MANCHESTER 1824

The University of Manchester

Introduction to *Code_Saturne*

Working with subroutines

J. Uribe

School of MACE, University of Manchester, UK

Introduction

Code_Saturne provides great capabilities and user customisation via FORTRAN subroutines.

- The running script runcase will compile all files in the **SRC** directory and link them with the existing libraries.

Introduction

Code_Saturne provides great capabilities and user customisation via FORTRAN subroutines.

- The running script runcase will compile all files in the **SRC** directory and link them with the existing libraries.
- Subroutines will have precedence to any settings on the xml files from the GUI.

Introduction

Code_Saturne provides great capabilities and user customisation via FORTRAN subroutines.

- The running script runcase will compile all files in the **SRC** directory and link them with the existing libraries.
- Subroutines will have precedence to any settings on the xml files from the GUI.
- The user's subroutines are in the **REFERENCE** directory inside **SRC**, arranged in modules.

Introduction

Code_Saturne provides great capabilities and user customisation via FORTRAN subroutines.

- The running script runcase will compile all files in the **SRC** directory and link them with the existing libraries.
- Subroutines will have precedence to any settings on the xml files from the GUI.
- The user's subroutines are in the **REFERENCE** directory inside **SRC**, arranged in modules.
- Different modules require different subroutines.

Introduction

Code_Saturne provides great capabilities and user customisation via FORTRAN subroutines.

- The running script runcase will compile all files in the **SRC** directory and link them with the existing libraries.
- Subroutines will have precedence to any settings on the xml files from the GUI.
- The user's subroutines are in the **REFERENCE** directory inside **SRC**, arranged in modules.
- Different modules require different subroutines.
- For full description of subroutines see the user's guide.

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.
- usclim.f90: Boundary conditions definitions. All types of boundaries can be set, but pre-defined ones are: inlet, outlet, symmetry and walls (periodicity set on the runcase file).

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.
- usclim.f90: Boundary conditions definitions. All types of boundaries can be set, but pre-defined ones are: inlet, outlet, symmetry and walls (periodicity set on the runcase file).
- usphyv.f90: Definition of physical properties (variable density, viscosity C_p etc.)

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.
- usclim.f90: Boundary conditions definitions. All types of boundaries can be set, but pre-defined ones are: inlet, outlet, symmetry and walls (periodicity set on the runcase file).
- usphyv.f90: Definition of physical properties (variable density, viscosity C_p etc.)
- usiniv.f90: Initialisation of variables on the domain.

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.
- usclim.f90: Boundary conditions definitions. All types of boundaries can be set, but pre-defined ones are: inlet, outlet, symmetry and walls (periodicity set on the runcase file).
- usphyv.f90: Definition of physical properties (variable density, viscosity C_p etc.)
- usiniv.f90: Initialisation of variables on the domain.
- usproj.f90: User postprocessing (profiles, budgets ...). Very useful since is called at each iteration. Plenty of examples.

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.
- usclim.f90: Boundary conditions definitions. All types of boundaries can be set, but pre-defined ones are: inlet, outlet, symmetry and walls (periodicity set on the runcase file).
- usphyv.f90: Definition of physical properties (variable density, viscosity C_p etc.)
- usiniv.f90: Initialisation of variables on the domain.
- usproj.f90: User postprocessing (profiles, budgets ...). Very useful since is called at each iteration. Plenty of examples.
- us*pst.f90: subroutines for the post-processing with ParaView (or Ensight) parts. User can create parts and assign variables to the new parts.

Base module subroutines

Most useful:

- usini1.f90: Set calculation parameters, physical and numerical. Most used parameters are listed with the default values.
- usclim.f90: Boundary conditions definitions. All types of boundaries can be set, but pre-defined ones are: inlet, outlet, symmetry and walls (periodicity set on the runcase file).
- usphyv.f90: Definition of physical properties (variable density, viscosity C_p etc.)
- usiniv.f90: Initialisation of variables on the domain.
- usproj.f90: User postprocessing (profiles, budgets ...). Very useful since is called at each iteration. Plenty of examples.
- us*pst.f90: subroutines for the post-processing with ParaView (or Ensight) parts. User can create parts and assign variables to the new parts.
- usts??.f90 : Source terms for each computed variable, including scalars (e.g. ΔP).

