pointer
- Equations set to add (the next equations set to add)
- Index of added equations set(the last successfully added equations set)
- Soultion mapping(which contains equations sets)
- Solver pointer

4.1.1. Solvers

Solver has the following attributes:

- Solution pointer
- Finished tag
- Solve type
- Output type
- Sparsity type
- Linear solver pointer
- Non-linear solver pointer
- Time integrationn solver pointer

- Eigenproblem solver pointer
- Solver matrices



4.2. Control

Control has the following attributes:

- Problem pointer
- Finished tag
- Control type

Chapter 3. Obtaining the Code and Setting up the Development Environment

1. Obtaining the Code and Libraries

To obtain the openCMISS source you need to check it out from the subversion repository. There are two parts to openCMISS to obtain - openCMISS itself and the various libraries it needs.

In your root openCMISS directory, make the `opencmiss` and `opemcmissextra` directories.

1.1. Obtain the Code

The openCMISS repository is at https://opencmiss.svn.sourceforge.net/svnroot/opencmiss/cm

To check out the main trunk of openCMISS issue the following command in the `opencmiss` directory:
```
svn co https://opencmiss.svn.sourceforge.net/svnroot/opencmiss/cm/trunk cm
```

If you are not familar with subversion, have a look at here [http://svnbook.red-bean.com].

1.2. Obtain the Libraries

The openCMISS libraries repository is at http://www.physiome.ox.ac.uk/svn/opencmissextras/cm

To check out the main trunk of the various libraries required with openCMISS issue the following command in the `opencmissextra` directory:
```
svn co http://www.physiome.ox.ac.uk/svn/opencmissextras/cm/trunk/external/
architecture cm/external/architecture
```

where architecture is the appropriate architecture for the machine. Possible architectures are:

- i386-win32
- i386-win32-debug
- i686-linux
- i686-linux-debug
- x86_64-linux
- x86_64-linux-debug
- rs6000-32-aix
- rs6000-32-aix-debug

Currently, the svn repository for openCMISS libraries is down. An alternative location for the libraries is at hpc. Go to `\\bioengsmb\cmiss\opencmissextras\cm` and copy the necessary files. The folder structure is the same as svn repository.

2. Project Set up

2.1. On AIX 5.3 (HPC)

2.1.1. Set environment

Set environment variable to point to openCMISS

```
setenv OPENCMISS_ROOT=<path to your opencmiss folder>
```

Set environment variable to point to openCMISS-extras
```
setenv OPENCMISSEXTRAS_ROOT=<path to your opencmissextras folder>
```

2.1.2. Set MPI

Create a file called `hostfile.list` in your home directory.

Inside the file, add several lines of “hpc.bioeng.auckland.ac.nz”

In `.rhost` file in the home direcotry, add “hpc <username>”

2.1.3. Compile

Change directory to opencmiss/cm

Change directory to examples/<example%gt;

Use `gmake`.

This should result in a binary that you can run in the `bin/rs6000-32-aix` folder.

2.2. On Ubuntu 8.04

2.2.1. Set environment

Set environment variable to point to opencmiss
```
setenv OPENCMISS_ROOT=<path to your opencmiss folder>
```

Set environment variable to point to opencmiss-extras
```
setenv OPENCMISSEXTRAS_ROOT=<path to your opencmissextras folder>
```

It is also helpful to add the following

```
setenv PATH ${OPENCMISSEXTRAS_ROOT}/cm/external/${archname}/bin:${PATH}

setenv PATH ${OPENCMISS_ROOT}/cm/bin/${archname}:${PATH}
```

where ${archname} is the appropriate architecture e.g., i686-linux, x86_64-linux.

If you are using totalview you will also need to add
```
setenv LM_LICENSE_FILES <path-to-the-flex-directory>
```

2.2.2. Install Compilers

Download Intel Fotran Compiler from here [http://www.intel.com/cd/software/products/asmo-na/eng/282048.htm].

Extract the file and follow the `install.htm` to install

2.2.3. Install MPI

Download MPICH2 from here. [http://www.mcs.anl.gov/research/projects/mpich2/]

Extract the file and follow the `README` to install.

Remember to set the path as suggested.

2.2.4. Compile

To build an example project:

```
make
```

To run the example project:
```
mpd & mpirun -n 2 path/to/the/execution/file
```

To debug the project using TotalView:
```
mpd & mpirun -tv 2 path/to/the/execution/file
```

2.3. On Windows XP (Visual Studio 2005)

2.3.1. Install Compilers

Download Intel Fortran Compiler from here [http://www.intel.com/cd/software/products/asmo-na/eng/278834.htm].

Execute the exe file and follow the installation wizard.

2.3.2. Install MPI

Download MPICH2 from here. [http://www.mcs.anl.gov/research/projects/mpich2/]

You can either download the source archive and follow the `README.windows` file to install or download the installer to install.

Set bin folder to the path

To start the MPI, run `smpd -start` in command window.

NOTE: as from MPICH2 version 1.0.7 the library names have changed. `libmpich2` has now become `libmpi`!

2.3.3. Compile and Debug

Build the Fortran project under the debug mode and generate the `opencmisstest-debug.exe` file.

In the C Project (since the Fortran projects do not support MPI cluster debugger), configure the debugging properties according to this [http://download.microsoft.com/download/6/8/d/68d7d82b-e477-4699-b403-72be2e6218b1/CCS03DebugParallelAppsVS05.doc].

The MPIShim location is in the path similar to C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x86\mpishim.exe.

Debug the C project.

2.3.4. Run

To run the project in the command window:
mpiexec -n 2 -localroot <path to the execution file>

2.4. On Windows Vista (Visual Studio 2008)

2.4.1. Install Compilers

Download Intel Fortran Compiler from here [http://www.intel.com/cd/software/products/asmo-na/eng/278834.htm].

Execute the exe file and follow the installation wizard.

2.4.2. Install MPI

Before you install MPICH2 under Vista you must turn off User Account Control

1. Goto Start -> Control Panel
2. Double-click on User Accounts
3. Click "Turn User Account Control on or off"
4. Untick "Use User Account Control (UAC) to help protect your computer" and click OK
5. Restart your computer.

Download from here [http://www.mcs.anl.gov/research/projects/mpich2/]. Choose the Win32 IA32 (binary) option.

Run the downloaded .msi file. Follow all instructions and install "For everybody".

Once you have installed MPICH2 you can turn User Account Control back on. Follow the instructions above and in 4. tick the "Use User Account Control ...".

NOTE: as from MPICH2 version 1.0.7 the library names have changed. libmpich2 has now become libmpi!

2.4.3. Compile

For each example, go into the VisualopenCMISS_08 folder. Double click the VisualopenCMISS project solution file to lauch Visual Studio.

3. Libraries Build (Optional)

3.1. Compiling PETSc

Note this is assuming you have the Intel Fortran compiler version 10.1.024. Adjust the version string as necessary.

3.1.1. Step1: Linux Environment installation and Compiler Environment Set up

Under windows system:

- Install Cygwin if you need to. Cywin can be found here [http://www.cygwin.com/]. Make sure you include the make and python modules when you install.
- Lauch a Command Prompt Window
- Run the ifortvars.bat batch file to setup your Intel Fortran environment. e.g., "C:\Program Files\Intel\Compiler \Fortran\10.1.024\IA32\Bin\ifortvars.bat"
- Run the Cygwin batch file to setup the unix environment e.g., "C:\Cygwin\Cygwin.bat"

For Linux

TODO

3.1.2. Step2: Compile PETSC

- Change to the opencmissextras PETSc directory e.g., if opencmissextras root is E:\opencmissextras and we are compiling PETSC version petsc-2.3.3-p8 then "cd /cygwin/e/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8"
- If you have MPICH2 version 1.0.7 or greater edit the python/BuildSystem/config/packages/MPI.py file. Find the self.liblist_mpich line. After the line "['fmpich2.lib','mpich2.lib']," add the line "['fmpich2.lib','mpi.lib'],".
- `PETSC_DIR=/cygdrive/e/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8; export PETSC_DIR`
- For a debug install issue the following commands

```
PETSC_ARCH=cygwin-c-debug; export PETSC_ARCH

config/configure.py  --prefix=/cygdrive/e/opencmissextras/cm/external/
i386-win32-debug    --with-shared=no    --with-cc='win32fe    cl'    --
with-fc='win32fe    ifort'    --with-cxx='win32fe    cl'    --download-
f-blas-lapack=1   LIBS=-L'/cygdrive/c/Program\   Files/Intel/Compiler/
Fortran/10.1.024/IA32/Lib' --with-debugging=yes
```

