

OpenPALM Tutorial: a first toy with *NSC2KE*

By F. Duchaine*, T. Morel* and E. Qu  merais**

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:

```
1 0          --> =0 2D, =1 AXISYMMETRIC
2 0          --> =0 Euler , =1 Navier-Stokes
3 166.       --> Reynolds by meter (the mesh is given in meter)
4 0.         --> inverse of Froude number (=0 no gravity)
5 1.8       --> inflow Mach number
6 1.         --> ratio pout/pin
7 1         --> wall =1 newmann b.c. on the temp.(adiabatic), =2 Dirichlet.(isothermal)
8 300.      --> inflow temperature (in Kelvin) for Sutherland laws
9 288.      --> if isothermal walls , wall temperature (in Kelvin)
10 0.0      --> angle of attack
11 3        --> Euler fluxes =1 roe, =2 osher,=3 kinetic
12 3        --> nordre = 1 first order scheme, =2 second order, =3 limited second order
13 0        --> =0 global time stepping (unsteady), =1 local Euler, =2 local N.S.
14 1.       --> cfl
15 10000    --> number of time step
16 10000    --> frequence for the solution to be saved
17 1.e10    --> maximum physical time for run (for unsteady problems)
18 -4.      --> order of magnitude for the residual to be reduced (for steady problems)
19 0        --> =0 start with uniform solution, =1 restart from INIT_NS
20 cccc turbulence cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
21 0        --> =0 no turbulence model, =1 k-epsilon model
22 0        --> =0 two-layer technique, =1 wall laws
23 1.e-2    --> delta in wall laws or limit of the one-eq. model. (in meter)
24 0        --> =0 start from uniform solution for k-epsilon, =1 from INIT_KE
25 -1.e10 1.e10 -1.e10 1.e10 --> xmin, xmax, ymin, ymax (BOX for k-epsilon r.h.s)
```

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

** D  partement Simulation Num  rique des   coulements et A  roacoustique (DSNA), ONERA, 29 Avenue de la Division Leclerc 92 322 Ch  tillon Cedex, France

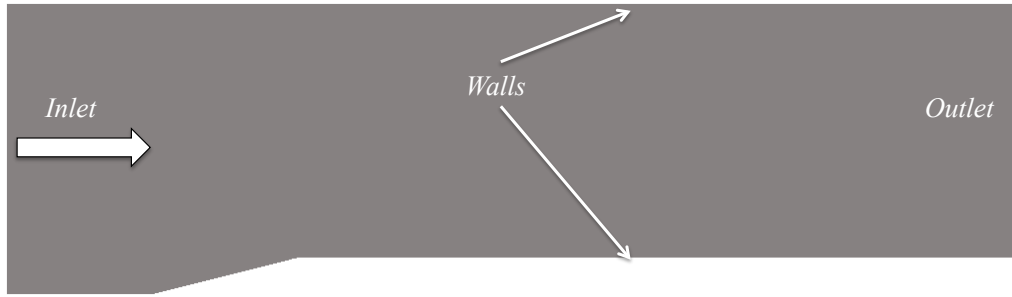


FIGURE 1. Global configuration.

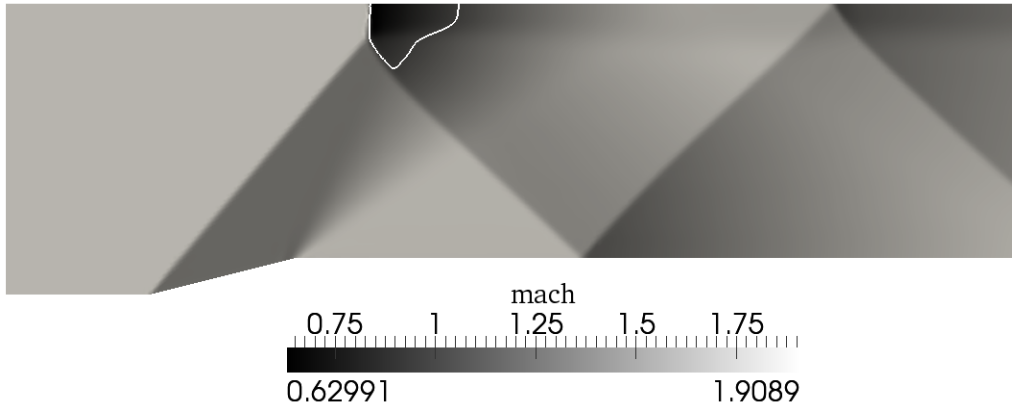


FIGURE 2. Mach number field and iso sonic line in the global configuration.

