The IMP4 wrapper for running IMP4 codes in the UAL framework

ITM-Project IMP 4

D. Reiser, A.H. Nielsen

The need for a Wrapper-Routine:

The IMP4 wrapper builds the bridge between the particular IMP4 code and the UAL communication standards via XML and CPO's.

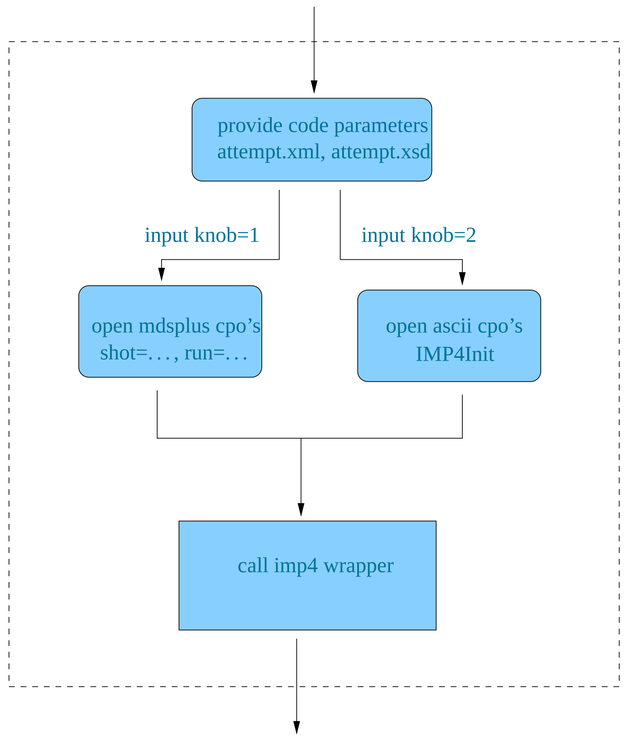
The IMP4 code itself is isolated to a large extent in the sense that it keeps its original structure and is provided as a subroutine using standard FORTRAN arguments for the basic necessary input parameters and the output of computations.

All UAL specific issues

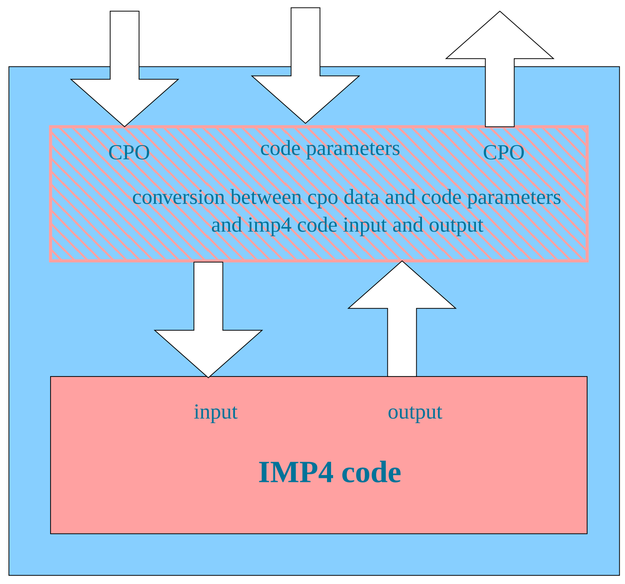
are handled by the IMP4 wrapper,

i.e. CPO read/write
and XML-parsing.

Running an IMP4 code via the IMP4 wrapper



Structure of IMP4 wrapper and IMP4 code



Basics about implementing codes in ITM environment:

