MARIN

BETTER SHIPS, BLUE OCEANS

SEACAL

Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

PhD Sebastian Costamagna
Scientific Software Specialist
R&D Dept. - MARIN
The Netherlands

MARITIME RESEARCH INSTITUTE NETHERLANDS

BETTER SHIPS, BLUE OCEANS

MARIN is a globally recognised top institute for hydrodynamic and nautical research. Our mission is 'Better Ships, Blue Oceans': we stand for clean, smart and safe shipping and sustainable use of the sea. We do this as an independent knowledge partner for the maritime sector, government and society.

SEAkeeping **CAL**culations

User group

>50 Users

>15 Institutions

Dev. Team

-T. Bunnik

-E. van Dalen

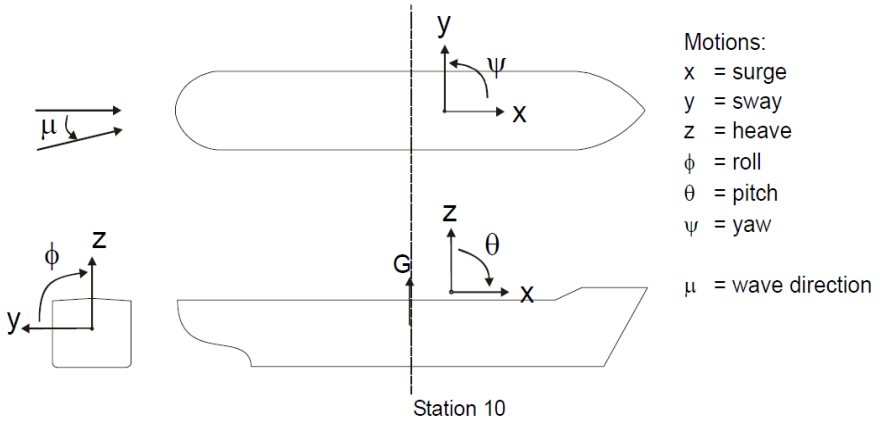
-R. van ‘t Veer

-S. Costamagna

SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

Frequency domain linear 3D diffraction theory. Advanced methods for calculations of roll damping, multi-hull/bodies motions, flexible ship responses, drift forces, effect of appendages, etc. on regular waves

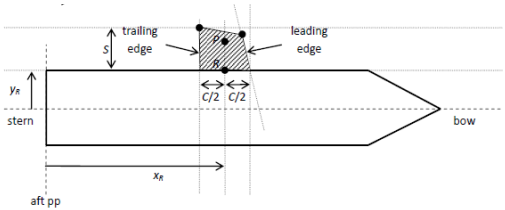
Wave direction and motions



Motions:
- x = surge
- y = sway
- z = heave
- ϕ = roll
- θ = pitch
- ψ = yaw

μ = wave direction

Appendages



Zero-speed Green's function Method

-Ship geometry described by ***panels*** (2k-15k)

-Excellent prediction at zero speed

-5 speeds, 13 headings and 30 frequencies

(=2000 ***conditions***) -> 12h on 4 CPUs

Rankine Source theory

-Ship and water FS described by ***panels*** (35k-85k)

-Excellent prediction at zero and forward speed

-5 speeds, 13 headings and 30 frequencies

(=2000 ***conditions***) -> 8h on 200 CPUs

Alternative method: CFD (non-linear effects)

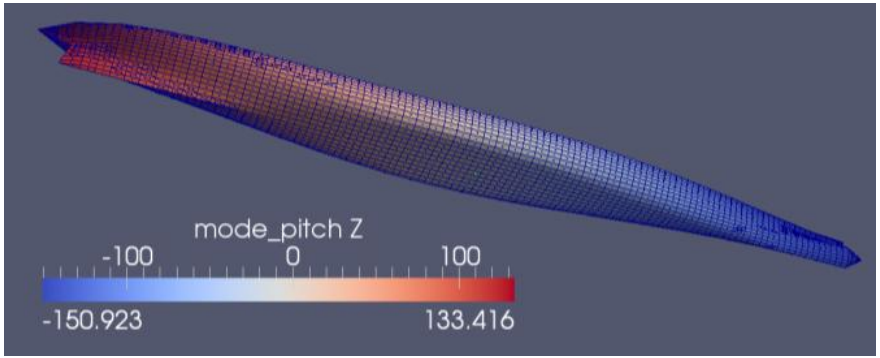
-Excellent prediction at zero and forward speed

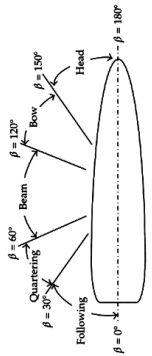
-1 speeds, 1 heading and 1 frequency

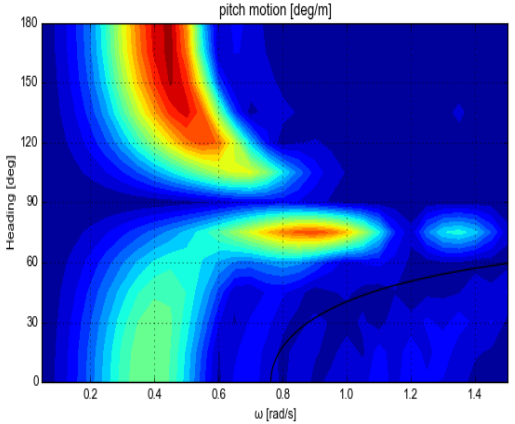
(=1 ***condition***)-> 36h on 200 CPUs (**x9.000 slower**…)

