OpenPALM Tutorial: a first toy with *NSC2KE*

By F. Duchaine*, T. Morel* and E. Quémerais**

Motivations and objectives

This report aims at providing an example of use of Open-PALM to construct a coupled application using the library CWIPI (M.P.Errera *et al.* 2010). To achieve this objective, a fluid-fluid simulation is done by coupling the open source code *NSC2KE* (Mohammadi 1994) with itself.

This report is organized as follow. First, the solver *NSC2KE* and the configuration are exposed. Then, the development of the coupled application *NSC2KE-NSC2KE* is detailed. Finally, results are given.

The *NSC2KE* solver and the configuration

Developed by Bijan Mohammadi at INRIA, *NSC2KE* (Mohammadi 1994) is a Finite-Volume Galerkin program computing 2D and axisymmetric flows on unstructured meshes. To solve the Euler part of the equations, a Roe, an Osher and a Kinetic solvers are available. To compute turbulent flows a $k-\epsilon$ model is available. Near-wall turbulence is computed either by wall-laws or by a two-layer approach. Time dependent problems can also be considered as a fourth order Runge-Kutta solver has been used. *NSC2KE* simulates a wide range of flow fields including:

- external and internal flows,
- subsonic to hypersonic inviscid flows,
- low-Reynolds subsonic to hypersonic viscous flows,
- subsonic to hypersonic fully separated turbulent flows,
- steady and unsteady flows.

The configuration retained for this simple test is a supersonic ramp at Mach number 1.8 followed by a long duct (Fig. 1).

The corresponding input file of *NSC2KE* (*DATA*) is:

```
0          --> =0 2D, =1 AXISYMMETRIC
0          --> =0 Euler, =1 Navier-Stokes
166.       --> Reynolds by meter (the mesh is given in meter)
0.         --> inverse of Froude number (=0 no gravity)
1.8        --> inflow Mach number
1.         --> ratio pout/pin
1          --> wall =1 newmann b.c. on the temp.(adiabatic), =2 Dirichlet.(isothermal)
300.       --> inflow temperature (in Kelvin) for Sutherland laws
288.       --> if isothermal walls , wall temperature (in Kelvin)
0.0        --> angle of attack
3          --> Euler fluxes =1 roe, =2 osher,=3 kinetic
3          --> nordre = 1 first order scheme, =2 second order, =3 limited second order
0          --> =0 global time steping (unsteady), =1 local Euler, =2 local N.S.
1.         --> cfl
10000      --> number of time step
10000      --> frequence for the  solution to be saved
1.e10      --> maximum physical time for run (for unsteady problems)
-4.        --> order of magnitude for the residual to be reduced (for steady problems)
0          --> =0 start with uniform solution, =1 restart from INIT_NS
cccc  turbulence  ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
0          --> =0 no turbulence model, =1 k-epsilon model
0          --> =0 two-layer technique, =1 wall laws
1.e-2      --> delta in wall laws or limit of the one-eq. model. (in meter)
0          --> =0 start from uniform solution for k-epsilon, =1 from INIT_KE
-1.e10 1.e10 -1.e10 1.e10      --> xtmin,xtmax,ytmin,ytmax (BOX for k-epsilon r.h.s)
```

* CERFACS, 42 avenue G. Coriolis, 31 057 Toulouse Cedex 01, France

** Département Simulation Numérique des Écoulements et Aéroacoustique (DSNA),ONERA, 29 Avenue de la Division Leclerc 92 322 Châtillon Cedex, France

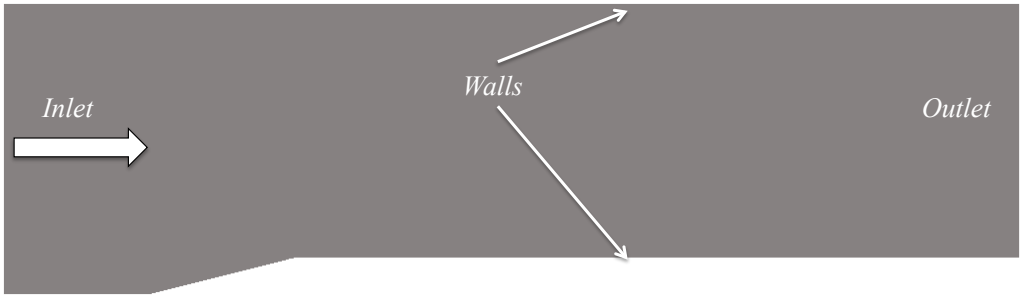

FIGURE 1. Global configuration.

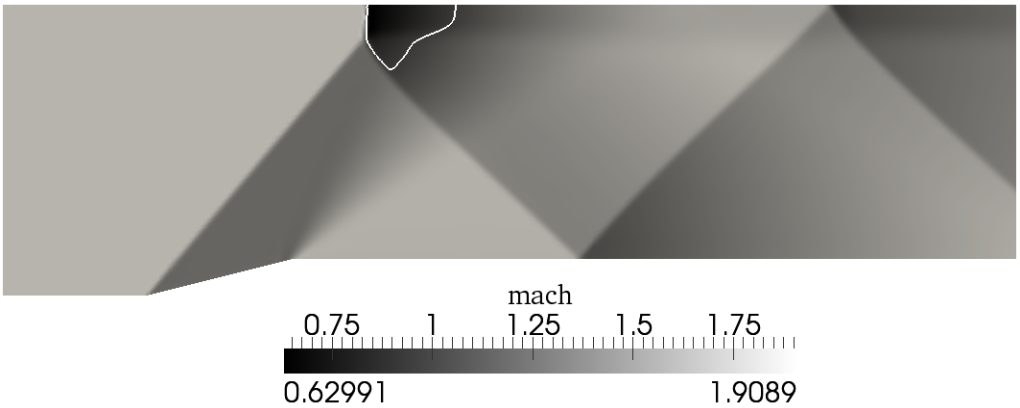

FIGURE 2. Mach number field and iso sonic line in the global configuration.