- The simulation code must be structured as Subroutine, but is kept unchanged otherwise
- The UAL specifics are located solely in the Wrapper-Routine
- Usually the Makefile needs extensive changes
- The UAL communication needs setting of several environment variables
- The communication via the Wrapper is compatible with MPI parallelized codes
- The Wrapper can be used directly on both, the Gateway and HPC-FF
- The Wrapper allows to create a Kepler actor out of the particular code submitting a job on the Gateway

```
subroutine imp4_wrapper(coreprof,equilibrium,coretransp &
  ,coretransp_out,code_parameters)

  !... the wrapper routine to connect IMP4_ATTEMPT with cpo framework

  USE EUITM_SCHEMAS
  USE EUITM_ROUTINES
  USE ITM_CONSTANTS

  USE imp4_wrapper_coeff

  implicit none

  include "mpif.h"

  integer(ITM_I4)          :: i,irho,nrho_core,nrho_equi,return_status
  REAL(R8)                 :: rho,drdrho, dqdrho,dndrho, a0, B0, R0
  REAL(R8),    allocatable :: rho_tor_core(:), rho_tor_equi(:)
  REAL(R8),    allocatable :: rho_vol_equi(:),n0(:),Te(:),q(:)

  type (type_coreprof),    pointer :: coreprof(:)
  type (type_equilibrium), pointer :: equilibrium(:)
  type (type_coretransp),  pointer :: coretransp(:)
  type (type_coretransp),  pointer :: coretransp_out(:)

  TYPE (type_param)                :: code_parameters

  integer(ITM_I4)                  :: np, ierr, rank

  !... first check for number np of processors available
  call   MPI_COMM_SIZE(MPI_COMM_WORLD,np,ierr)
  !... get rank pid of every processor in communicator MPI_COMM_WORLD
  call   MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)

  !  IMP4 Code specific: default values of need parameters in IMP4 code4
  c_Ns = 8
  c_Nx = 32
  c_Ny = 64
  c_Nt = 100
  c_ws = 1
  c_wx = 1
  c_mi = 1
  c_a  = 0.4

  !-- assign code parameters to internal variables
  call assign_code_parameters(code_parameters, return_status)

  ! We can read B0 and R0 from coreprof
  B0 = coreprof(1)%toroid_field%b0
  R0 = coreprof(1)%toroid_field%r0
  a0 = equilibrium(1)%eqgeometry%a_minor
```

```
  ! Find the size of rho_tor, number of radial points
  nrho_core = size(coreprof(1)%rho_tor)

  ! Find the size of rho_tor, number of radial points
  nrho_equi = size(equilibrium(1)%profiles_1d%rho_tor)
  ! allocate arrys to hold profiles and coordinates
  allocate(rho_tor_core(nrho_core), rho_tor_equi(nrho_equi) &
    ,rho_vol_equi(nrho_equi),n0(nrho_core),Te(nrho_core),q(nrho_equi))

  ! Read rho_tor from coreprof
  rho_tor_core = coreprof(1)%rho_tor

  !Read rho_tor and rho_vol from equilibrium
  rho_tor_equi =    equilibrium(1)%profiles_1d%rho_tor
  rho_vol_equi = a0*equilibrium(1)%profiles_1d%rho_vol

  ! Read no and Te profile from coreprof
  n0           = coreprof(1)%ne%value
  Te           = coreprof(1)%te%value
  ! Read q-profile from equilibrium
  q            = equilibrium(1)%profiles_1d%q

  ! Find the parameters needed by ATTEMPT
  ! Convert the reference point, c_a, to units of rho_tor
  call L3interp(rho_tor_equi,rho_vol_equi,nrho_equi,rho,c_a,1)

  ! Interpolate to find n, Te and q at the reference point
  call L3interp(n0,rho_tor_core ,nrho_core,c_n0,rho,1)
  call L3interp(Te,rho_tor_core ,nrho_core,c_Te,rho,1)
  call L3interp(q, rho_tor_equi ,nrho_equi,c_q0,rho,1)

  ! Calculate dr/drho , dqdrho and dn/drho
  call L3deriv(n0,rho_tor_core,nrho_core,dndrho,rho,1)
  call L3deriv(q ,rho_tor_equi,nrho_equi,dqdrho,rho,1)
  call l3deriv(rho_vol_equi,rho_tor_equi,nrho_equi,drdrho,c_a,1)

  ! Calculate -n0/dn0/dr and r*dqdr/q
  c_Lp = -c_n0/dndrho*drdrho
  c_sh =  c_a /c_q0*dqdrho/drdrho

  c_R0 = R0
  c_B0 = B0

  !... start the simulation run
  call IMP4_ATTEMPT(c_Ns,c_Nx,c_Ny,c_Nt,c_B0,c_Lp,c_a,c_sh,c_q0 &
     ,c_R0,c_mi,c_n0,c_Te,c_ws,c_wx,c_flux,c_Deff)

  ! Return with Deff
  allocate(coretransp_out(1))
  coretransp_out(1)%time = 0.0
  write(*,*) 'coretransp_out(1)%time = ', coretransp_out(1)%time
  allocate(coretransp_out(1)%ne_transp%diff_eff(nrho_core,3))
  coretransp_out(1)%ne_transp%diff_eff(1:nrho_core,1) = c_Deff

  return

end subroutine imp4_wrapper
```