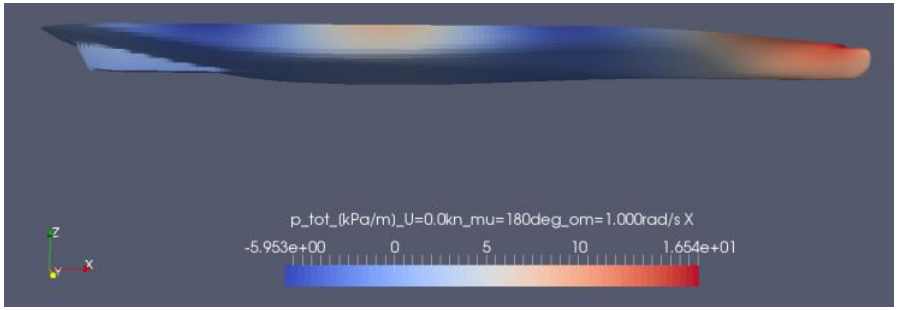
SEAkeeping **CAL**culations

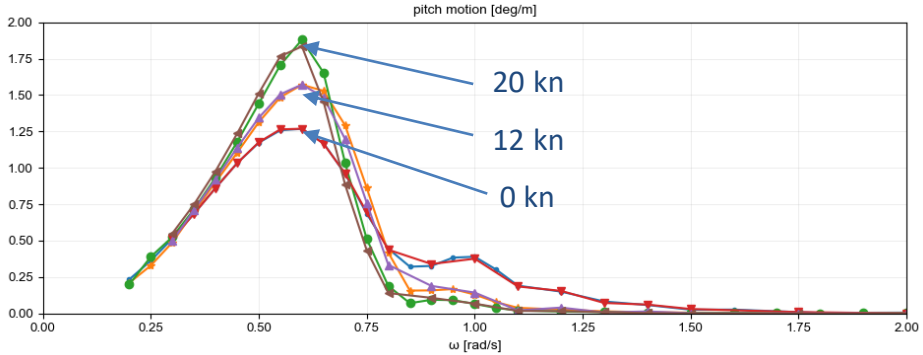
SEACAL: Zero-speed Green's function method







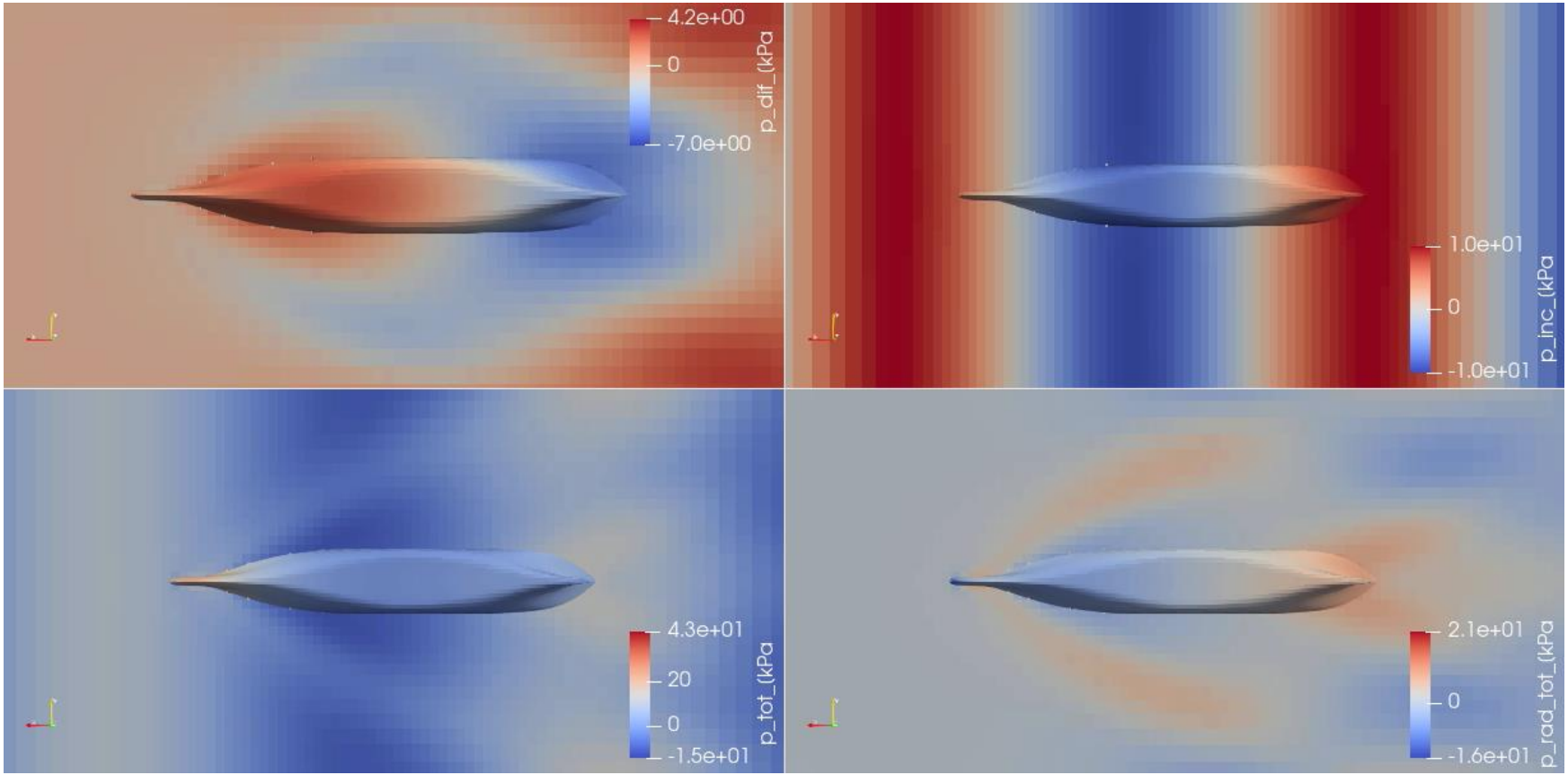




SEAkeeping **CAL**culations

Response Amplitude Operator (RAO)