A transient computation is made until the convergence to a stationary state. Figure 2 shows the field of Mach number for this configuration which illustrates the complexity of the flow structure.

The idea of the fluid-fluid coupling is to treat the ramp with one instance of the solver and the duct with an other one. The coupling is done through the boundary conditions of the domains. Figure 3 represents the corresponding domains. We see that this two domains overlap so that the information exchanged from one code to the other is interpolated from the interior grid of the source solver to the boundary nodes of the target solver.

Development of the coupled application *NSC2KE-NSC2KE*

The development of the coupled application relies on two steps: (1) the preparation of the coupling in PrePALM and (2) the instrumentation in the solver.

PrePALM job

To start with, a unit is made with the solver *NSC2KE*. This is simply done by replacing *program* in *nsc2ke.f* by *subroutine* and by creating an identity card. It is interesting to note that, thanks to the flexibility of the Open-PALM coupler, only one source code, thus only one *Id card*, is used for this tutorial. This *Id card* reads:

```
1 !PALM_UNIT -name nsc2ke\
2 !          -functions {f90 nsc2ke}\
3 !          -object_files {../NSC2KE_FLO/lib_nsc2ke.a ../NSC2KE_FLO/mod_interf.o \
4 !                         ../NSC2KE_FLO/interf_cpl.o}\
5 !          -comment {NSC2KE}
6 !
7 !PALM_CWIPI_COUPLING -name toy
8 !
9 !PALM_CWIPI_OBJECT -name allexch -coupling toy -intent INOUT
```

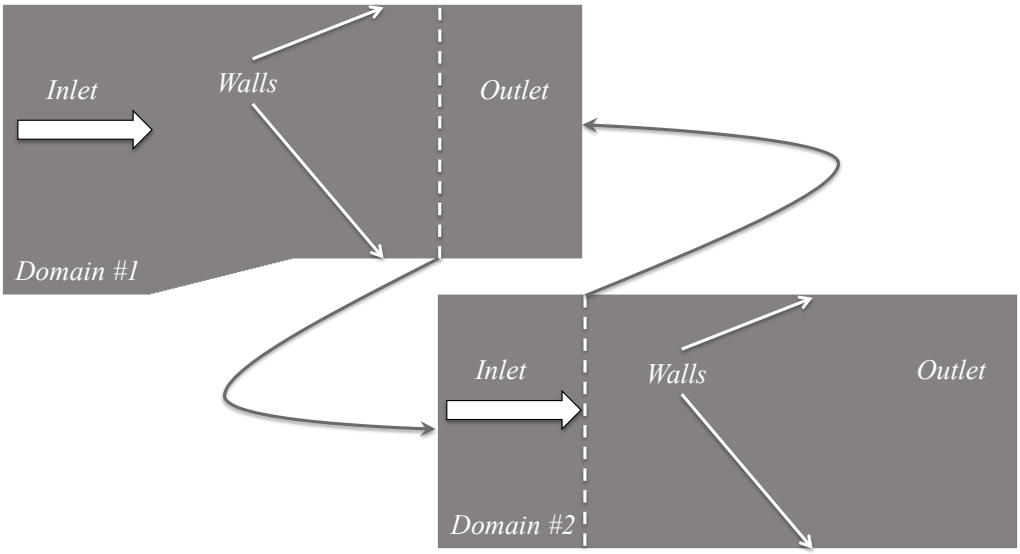

FIGURE 3. Fluid-fluid coupling configuration.

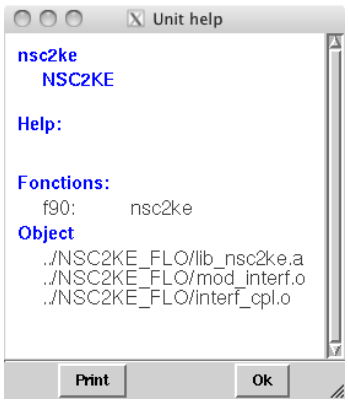

FIGURE 4. Identity card of *NSC2KE* as it appears in Open-PALM.

The keyword *PALM_CWIPI_COUPLING* allows to define a *CWIPI* coupling environment for the unit *NSC2KE*. It is very important to underline here that this name (*toy*) doesn't (*totally*) correspond to the one use by CWIPI to exchange data between the codes. In fact, each unit defines its own keyword (in the case of this tutorial, each unit defines *toy*). Based on this keyword, Open-PALM (PrePALM in fact) will then reconstructs the coupling name associated to the two codes: *toy.toy*. In this example, the object *allexch*, associated to the CWIPI coupling keyword *toy* is exchanged between the codes.

Figure 4 presents the identity card of *NSC2KE* as it appears in Open-PALM. We see that for the moment, nothing indicates than *CWIPI* objects are exchanged between the codes.