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 another 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_CWIPLCOUPLING -name toy
8 !
9 !PALM_CWIPLOBJECT -name allexch -coupling toy -intent INOUT

```

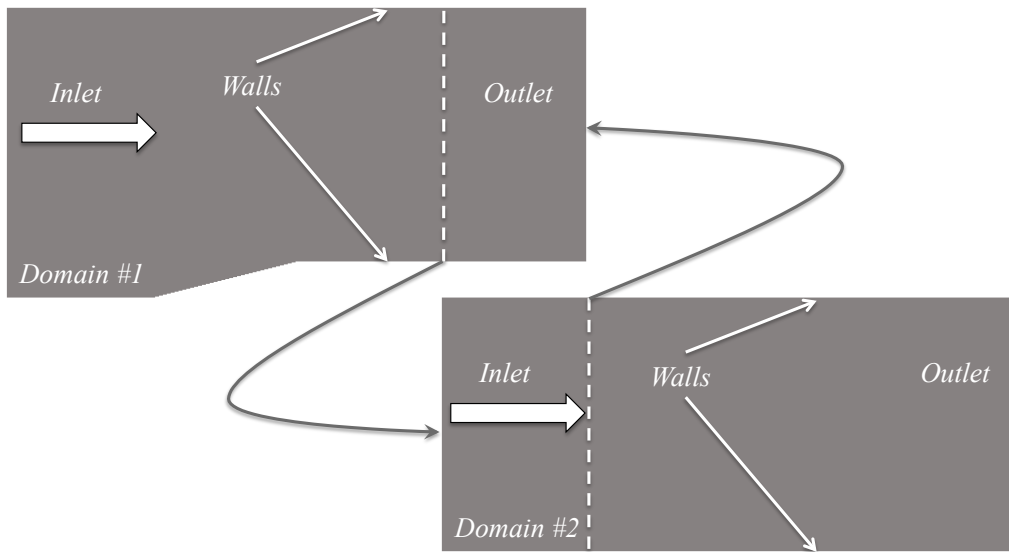
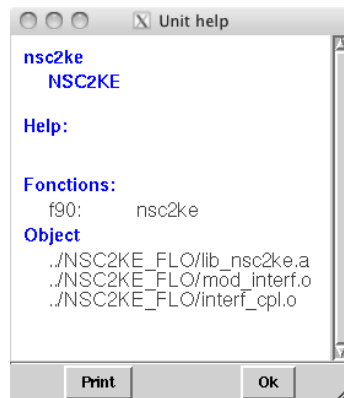


FIGURE 3. Fluid-fluid coupling configuration.

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

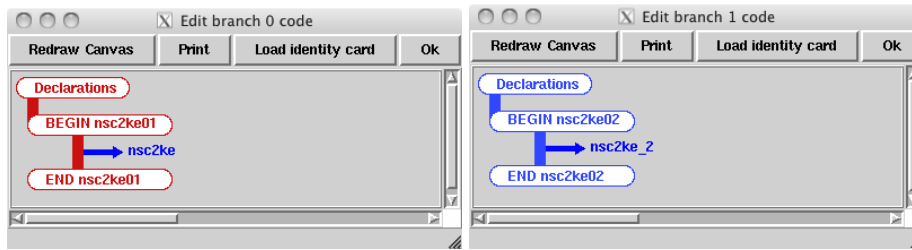
FIGURE 5. Integration of the *NSC2KE* units in the two branches.

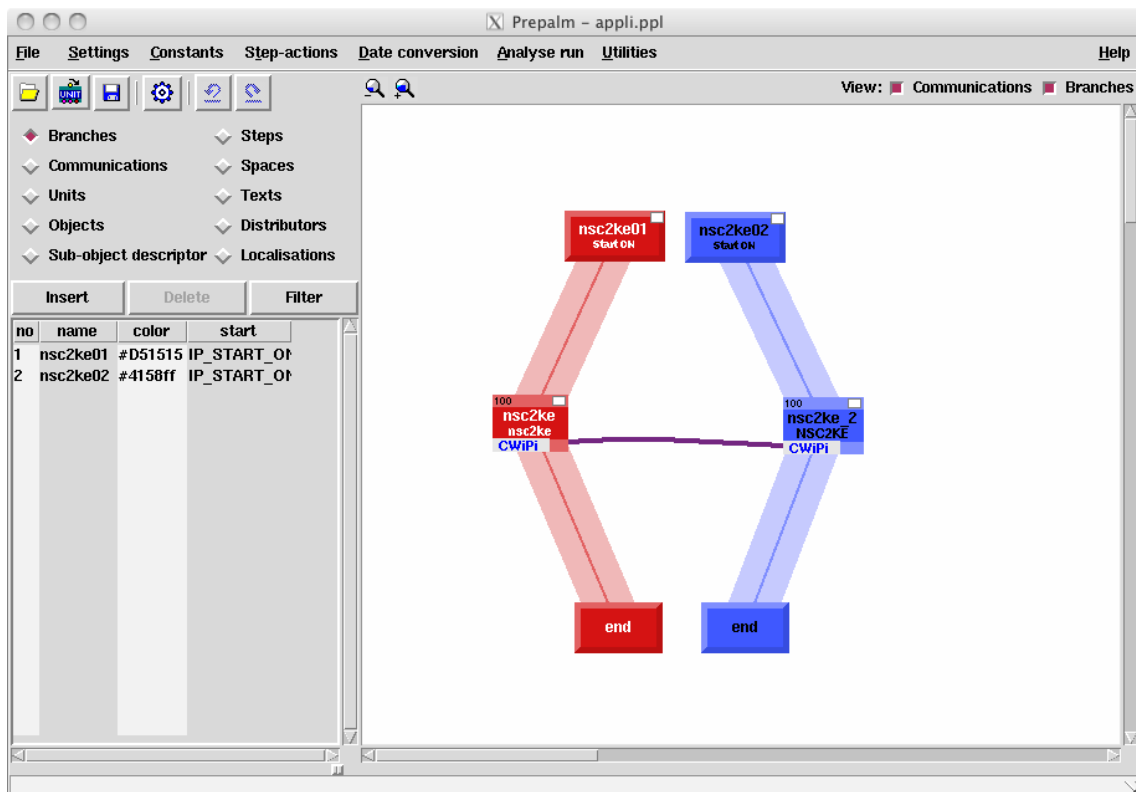
FIGURE 6. Insertion of the *NSC2KE* unit in a branch.

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:

```

1 module mod_interf
2   integer                :: getnum
3   integer                :: nbexch
4   integer                :: itcompt
5   integer                :: fcpl
6   double precision, dimension(:), allocatable :: coord
7   double precision, dimension(:), allocatable :: exchcoord
8   double precision, dimension(:), allocatable :: sendfield
9   double precision, dimension(:), allocatable :: recvfield
10  integer, dimension(:), allocatable :: elems
11  integer, dimension(:), allocatable :: conni
12  integer, dimension(:), allocatable :: indexch
13 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*:

```

1   subroutine interf_init
2     use palmlib
3     use cwipi
4     use mod_interf
5     implicit none
6     integer :: il_err
7     integer :: outfreq
8     double precision :: dimtol
9     character(len=PLNAME) :: cl_coupling_name
10    character(len=PLNAME) :: output_format
11    character(len=PLNAME) :: output_format_option
12  c
13    itcompt = 0
14    open(51, file="coupling.choices", status="old")
15    read(51,*) fcpl
16    read(51,*) outfreq
17    read(51,*) dimtol
18    read(51,*) getnum
19    close(51)
20  c
21    cl_coupling_name = 'toy'
22    output_format = 'Enight Gold'
23    output_format_option = 'text'
24    call PCW_Init(il_err)
25    call PCW_Create_coupling(cl_coupling_name,
26 &                          cwipi_cpl_parallel_with_part,
27 &                          2,
28 &                          dimtol,
29 &                          cwipi_static_mesh,
30 &                          cwipi_solver_cell_vertex,
31 &                          outfreq,
32 &                          output_format,
33 &                          output_format_option,
34 &                          il_err)
35  c
36    call cwipi_set_output_listing_f(PLOUT)
37    call cwipi_dump_appli_properties_f
38  c
39  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
3      c
4          use palmlib
5          use cwipi
6          use mod_interf
7          implicit none
8          integer      :: i
9          integer      :: j
10         integer      :: il_err
11         integer      :: nsm
12         integer      :: ntm
13         integer      :: ns
14         integer      :: nt
15         real          :: coor(2,nsm)
16         integer      :: nu(3,ntm)
17         integer      :: logfr(nsm)
18
19     c
20     character(len=PLLNAME) :: cl_coupling_name
21     integer                :: stride
22     integer                :: nNotLocatedPoints
23
24     c
25     nbexch = 0
26     if (getnum.eq.1) then      ! Output is coupled
27         do i = 1, ns
28             if (logfr(i).eq.4) nbexch = nbexch + 1
29         enddo
30     elseif (getnum.eq.2) then ! Inlet is coupled
31         do i = 1, ns
32             if (logfr(i).eq.6) nbexch = nbexch + 1
33         enddo
34     endif
35
36     c
37     allocate ( exchcoord(3*nbexch) )
38     allocate ( indexch(nbexch) )
39
40     c
41     j = 0
42     if (getnum.eq.1) then      ! Output is coupled
43         do i = 1, ns
44             if (logfr(i).eq.4) then
45                 j = j + 1
46                 exchcoord((j-1)*3+1) = coor(1,i)
47                 exchcoord((j-1)*3+2) = coor(2,i)
48                 exchcoord((j-1)*3+3) = 0.0d0
49                 indexch(j) = i
50             endif
51         enddo
52     elseif (getnum.eq.2) then ! Inlet is coupled
53         do i = 1, ns
54             if (logfr(i).eq.6) then
55                 j = j + 1
56                 exchcoord((j-1)*3+1) = coor(1,i)
57                 exchcoord((j-1)*3+2) = coor(2,i)
58                 exchcoord((j-1)*3+3) = 0.0d0
59                 indexch(j) = i
60             endif
61         enddo
62     endif
63
64     c
65     enddo
66
67     c
68     enddo
69
70     c
71     enddo
72
73     c
