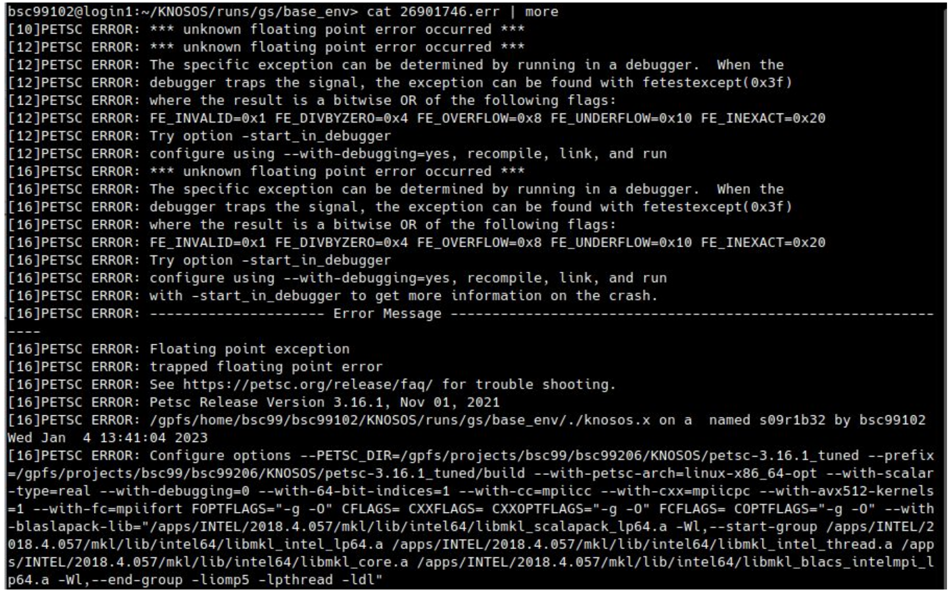
KNOSOS

KiNetic Orbit-averaging SOlver for Stellarators

Division by Zero error

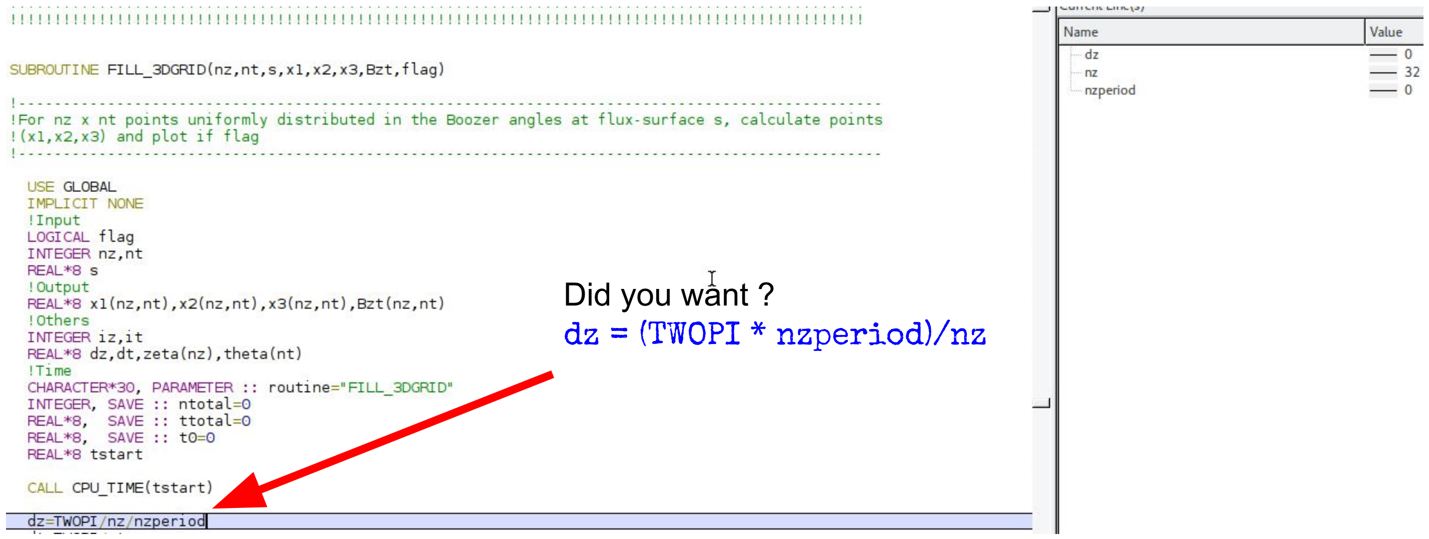
- Pass the `-fp_trap` flag to PETSc
- This flag is specific to PETSc and not `mpiifort` or the Debugger (DDT)
- `srun ./knosos.x -fp_trap` output or use `PetscSetFPTrap(PetscFPTrap flag)` inside the program where flag is `PETSC_FP_TRAP_ON`



```
bsc99102@login1:~/KNOSOS/runs/gs/base_env> cat 26901746.err | more
[10]PETSC ERROR: *** unknown floating point error occurred ***
[12]PETSC ERROR: *** unknown floating point error occurred ***
[12]PETSC ERROR: The specific exception can be determined by running in a debugger.  When the
[12]PETSC ERROR: debugger traps the signal, the exception can be found with fetestexcept(0x3f)
[12]PETSC ERROR: where the result is a bitwise OR of the following flags:
[12]PETSC ERROR: FE_INVALID=0x1 FE_DIVBYZERO=0x4 FE_OVERFLOW=0x8 FE_UNDERFLOW=0x10 FE_INEXACT=0x20
[12]PETSC ERROR: Try option -start_in_debugger
[12]PETSC ERROR: configure using --with-debugging=yes, recompile, link, and run
[16]PETSC ERROR: *** unknown floating point error occurred ***
[16]PETSC ERROR: The specific exception can be determined by running in a debugger.  When the
[16]PETSC ERROR: debugger traps the signal, the exception can be found with fetestexcept(0x3f)
[16]PETSC ERROR: where the result is a bitwise OR of the following flags:
[16]PETSC ERROR: FE_INVALID=0x1 FE_DIVBYZERO=0x4 FE_OVERFLOW=0x8 FE_UNDERFLOW=0x10 FE_INEXACT=0x20
[16]PETSC ERROR: Try option -start_in_debugger
[16]PETSC ERROR: configure using --with-debugging=yes, recompile, link, and run
[16]PETSC ERROR: with -start_in_debugger to get more information on the crash.
[16]PETSC ERROR: --------------------- Error Message ----------------------------------------------------------
----
[16]PETSC ERROR: Floating point exception
[16]PETSC ERROR: trapped floating point error
[16]PETSC ERROR: See https://petsc.org/release/faq/ for trouble shooting.
[16]PETSC ERROR: Petsc Release Version 3.16.1, Nov 01, 2021
[16]PETSC ERROR: /gpfs/home/bsc99/bsc99102/KNOSOS/runs/gs/base_env/./knosos.x on a  named s09r1b32 by bsc99102
Wed Jan  4 13:41:04 2023
[16]PETSC ERROR: Configure options --PETSC_DIR=/gpfs/projects/bsc99/bsc99206/KNOSOS/petsc-3.16.1_tuned --prefix
=/gpfs/projects/bsc99/bsc99206/KNOSOS/petsc-3.16.1_tuned/build --with-petsc-arch=linux-x86_64-opt --with-scalar
-type=real --with-debugging=0 --with-64-bit-indices=1 --with-cc=mpiicc --with-cxx=mpiicpc --with-avx512-kernels
=1 --with-fc=mpiifort FOPTFLAGS="-g -O" CFLAGS= CXXFLAGS= CXXOPTFLAGS="-g -O" FCFLAGS= COPTFLAGS="-g -O" --with
-blaslapack-lib="/apps/INTEL/2018.4.057/mkl/lib/intel64/libmkl_scalapack_lp64.a -Wl,--start-group /apps/INTEL/2
018.4.057/mkl/lib/intel64/libmkl_intel_lp64.a /apps/INTEL/2018.4.057/mkl/lib/intel64/libmkl_intel_thread.a /app
s/INTEL/2018.4.057/mkl/lib/intel64/libmkl_core.a /apps/INTEL/2018.4.057/mkl/lib/intel64/libmkl_blacs_intelmpi_l
p64.a -Wl,--end-group -liomp5 -lpthread -ldl"
```