To couple the two instances of *NSC2KE*, two branches are drawn on the PrePALM canvas and the unit is inserted in both branch (Fig. 5). The *Execution working directory* (Fig. 6) allows to specify the directory where the unit is executed. In the present tutorial, one instance of *NSC2KE* is executed in the directory CAS01/ and the second one in CAS02/. The resulting application in PrePALM is presented on Fig. 7.

The CWIPI functionalities are directly accessible in the canvas: when a unit contains reference to CWIPI, the representation of the unit in the PrePALM canvas contains a small rectangle with the inscription *CWIPI* (Fig. 7). A left click on one of the unit allow to *insert* a CWIPI coupling between this unit and the other one in the context defined by the keyword *toy*. When the coupling is inserted, a link between the units appears in this canvas (Fig. 7). A left click on this link allows to edit

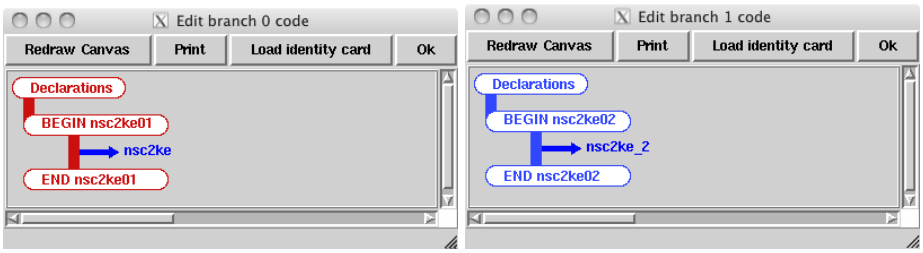

FIGURE 5. Integration of the *NSC2KE* units in the two branches.

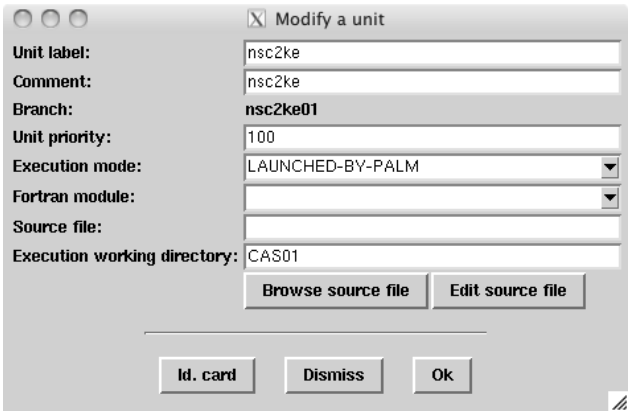

FIGURE 6. Insertion of the *NSC2KE* unit in a branch.

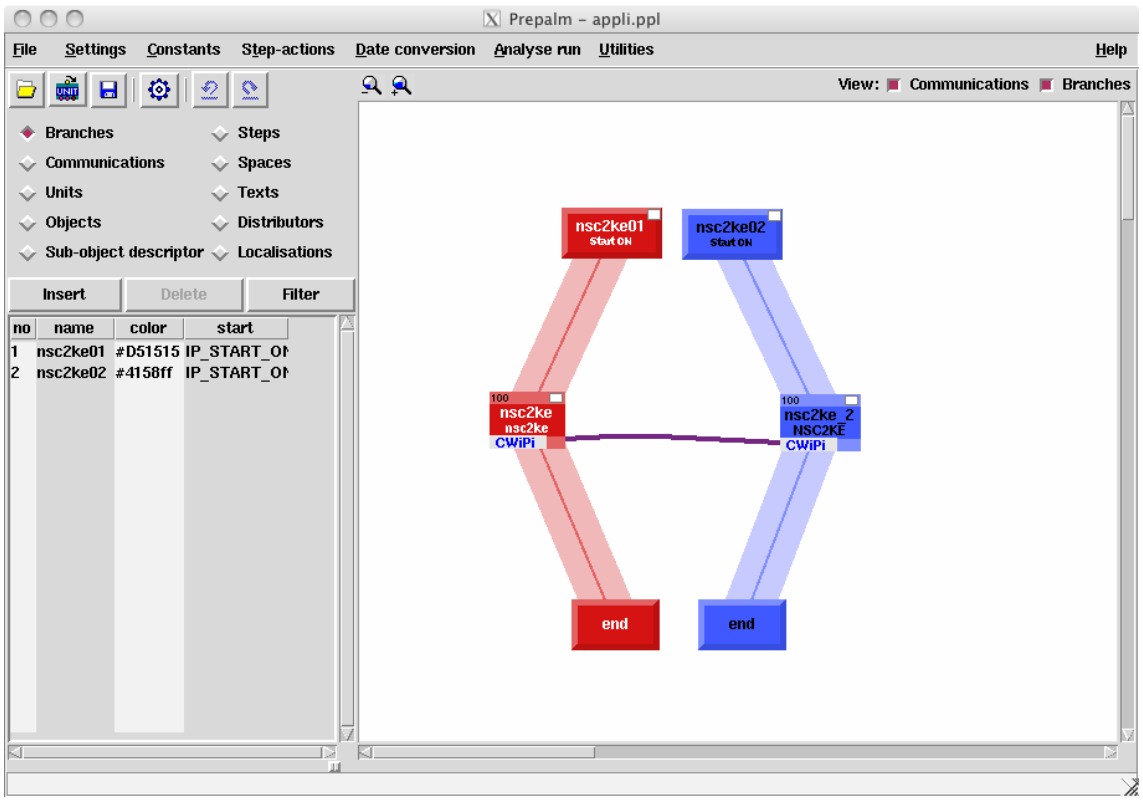

FIGURE 7. Open-PALM application with CWIPI settings.