Structure

Subroutines start with the list of arguments passed from the calling routine.

```
subroutine usclim &
!================

 ( idbia0 , idbra0 ,                                                  &
   ndim   , ncelet , ncel   , nfac   , nfabor , nfml   , nprfml , &
   nnod   , lndfac , lndfbr , ncelbr ,                                &
   nvar   , nscal  , nphas  ,                                         &
   nideve , nrdeve , nituse , nrtuse ,                                &
   ifacel , ifabor , ifmfbr , ifmcel , iprfml , maxelt , lstelt , &
   ipnfac , nodfac , ipnfbr , nodfbr ,                                &
   icodcl , itrifb , itypfb ,                                         &
   idevel , ituser , ia     ,                                         &
   xyzcen , surfac , surfbo , cdgfac , cdgfbo , xyznod , volume , &
   dt     , rtp    , rtpa   , propce , propfa , propfb ,           &
   coefa  , coefb  , rcodcl ,                                         &
   w1     , w2     , w3     , w4     , w5     , w6     , coefu  , &
   rdevel , rtuser , ra     )
!===============================================================================
! Purpose:
! -------

!    User subroutine.

!    Fill boundary conditions arrays (icodcl, rcodcl) for unknown variables.
```

Structure

Then, a table with the explanation of the arguments is presented:

```
!-------------------------------------------------------------------------------
! Arguments
!------------------.----.-----.-----------------------------------------------.
! name              !type!mode ! role                                          !
!------------------!----!-----!-----------------------------------------------!
! idbia0            ! i  ! <-- ! number of first free position in ia           !
! idbra0            ! i  ! <-- ! number of first free position in ra           !
! ndim              ! i  ! <-- ! spatial dimension                             !
! ncelet            ! i  ! <-- ! number of extended (real + ghost) cells       !
! ncel              ! i  ! <-- ! number of cells                               !
! nfac              ! i  ! <-- ! number of interior faces                      !
! nfabor            ! i  ! <-- ! number of boundary faces                      !
! nfml              ! i  ! <-- ! number of families (group classes)            !
! nprfml            ! i  ! <-- ! number of properties per family (group class) !
! nnod              ! i  ! <-- ! number of vertices                            !
! lndfac            ! i  ! <-- ! size of nodfac indexed array                  !
! lndfbr            ! i  ! <-- ! size of nodfbr indexed array                  !
! ncelbr            ! i  ! <-- ! number of cells with faces on boundary        !
! nvar              ! i  ! <-- ! total number of variables                     !
! nscal             ! i  ! <-- ! total number of scalars                       !
! nphas             ! i  ! <-- ! number of phases                              !
! ifacel(2, nfac)   ! ia ! <-- ! interior faces -> cells connectivity          !
! ifabor(nfabor)    ! ia ! <-- ! boundary faces -> cells connectivity          !
! ifmfbr(nfabor)    ! ia ! <-- ! boundary face family numbers                  !
!------------------!----!-----!-----------------------------------------------!

!     Type: i (integer), r (real), s (string), a (array), l (logical),
!           and composite types (ex: ra real array)
!     mode: <-- input, --> output, <-> modifies data, --- work array
!===============================================================================
```

Structure

Common blocks and Arguments declarations:

```
  implicit none
!===============================================================================
! Common blocks
!===============================================================================
include "paramx.h"
include "pointe.h"
include "numvar.h"
include "optcal.h"
include "cstphy.h"
include "cstnum.h"
include "entsor.h"
include "parall.h"
include "period.h"
include "ihmpre.h"
!===============================================================================
! Arguments
integer          idbia0 , idbra0
integer          ndim   , ncelet , ncel   , nfac   , nfabor
integer          nfml   , nprfml
integer          nnod   , lndfac , lndfbr , ncelbr
integer          nvar   , nscal  , nphas
integer          nideve , nrdeve , nituse , nrtuse
integer          ifacel(2,nfac) , ifabor(nfabor)
integer          ifmfbr(nfabor) , ifmcel(ncelet)
integer          iprfml(nfml,nprfml), maxelt, lstelt(maxelt)
integer          ipnfac(nfac+1), nodfac(lndfac)
integer          ipnfbr(nfabor+1), nodfbr(lndfbr)
integer          icodcl(nfabor,nvar)
integer          itrifb(nfabor,nphas), itypfb(nfabor,nphas)
integer          idevel(nideve), ituser(nituse), ia(*)
```

Structure

Local variables and initialisation:

```
! Local variables

integer           idebia, idebra
integer           ifac, iel, ii, ivar, iphas
integer           ilelt, nlelt
double precision  uref2, d2s3
double precision  rhomoy, dh, ustar2
double precision  xintur
double precision  xkent, xeent

!===============================================================================


!===============================================================================
! 1.  Initialization
!===============================================================================

idebia = idbia0
idebra = idbra0

d2s3 = 2.d0/3.d0
```

Continue with the rest .....