For a non-debug install issue the following commands

```
PETSC_ARCH=cygwin-c-opt; export PETSC_ARCH

config/configure.py  --prefix=/cygdrive/e/opencmissextras/cm/external/
i386-win32 --with-shared=no --with-cc='win32fe cl' --with-fc='win32fe
ifort'  --with-cxx='win32fe  cl'  --download-f-blas-lapack=1  LIBS=-L'/
cygdrive/c/Program\ Files/Intel/Compiler/Fortran/10.1.024/IA32/Lib' --
with-debugging=no
```

- `make -e all`
- `make -e install`

Chapter 4. Object Interface

1. Objectives

- Simple interface for a variety of scripting and programming languages
- User friendly in that the library should sensibly set default parameters to minimise the amount of information the programmer/user has to send
- Should be extensible so that extra parameters can be added at a later stage without causing problems

2. General Rules

- Objects identified by a unique user number
- Objects are initialised with a `OBJECT_INITIALISE` call
- Objects are finalised with a `OBJECT_FINALISE` call
- Objects are created with a pair of `OBJECT_CREATE_START` and `OBJECT_CREATE_FINISH` calls
- Objects are destroyed with a `OBJECT_DESTROY` call
- Object parameters are set with a number of `OBJECT_PARAMETER_SET` calls between the START and FINISH calls
- START call initialises OBJECT and sets default parameters. SET calls modify default parameters. FINISH call finalises the object

3. Example(Basis functions)

```
!Initialise the basis functions.
CALL BASES_INITIALISE(ERR,ERROR,*999)
!Start the creation of a basis with a user number of 1.
!The default is tri-linear Lagrange
CALL BASIS_CREATE_START(1,BASIS,ERR,ERROR,*999)
!Set the number of xi directions to 2.
CALL BASIS_NUMBER_OF_XI_SET(BASIS,2,ERR,ERROR,*999)
!Set the interpolation to be cubic Hermite*quadratic Lagrange.
CALL BASIS_INTERPOLATION_XI_SET(BASIS, &
& (/CUBIC_HERMITE,QUADRATIC_LAGRANGE/),ERR,ERROR,*999)
!Finish the creation of the basis.
CALL BASIS_CREATE_FINISH(BASIS,ERR,ERROR,*999)
!Destroy the basis with the user number 1.
CALL BASIS_DESTROY(1,ERR,ERROR,*999)
!Finalise the basis functions.
CALL BASES_FINALISE(ERR,ERROR,*999)
```

Chapter 5. Examples

1. Laplace equation

```
!Initialise CMISS.
CALL CMISS_INITIALISE(ERR,ERROR,*999)

!Start the creation a coordinate system (default 3D RC)
CALL COORDINATE_SYSTEM_CREATE_START(1,COORDINATE_SYSTEM,ERR,ERROR,*999)
!Set the coordinate system to be 2D
CALL COORDINATE_SYSTEM_DIMENSION_SET(COORDINATE_SYSTEM,2,ERR,ERROR,*999)
!Finish the creation a coordinate system
CALL COORDINATE_SYSTEM_CREATE_FINISH(COORDINATE_SYSTEM,ERR,ERROR,*999)

!Start the creation of a region with a coordinate system.
CALL REGION_CREATE_START(1,COORDINATE_SYSTEM,REGION,ERR,ERROR,*999)
!Finish the creation of a region
CALL REGION_CREATE_FINISH(REGION,ERR,ERROR,*999)

!Start the creation of a basis (default tri-linear Lagrange)
CALL BASIS_CREATE_START(1,BASIS,ERR,ERROR,*999)
!Set the number of Xi directions to 2 (bi-linear Lagrange)
CALL BASIS_NUMBER_XI_SET(BASIS,2,ERR,ERROR,*999)
!Finish the creation of a basis
CALL BASIS_CREATE_FINISH(BASIS,ERR,ERROR,*999)



!Start the creation of a generated mesh in the region
CALL GENERATED_MESH_CREATE_START(1,REGION,GENERATED_MESH,ERR,ERROR,*999)
!Set up a regular 100x100 mesh
CALL GENERATED_MESH_TYPE_SET(GENERATED_MESH,GENERATED_MESH_REGULAR_TYPE, &
    & ERR,ERROR,*999)
CALL GENERATED_MESH_BASIS_SET(GENERATED_MESH,BASIS,ERR,ERROR,*999)
CALL GENERATED_MESH_EXTENT_SET(GENERATED_MESH,(/2.0_DP,1.0_DP/),ERR,ERROR,*999)
CALL GENERATED_MESH_NUMBER_OF_ELEMENTS_SET(GENERATED_MESH,(/100,100/), &
    & ERR,ERROR,*999)
!Finish the creation of a generated mesh in the region
CALL GENERATED_MESH_CREATE_FINISH(GENERATED_MESH,MESH,ERR,ERROR,*999)

!Start the creation a decomposition on a mesh
CALL DECOMPOSITION_CREATE_START(1,MESH,DECOMPOSITION,ERR,ERROR,*999)
!Set the decomposition so that the domains are calculated
CALL DECOMPOSITION_TYPE_SET(DECOMPOSITION,DECOMPOSITION_CALCULATED_TYPE, &
    & ERR,ERROR,*999)
CALL DECOMPOSITION_NUMBER_OF_DOMAINS_SET(DECOMPOSITION,5,ERR,ERROR,*999)
!Finish the creation of decomposition
CALL DECOMPOSITION_CREATE_FINISH(DECOMPOSITION,ERR,ERROR,*999)



!Start the creation of a field on a region (default geometric)
CALL FIELD_CREATE_START(1,REGION,GEOMETRIC_FIELD,ERR,ERROR,*999)
!Set the decomposition the field will use
CALL FIELD_MESH_DECOMPOSITION_SET(GEOMETRIC_FIELD,DECOMPOSITION,ERR,ERROR,*999)
!Set mesh components each of the field variable components will use. NB. These
```

```
!are shown for example as each field variable component will default to the
!first mesh component.
CALL FIELD_COMPONENT_MESH_COMPONENT_SET(GEOMETRIC_FIELD, &
    & FIELD_STANDARD_VARIABLE_TYPE,1,1,ERR,ERROR,*999)
CALL FIELD_COMPONENT_MESH_COMPONENT_SET(GEOMETRIC_FIELD, &
    & FIELD_STANDARD_VARIABLE_TYPE,2,1,ERR,ERROR,*999)
!Finish the creation of a field
CALL FIELD_CREATE_FINISH(GEOMETRIC_FIELD,ERR,ERROR,*999)

!Set a field value in a parameter set
CALL FIELD_PARAMETER_SET_UPDATE_NODE(GEOMETRIC_FIELD, &
    & FIELD_VALUES_SET_TYPE,1,1,1,FIELD_STANDARD_VARIABLE_TYPE, &
    & 1.0_DP,ERR,ERROR,*999)
!Start the update of field values in a parameter set
CALL FIELD_PARAMETER_SET_UPDATE_START(GEOMETRIC_FIELD, &
    & FIELD_VALUES_SET_TYPE,ERR,ERROR,*999)
!!DO SOME CALCULATIONS
!Finish the update of field values in a parameter set
CALL FIELD_PARAMETER_SET_UPDATE_FINISH(GEOMETRIC_FIELD, &
    & FIELD_VALUES_SET_TYPE,ERR,ERROR,*999)



!Start the creation of an equations set on a region
CALL EQUATIONS_SET_CREATE_START(1,REGION,GEOMETRIC_FIELD,EQUATIONS_SET, &
    & ERR,ERROR,*999)
!Set the equations set to be standard Laplace#s equation problem
CALL EQUATIONS_SET_SPECIFICATION_SET(EQUATIONS_SET, &
    & EQUATIONS_SET_CLASSICAL_FIELD_CLASS,EQUATIONS_SET_LAPLACE_EQUATION_TYPE, &
    & EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE,ERR,ERROR,*999)
!Finish the creation of an equations set
CALL EQUATIONS_SET_CREATE_FINISH(EQUATIONS_SET,ERR,ERROR,*999)

!Start the creation of the equations set dependent field
CALL EQUATIONS_SET_DEPENDENT_CREATE_START(EQUATIONS_SET,ERR,ERROR,*999)
!Finish the create of the problems dependent field
CALL EQUATIONS_SET_DEPENDENT_CREATE_FINISH(EQUATIONS_SET,ERR,ERROR,*999)

!Start the creation of the equations set fixed conditions
CALL EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(EQUATIONS_SET,ERR,ERROR,*999)
!Set BC#s
CALL EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF(EQUATIONS_SET,1, &
    & EQUATIONS_SET_FIXED_BOUNDARY_CONDITION,0.0_DP,ERR,ERROR,*999)
CALL EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF(PROBLEM,10, &
    & EQUATIONS_SET_FIXED_BOUNDARY_CONDITION,1.0_DP,ERR,ERROR,*999)
!Finish the create of the equations set fixed conditions
CALL EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(EQUATIONS_SET,ERR,ERROR,*999)



 !Start the creation of the equations for the equations set
CALL EQUATIONS_SET_EQUATIONS_CREATE_START(EQUATIONS_SET,ERR,ERROR,*999)
!Set the equations matrices sparsity type
CALL EQUATIONS_SET_SPARSITY_TYPE_SET(EQUATIONS_SET,EQUATIONS_SET_SPARSE_MATRICES, &
    & ERR,ERROR,*999)
!Set the equations output type
CALL EQUATIONS_SET_OUTPUT_TYPE_SET(EQUATIONS_SET,EQUATIONS_SET_TIMING_OUTPUT, &
    & ERR,ERROR,*999)
```

```
!Finish the creation of the equations for the equations set
CALL EQUATIONS_SET_EQUATIONS_CREATE_FINISH(EQUATIONS_SET,ERR,ERROR,*999)

!Start the creation of a problem
CALL PROBLEM_CREATE_START(1,PROBLEM,ERR,ERROR,*999)
!Set the problem set to be standard Laplace#s equation problem
CALL PROBLEM_SPECIFICATION_SET(PROBLEM,PROBLEM_CLASSICAL_FIELD_CLASS, &
    & PROBLEM_LAPLACE_EQUATION_TYPE,PROBLEM_STANDARD_LAPLACE_SUBTYPE,ERR,ERROR,*999)
!Finish the creation of a problem
CALL PROBLEM_CREATE_FINISH(PROBLEM,ERR,ERROR,*999)

!Start the creation of the control for a problem
CALL PROBLEM_CONTROL_CREATE_START(PROBLEM,ERR,ERROR,*999)
!Finish the creation of the control a problem
CALL PROBLEM_CONTROL_CREAT_CREATE_FINISH(PROBLEM,ERR,ERROR,*999)



 !Start the creation of the solutions for the problem
CALL PROBLEM_SOLUTIONS_CREATE_START(PROBLEM,ERR,ERROR,*999)
!Add in the equations set
CALL PROBLEM_SOLUTIONS_EQUATIONS_SET_ADD(PROBLEM,1,EQUATIONS_SET,ERR,ERROR,*999)
!Set the solutions output type
CALL PROBLEM_SOLUTIONS_OUTPUT_TYPE_SET(PROBLEM,1,PROBLEM_SOLUTION_MATRIX_OUTPUT, &
    & ERR,ERROR,*999)
!Finish the creation of the solutions for the problem
CALL PROBLEM_SOLUTIONS_CREATE_FINISH(PROBLEM,ERR,ERROR,*999)

!Start the creation of the solvers for the problem
CALL PROBLEM_SOLVER_CREATE_START(PROBLEM,ERR,ERROR,*999)
!Set the type of preconditioner
CALL PROBLEM_SOLVER_ITERATIVE_PRECONDITIONER_TYPE_SET(PROBLEM,1, &
    & PROBLEM_SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER,ERR,ERROR,*999)
!Set the number of iterations
CALL PROBLEM_SOLVER_ITERATIVE_MAX_ITERATIONS_SET(PROBLEM,1,100,ERR,ERROR,*999)
!Set the output type
CALL PROBLEM_SOlVER_OUTPUT_TYPE_SET(PROBLEM,1,PROBLEM_SOLVER_TIMING_OUTPUT, &
    & ERR,ERROR,*999)
!Finish the creation of the solvers for the problem
CALL PROBLEM_SOLVER_CREATE_FINISH(PROBLEM,ERR,ERROR,*999)

!Solve the problem
CALL PROBLEM_SOLVE(PROBLEM,ERR,ERROR,*999)

!Finalise CMISS.
CALL CMISS_FINALISE(ERR,ERROR,*999)
```