or delete this coupling as well as to insert a communication. PrePALM automatically detects the possible communications thanks to the *Id cards* of the units. In the present case, PrePALM propose the communication "*allexch <==> allexch*" meaning that their is a synchronized exchanged of the sent and received objects, ie the send and receive operations are done in a unique instruction. The job with PrePALM is terminated by creating the PALM files (*Make PALM files*) in MPI-1 mode. Indeed, for the moment, the CWIPI functionalities are only available with the MPI-1 mode of Open-PALM.

Job in NSC2KE

As mentioned in the *Id card* of the unit *NSC2KE* (Fig. 4), the source code is composed of the standard library of the solver (*lib_nsc2ke.a*) and two additional files (*mod_interf.f90* and *inter_cpl.f*). The library contains the originals sources of the code with some modifications to allow data exchanges. The module *mod_interf.f90* describes the variables used for the coupling with Open-PALM:

```
module mod_interf
  integer                                          :: getnum
  integer                                          :: nbexch
  integer                                          :: itcompt
  integer                                          :: fcpl
  double precision, dimension(:), allocatable :: coord
  double precision, dimension(:), allocatable :: exchcoord
  double precision, dimension(:), allocatable :: sendfield
  double precision, dimension(:), allocatable :: recvfield
  integer,          dimension(:), allocatable :: elems
  integer,          dimension(:), allocatable :: conni
  integer,          dimension(:), allocatable :: indexch
end module mod_interf
```

The file *inter_cpl.f* contains the routines *interf_init*, *interf_def_mesh*, *interf_exchange*, *interf_end* defined for the coupling and called by *NSC2KE* during its execution. The routine *interf_init* allows to initiate the coupling in the units and is called at the beginning of *NSC2KE*:

```
      subroutine interf_init
        use palmlib
        use cwipi
        use mod_interf
        implicit none
        integer :: il_err
        integer :: outfreq
        double precision :: dimtol
        character(len=PL_LNAME) :: cl_coupling_name
        character(len=PL_LNAME) :: output_format
        character(len=PL_LNAME) :: output_format_option
c
        itcompt = 0
        open(51,file="coupling.choices",status="old")
        read(51,*)fcpl
        read(51,*)outfreq
        read(51,*)dimtol
        read(51,*)getnum
        close(51)
c
        cl_coupling_name = 'toy'
        output_format = 'Ensight Gold'
        output_format_option ='text'
        call PCW_Init(il_err)
        call PCW_Create_coupling(cl_coupling_name,
     &                 cwipi_cpl_parallel_with_part,
     &                 2,
     &                 dimtol,
     &                 cwipi_static_mesh,
     &                 cwipi_solver_cell_vertex,
     &                 outfreq,
     &                 output_format,
     &                 output_format_option,
     &                 il_err)
c
       call cwipi_set_output_listing_f(PL_OUT)
       call cwipi_dump_appli_properties_f
c
      end subroutine interf_init
```

In order to allow a flexible coding, the definition file *coupling.choices* readed by the units contains the coupling frequency *fcpl* (ie the number of iteration of the codes between two data exchange), the frequency of the CWIPI outputs *outfreq* and the geometric tolerance *dimtol* used by CWIPI to localize the grid points. In this tutorial, *dimtol* is fixed to 0.01. The integer *getnum* allows to distinguish which entity of the solver will treat the ramp or the duct. As discussed latter, the unit with *getnum=1* will be in charge of the ramp and the one with *getnum=2* in charge of the duct. The connection of the unit to the CWIPI environment is done with the instruction *PCW_Init* (line 24). Then, the instruction *PCW_create_coupling* (line 25) allows the connection between the two units. It is important that the coupling name defined in the *Id card* correspond to the name given in the *PCW_create_coupling* (ie toy in this tutorial). Finally, the instruction *cwipi_set_output_listing_f* (line 36) redirects the output writes of CWIPI in the generic files of Open-PALM accessible with *PL_OUT*.