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).
- Due to historic reasons, the names of the subroutines always have 6 letters.

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).
- Due to historic reasons, the names of the subroutines always have 6 letters.
- Most integer variables start with "i", i.e. `idebia`.

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).
- Due to historic reasons, the names of the subroutines always have 6 letters.
- Most integer variables start with "i", i.e. `idebia`.
- Variable names are also maximum 6 characters long (legacy from FORTRAN77).

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).
- Due to historic reasons, the names of the subroutines always have 6 letters.
- Most integer variables start with "i", i.e. `idebia`.
- Variable names are also maximum 6 characters long (legacy from FORTRAN77).
- Compilation not only of user subroutines but ANY f90 or c file in **SRC** (this could include *Code_Saturne* original source routines).

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).
- Due to historic reasons, the names of the subroutines always have 6 letters.
- Most integer variables start with "i", i.e. `idebia`.
- Variable names are also maximum 6 characters long (legacy from FORTRAN77).
- Compilation not only of user subroutines but ANY f90 or c file in **SRC** (this could include *Code_Saturne* original source routines).
- If an input file is needed, place it in **DATA** and link it by editing the `runcase` file.

Structure

Important points to take into account:

- *Code_Saturne* is in double precision. Declare your variables as DP (and assign values as `1.D0`).
- Due to historic reasons, the names of the subroutines always have 6 letters.
- Most integer variables start with "i", i.e. `idebia`.
- Variable names are also maximum 6 characters long (legacy from FORTRAN77).
- Compilation not only of user subroutines but ANY f90 or c file in **SRC** (this could include *Code_Saturne* original source routines).
- If an input file is needed, place it in **DATA** and link it by editing the `runcase` file.
- To check potential compilation errors before running, use the command `code_saturne compile` inside the **SRC** directory.

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.
- `NPHAS`: Number of phases.

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.
- `NPHAS`: Number of phases.
- `NCEL, NCELET`: Number of cells and cells plus ghost cells, per MPI task.

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.
- `NPHAS`: Number of phases.
- `NCEL, NCELET`: Number of cells and cells plus ghost cells, per MPI task.
- `NFAC, NFABOR`[1]: Number of internal and boundary faces.

[1] Bord = boundary in french

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.
- `NPHAS`: Number of phases.
- `NCEL, NCELET`: Number of cells and cells plus ghost cells, per MPI task.
- `NFAC, NFABOR`[1]: Number of internal and boundary faces.
- `NVAR`: Number of variables (depends on case and the physics).

[1] Bord = boundary in french

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.
- `NPHAS`: Number of phases.
- `NCEL, NCELET`: Number of cells and cells plus ghost cells, per MPI task.
- `NFAC, NFABOR`[1]: Number of internal and boundary faces.
- `NVAR`: Number of variables (depends on case and the physics).
- `NPROCE`: Number of properties defined at at the cell centres.

[1] Bord = boundary in french

Main Variables

Array dimensions (set by the kernel, not to be changed by the user)

- `NDIM` : Number of dimensions.
- `NPHAS`: Number of phases.
- `NCEL, NCELET`: Number of cells and cells plus ghost cells, per MPI task.
- `NFAC, NFABOR`[1]: Number of internal and boundary faces.
- `NVAR`: Number of variables (depends on case and the physics).
- `NPROCE`: Number of properties defined at at the cell centres.
- `NPROFA, NPROFB`: Number of properties defined at the internal and at the boundary faces.

[1] Bord = boundary in french

Main Variables

Geometrical arrays (set by the preprocessor, not be changed by the user)

- `XYZCEN(NDIM,NCELET)`: Coordinates of the cell centres.
- `CDGFAC(NDIM,NFAC)`: Coordinates of the internal
- `CDGFBO(NDIM,NFABOR)`: Coordinates of the boundary faces.
- `SURFAC(NDIM,NFAC)`: Vector normal to the internal face whose length is the area of the face.
- `SURFBO(NDIM,NFABOR)`: Vector normal to the boundary face whose length is the area of the face.
- `VOLUME(NCELET)`: Volume of each cell.
- `IFACEL(2,NFAC)`: Connectivity between faces and cells. Contains cells at each side of the face.
- `IFABOR(NFABOR)`: Connectivity between faces and cells at the boundaries (only one cell per face).