Chapter 6. Library Commands

1. Top level

1.1. cmiss

CMISS_FINALISE(ERR, ERROR,*)

- Description: Finalises CMISS.
- Parameters
  - ERR: The error string
  - ERROR: The error code
  - *:

CMISS_INITIALISE(ERR, ERROR,*)

- Description: Initialises CMISS.
- Parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

CMISS_WRITE_ERROR(ERR, ERROR)

- Description: Writes the error string to screen.
- Parameters
  - ERR: The error code
  - ERROR: The error string

2. Basis functions

BASIS_ROUTINES::BASIS_INTERPOLATE_GAUSS

BASIS_ROUTINES::BASIS_INTERPOLATE_XI

BASIS_ROUTINES::BASIS_INTERPOLATION_XI_SET

BASIS_ROUTINES::BASIS_NUMBER_OF_XI_SET

BASIS_ROUTINES::BASIS_TYPE_SET

BASIS_ROUTINES::BASIS_COLLAPSED_XI_SET

BASIS_ROUTINES::BASIS_QUADRATURE_NUMBER_OF_GAUSS_XI_SET

BASIS_ROUTINES::BASIS_QUADRATURE_ORDER_SET

BASIS_ROUTINES::BASIS_QUADRATURE_TYPE_SET

BASIS_ROUTINES::SIMPLEX_LINEAR_EVALUATE

BASIS_ROUTINES::SIMPLEX_QUADRATIC_EVALUATE

BASIS_ROUTINES::SIMPLEX_CUBIC_EVALUATE

BASIS_ROUTINES::BASIS_LHTP_BASIS_EVALUATE
BASES_FINALISE(ERR, ERROR,*)

- Description: Finalises the bases and deallocates all memory.
- Parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

BASES_INITIALISE(ERR, ERROR,*)

- Description: Initialises the bases.
- Parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

BASIS_CREATE_FINISH(BASIS, ERR, ERROR,*)

- Description: Finishes the creation of a new basis.
- Parameters
  - BASIS: A pointer to the basis to finish the creation of
  - ERR: The error code
  - ERROR: The error string
  - *:

BASIS_CREATE_START(USER_NUMBER, BASIS, ERR, ERROR,*)

- Description: Starts the creation of a new basis.
- Parameters
  - USER_NUMBER: The user number of the basis to start the creation of
  - BASIS: A pointer to the created basis. Must not be associated on entry.
  - ERR: The error code
  - ERROR: The error string
  - *:

BASIS_NUMBER_OF_NODES_GET(BASIS, NUMBER_OF_NODES, ERR, ERROR,*)

- Description: Returns the number of local nodes in the specified basis.
- Parameters
  - BASIS: A pointer to the basis to get the number of nodes

- NUMBER_OF_NODES: On exit, the number of local nodes in the basis
- ERR: The error code
- ERROR: The error string
- *:

BASIS_USER_NUMBER_FIND(USER_NUMBER, BASIS, ERR, ERROR,*)

- Description: Finds and returns in BASIS a pointer to the basis with the number given in USER_NUMBER. If no basis with that number exits BASIS is left nullified.
- Parameters
  - USER_NUMBER: The user number of the basis to find.
  - BASIS: On exit, a pointer to the found basis. If no basis with the given user number exists the pointer is NULL.
  - ERR: The error code
  - ERROR: The error string
  - *:

3. Coordinate Systems

COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE

COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE

COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC

COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC

COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE

COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET

COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET

COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET

COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET

COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET

COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET

COORDINATE_ROUTINES::DXZ

COORDINATE_ROUTINES::D2ZX

COORDINATE_ROUTINES::DZX

COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_CONVERT_TO_RC

COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY
COORDINATE_METRICS_CALCULATE(COORDINATE_SYSTEM, JACOBIAN_TYPE, METRICS, ERR, ERROR,*)

- Description:

- Parameters
  - COORDINATE_SYSTEM:
  - JACOBIAN_TYPE:
  - METRICS:
  - ERR:
  - ERROR:
  - *:

COORDINATE_SYSTEM_FOCUS_GET(COORDINATE_SYSTEM, ERR, ERROR)

- Description:
- Parameters
  - COORDINATE_SYSTEM:
  - ERR:
  - ERROR:

COORDINATE_SYSTEM_CREATE_START(USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)

- Description:
- Parameters
  - USER_NUMBER:
  - COORDINATE_SYSTEM:
  - ERR:
  - ERROR:
  - *:

COORDINATE_SYSTEM_CREATE_FINISH(COORDINATE_SYSTEM, ERR, ERROR,*)

- Description:
- Parameters
  - COORDINATE_SYSTEM:
  - ERR:
  - ERROR:
  - *:

COORDINATE_DERIVATIVE_NORM(COORDINATE_SYSTEM, PART_DERIV_INDEX, INTERPOLATED_POINT, DERIV_NORM, ERR, ERROR,*)