Once the coupling have been created, the next step is to define the mesh for CWIPI. This is done with the routine *interf_def_mesh* called in the routine *mailla* of *NSC2KE* which reads the mesh.

```
1       subroutine interf_def_mesh(nsm,ntm,ns,nt,coor,nu,logfr)
2   c
3           use palmlib
4           use cwipi
5           use mod_interf
6           implicit none
7           integer           :: i
8           integer           :: j
9           integer           :: il_err
10          integer           :: nsm
11          integer           :: ntm
12          integer           :: ns
13          integer           :: nt
14          real              :: coor(2,nsm)
15          integer           :: nu(3,ntm)
16          integer           :: logfr(nsm)
17  c
18          character(len=PL_LNAME) :: cl_coupling_name
19          integer                 :: stride
20          integer                 :: nNotLocatedPoints
21  c
22  c*****************************************************
23  c
24          nbexch = 0
25          if (getnum.eq.1) then      ! Output is coupled
26            do i = 1, ns
27              if (logfr(i).eq.4) nbexch = nbexch + 1
28            enddo
29          elseif (getnum.eq.2) then  ! Inlet is coupled
30            do i = 1, ns
31              if (logfr(i).eq.6) nbexch = nbexch + 1
32            enddo
33          endif
34  c
35          allocate ( exchcoord(3*nbexch) )
36          allocate ( indexch(nbexch) )
37  c
38          j = 0
39          if (getnum.eq.1) then      ! Output is coupled
40            do i = 1, ns
41              if (logfr(i).eq.4) then
42                j = j + 1
43                exchcoord((j-1)*3+1) = coor(1,i)
44                exchcoord((j-1)*3+2) = coor(2,i)
45                exchcoord((j-1)*3+3) = 0.0d0
46                indexch(j)           = i
47              endif
48            enddo
49          elseif (getnum.eq.2) then  ! Inlet is coupled
50            do i = 1, ns
51              if (logfr(i).eq.6) then
52                j = j + 1
53                exchcoord((j-1)*3+1) = coor(1,i)
54                exchcoord((j-1)*3+2) = coor(2,i)
55                exchcoord((j-1)*3+3) = 0.0d0
56                indexch(j)           = i
57              endif
58            enddo
```

```
59          endif
60  c
61  c*************************************************
62  c
63          allocate ( coord (3*ns) )
64          allocate ( elems (3*nt) )
65          allocate ( conni (nt+1) )
66          conni = 0
67          coord = 0.0d0
68  c
69          do i = 1, ns
70            coord((i-1)*3+1) = coor(1,i)
71            coord((i-1)*3+2) = coor(2,i)
72            coord((i-1)*3+3) = 0.0d0
73          enddo
74  c
75          do i = 1, nt
76            elems((i-1)*3+1) = nu(1,i)
77            elems((i-1)*3+2) = nu(2,i)
78            elems((i-1)*3+3) = nu(3,i)
79            conni(i+1)       = conni(i) + 3
80          enddo
81  c
82  c*************************************************
83  c
84          cl_coupling_name = 'toy'
85          call PCW_Define_mesh(cl_coupling_name,
86       &                       ns, nt, coord,
87       &                       conni, elems,
88       &                       il_err)
89  c
90          call PCW_set_points_to_locate (cl_coupling_name,
91       &                                 nbexch,
92       &                                 exchcoord,
93       &                                 il_err)
94  c
95  c*************************************************
96  c
97          allocate ( sendfield (6*ns) )
98          allocate ( recvfield (6*nbexch) )
99  c
100 c*************************************************
101 c
102       end subroutine interf_def_mesh
```

The aim of the routine *interf_def_mesh* is thus twofolds: it defines (1) the source meshes of the domains and (2) the target points of the boundaries to couple. In this tutorial, it have been decided that the whole mesh representing the domains are used as sources. The meshes are expressed in terms of a number of nodes *ns*, a number of elements *nt*, a table of coordinates *coord*, a connectivity table *elems* and a connectivity descriptor *conni*. The meshes are shared with CWIPI using the instruction *PCW_Define_mesh* (line 85). The target points at the outlet (when *getnum=1*) and at the inlet (when *getnum=2*) are located with the flag used for the definition of the boundary conditions in *NSC2KE* (ie, 4 for an outlet and 6 for an inlet). The number of boundary points *nbexch* as well as the coordinates of these points *exchcoord* are transmitted to CWIPI with the instruction *PCW_set_points_to_locate*. Lines 97 and 98 are the allocation of the arrays used to send the data on the whole domain (*sendfield*) and to receive them on the coupled boundaries (*recvfield*).

Then, the routine *interf_exchange* is called at the end of the iteration loop in *NSC2KE*:

```
1         subroutine interf_exchange(nn,ns,nvar,t,kt,gam1,pres,presout,ua)
2          use palmlib
3          use cwipi
4          use mod_interf
5          implicit none
6   c
7          integer :: nn
8          integer :: ns
9          integer :: nvar
10         real    :: t
11         integer :: kt
12         real    :: gam1
13         real    :: pres(nn)
14         real    :: presout(nn)
15         real    :: ua(nvar,nn)
```

```
16  c
17          integer :: i
18          integer :: j
19          integer :: stride
20          integer :: il_err
21          integer :: nNotLocatedPoints
22          double precision :: time
23          double precision :: ua1
24          double precision :: ua2
25          double precision :: ua3
26          double precision :: ua4
27          character(len=PL_LNAME) :: cl_coupling_name
28          character(len=PL_LNAME) :: cl_exchange_name
29          character(len=PL_LNAME) :: cl_sending_field_name
30          character(len=PL_LNAME) :: cl_receiving_field_name
31  c
32          if(mod(kt,fcpl).eq.0) then
33  c
34            do i = 1, ns
35              sendfield(6*(i-1)+1) = ua(1,i)
36              sendfield(6*(i-1)+2) = ua(2,i)
37              sendfield(6*(i-1)+3) = ua(3,i)
38              sendfield(6*(i-1)+4) = ua(4,i)
39              sendfield(6*(i-1)+5) = ua(5,i)
40              sendfield(6*(i-1)+6) = ua(6,i)
41            enddo
42  c
43            time    = t
44            stride  = 6
45            itcompt = itcompt + 1
46            cl_coupling_name        = 'toy'
47            cl_exchange_name        = 'allexch'
48            cl_sending_field_name   = 'send'
49            cl_receiving_field_name = 'rec'
50            call PCW_Sendrecv (cl_coupling_name,
51       &                       cl_exchange_name,
52       &                       stride,
53       &                       itcompt,
54       &                       time,
55       &                       cl_sending_field_name,
56       &                       sendfield,
57       &                       cl_receiving_field_name,
58       &                       recvfield,
59       &                       nNotLocatedPoints,
60       &                       il_err)
61  c
62            if (getnum.eq.2) then
63              do i = 1, nbexch
64                j = indexch(i)
65                ua(1,j) = recvfield(6*(i-1)+1)
66                ua(2,j) = recvfield(6*(i-1)+2)
67                ua(3,j) = recvfield(6*(i-1)+3)
68                ua(4,j) = recvfield(6*(i-1)+4)
69              enddo
70            else
71              presout = pres
72              do i = 1, nbexch
73                j = indexch(i)
74                ua1        = recvfield(6*(i-1)+1)
75                ua2        = recvfield(6*(i-1)+2)
76                ua3        = recvfield(6*(i-1)+3)
77                ua4        = recvfield(6*(i-1)+4)
78                presout(j) = gam1*(ua4-0.5*(ua2*ua2+ua3*ua3)/ua1)
79              enddo
80            endif
81  c
82          endif
83  c
84        end subroutine interf_exchange
```