74     enddo
75
76     c
77     enddo
78
79     c
80     enddo
81
82     c
83     enddo
84
85     c
86     enddo
87
88     c
89     enddo
90
91     c
92     enddo
93
94     c
95     enddo
96
97     c
98     enddo
99
100    c
101    enddo
102
103    c
104    enddo
105
106    c
107    enddo
108
109    c
110    enddo
111
112    c
113    enddo
114
115    c
116    enddo
117
118    c
119    enddo
120
121    c
122    enddo
123
124    c
125    enddo
126
127    c
128    enddo
129
130    c
131    enddo
132
133    c
134    enddo
135
136    c
137    enddo
138
139    c
140    enddo
141
142    c
143    enddo
144
145    c
146    enddo
147
148    c
149    enddo
150
151    c
152    enddo
153
154    c
155    enddo
156
157    c
158    enddo
159
160    c
161    enddo
162
163    c
164    enddo
165
166    c
167    enddo
168
169    c
170    enddo
171
172    c
173    enddo
174
175    c
176    enddo
177
178    c
179    enddo
180
181    c
182    enddo
183
184    c
185    enddo
186
187    c
188    enddo
189
190    c
191    enddo
192
193    c
194    enddo
195
196    c
197    enddo
198
199    c
200    enddo
201
202    c
203    enddo
204
205    c
206    enddo
207
208    c
209    enddo
210
211    c
212    enddo
213
214    c
215    enddo
216
217    c
218    enddo
219
220    c
221    enddo
222
223    c
224    enddo
225
226    c
227    enddo
228
229    c
230    enddo
231
232    c
233    enddo
234
235    c
236    enddo
237
238    c
239    enddo
240
241    c
242    enddo
243
244    c
245    enddo
246
247    c
248    enddo
249
250    c
251    enddo
252
253    c
254    enddo
255
256    c
257    enddo
258
259    c
260    enddo
261
262    c
263    enddo
264
265    c
266    enddo
267
268    c
269    enddo
270
271    c
272    enddo
273
274    c
275    enddo
276
277    c
278    enddo
279
280    c
281    enddo
282
283    c
284    enddo
285
286    c
287    enddo
288
289    c
290    enddo
291
292    c
293    enddo
294
295    c
296    enddo
297
298    c
299    enddo
300
301    c
302    enddo
303
304    c
305    enddo
306
307    c
308    enddo
309
310    c
311    enddo
312
313    c
314    enddo
315
316    c
317    enddo
318
319    c
320    enddo
321
322    c
323    enddo
324
325    c
326    enddo
327
328    c
329    enddo
330
331    c
332    enddo
333
334    c
335    enddo
336
337    c
338    enddo
339
340    c
341    enddo
342
343    c
344    enddo
345
346    c
347    enddo
348
349    c
350    enddo
351
352    c
353    enddo
354
355    c
356    enddo
357
358    c
359    enddo
360
361    c
362    enddo
363
364    c
365    enddo
366
367    c
368    enddo
369
370    c
371    enddo
372
373    c
374    enddo
375
376    c
377    enddo
378
379    c
380    enddo
381
382    c
383    enddo
384
385    c
386    enddo
387
388    c
389    enddo
390
391    c
392    enddo
393
394    c
395    enddo
396
397    c
398    enddo
399
400    c
401    enddo
402
403    c
404    enddo
405
406    c
407    enddo
408
409    c
410    enddo
411
412    c
413    enddo
414
415    c
416    enddo
417
418    c
419    enddo
420
421    c
422    enddo
423
424    c
425    enddo
426
427    c
428    enddo
429
430    c
431    enddo
432
433    c
434    enddo
435
436    c
437    enddo
438
439    c
440    enddo
441
442    c
443    enddo
444
445    c
446    enddo
447
448    c
449    enddo
450
451    c
452    enddo
453
454    c
455    enddo
456
457    c
458    enddo
459
460    c
461    enddo
462
463    c
464    enddo
465
466    c
467    enddo
468
469    c
470    enddo
471
472    c
473    enddo
474
475    c
476    enddo
477
478    c
479    enddo
480
481    c
482    enddo
483
484    c
485    enddo
486
487    c
488    enddo
489
490    c
491    enddo
492
493    c
494    enddo
495
496    c
497    enddo
498
499    c
500    enddo
501
502    c
503    enddo
504
505    c
506    enddo
507
508    c
509    enddo
509