SEACAL: Rankine source theory



SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

MARIN

Hydrostatics

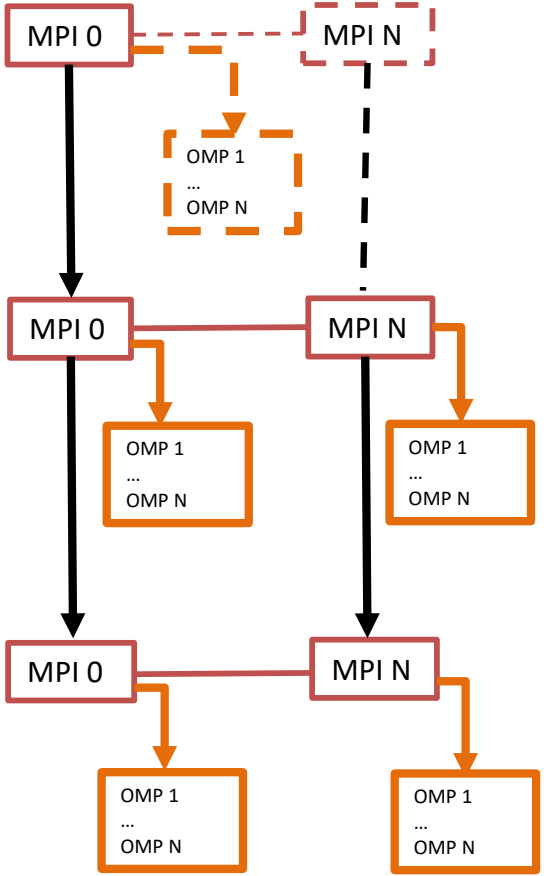
Hydrodynamics

RankineSrc
- AddNodes(MPI)
- Panels(MPI)
- GreenFunct(OMP)

RadDifSol
- Conditions(MPI)
- Solver(OMP)

Responses

DriftForces(MPI)
TimeEvolution(OMP)



MPI: **Loop Condition: U, mu, om, Aphi**

OMP: **Loop Panel: Ship, Water FS**

```
# i_cond, i_mu, i_U, i_om, mpiproc_cond(i_cond)
      1      1     1     1      0
      2      1     1     2      1
      3      1     1     3      2
      4      1     2     1      3
      5      1     2     2      4
      6      1     2     3      5
```

```
# mpi_size            6
#N_FreeSurf_panel,N_hydro_panel_total
       25376        7048
#FSPanels_per_Rank,HydroPanels_per_Rank
        4229        1174
#i_rank,N_FreeSurf_panel_rank_sta,N_FreeSurf_panel_rank_end
           1           1        4229
           2        4230        8458
           3        8459       12687
           4       12688       16916
           5       16917       21145
           6       21146       25376
#i_rank,N_hydro_panel_rank_sta,N_hydro_panel_rank_end
           1           1        1174
           2        1175        2348
           3        2349        3522
           4        3523        4696
           5        4697        5870
           6        5871        7048
```

SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

Cmake (software compilation process)

Compiler flags (platform/compiler/version dependent). External Libs. MSVS solution file (Win). Eclipse solution file (Linux). Installation step.

Windows: Compiler: ifort (Intel); MPI: MS-MPI & IntelMPI

```
/NODEFAULTLIB                                                          #extLib compiler flags
/fpp /traceback /Qopenmp /Qmkl:parallel /libs:dll
set_source_files_properties(XXX PROPERTIES COMPILE_FLAGS/Qoverride-limits) #OMP compilation
/STACK:268435456,268435456                                             #OMP run-time
set_source_files_properties(YYY PROPERTIES HEADER_FILE_ONLY TRUE)
target_compile_options(${MARIN_MODULE} PUBLIC XXX/Include/mpi.f90)     #MS-MPI Lib
target_link_libraries(${MARIN_MODULE} XXX/Lib/x64/msmpifec.lib)        #MS-MPI Lib
target_link_libraries(${MARIN_MODULE} XXX/Lib/x64/msmpi.lib)           #MS-MPI Lib

INSTALL(FILES <INTEL>/windows/mpi/intel64/bin/release/impi.dll"
  DESTINATION "${CMAKE_HOME_DIRECTORY}/eoe/bin/win64" RENAME msmpi.dll)    #INTEL-MPI Lib
```

Linux: mpiifort (Intel) & mpifort (GNU)

```
-fopenmp -fbacktrace -heap-arrays -ffree-line-length-512 -foverride-limits # mpifort (gnu)

 Compiler optimizations                                                    #NO SIMD | NO AVX512; Only –O2
---------------------------------------
ulimit -s unlimited       #omp run-time
setenv MKL_NUM_THREADS 1
setenv OMP_NUM_THREADS 1
setenv OMP_STACKSIZE 250M #omp run-time
```

MPI parallelization

```
#ifdef use_mpi_f08
#ifndef _WIN32
use mpi_f08
#endif
#endif

#ifdef use_mpi_f08
#ifdef _WIN32
include 'mpif.h'
#endif
#endif
```

Example: Loop over conditions

```
iCond_MPI : do i_cond=1,N_U*N_omega_tot*N_mu ! loop over all input conditions

    if (mpiproc_cond(i_cond) .ne. my_rank) cycle iCond_MPI ! skip if not my rank

    i_U  = i_U_all(i_cond)     ! retrieve input indices
    i_om = i_om_all(i_cond)
    i_mu = i_mu_all(i_cond)

    U  = thisSim%parHYD%shipSpeed(i_U)      ! use input indices
    mu = thisSim%waveSys%waveDir(i_mu)
```

```
module MPI_Calc
public :: &
 InitEndMPI,          &     ! start and end MPI interface
 prepareMPI,          &     ! distribute conditions over processes
 prepareMPIwAphi,     &     ! distribute conditions, including roll input, over processes
 prepareMPI_panels,   &     ! distribute panels over processes
 prepareMPI_AddNode,  &     ! distribute added nodes processes
 get_MPIprocCond,     &     ! retrieve process number on which a condition is calculated
```

MPI-3 shared memory (easy to implement!)

SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

MARIN

OMP parallelization

Current approach: OMP Low level panel loop

```
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(i_panel)
!$OMP DO
  do i_panel=1,N
    call NumIntGreen_interface(&
          &A=A(i_panel,1:3), & ! unit = m,m,m
          &B=B(i_panel,1:3), & ! unit = m,m,m
          &C=C(i_panel,1:3), & ! unit = m,m,m
          &D=D(i_panel,1:3), & ! unit = m,m,m
          &P=P, & ! unit = m,m,m
          &k_e_inf_h=k_e_inf_h, & ! unit = -
          &k_e_inf=k_e_inf, & ! unit = -
          &lambda_e_inf=lambda_e_inf, & ! unit = m
          &area123=area123(i_panel), &
          &area341=area341(i_panel), &
          &area_max=area_max, & ! unit = m2
          &IFS=IFS, IBS=IBS, &
          &G=G(i_panel), Gx=Gx(i_panel), Gy=Gy(i_panel), Gz=Gz(i_panel))
  enddo
!$OMP END DO
!$OMP END PARALLEL
```

```
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(i_panel, intG)
!$OMP DO
do i_panel=1,N
   call Rankine_singularity_G(intG, A(i_panel,1:3), B(i_panel,1:3),C(i_panel,1:3),
                                    D(i_panel,1:3), P, epsilon, this_analytical)
   intG_row(i_panel)=intG
enddo
!$OMP END DO
!$OMP END PARALLEL
```

Previous approach: OMP high level input cond. loop

```
#ifdef highlevel_omp
!$omp parallel do schedule (dynamic), &
!$omp&  private(thisThread, i_cond, i_U, i_mu, i_om, U, mu, tau, &
!$omp&  phi_wl, pres_wl, prs_wl_db, prs_wl_uniform, prs, prs_db, &
!$omp&  prs_uniform, &
!$omp&  zeta, zeta_wl_db, zeta_wl_uniform, &
!$omp&  int_gradG2, &
!$omp&  j_count, j_count_start, i_count, i_count_start, singular, &
!$omp&  suppressIrregFreq, fid_hydfile, fn, &
!$omp&  mat_temp,i_mode,i_ship,i_panel,j_ship,j_panel, &
!$omp&  exists, stat, N_message, messages, sResult, error, error_marin, &
!$omp&  all_A, all_B, all_C, all_D, &
!$omp&  all_area123, all_area341, &
!$omp&  all_G, all_Gx, all_Gy, all_Gz, &
!$omp&  puvw, row_phi_uvw), &
!$omp&  firstprivate(tempRegWave, &
!$omp&  A_j, area_max, DD, h, k_e_inf, k_e_inf_h, KR, KZ, &
!$omp&  lambda_e_inf, omega_0, omega_e,  &
!$omp&  RP, R, RSD, PVI, X, Y, Z, factor, &
!$omp&  P_add_wl, TERM, v_s, grad_h, P_add, &
!$omp&  v, v_inc, v_wl, &
!$omp&  grad_v, int_dG2dn_i, int_G2 , phi, D_R_hydro, &
!$omp&  rhsMat_dble, rhsMat_grad_dble, rhsMat_even_dble, &
!$omp&  rhsMat_even_grad_dble, rhsMat_odd_dble, rhsMat_odd_grad_dble, &
!$omp&  sol_even_dble, sol_even_grad_dble, sol_odd_dble, sol_odd_grad_dble, &
!$omp&  sol_GMRES_dble, sol_grad_dble, last_omega_e, ipiv, ipiv_even, ipiv_odd, &
!$omp&  symIndex, ALL_sym, P_i, P_j, sol_GMRES_RANKINE, &
!$omp&  i_pnl,j_pnl)
#endif
  do i_cond=1,N_U*N_omega_tot*N_mu ! i_cond
```

```
integer :: N_omega = HUGE_R      ! implied SAVE ; SUPER Dangerous!
```