When the current iteration correspond to a multiple of the coupling frequency (line 32), this routine send the whole field *ua* of the solver on all the points of the grid. The array *ua* is composed of:

- *ua(1,.)*: density (ρ),
- *ua(2,.)*: horizontal momentum (ρu),
- *ua(3,.)*: vertical momentum (ρv),
- *ua(4,.)*: total energy per unit of volume ($\mathcal{E}$),

- *ua(5,.)*: kinetic energy of turbulence k (k),
- *ua(6,.)*: rate of dissipation of k (ϵ),

The units send the array *sendfield* and receive the interpolated data on the boundary nodes in the array *recvfield* at the same time with the instruction *PCW_Sendrecv*. Note that the name of the object *allexch* has to correspond to the one defined in the *Id card* of the unit. The domain #1 has to calculate the pressure from the received data in order to impose it at its boundary nodes. As *NSC2KE* works with non-dimensional variables, the pressure P is calculated with (line 78):

$$P = (\gamma - 1)\left(\mathcal{E} - \frac{1}{2}\frac{(\rho u)^2 + (\rho v)^2}{\rho}\right) \tag{0.1}$$

The computed pressure can then be imposed at the outlet boundary condition in the routine *cdl* of *NSC2KE* through the variable *pstar*. On the other hand, domain #2 can directly impose the received data ρ, ρu, ρv and $\mathcal{E}$ at the inlet (lines 65 to 68).

Finally the routine *interf_end* is called at the end of *NSC2KE* in order to deallocate the arrays used for the coupling as well as to delete the coupling:

```
1       subroutine interf_end
2         use palmlib
3         use cwipi
4         use mod_interf
5         implicit none
6         integer :: il_err
7         character(len=PL_LNAME) :: cl_coupling_name
8   c
9         deallocate ( exchcoord )
10        deallocate ( indexch   )
11        deallocate ( coord     )
12        deallocate ( elems     )
13        deallocate ( conni     )
14        deallocate ( sendfield )
15        deallocate ( recvfield )
16  c
17        cl_coupling_name = 'toy'
18        call PCW_Delete_coupling(cl_coupling_name,il_err)
19  c
20      end subroutine interf_end
```

Results

Figure 2 shows that with an inlet Mach number of 1.8, the flow is almost supersonic everywhere in the configuration. The main consequence is that no information goes upstream in the flow. Thus, the effort made to impose the pressure from domain #2 to domain #1 is not necessary in this particular case. This is clearly illustrated when several coupling frequencies are tested. Indeed, for the frequencies tested *fcpl* = {*1; 10; 100; 1000*}, the results in terms of Mach number and pressure fields are exactly the same and are superposable to the computation without coupling. Figures 9 and 10 present the Mach number and pressure field in the couple configuration, respectively. Figure 11 shows the corresponding profiles of Mach number and pressure on the center line of the configuration. This figure clearly illustrates that the results obtained with the coupled simulations are in a very good accordance with those obtained on the whole configuration.

In order to illustrate the importance to take into account the feed back in pressure in domain #1, a case with a subsonic inlet is simulated. The inlet Mach number is fixed to 0.615. Figure 12 shows that the coupled simulation without the pressure adjustment on the outlet of domain #1 gives erroneous results. The profiles of Mach number and pressure along the center line of the configuration (Fig. 13) confirm the importance of the pressure feed back in such a subsonic fluid-fluid coupling.