Comments on implementation and the svn-repository:

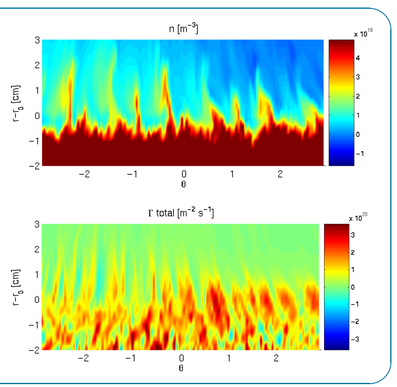
The wrapper and running example of the drift-fluid turbulence code ATTEMPT [Physics of Plasmas 14, 082314 (2007)] can be downloaded from a GFORGE svn-repository.

http://gforge.efda-itm.eu/svn/cpo_interface/tags

A detailed README file gives step by step instructions how to install the code

The code is MPI parallelized and can be used on the Gateway and HPC-FF platform

Standard output are turbulent diffusion coefficients, but also a large number of profiles etc. are provided



Quick guide to a test run - the README file

```
## download the files

## Download only the folders you need from the repository
svn checkout --username your_username
http://gforge.efda-itm.eu/svn/cpo_interface/branches/IMP4_ATTEMPT.kit

## go to the branch
cd IMP4_ATTEMPT.kit

## IMPORTANT NOTE!!
##
## if you want to use cpo data structure (on gateway only) change
## the hard-wired variable input_knob to 1 in source/testbed/imp4.f90
## in that case you also have to adjust the hard-wired variables
## shot and run for the particular data set you want to use
##
## if you want to use ascii substitute of cpo data structure (necessary for hpc-ff)
## change the hard-wired variable input_knob to 2 in source/testbed/imp4.f90
## in that case an ASCII data file named IMP4Init must be present in the
## working directory where you start the code
##

## compile the code and libraries
## hdf5=yes if hdf5 is available and used
## hdf5=no if hdf5 is not available or not necessary

cd code

gmake -f Makefile.machine "hdf5=yes" imp4.x
gmake -f Makefile.machine "hdf5=yes" libimp4

## start a test run in directory run/test

cd ../runs/test/

## put the executable into the working directory

ln -s ../../code/imp4.x .
## or
cp ../../code/imp4.x .

## set the environment variables as in the example
## of file settings.gateway

source /afs/efda-itm.eu/project/switm/scripts/set_itm_env 4.07b
source /afs/efda-itm.eu/project/switm/scripts/set_itm_data_env dreiser Textor 4.07b
```

```
## before starting the job the code parameters
## in the file attempt.xsl have to adjusted for
## the particular run (description below)

## start the batch script
## on the gateway

qsub batch.job.gateway

## start the batch script
## on the hpc-ff

msub batch.job.hpcff

################################################################################
################################################################################
################################################################################

## Description of input files for runing the job

## the input file attempt.xml


################################################################################
 <?xml-stylesheet type="text/xsl" href="./input_ATTEMPT.xsl"
 charset="ISO-8859-1"?>

 <parameters>

   <ATTEMPT>
     <c_Ns> 16     </c_Ns>
     <c_Nx> 64     </c_Nx>
     <c_Ny> 128    </c_Ny>
     <c_Nt> 1000   </c_Nt>
     <c_B0> 2.0    </c_B0>
     <c_a>  0.4    </c_a>
     <c_R0> 1.75   </c_R0>
     <c_mi> 2.0    </c_mi>
     <c_ws> 1      </c_ws>
     <c_wx> 1      </c_wx>
   </ATTEMPT>

 </parameters>
################################################################################
```