Solved Variables

Solved variables are stored in two main arrays:

- `RTP(NCELET,NVAR)` : Variables at the current time step.
- `RTPA(NCELET,NVAR)` : Variables at the previous time step.

Solved Variables

Solved variables are stored in two main arrays:

- `RTP(NCELET,NVAR)` : Variables at the current time step.
- `RTPA(NCELET,NVAR)` : Variables at the previous time step.

Each variable has an integer that points to a part of the RTP array. There is one index per phase (even though is not a multiphase code). The number of variables `NVAR` depends on the case setup (turbulence model, scalars, specific physics etc.).

Solved Variables

Solved variables are stored in two main arrays:

- `RTP(NCELET,NVAR)` : Variables at the current time step.
- `RTPA(NCELET,NVAR)` : Variables at the previous time step.

Each variable has an integer that points to a part of the `RTP` array. There is one index per phase (even though is not a multiphase code). The number of variables `NVAR` depends on the case setup (turbulence model, scalars, specific physics etc.).

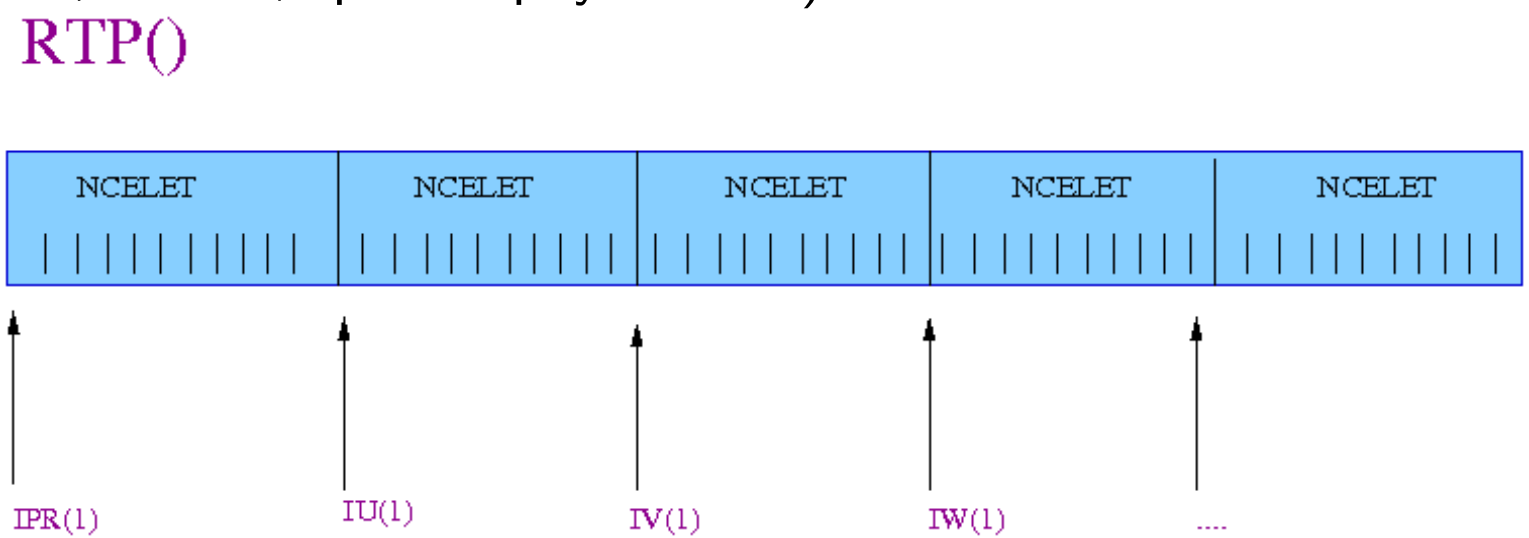
- `IPR(IPHAS)` : Pressure for phase IPHAS.
- `IU(IPHAS)` : Velocity in the X direction.
- `IV(IPHAS)` : Velocity in the Y direction.
- `IW(IPHAS)` : Velocity in the W direction.
- ...

Solved Variables

Solved variables are stored in two main arrays:

- `RTP(NCELET,NVAR)` : Variables at the current time step.
- `RTPA(NCELET,NVAR)` : Variables at the previous time step.

Each variable has an integer that points to a part of the RTP array. There is one index per phase (even though is not a multiphase code). The number of variables `NVAR` depends on the case setup (turbulence model, scalars, specific physics etc.).