Conclusion

This first tutorial aims at providing an example of use of Open-PALM to construct a coupled application using the library CWIPI (M.P.Errera *et al.* 2010). To achieve this objective, a fluid-fluid

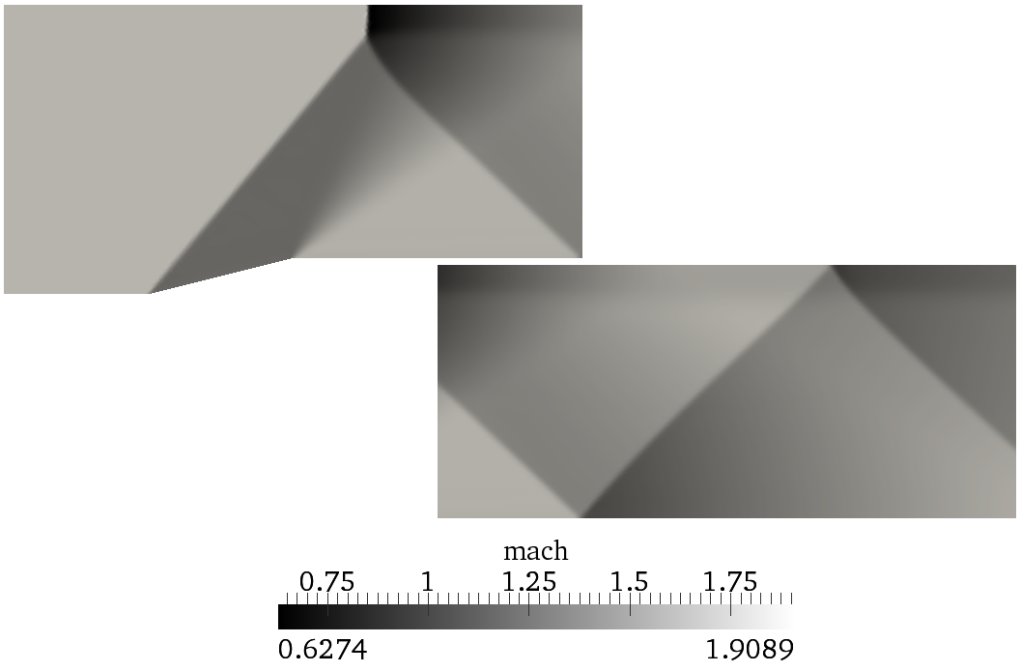

FIGURE 8. Mach number field in the couple configuration.

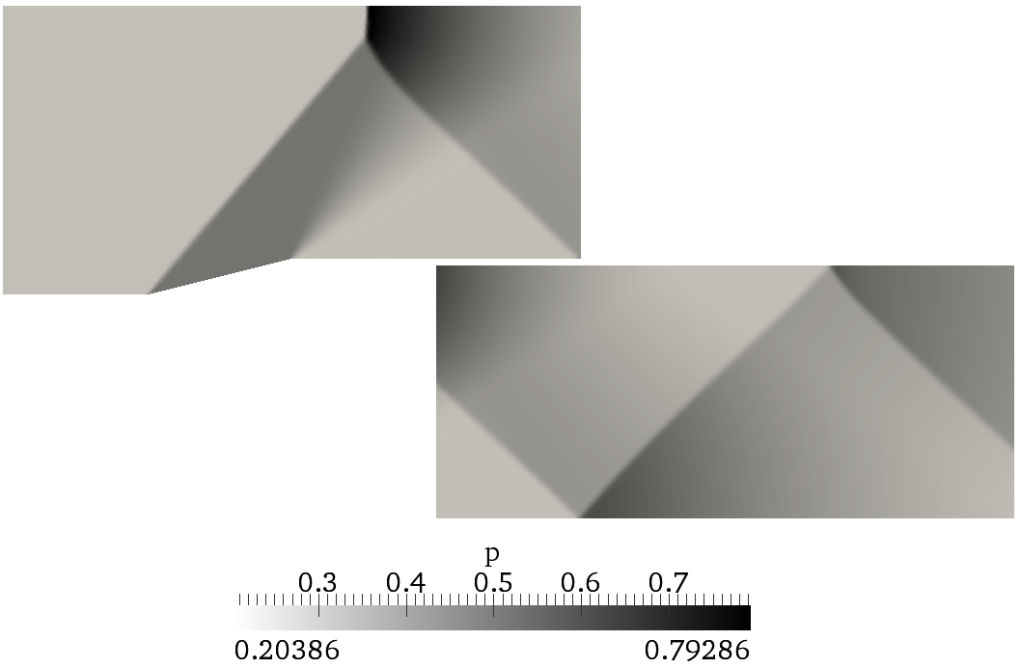

FIGURE 9. Pressure field in the couple configuration.

simulation is done by coupling the open source code *NSC2KE* (Mohammadi 1994) with itself. This code is a sequential solver but the implementation on a parallel solver is quite the same as described in this report. The work done during this study has allowed to use an interesting functionality of CWIPI. Indeed, a natural way to couple two solvers is to define an interface (ie a surface in 3D, a line in 2D) between them and to communicate on this interface. This leads to define meshes that support data to exchange and then directly communicate quantities on these meshes. Due to the

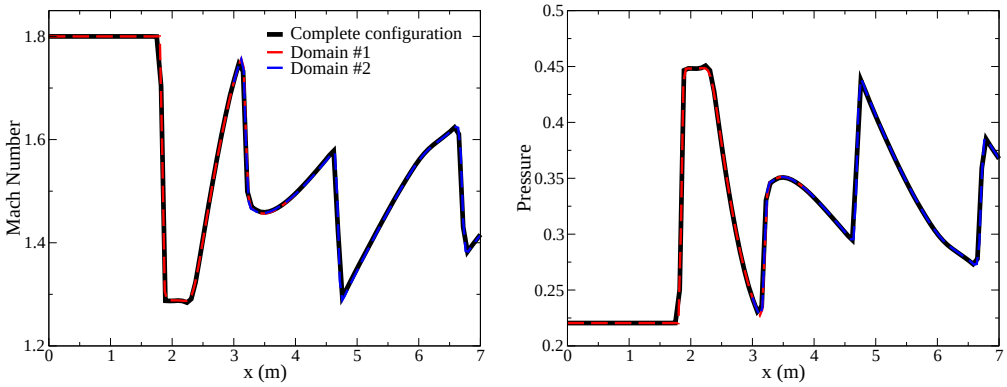

FIGURE 10. Mach number and pressure profiles along the center line of the uncouple (solid line) and couple configurations (dashed lines).