- Description:
- Parameters

- COORDINATE_SYSTEM:
- PART_DERIV_INDEX:
- INTERPOLATED_POINT:
- DERIV_NORM:
- ERR:
- ERROR:
- *:

COORDINATE_INTERPOLATION_ADJUST(COORDINATE_SYSTEM, PARTIAL_DERIVATIVE_INDEX, VALUE, ERR, ERROR,*)

- Description:
- Parameters
  - COORDINATE_SYSTEM:
  - PARTIAL_DERIVATIVE_INDEX:
  - VALUE:
  - ERR:
  - ERROR:
  - *:

COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(COORDINATE_SYSTEM, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

- Description:
- Parameters
  - COORDINATE_SYSTEM:
  - INTERPOLATION_PARAMETERS:
  - ERR:
  - ERROR:
  - *:

COORDINATE_SYSTEM_USER_NUMBER_FIND(USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)

- Description:
- Parameters
  - USER_NUMBER:
  - COORDINATE_SYSTEM:
  - ERR:

- ERROR:
- *:

COORDINATE_SYSTEMS_FINALISE(ERR, ERROR,*)

- Description:
- Parameters
  - ERR:
  - ERROR:
  - *:

COORDINATE_SYSTEMS_INITIALISE(ERR, ERROR,*)

- Description:
- Parameters
  - ERR:
  - ERROR:
  - *:

4. Regions

REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET

REGION_ROUTINES::REGION_LABEL_SET
REGION_COORDINATE_SYSTEM_GET(REGION, ERR, ERROR)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:

REGION_CREATE_FINISH(REGION, ERR, ERROR,*)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:
  - *:

REGION_CREATE_START(USER_NUMBER, REGION, ERR, ERROR,*)

- Description:

- Parameters
  - USER_NUMBER:
  - REGION:
  - ERR:
  - ERROR:
  - *:

REGION_LABEL_GET(REGION, ERR, ERROR)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:

REGION_SUB_REGION_CREATE_START(USER_NUMBER, PARENT_REGION, SUB_REGION, ERR, ERROR,*)

- Description:
- Parameters
  - USER_NUMBER:
  - PARENT_REGION:
  - SUB_REGION:
  - ERR:
  - ERROR:
  - *:

REGION_SUB_REGION_CREATE_FINISH(REGION, ERR, ERROR,*)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:
  - *:

REGION_USER_NUMBER_FIND(USER_NUMBER, REGION, ERR, ERROR,*)

- Description:
- Parameters
  - USER_NUMBER:

- REGION:
- ERR:
- ERROR:
- *:

REGIONS_INITIALISE(ERR, ERROR,*)

- Description:
- Parameters
  - ERR:
  - ERROR:
  - *:

REGIONS_FINALISE(ERR, ERROR,*)

- Description:
- Parameters
  - ERR:
  - ERROR:
  - *:

4.1. Node

NODE_CHECK_EXISTS(USER_NUMBER, REGION, NODE_EXISTS, GLOBAL_NUMBER, ERR, ERROR,*)

- Description:
- Parameters
  - USER_NUMBER:
  - REGION:
  - NODE_EXISTS:
  - GLOBAL_NUMBER:
  - ERR:
  - ERROR:
  - *:

NODE_DESTROY(NODE, ERR, ERROR,*)

- Description:
- Parameters
  - NODE:
  - ERR:

- ERROR:
- *:

NODE_INITIAL_POSITION_SET(GLOBAL_NUMBER, INITIAL_POSITION, NODES, ERR, ERROR,*)

- Description:
- Parameters
  - GLOBAL_NUMBER:
  - INITIAL_POSITION:
  - NODES:
  - ERR:
  - ERROR:
  - *:

NODE_NUMBER_SET(GLOBAL_NUMBER, USER_NUMBER, NODES, ERR, ERROR,*)

- Description:
- Parameters
  - GLOBAL_NUMBER:
  - USER_NUMBER:
  - NODES:
  - ERR:
  - ERROR:
  - *:

NODES_CREATE_FINISH(REGION, ERR, ERROR,*)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:
  - *:

NODES_CREATE_START(NUMBER_OF_NODES, REGION, NODES, ERR, ERROR,*)

- Description:
- Parameters
  - NUMBER_OF_NODES:

- REGION:
- NODES:
- ERR:
- ERROR:
- *:

NODES_FINALISE(REGION, ERR, ERROR,*)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:
  - *:

NODES_INITIALISE(REGION, ERR, ERROR,*)

- Description:
- Parameters
  - REGION:
  - ERR:
  - ERROR:
  - *:

4.2. Mesh

MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET

MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET
DECOMPOSITION_CREATE_FINISH(MESH, DECOMPOSITION, ERR, ERROR,*)

- Description: Finishes the creation of a domain decomposition on a given mesh.
- Parameters
  - MESH: A pointer to the mesh to decompose.
  - DECOMPOSITION: A pointer to the decomposition to finish creating
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITION_CREATE_START(USER_NUMBER, MESH, DECOMPOSITION, ERR, ERROR,*)

- Description: Starts the creation of a domain decomposition for a given mesh.
- Parameters
  - USER_NUMBER: The user number of the decomposition
  - MESH: A pointer to the mesh to decompose
  - DECOMPOSITION: On return a pointer to the created decomposition. Must not be associated on entry.
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITION_DESTROY(USER_NUMBER, MESH, ERR, ERROR,*)

- Description: Destroys a domain decomposition identified by a user number and deallocates all memory.
- Parameters
  - USER_NUMBER: The user number of the decomposition to destroy.
  - MESH: A pointer to the mesh containing the decomposition.
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITION_NUMBER_OF_DOMAINS_SET(DECOMPOSITION, NUMBER_OF_DOMAINS, ERR, ERROR,*)

- Description: Sets/changes the number of domains for a decomposition.
- Parameters
  - DECOMPOSITION: A pointer to the decomposition to set the number of domains for.
  - NUMBER_OF_DOMAINS: The number of domains to set.
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITION_TYPE_SET(DECOMPOSITION, TYPE, ERR, ERROR,*)

- Description: Sets/changes the decomposition type for a decomposition.
- Parameters
  - DECOMPOSITION: A pointer to the decomposition to set the type for
  - TYPE: The decomposition type to set MESH_ROUTINES::DecompositionTypes,MESH_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITION_USER_NUMBER_FIND(USER_NUMBER, MESH, DECOMPOSITION, ERR, ERROR,*)

- Description: Finds and returns in DECOMPOSITION a pointer to the decomposition identified by USER_NUMBER in the given MESH. If no decomposition with that USER_NUMBER exists DECOMPOSITION is left nullified.
- Parameters
  - USER_NUMBER: The user number of the decomposition to find
  - MESH: A pointer to the mesh containing the decomposition to find
  - DECOMPOSITION: On return a pointer to the decomposition with the specified user number. If no decomposition with that user number exists then DECOMPOSITION is returned NULL.
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITIONS_FINALISE(MESH, ERR, ERROR,*)

- Description: Finalises the domain decompositions for a given mesh.
- Parameters
  - MESH: A pointer to the mesh to finalise the decomposition for
  - ERR: The error code
  - ERROR: The error string
  - *:

DECOMPOSITIONS_INITIALISE(MESH, ERR, ERROR,*)

- Description: Initialises the domain decompositions for a given mesh.
- Parameters
  - MESH: A pointer to the mesh to initialise the decompositions for
  - ERR: The error code
  - ERROR: The error string
  - *:

MESH_CREATE_FINISH(REGION, MESH, ERR, ERROR,*)

- Description: Finishes the process of creating a mesh on a region.
- Parameters
  - REGION: A pointer to the region containing the mesh
  - MESH: A pointer to the mesh to finish creating
  - ERR: The error code
  - ERROR: The error string

- *:

MESH_CREATE_REGULAR(USER_NUMBER, REGION, ORIGIN, MAXIMUM_EXTENT, NUMBER_ELEMENTS_XI, BASIS, MESH, ERR, ERROR,*)

- Description: Creates the regular mesh with the given USER_NUMBER in the specifed REGION. The mesh starts at the ORIGIN(:) and has a maximum extent position of MAXIMUM_EXTENT(:) with the NUMBER_OF_ELEMENTS(:) in each direction. Each element is of the specified BASIS type. A pointer to the finished mesh is returned in MESH.
- Parameters
  - USER_NUMBER: The user number of the mesh to create
  - REGION: A pointer to the region containing the mesh
  - ORIGIN: ORIGIN(i). ORIGIN(i) contains the i'th coordinate in the region of the origin of the regular mesh.
  - MAXIMUM_EXTENT: MAXIMUM_EXTENT(i). MAXIMUM_EXTENT(i) contains the i'th extent (or size) of the regular mesh.
  - NUMBER_ELEMENTS_XI: NUMBER_ELEMENTS_XI(i). NUMBER_ELEMENTS_XI(i) contains the number of elements in the i'th direction
  - BASIS: A pointer to the basis to use for each element in the regular mesh
  - MESH: On return, a pointer to the generated mesh
  - ERR: The error code
  - ERROR: The error string
  - *:

MESH_CREATE_START(USER_NUMBER, REGION, NUMBER_OF_DIMENSIONS, MESH, ERR, ERROR,*)

- Description: Starts the process of creating a mesh defined by a user number with the specified NUMBER_OF_DIMENSIONS in the region identified by REGION.
- Parameters
  - USER_NUMBER: The user number of the mesh to create
  - REGION: A pointer to the region to create the mesh on
  - NUMBER_OF_DIMENSIONS: The number of dimensions in the mesh.
  - MESH: On exit, a pointer to the created mesh. Must not be associated on entry.
  - ERR: The error code
  - ERROR: The error string
  - *:

MESH_DESTROY(USER_NUMBER, REGION, ERR, ERROR,*)

- Description: Destroys the mesh identified by a user number on the given region and deallocates all memory.
- Parameters

- USER_NUMBER: The user number of the mesh to destroy
- REGION: A pointer to the region containing the mesh
- ERR: The error code
- ERROR: The error string
- *:

MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(MESH, MESH_COMPONENT_NUMBER, ERR, ERROR,*)

- Description: Finishes the process of creating elements for a specified mesh component in a mesh topology.
- Parameters
  - MESH: A pointer to the mesh to finish creating the elements for
  - MESH_COMPONENT_NUMBER: The mesh component number to finish creating the elements for
  - ERR: The error code
  - ERROR: The error string
  - *:

MESH_TOPOLOGY_ELEMENTS_CREATE_START(MESH, MESH_COMPONENT_NUMBER, BASIS, ELEMENTS, ERR, ERROR,*)

- Description: Starts the process of creating elements in the mesh component identified by MESH and component_idx. The elements will be created with a default basis of BASIS. ELEMENTS is the returned pointer to the MESH_ELEMENTS data structure.
- Parameters
  - MESH: A pointer to the mesh to start creating the elements on
  - MESH_COMPONENT_NUMBER: The mesh component number
  - BASIS: A pointer to the default basis to use
  - ELEMENTS: On return, a pointer to the created mesh elements
  - ERR: The error code
  - ERROR: The error string
  - *:

MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(GLOBAL_NUMBER, ELEMENTS, BASIS, ERR, ERROR,*)

- Description: Changes/sets the basis for a mesh element identified by a given global number.
- Parameters
  - GLOBAL_NUMBER: The global number of the element to set the basis for
  - ELEMENTS: A pointer to the elements to set the basis for before number?
  - BASIS: A pointer to the basis to set

- ERR: The error code
- ERROR: The error string
- *:

MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(GLOBAL_NUMBER, ELEMENTS, USER_ELEMENT_NODES, ERR, ERROR,*)

- Description: Changes/sets the element nodes for a mesh element identified by a given global number.
- Parameters
  - GLOBAL_NUMBER: The global number of the element to set the nodes for
  - ELEMENTS: A pointer to the elements to set before number?
  - USER_ELEMENT_NODES: USER_ELEMENT_NODES(i). USER_ELEMENT_NODES(i) is the i'th user node number for the element
  - ERR: The error code
  - ERROR: The error string
  - *:

MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(GLOBAL_NUMBER, USER_NUMBER, ELEMENTS, ERR, ERROR,*)

- Description: Changes/sets the user number for a global element identified by a given global number.
- Parameters
  - GLOBAL_NUMBER: The global number of the elements to set.
  - USER_NUMBER: The user number of the element to set
  - ELEMENTS: A pointer to the elements to set the user number for This should be the first parameter.
  - ERR: The error code
  - ERROR: The error string
  - *:

MESHES_FINALISE(REGION, ERR, ERROR,*)

- Description: Finalises the meshes in the given region.
- Parameters
  - REGION: A pointer to the region to finalise the meshes for.
  - ERR: The error code
  - ERROR: The error string
  - *:

MESHES_INITIALISE(REGION, ERR, ERROR,*)

- Description: Initialises the meshes for the given region.

- Parameters
  - REGION: A pointer to the region to initialise the meshes for.
  - ERR: The error code
  - ERROR: The error string
  - *:

4.3. Field

FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_SET

FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET

FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET

FIELD_ROUTINES::FIELD_DIMENSION_SET

FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET

FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET

FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET

FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET

FIELD_ROUTINES::FIELD_SCALING_TYPE_SET

FIELD_ROUTINES::FIELD_TYPE_SET
FIELD_CREATE_FINISH(REGION, FIELD, ERR, ERROR,*)

- Description: Finishes the creation of a field on a region.
- Parameters
  - REGION: A pointer to the region containing the field
  - FIELD: A pointer to the field to finish the creation of
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_CREATE_START(USER_NUMBER, REGION, FIELD, ERR, ERROR,*)

- Description: Starts the creation of a field defined by a user number in the specified region.
- Parameters

- USER_NUMBER: The user number for the field
- REGION: A pointer to the region in which to create the field
- FIELD: On return a pointer to the field being created
- ERR: The error code
- ERROR: The error string
- *:

FIELD_DESTROY(FIELD, ERR, ERROR,*)

- Description: Destroys a field identified by a pointer to a field.
- Parameters
  - FIELD: A pointer to the field to destroy
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATE_GAUSS(PARTIAL_DERIVATIVE_TYPE, QUADRATURE_SCHEME, GAUSS_POINT_NUMBER, INTERPOLATED_POINT, ERR, ERROR,*)

- Description: Interpolates a field at a gauss point to give an interpolated point. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name XEXG, ZEXG.
- Parameters
  - PARTIAL_DERIVATIVE_TYPE: The partial derivative type of the provided field interpolation
  - QUADRATURE_SCHEME: The quadrature scheme of the Gauss points BASIS_ROUTINES::QuadratureSchemes,BASIS_ROUTINES
  - GAUSS_POINT_NUMBER: The number of the Gauss point to interpolate the field at
  - INTERPOLATED_POINT: The pointer to the interpolated point which will contain the field interpolation information at the specified Gauss point
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATE_XI(PARTIAL_DERIVATIVE_TYPE, XI, INTERPOLATED_POINT, ERR, ERROR,*)

- Description: Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.

- Parameters
  - PARTIAL_DERIVATIVE_TYPE: The partial derivative type of the provide field interpolation
  - XI: XI(ni). The ni'th Xi coordinate to evaluate the field at
  - INTERPOLATED_POINT: The pointer to the interpolated point which will contain the field interpolation information at the specified Xi point
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATED_POINT_FINALISE(INTERPOLATED_POINT, ERR, ERROR,*)

- Description: Finalises the interpolated point and deallocates all memory.
- Parameters
  - INTERPOLATED_POINT: A pointer to the interpolated point to finalise
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATED_POINT_INITIALISE(INTERPOLATION_PARAMETERS, INTERPOLATED_POINT, ERR, ERROR,*)

- Description: Initialises the interpolated point for an interpolation parameters.
- Parameters
  - INTERPOLATION_PARAMETERS: A pointer to the interpolation parameters to initialise the interpolated point for
  - INTERPOLATED_POINT: On exit, A pointer to the interpolated point that has been initialised
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATED_POINT_METRICS_CALCULATE(JACOBIAN_TYPE, INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

- Description: Calculates the interpolated point metrics and the associated interpolated point.
- Parameters
  - JACOBIAN_TYPE: The Jacobian type of the calculation COORDINATE_ROUTINES_JacobianTypes,COORDINATE_ROUTINES
  - INTERPOLATED_POINT_METRICS: A pointer to the interpolated point metrics
  - ERR: The error code
  - ERROR: The error string

- *:

FIELD_INTERPOLATED_POINT_METRICS_FINALISE(INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

- Description: Finalises the interpolated point metrics and deallocates all memory.
- Parameters
  - INTERPOLATED_POINT_METRICS: A pointer to the interpolated point metrics to finalise
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(INTERPOLATED_POINT, INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

- Description: Initialises the interpolated point metrics for an interpolated point.
- Parameters
  - INTERPOLATED_POINT:
  - INTERPOLATED_POINT_METRICS: On exit, a pointer to the interpolated point metrics that have been initialised
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(PARAMETER_SET_NUMBER, ELEMENT_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