```

```

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 has 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=PLLNAME) :: cl_coupling_name
28     character(len=PLLNAME) :: cl_exchange_name
29     character(len=PLLNAME) :: cl_sending_field_name
30     character(len=PLLNAME) :: 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 &                          rcvfield ,
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) = rcvfield(6*(i-1)+1)
66                 ua(2,j) = rcvfield(6*(i-1)+2)
67                 ua(3,j) = rcvfield(6*(i-1)+3)
68                 ua(4,j) = rcvfield(6*(i-1)+4)
69             enddo
70         else
71             presout = pres
72             do i = 1, nbexch
73                 j = indexch(i)
74                 ua1 = rcvfield(6*(i-1)+1)
75                 ua2 = rcvfield(6*(i-1)+2)
76                 ua3 = rcvfield(6*(i-1)+3)
77                 ua4 = rcvfield(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):

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

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=PLNAME) :: 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

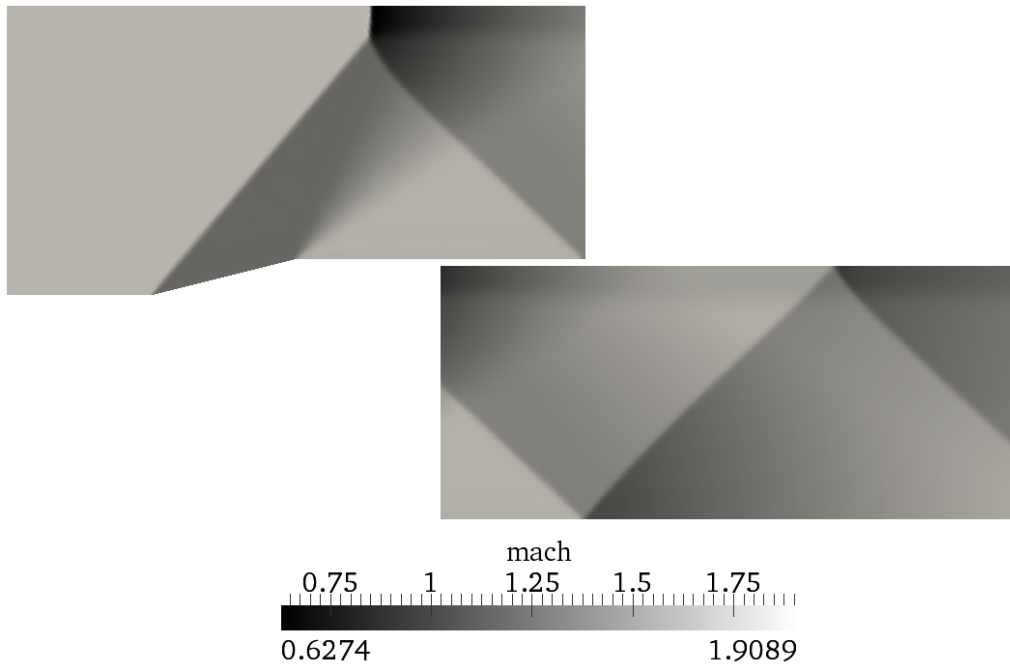


FIGURE 8. Mach number field in the couple configuration.

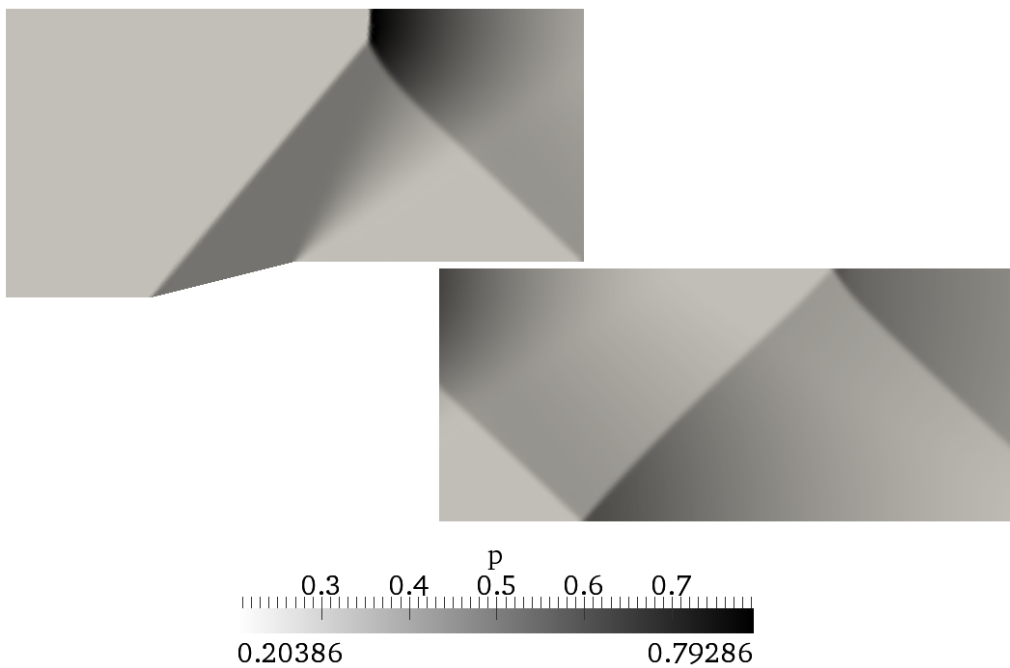


FIGURE 9. Pressure field in the couple configuration.

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

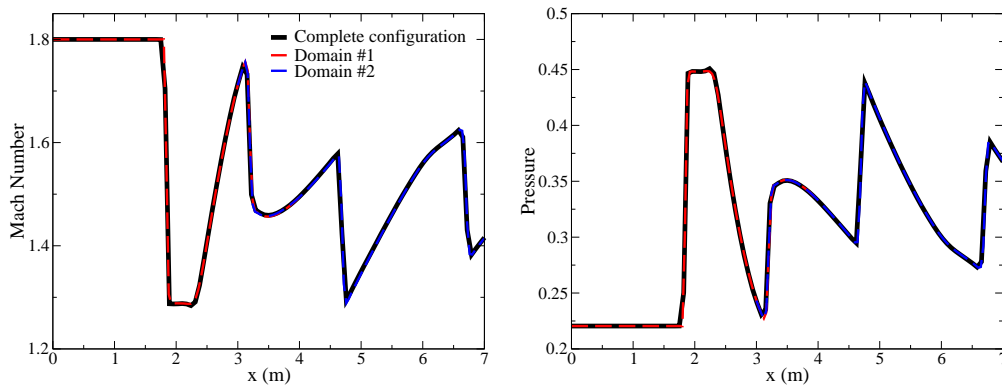