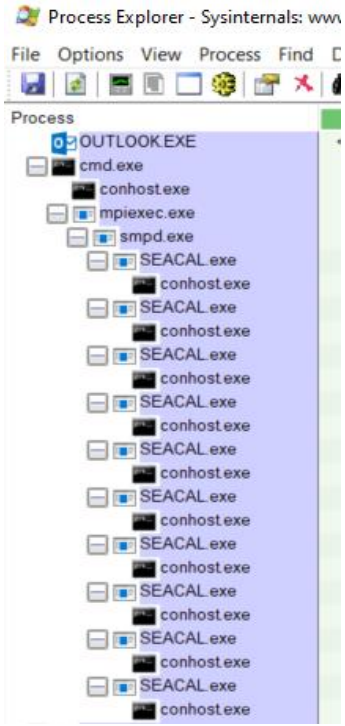
Thread safe; OMP CRITICAL

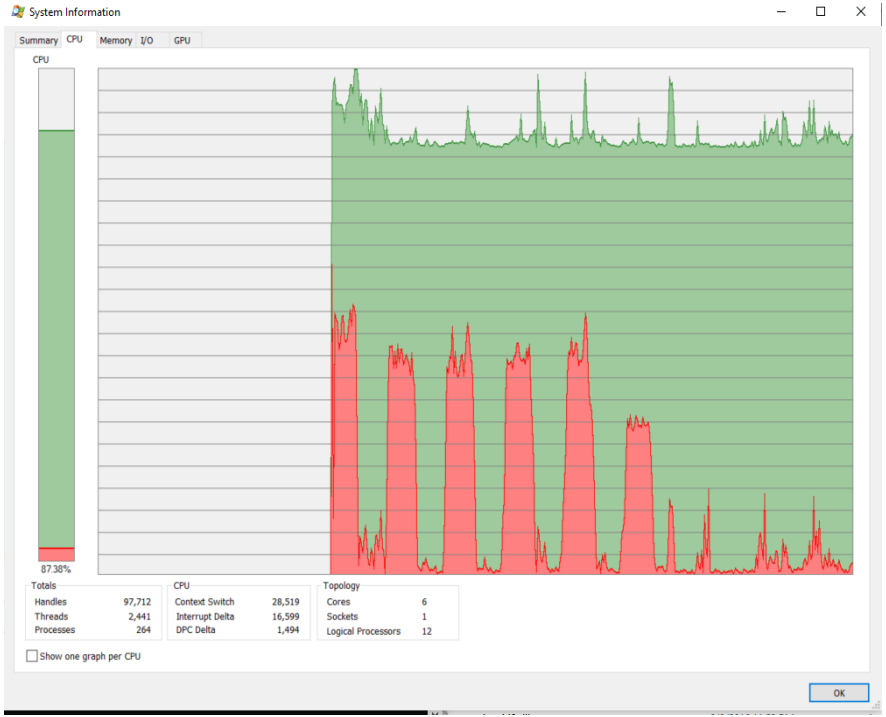
Missing feature: OMP private copies of structure child's `thisSim%A(i,j,k)`

```
allocate(tempMesh_i(n_threads,N_ship))
! Create local copies for each omp thread
do i_threads =1, n_threads
   do i_ship = 1, N_ship
      call Copy(thisMeshOrig=thisSim%ship(i_ship)%hydroMesh%mesh,
                thisMeshCopy=tempMesh_i(i_threads,i_ship))
   enddo
enddo
```

Windows

```
$>start /B smpd
$>mpiexec.exe -n 10 -host localhost SEACAL -c XXX
$>taskkill /F /IM smpd.exe /T
```



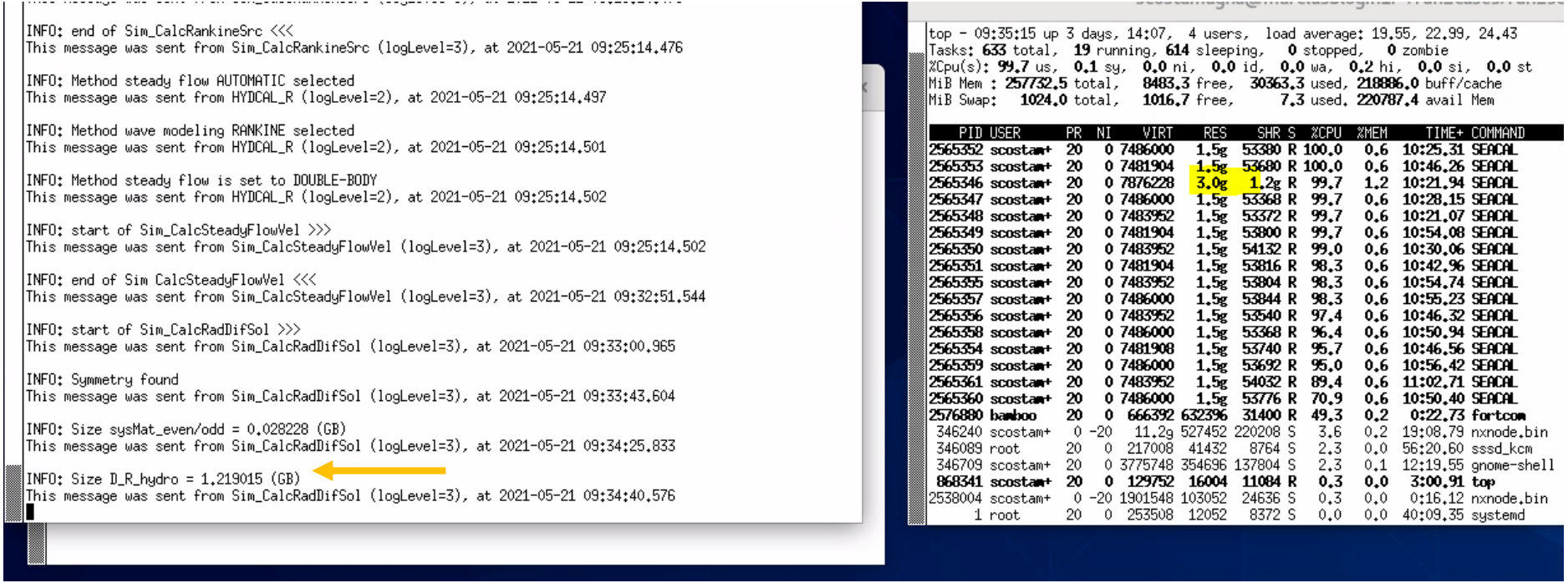


SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

MARIN

Linux

```
(IntelMPI) $>mpirun -n 16 SEACAL -c XXX
(OpenMPI)  $>mpirun --use-hwthread-cpus -n 16 SEACAL -c XXX
```