- Description: Gets the interpolation parameters for a particular element. Old CMISS name XPXE, ZPZE.
- Parameters
  - PARAMETER_SET_NUMBER: The field parameter set number to get the element parameters for
  - ELEMENT_NUMBER: The element number to get the element parameters for
  - INTERPOLATION_PARAMETERS: A pointer to the interpolation parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATION_PARAMETERS_FINALISE(INTERPOLATION_PARAMETERS, ERR, ERROR,*)

- Description: Finalises the interpolation parameters and deallocates all memory.
- Parameters
  - INTERPOLATION_PARAMETERS: A pointer to the interpolation parameters to finalise

- ERR: The error code
- ERROR: The error string
- *:

FIELD_INTERPOLATION_PARAMETERS_INITIALISE(FIELD, VARIABLE_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

- Description: Initialises the interpolation parameters for a field variable.
- Parameters
  - FIELD: A pointer to the field to initialise the interpolation parameters for
  - VARIABLE_NUMBER: The field variable number to initialise the interpolation parameters for
  - INTERPOLATION_PARAMETERS: On exit, a pointer to the initialised interpolation parameters.
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_INTERPOLATION_PARAMETERS_LINE_GET(PARAMETER_SET_NUMBER, LINE_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

- Description: Gets the interpolation parameters for a particular line. Old CMISS name XPXE, ZPZE.
- Parameters
  - PARAMETER_SET_NUMBER: The field parameter set number to get the line parameters for
  - LINE_NUMBER: The line number to get the line parameters for
  - INTERPOLATION_PARAMETERS: A pointer to the interpolation parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_GEOMETRIC_PARAMETERS_UPDATE_FROM_INITIAL_MESH(FIELD, ERR, ERROR,*)

- Description: Updates the geometric field parameters from the initial nodal positions of the mesh. Any derivative values for the nodes are calculated from an average straight line approximation.
- Parameters
  - FIELD: A pointer to the field to update the geometric parameters for
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_NEXT_NUMBER_FIND(REGION, NEXT_NUMBER, ERR, ERROR,*)

- Description: Finds the next available user number for the fields defined on the given region.

- Parameters
  - REGION: A pointer to the region
  - NEXT_NUMBER: On exit, the next field user number in the region
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_ADD(FIELD, FIELD_FROM_SET_TYPE, FIELD_TO_SET_TYPE, ERR, ERROR,*)

- Description: Adds the parameter set from one parameter set type to another parameter set type.
- Parameters
  - FIELD: A pointer to the field to add the parameter sets for
  - FIELD_FROM_SET_TYPE: The field parameter set identifier to add the parameters from
  - FIELD_TO_SET_TYPE: The field parameter set identifier to add the parameters to
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_COPY(FIELD, FIELD_FROM_SET_TYPE, FIELD_TO_SET_TYPE, ERR, ERROR,*)

- Description: Copys the parameter set from one parameter set type to another parameter set type.
- Parameters
  - FIELD: A pointer to the field to copy the parameters set for
  - FIELD_FROM_SET_TYPE: The field parameter set identifier to copy the parameters from
  - FIELD_TO_SET_TYPE: The field parameter set identifier to copy the parameters to
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_CREATE(FIELD, FIELD_SET_TYPE, ERR, ERROR,*)

- Description: Creates a new parameter set of type set type for a field.
- Parameters
  - FIELD: A pointer to the field to create the parameter set for
  - FIELD_SET_TYPE: The field parameter set identifier
  - ERR: The error code
  - ERROR: The error string

- *:

FIELD_PARAMETER_SET_GET(FIELD, FIELD_SET_TYPE, PARAMETERS, ERR, ERROR,*)

- Description: Returns a pointer to the specified field parameter set array. The pointer must be restored with a call to FIELD_PARAMETER_SET_RESTORE call. Note: the values can be used for read operations but a FIELD_PARAMETER_SET_UPDATE call must be used to change any values.
- Parameters
  - FIELD: A pointer to the field to get the parameter set from
  - FIELD_SET_TYPE: The field parameter set identifier
  - PARAMETERS: On exit, a pointer to the field parameter set data
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_RESTORE(FIELD, FIELD_SET_TYPE, PARAMETERS, ERR, ERROR,*)

- Description: Restores the specified field parameter set array that was obtained with FIELD_PARAMETER_SET_GET.
- Parameters
  - FIELD: A pointer to the field to restore the parameter set from
  - FIELD_SET_TYPE: The field parameter set identifier
  - PARAMETERS: The pointer to the field parameter set data obtained with the parameter set get call
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_UPDATE_CONSTANT(FIELD, FIELD_SET_TYPE, COMPONENT_NUMBER, VARIABLE_NUMBER, VALUE, ERR, ERROR,*)

- Description: Updates the given parameter set with the given value for the constant of the field variable component.
- Parameters
  - FIELD: A pointer to the field to update
  - FIELD_SET_TYPE: The field parameter set identifier
  - COMPONENT_NUMBER: The field variable component number to update
  - VARIABLE_NUMBER: The field variable to update
  - VALUE: The value to update to
  - ERR: The error code

- ERROR: The error string
- *:

FIELD_PARAMETER_SET_UPDATE_DOF(FIELD, FIELD_SET_TYPE, DOF_NUMBER, VALUE, ERR, ERROR,*)

- Description: Updates the given parameter set with the given value for a particular dof of the field.
- Parameters
  - FIELD: A pointer to the field to update
  - FIELD_SET_TYPE: The field parameter set identifier
  - DOF_NUMBER: The dof number to update
  - VALUE: The value to update to
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_UPDATE_ELEMENT(FIELD, FIELD_SET_TYPE, ELEMENT_NUMBER, COMPONENT_NUMBER, VARIABLE_NUMBER,&VALUE, ERR, ERROR,*)

- Description: Updates the given parameter set with the given value for a particular element of the field variable component.
- Parameters
  - FIELD: A pointer to the field to update
  - FIELD_SET_TYPE: The field parameter set identifier
  - ELEMENT_NUMBER: The element number to update
  - COMPONENT_NUMBER: The field variable component to update
  - VARIABLE_NUMBER: The field variable to update
  - *:
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_UPDATE_FINISH(FIELD, FIELD_SET_TYPE, ERR, ERROR,*)

- Description: Finishes the the parameter set update for a field.
- Parameters
  - FIELD: A pointer to the field to finish the update for
  - FIELD_SET_TYPE: The field parameter set identifier to finish the update for
  - ERR: The error code

- ERROR: The error string
- *:

FIELD_PARAMETER_SET_UPDATE_NODE(FIELD, FIELD_SET_TYPE, DERIVATIVE_NUMBER, NODE_NUMBER, COMPONENT_NUMBER, VARIABLE_NUMBER,&VALUE, ERR, ERROR,*)

- Description: Updates the given parameter set with the given value for a particular node and derivative of the field variable component.
- Parameters
  - FIELD: A pointer to the field to update
  - FIELD_SET_TYPE: The field parameter set identifier
  - DERIVATIVE_NUMBER: The node derivative number to update
  - NODE_NUMBER: The node number to update
  - COMPONENT_NUMBER: The field variable component number to update
  - VARIABLE_NUMBER: The field variable to update
  - *:
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELD_PARAMETER_SET_UPDATE_START(FIELD, FIELD_SET_TYPE, ERR, ERROR,*)

- Description: Starts the the parameter set update for a field.
- Parameters
  - FIELD: A pointer to the field to start the update for
  - FIELD_SET_TYPE: The field parameter set identifier to update
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELDS_FINALISE(REGION, ERR, ERROR,*)

- Description: Finalises the fields for the given region and deallocates all memory.
- Parameters
  - REGION: A pointer to the region to finalise the fields for
  - ERR: The error code
  - ERROR: The error string
  - *:

FIELDS_INITIALISE(REGION, ERR, ERROR,*)

- Description: Initialises the fields for the given region.
- Parameters
  - REGION: A pointer to the region to initialise the fields for
  - ERR: The error code
  - ERROR: The error string
  - *:

4.4. Equations Set

EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF

EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICATION_SET
EQUATIONS_SET_ASSEMBLE(EQUATIONS_SET, ERR, ERROR,*)

- Description: Assembles the equations for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to initialise the analytic solution for.
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_BACKSUBSTITUTE(EQUATIONS_SET, ERR, ERROR,*)

- Description: Backsubstitutes with an equations set to calculate unknown right hand side vectors.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to backsubstitute
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_CREATE_FINISH(EQUATIONS_SET, ERR, ERROR,*)

- Description: Finishes the process of creating an equation set on a region.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to finish creating
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_CREATE_START(USER_NUMBER, REGION, GEOM_FIBRE_FIELD, EQUATIONS_SET, ERR, ERROR,*)