Physical properties

Physical properties are stored by default at the cell centres but some can be stored at the faces for some specific physics modules.

- `PROPCE(NCELET,NPROPCE)`: Properties at the cell centres.
- `PROPFA(NFAC,NPROPFA)`: Properties at the face centres.
- `PROPFB(NFABOR, NPROPB)`: Properties at the boundary faces.

Physical properties

Physical properties are stored by default at the cell centres but some can be stored at the faces for some specific physics modules.

- `PROPCE(NCELET,NPROPCE)`: Properties at the cell centres.
- `PROPFA(NFAC,NPROPFA)`: Properties at the face centres.
- `PROPFB(NFABOR, NPROPB)`: Properties at the boundary faces.

To access the property, you need to find the pointer inside the properties pointer array: `IPPROC(IPROP)`, `IPPROF(IPROP)`, `IPPROB(IPROP)`.

Physical properties

Physical properties are stored by default at the cell centres but some can be stored at the faces for some specific physics modules.

- `PROPCE(NCELET,NPROPCE)`: Properties at the cell centres.
- `PROPFA(NFAC,NPROPFA)`: Properties at the face centres.
- `PROPFB(NFABOR, NPROPB)`: Properties at the boundary faces.

To access the property, you need to find the pointer inside the properties pointer array: `IPPROC(IPROP)`, `IPPROF(IPROP)`, `IPPROB(IPROP)`.

Each property has a pointer

- `IROMC(IPHAS)`: Density for phase IPHAS.
- `IVISCL(IPHAS)`: Molecular viscosity for phase IPHAS.
- `IVISCT(IPHAS)`: Turbulent viscosity for phase IPHAS.

Physical properties

Physical properties are stored by default at the cell centres but some can be stored at the faces for some specific physics modules.

- `PROPCE(NCELET,NPROPCE)`: Properties at the cell centres.
- `PROPFA(NFAC,NPROPFA)`: Properties at the face centres.
- `PROPFB(NFABOR, NPROPB)`: Properties at the boundary faces.

To access the property, you need to find the pointer inside the properties pointer array: `IPPROC(IPROP)`, `IPPROF(IPROP)`, `IPPROB(IPROP)`.

Each property has a pointer

- `IROMC(IPHAS)`: Density for phase IPHAS.
- `IVISCL(IPHAS)`: Molecular viscosity for phase IPHAS.
- `IVISCT(IPHAS)`: Turbulent viscosity for phase IPHAS.

For example, to access the density ρ at a given cell index IEL use:

$\rho =$`PROPCE(IEL,IPPROC(IROM(IPHAS)))`

Memory management

All information is stored in two macro-arrays:

- `IA(LONG(IA))` : Integer array.
- `RA(LONG(RA))` : Real array.

In FORTRAN 77, memory cannot be dynamically allocated so the size of the arrays has to be determined before the calculation. They can be specified in the GUI or in usini1.f90. If they are set to zero, the code tries to estimate them based on the number of cells and variables. Each subroutine that allocates new memory checks if the size of IA and RA is large enough, if not, it stops with an error message.

Examples

Change the density ρ as a function of the temperature T.

$$\rho = T(aT + b) + c$$

```
      iphas = 1
      ivart = isca(iscalt(iphas))
      ipcrom = ipproc(irom(iphas))
!
      vara  = -4.0668d-3
      varb  = -5.0754d-2
      varc  =  1000.9d0
!
      do iel = 1, ncel
        xrtp = rtp(iel,ivart)
        propce(iel,ipcrom) = xrtp * (vara*xrtp+varb) + varc
      enddo
```

Physical properties (ρ, μ, C_p...) can be changed via the usphyv.f90 subroutine.

Examples

Initialize temperature T at the beginning of the calculation:

```
iphas = 1
ivart = isca(iscalt(iphas))
if (isuite.eq.0) then
  do iel = 1, ncel
    rtp(iel,ivart) = 25.d0
  enddo
endif
```

- ISUITE : Index for calculation restart. ($= 1$, restart.)

To initialize velocity as a cosine function:

```
iphas = 1
ivaru = iu(iphas)
xpi = 3.14159d0
A = 2.d0
if (isuite.eq.1) then
  do iel = 1, ncel
    rtp(iel,ivaru) = A*cos(2.d0*xpi*ttcabs)
  enddo
endif
```

- TTCABS : Current time (in seconds).