D_R_hydro Matrix => 1.2 GB MPI-3 shared memory

Run-Time Memory Usage Monitor

Windows

```
#Nr,PhysMTot(GB), PhysMAvail(GB), PhysMAvail(%)
#Sim_Init ONE INIT started
   1    34.131    21.779    64.000
#Sim_Init ONE INIT finished
   2    34.131    21.779    64.000
#SEACAL started
   3    34.131    21.779    64.000
…
#RESCAL finished
  10    34.131    20.803    61.000
#SEACAL finished
  11    34.131    20.898    62.000
```

https://forums.developer.nvidia.com/t/how-to-use-the-function-globalmemorystatus-in-pg/132018

Linux

```
#Nr|mpirank|MemTot(GB)|memUsage(GB)|memPeak(GB)|FreeMem(GB)|FreeMem(%)
#SEACAL started
   1     10     377.3       4.3       4.3     355.4      94.2
#HYDMES finished
   2     10     377.3       4.3       4.3     354.8      94.0
…
#RESCAL finished
 176     10     377.3       6.8       7.4     305.4      80.9
#SEACAL finished
 177     10     377.3       6.4       7.4     314.2      83.3
```

```
!--- get process ID

pid=getpid()
write(pid_char,'(I8)') pid
filename='/proc/'//trim(adjustl(pid_char))//'/status'
```

-Low memory available: Hard to predict problems when big chunk (contiguous) memory allocation required

-Built-in functions for retrieving memory consumption info missing

-Adding extra elements to an array should be more easy... (other than via move_alloc)

Temp files: file format

```
AddNode_hyd_rank<X>.h5                          ; info for reference points stored per point
AddNodeWL_hyd_rank<X>.h5                        ; info for water line points stored altogether
J-Class_FreeSurf_U_<>kn_mu_<>deg_om0_<>.hyd     ; info for a given condition for FS panels
J-Class_HydroMesh_U_<>kn_mu_<>deg_om0_<>.hyd    ; info for a given condition for Hydro Mesh
J-Class_Sigma_U_<>kn_mu_<>deg_om0_<>.hyd        ; info of Sigma matrix for a given condition
```

.hyd (fort binary) file per condition (U, mu, om)
(allowing “re-start” with adding extra cond’s)

```
J-Class_FreeSurf_U_0.0kn_mu_225deg_om0_0.300.hyd
J-Class_FreeSurf_U_0.0kn_mu_225deg_om0_0.500.hyd
J-Class_FreeSurf_U_0.0kn_mu_225deg_om0_0.700.hyd
J-Class_FreeSurf_U_0.0kn_mu_225deg_om0_0.900.hyd
J-Class_FreeSurf_U_0.0kn_mu_225deg_om0_1.100.hyd
J-Class_FreeSurf_U_0.0kn_mu_45deg_om0_0.300.hyd
J-Class_FreeSurf_U_0.0kn_mu_45deg_om0_0.500.hyd
J-Class_FreeSurf_U_0.0kn_mu_45deg_om0_0.700.hyd
J-Class_FreeSurf_U_0.0kn_mu_45deg_om0_0.900.hyd
J-Class_FreeSurf_U_0.0kn_mu_45deg_om0_1.100.hyd
```

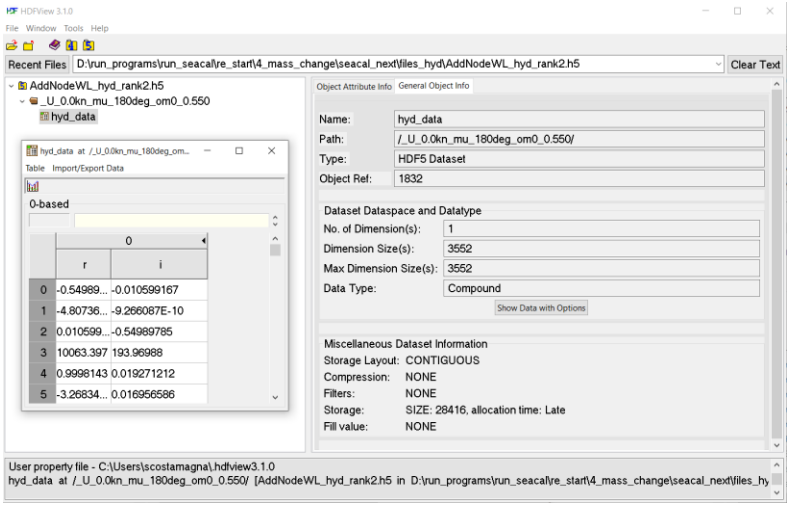
Too many files (>2000)
Too much disk space (>10Gb)

Switching to hdf5
(per MPI rank)

```
AddNodeWL_hyd_rank0.h5
AddNodeWL_hyd_rank10.h5
AddNodeWL_hyd_rank11.h5
AddNodeWL_hyd_rank12.h5
AddNodeWL_hyd_rank13.h5
AddNodeWL_hyd_rank14.h5
AddNodeWL_hyd_rank1.h5
AddNodeWL_hyd_rank2.h5
AddNodeWL_hyd_rank3.h5
AddNodeWL_hyd_rank4.h5
AddNodeWL_hyd_rank5.h5
AddNodeWL_hyd_rank6.h5
AddNodeWL_hyd_rank7.h5
AddNodeWL_hyd_rank8.h5
AddNodeWL_hyd_rank9.h5
```

SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

AddNodeWL_hyd_rank0.h5
AddNodeWL_hyd_rank10.h5
AddNodeWL_hyd_rank11.h5
AddNodeWL_hyd_rank12.h5
AddNodeWL_hyd_rank13.h5
AddNodeWL_hyd_rank14.h5
AddNodeWL_hyd_rank1.h5
AddNodeWL_hyd_rank2.h5
AddNodeWL_hyd_rank3.h5
AddNodeWL_hyd_rank4.h5
AddNodeWL_hyd_rank5.h5
AddNodeWL_hyd_rank6.h5
AddNodeWL_hyd_rank7.h5
AddNodeWL_hyd_rank8.h5
AddNodeWL_hyd_rank9.h5



-H5 (hdf5) files with *groups* per (U, mu, om) condition

PROBLEM to deal with… : opening/closing h5 files repeatedly
time consuming, fragmented file resulting in huge file size (x20)

```
-hdf5 lib compiled with FORTRAN2003 flag ON
-H5_INTERFACE Module file
```

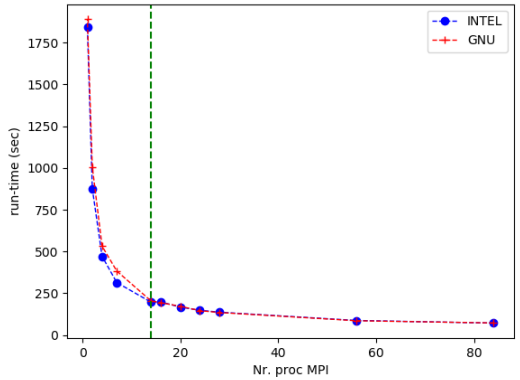
SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions

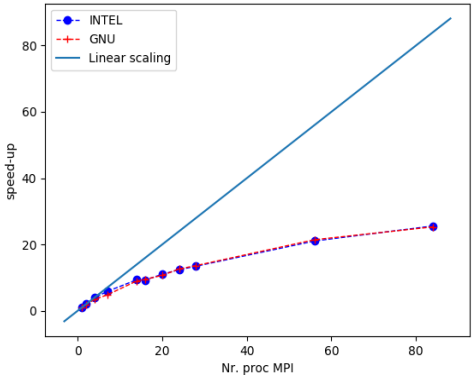
Parallel settings selection | SEACAL inputs

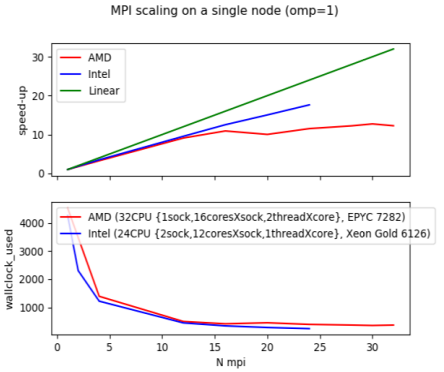
i) Nr. Total MPI
ii) MPI x Node
iii) openMP x Node

Based on
-Nr. input conditions (mu*om*U*Aphi)
-Nr. Panels (Ship + FreeSurf (Rankine))
-HARDWARE

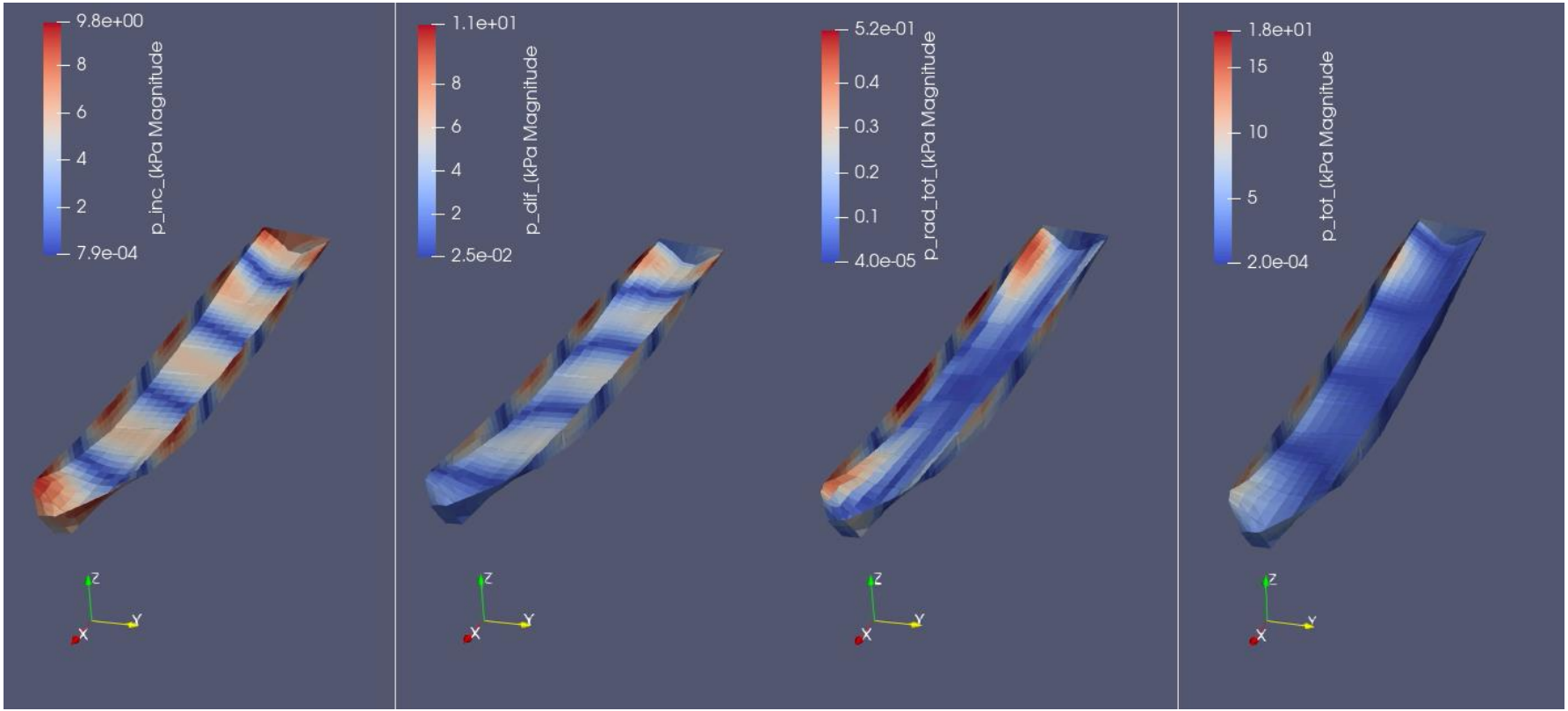
Total nr. of panels (Hydro+FreeSurf)	Estimated memory consumption (Gb)	queue = LARGE $N_{MPIperNod}$	queue = Normal, Short $N_{MPIperNod}$
5.000	0.545571918	24	24
10.000	1.1370817	24	24
15.000	1.878433825	24	24
20.000	2.7683268	24	24
25.000	3.806760625	24	24
30.000	4.9937353	24	18
35.000	6.329250825	24	14
40.000	7.8133072	24	12
50.000	11.2270425	24	8
60.000	15.2349412	24	6
80.000	25.0332288	15	3
100.000	37.20817	10	2
120.000	51.7597648	7	1
140.000	68.6880132	6	1
160.000	87.9929152	4	1
200.000	133.73268	2	-
250.000	204.2760625	1	-
300.000	289.67353	1	-
340.000	368.6864452	1	-