- Description: Starts the process of creating an equations set defined by USER_NUMBER in the region identified by REGION.
- Parameters
  - USER_NUMBER: The user number of the equations set
  - REGION: A pointer to the region to create the equations set on
  - GEOM_FIBRE_FIELD: A pointer to the either the geometry or, in appropriate, the fibre field for the equation set
  - EQUATIONS_SET: On return, a pointer to the equations set
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_DESTROY(USER_NUMBER, REGION, ERR, ERROR,*)

- Description: Destroys an equations set identified by a user number on the give region and deallocates all memory.
- Parameters
  - USER_NUMBER: The user number of the equations set to destroy
  - REGION: The region of the equations set to destroy
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_FIXED_CONDITIONS_APPLY(EQUATIONS_SET, ERR, ERROR,*)

- Description: Applies the fixed conditions in an equation set to the dependent field in an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to apply the fixed conditions for.
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(EQUATIONS_SET, ERR, ERROR,*)

- Description: Finish the creation of fixed conditions for an equation set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to create the fixed conditions for.
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(EQUATIONS_SET, ERR, ERROR,*)

- Description: Start the creation of fixed conditions for a problem.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to start the creation of the fixed conditions for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_FIXED_CONDITIONS_DESTROY(EQUATIONS_SET, ERR, ERROR,*)

- Description: Destroy the fixed conditions for an equations set and deallocate all memory.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to destroy the fixed conditions for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_MATERIALS_COMPONENT_INTERPOLATION_SET(EQUATIONS_SET, COMPONENT_NUMBER, INTERPOLATION_TYPE, ERR, ERROR,*)

- Description: Sets/changes the field component interpolation for a materials field of a problem.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to set the materials component interpolation for
  - COMPONENT_NUMBER: The component of the material field to set the interpolation for
  - INTERPOLATION_TYPE: The interpolation type to set
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_SET(EQUATIONS_SET, COMPONENT_NUMBER, MESH_COMPONENT_NUMBER, ERR, ERROR,*)

- Description: Sets/changes the field component mesh component for a materials field of a problem.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to set the materials field component mesh component for
  - COMPONENT_NUMBER: The component number of the equations set materials field to set the mesh component for
  - MESH_COMPONENT_NUMBER: The mesh component number to set
  - ERR: The error code

- ERROR: The error string
- *:

EQUATIONS_SET_MATERIALS_CREATE_FINISH(EQUATIONS_SET, ERR, ERROR,*)

- Description: Finish the creation of materials for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to finish the creation of the materials field for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_MATERIALS_CREATE_START(EQUATIONS_SET, ERR, ERROR,*)

- Description: Start the creation of materials for a problem.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to start the creation of the materials field for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_MATERIALS_DESTROY(EQUATIONS_SET, ERR, ERROR,*)

- Description: Destroy the materials for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to destroy the materials for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_MATERIALS_SCALING_SET(EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

- Description: Sets/changes the field scaling for a materials field of an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to set the scaling on the material field
  - SCALING_TYPE: The scaling type to set
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_DEPENDENT_COMPONENT_MESH_COMPONENT_SET(EQUATIONS_SET, VARIABLE_NUMBER, COMPONENT_NUMBER,&MESH_COMPONENT_NUMBER, ERR, ERROR,*)

- Description: Sets/changes the field component mesh component for a dependent field of an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to set the dependent field component mesh component
  - VARIABLE_NUMBER: The dependent field variable number to set this should be variable type???
  - COMPONENT_NUMBER: The dependent field component number to set
  - *:
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_DEPENDENT_CREATE_FINISH(EQUATIONS_SET, ERR, ERROR,*)

- Description: Finish the creation of a dependent variables for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to finish the creation of
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_DEPENDENT_CREATE_START(EQUATIONS_SET, ERR, ERROR,*)

- Description: Start the creation of dependent variables for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to start the creation of a dependent field on
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_DEPENDENT_DEPENDENT_FIELD_GET(EQUATIONS_SET, DEPENDENT_FIELD, ERR, ERROR,*)

- Description: Returns a pointer to the dependent field of the dependent variables for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equation set to get the dependent field for
  - DEPENDENT_FIELD: On return, a pointer to the dependent field for the equations set. Must not be associated on entry.
  - ERR: The error code

- ERROR: The error string
- *:

EQUATIONS_SET_DEPENDENT_DESTROY(EQUATIONS_SET, ERR, ERROR,*)

- Description: Destroy the dependent variables for an equations sety.
- Parameters
  - EQUATIONS_SET: The pointer to the equations set to destroy
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_DEPENDENT_SCALING_SET(EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

- Description: Sets/changes the field scaling for a dependent field of an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to the dependent field scaling for.
  - SCALING_TYPE: The scaling type to set.
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_EQUATIONS_CREATE_FINISH(EQUATIONS_SET, ERR, ERROR,*)

- Description: Finish the creation of equations for the equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to finish the creation of the equations for.
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_EQUATIONS_CREATE_START(EQUATIONS_SET, ERR, ERROR,*)

- Description: Start the creation of equations for the equation set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to create equations for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_EQUATIONS_SPARSITY_TYPE_SET(EQUATIONS_SET, SPARSITY_TYPE, ERR, ERROR,*)

- Description: Sets/changes the sparsity type for the equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations to set the sparsity type for
  - SPARSITY_TYPE: The sparsity type to set EQUATIONS_ROUTINES_SparsityTypes,PROBLEM_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET(EQUATIONS_SET, OUTPUT_TYPE, ERR, ERROR,*)

- Description: Sets/changes the output type for the equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to set the output type for
  - OUTPUT_TYPE: The output type to set EQUATIONS_ROUTINES_OutputTypes,EQUATIONS_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_SOURCE_CREATE_FINISH(EQUATIONS_SET, ERR, ERROR,*)

- Description: Finish the creation of a source for an equation set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to start the creation of a souce for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_SOURCE_CREATE_START(EQUATIONS_SET, ERR, ERROR,*)

- Description: Start the creation of a source for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to start the creation of a source for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_SOURCE_DESTROY(EQUATIONS_SET, ERR, ERROR,*)

- Description: Destroy the source for an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to destroy the source for
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SET_SOURCE_SCALING_SET(EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

- Description: Sets/changes the field scaling for a source field of an equations set.
- Parameters
  - EQUATIONS_SET: A pointer to the equations set to set the scaling on the source field
  - SCALING_TYPE: The scaling type to set
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SETS_FINALISE(REGION, ERR, ERROR,*)

- Description: Finalises all equations sets on a region and deallocates all memory.
- Parameters
  - REGION: A pointer to the region to finalise the problems for.
  - ERR: The error code
  - ERROR: The error string
  - *:

EQUATIONS_SETS_INITIALISE(REGION, ERR, ERROR,*)

- Description: Intialises all equations sets on a region.
- Parameters
  - REGION: A pointer to the region to initialise the equations sets for
  - ERR: The error code
  - ERROR: The error string
  - *:

5. Problems

PROBLEM_ROUTINES::PROBLEM_DESTROY

PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET
PROBLEM_CREATE_FINISH(PROBLEM, ERR, ERROR,*)

- Description: Finishes the process of creating a problem.
- Parameters
  - PROBLEM: A pointer to the problem to finish creating.
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_CREATE_START(USER_NUMBER, PROBLEM, ERR, ERROR,*)

- Description: Starts the process of creating a problem defined by USER_NUMBER.
- Parameters
  - USER_NUMBER: The user number of the problem to create
  - PROBLEM: On return, a pointer to the created problem. Must not be associated on entry.
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_CONTROL_CREATE_FINISH(PROBLEM, ERR, ERROR,*)

- Description: Finish the creation of the control for the problem.
- Parameters
  - PROBLEM: A pointer to the problem to finish the control for
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_CONTROL_CREATE_START(PROBLEM, ERR, ERROR,*)

- Description: Start the creation of a problem control for a problem.
- Parameters
  - PROBLEM: A pointer to the problem to start the creation of a control for.
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_CONTROL_DESTROY(PROBLEM, ERR, ERROR,*)

- Description: Destroy the control for a problem.
- Parameters
  - PROBLEM: A pointer to the problem to destroy the control for.

- ERR: The error code
- ERROR: The error string
- *:

PROBLEM_SOLVE(PROBLEM, ERR, ERROR,*)

- Description: Solves a problem.
- Parameters
  - PROBLEM: A pointer to the problem to solve.
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_SOLUTIONS_CREATE_FINISH(PROBLEM, ERR, ERROR,*)

- Description: Finish the creation of solutions for a problem.
- Parameters
  - PROBLEM: A pointer to the problem
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_SOLUTIONS_CREATE_START(PROBLEM, ERR, ERROR,*)

- Description: Start the creation of a solution for the problem.
- Parameters
  - PROBLEM: A pointer to the problem to create a solution for
  - ERR: The error code
  - ERROR: The error string !<The error string
  - *:

PROBLEM_SOLUTION_EQUATIONS_SET_ADD(PROBLEM, SOLUTION_INDEX, EQUATIONS_SET, EQUATIONS_SET_INDEX, ERR, ERROR,*)

- Description: Adds an equations set to a problem solution.
- Parameters
  - PROBLEM: A pointer to the problem to add the equations set to a solution
  - SOLUTION_INDEX: The solution index in the problem to add the equations set to
  - EQUATIONS_SET: A pointer to the equations set to add
  - EQUATIONS_SET_INDEX: On return, the index of the equations set that has been added.