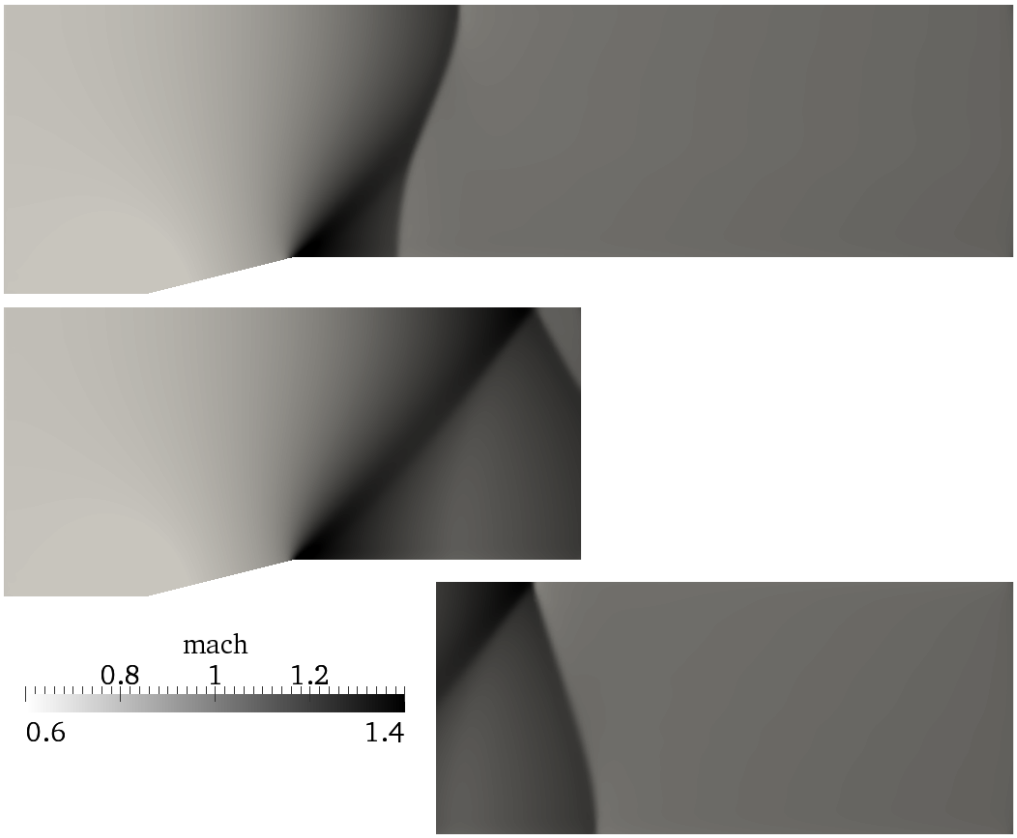

FIGURE 11. Mach number field in the uncouple and couple configuration without pressure feed back at inlet Mach number 0.615.

fact that, in this study, the domains to couple overlap, the functionality of *points to locate* has allowed to define sources meshes that differ from target points.

REFERENCES

MOHAMMADI, B. 1994 Fluid dynamics computation with nsc2ke. an user-guide. *Tech. Rep.* RT-0164. INRIA report.

M.P.ERRERA, QUÉMERAIS, E. & BAQUÉ, B. 2010 Approche multi-physique par couplage de

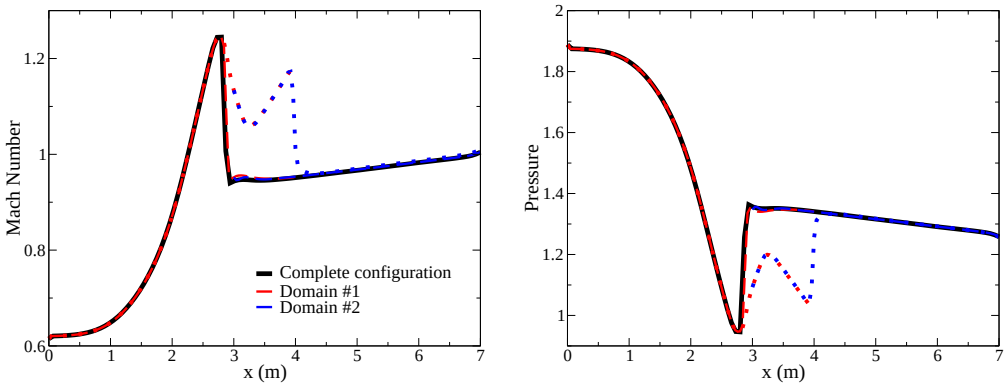

FIGURE 12. Mach number and pressure profiles along the center line of the uncouple (solid line) and couple configurations with the feed back in pressure (dashed lines) and without (dotted lines) for the simulations at inlet Mach number 0.615.

codes. application en aérothermique. In *1er Colloque International Francophone d'Energétique et Mécanique*. Saly, Sénégal.