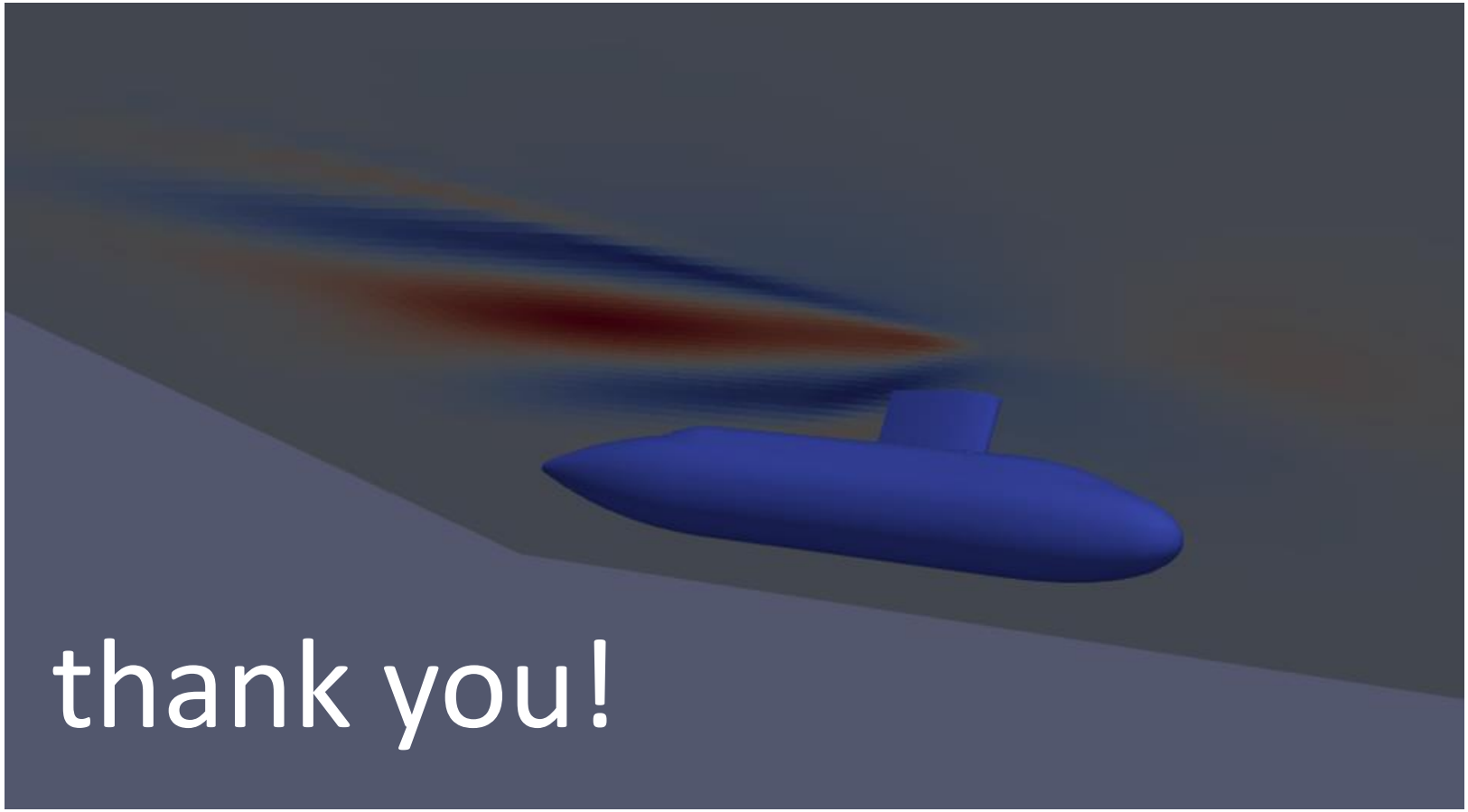






SEACAL: Cross Platform Parallel Hybrid Code for Calculations of Ship Motions




thank you!