- ERR: The error code
- ERROR: The error string
- *:

PROBLEM_SOLVER_CREATE_FINISH(PROBLEM, ERR, ERROR,*)

- Description: Finish the creation of the solver for the problem solutions.
- Parameters
  - PROBLEM: A pointer to the problem to finish the solvers for
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_SOLVER_CREATE_START(PROBLEM, ERR, ERROR,*)

- Description: Start the creation of a problem solver for a problem.
- Parameters
  - PROBLEM: A pointer to the problem to start the creation of a solution for.
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_SOLVER_DESTROY(PROBLEM, ERR, ERROR,*)

- Description: Destroy the solvers for a problem.
- Parameters
  - PROBLEM: A pointer to the problem to destroy the solver for.
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEM_SOLVER_GET(PROBLEM, SOLUTION_INDEX, SOLVER, ERR, ERROR,*)

- Description: Returns a pointer to the solver for a solution on a problme.
- Parameters
  - PROBLEM: A pointer to the problem to get the solver for.
  - SOLUTION_INDEX: The solution index to get the solver for.
  - SOLVER: On return, a pointer to the solver. Must not be associated on entry.
  - ERR: The error code
  - ERROR: The error string

- *:

PROBLEMS_FINALISE(ERR, ERROR,*)

- Description: Finalises all problems and deallocates all memory.
- Parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

PROBLEMS_INITIALISE(ERR, ERROR,*)

- Description: Intialises all problems.
- Parameters
  - ERR: The error code
  - ERROR: The error string
  - *:

5.1. Solver

SOLVER_CREATE_FINISH(SOLVER, ERR, ERROR,*)

- Description: Finishes the process of creating a solver for a problem solution.
- Parameters
  - SOLVER: A pointer the solver to finish the creation of.
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_CREATE_START(SOLUTION, SOLVE_TYPE, SOLVER, ERR, ERROR,*)

- Description: Starts the process of creating a solver for a problem solution.
- Parameters
  - SOLUTION: A pointer to the solution to create the solver for.
  - SOLVE_TYPE: The type of solver to create SOLVER_ROUTINES::SolverTypes,SOLVER_ROUTINES
  - SOLVER: On exit, a pointer to the problem solver.
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_DESTROY(SOLVER, ERR, ERROR,*)

- Description: Destroy a problem solver.

- Parameters
  - SOLVER: A pointer the solver to destroy
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LIBRARY_SET(SOLVER, SOLVER_LIBRARY, ERR, ERROR,*)

- Description: Sets/changes the type of library to use for the solver.
- Parameters
  - SOLVER: A pointer the solver to set the type of
  - SOLVER_LIBRARY: The type of library to use for the solver SOLVER_ROUTINES::SolverLibraries,SOLVER_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_DIRECT_TYPE_SET(SOLVER, DIRECT_SOLVER_TYPE, ERR, ERROR,*)

- Description: Sets/changes the type of direct linear solver.
- Parameters
  - SOLVER: A pointer the problem solver to set the direct linear solver type
  - DIRECT_SOLVER_TYPE: The type of direct linear solver to set SOLVER_ROUTINES::DirectLinearSolverTypes,SOLVER_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET(SOLVER, ABSOLUTE_TOLERANCE, ERR, ERROR,*)

- Description: Sets/changes the maximum absolute tolerance for an iterative linear solver.
- Parameters
  - SOLVER: A pointer the problem solver to set
  - ABSOLUTE_TOLERANCE: The absolute tolerance to set
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET(SOLVER, DIVERGENCE_TOLERANCE, ERR, ERROR,*)

- Description: Sets/changes the maximum divergence tolerance for an iterative linear solver.
- Parameters
  - SOLVER: A pointer the problem solver to set
  - DIVERGENCE_TOLERANCE: The divergence tolerance to set
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(SOLVER, MAXIMUM_ITERATIONS, ERR, ERROR,*)

- Description: Sets/changes the maximum number of iterations for an iterative linear solver.
- Parameters
  - SOLVER: A pointer the problem solver to set the maximum number of iterations
  - MAXIMUM_ITERATIONS: The maximum number of iterations
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(SOLVER, ITERATIVE_PRECONDITIONER_TYPE, ERR, ERROR,*)

- Description: Sets/changes the type of preconditioner for an iterative linear solver.
- Parameters
  - SOLVER: A pointer the problem solver to set the iterative linear solver type
  - ITERATIVE_PRECONDITIONER_TYPE: The type of iterative preconditioner to set SOLVER_ROUTINES::IterativeLinearSolverTypes,SOLVER_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(SOLVER, RELATIVE_TOLERANCE, ERR, ERROR,*)

- Description: Sets/changes the relative tolerance for an iterative linear solver.
- Parameters
  - SOLVER: A pointer the solver to set
  - RELATIVE_TOLERANCE: The relative tolerance to set
  - ERR: The error code
  - ERROR: The error string

- *:

SOLVER_LINEAR_ITERATIVE_TYPE_SET(SOLVER, ITERATIVE_SOLVER_TYPE, ERR, ERROR,*)

- Description: Sets/changes the type of iterative linear solver.
- Parameters
  - SOLVER: A pointer the solver to set the iterative linear solver type
  - ITERATIVE_SOLVER_TYPE: The type of iterative linear solver to set SOLVER_ROUTINES::IterativeLinearSolverTypes,SOLVER_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_LINEAR_TYPE_SET(SOLVER, LINEAR_SOLVER_TYPE, ERR, ERROR,*)

- Description: Sets/changes the type of linear solver.
- Parameters
  - SOLVER: A pointer the solver to set the linear solver type
  - LINEAR_SOLVER_TYPE: The type of linear solver to set SOLVER_ROUTINES::LinearSolverTypes,SOLVER_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_OUTPUT_TYPE_SET(SOLVER, OUTPUT_TYPE, ERR, ERROR,*)

- Description: Sets/changes the output type for a solver.
- Parameters
  - SOLVER: A pointer the problem solver to set the iterative linear solver type
  - OUTPUT_TYPE: The type of solver output to be set SOLVER_ROUTINES::OutputTypes,SOLVER_ROUTINES
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_SPARSITY_TYPE_SET(SOLVER, SPARSITY_TYPE, ERR, ERROR,*)

- Description: Sets/changes the sparsity type for a solver.
- Parameters
  - SOLVER: A pointer the problem solver to set the iterative linear solver type

- SPARSITY_TYPE: The type of solver sparsity to be set SOLVER_ROUTINES::SparsityTypes,SOLVER_ROUTINES
- ERR: The error code
- ERROR: The error string
- *:

SOLVER_SOLVE(SOLVER, ERR, ERROR,*)

- Description: Solve the problem.
- Parameters
  - SOLVER: A pointer the solver to solve
  - ERR: The error code
  - ERROR: The error string
  - *:

SOLVER_VARIABLES_UPDATE(SOLVER, ERR, ERROR,*)

- Description: Updates the dependent variables from the solver solution.
- Parameters
  - SOLVER: A pointer the solver to update the variables from
  - ERR: The error code
  - ERROR: The error string
  - *:

Chapter 7. Coding Style

- Within a module all named constants and procedure names should be prefixed by a name indicating that module so as to maintain a namespace.
- All dynamic arrays should be `ALLOCATABLE` rather than `POINTER` unless full pointer functionality is required.
- A double space should be used for an indent. Tabs should not be used to indent code.
- All pointers should be checked to see if they are `ASSOCIATED` before de-referencing them.
- If there is just a single statement following an `IF` clause use the inline form of the `IF` statement and do not use `THEN` and `ENDIF`.
- There should be a space before the first continuation character and a space after the second continuation character when continuing lines.
- Use standard loop variable names e.g., `nn`, `component_idx`, when looping rather than temporary variable names.
- When using case statements put in all known values of the the case variable and use a `CALL FLAG_ERROR("Not implemented", ...` statement if the code for the case variable has yet to be coded.
- Use a `!================...` line between subroutines and functions
- For dummy array arguments the dimension qualifier should be with the array name i.e., use `INTEGER(INTG) :: FRED(N)` rather than `INTEGER(INTG), DIMENSION(N) :: FRED`.