FIGURE 10. Mach number and pressure profiles along the center line of the uncouple (solid line) and couple configurations (dashed lines).

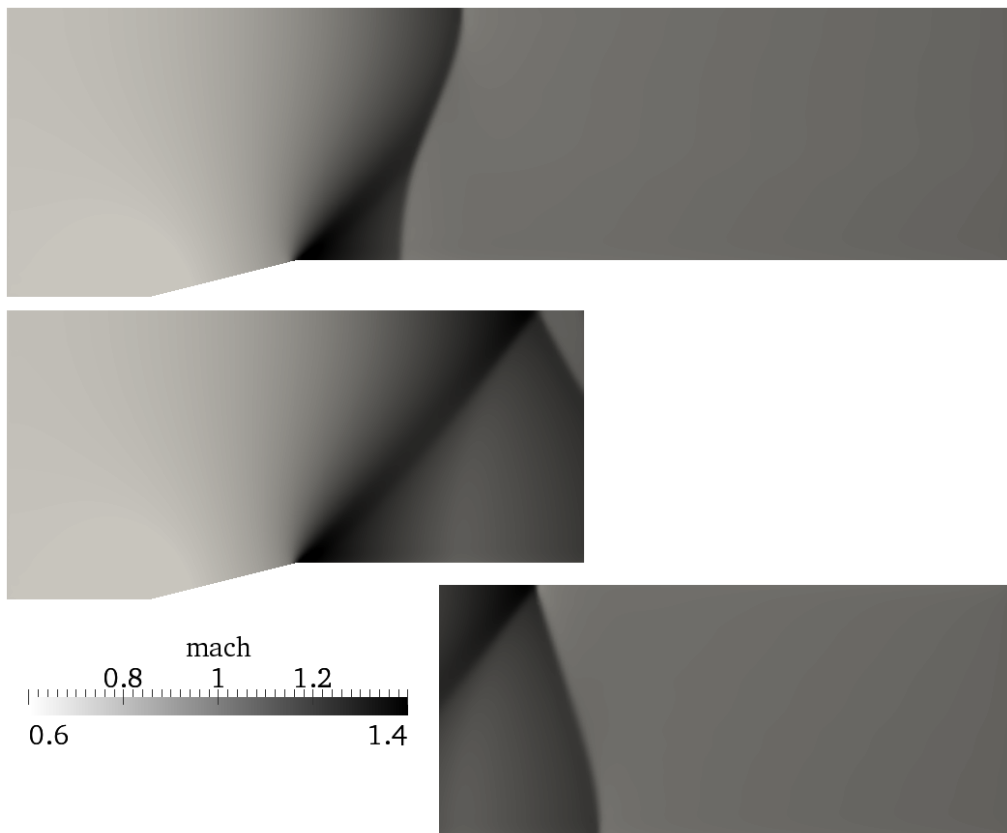


FIGURE 11. Mach number field in the uncouple and couple configuration without pressure feed back at inlet Mach number 0.615.

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

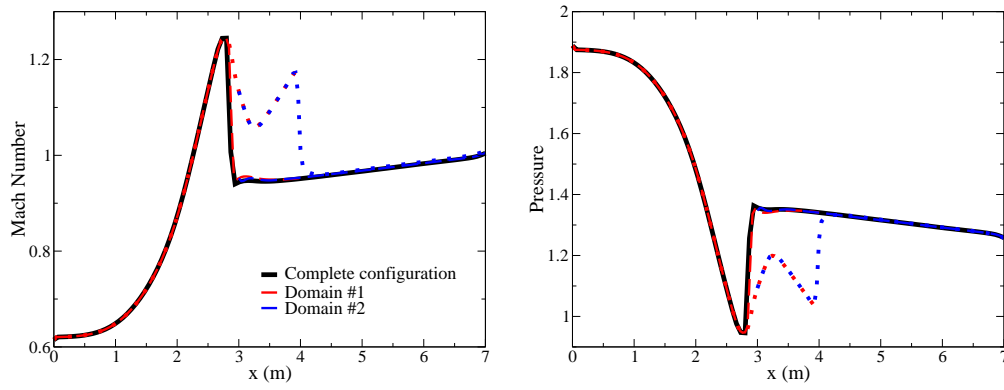


FIGURE 12. Mach number and pressure profiles along the center line of the uncouple (solid line) and couple configurations with the feed back in pressure (dashed lines) and without (dotted lines) for the simulations at inlet Mach number 0.615.

codes. application en a  rothermique. In *1er Colloque International Francophone d'Energetique et Mecanique*. Saly, S  n  gal.