WHERE does the error occur in the code ?

- Run DDT like: `ddt ./knosos.x -fp_trap`
- All 22 Processes stop in `fill_3dgrid (configuration.f90:1617)` with signal `SIGFPE` (Arithmetic exception) – floating point division by zero.



```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


SUBROUTINE FILL_3DGRID(nz,nt,s,x1,x2,x3,Bzt,flag)

!----------------------------------------------------------------------------------------------
!For nz x nt points uniformly distributed in the Boozer angles at flux-surface s, calculate points
!(x1,x2,x3) and plot if flag
!----------------------------------------------------------------------------------------------

  USE GLOBAL
  IMPLICIT NONE
  !Input
  LOGICAL flag
  INTEGER nz,nt
  REAL*8 s
  !Output
  REAL*8 x1(nz,nt),x2(nz,nt),x3(nz,nt),Bzt(nz,nt)
  !Others
  INTEGER iz,it
  REAL*8 dz,dt,zeta(nz),theta(nt)
  !Time
  CHARACTER*30, PARAMETER :: routine="FILL_3DGRID"
  INTEGER, SAVE :: ntotal=0
  REAL*8,  SAVE :: ttotal=0
  REAL*8,  SAVE :: t0=0
  REAL*8 tstart

  CALL CPU_TIME(tstart)

  dz=TWOPI/nz/nzperiod
```

Name	Value
dz	0
nz	32
nzperiod	0

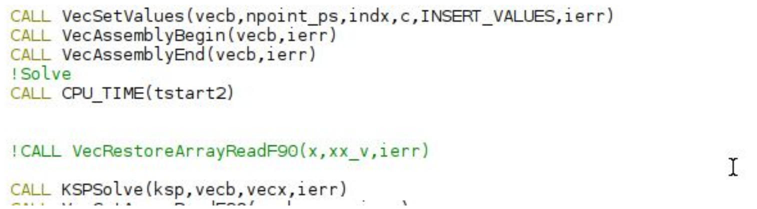
Did you want ?
`dz = (TWOPI * nzperiod)/nz`

The value of `dz`

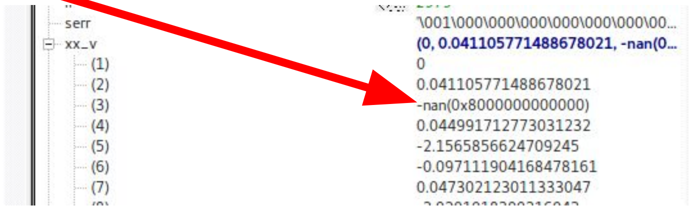
- `dz = Inf` on every process after `dz = TWOPI/nz/nzperiod` executes
- Correct Solution is: `dz = (TWOPI * nzperiod)/nz`
- If not corrected, error propagates for e.g., `zeta(iz)=(iz-1.)*dz` contains `NaN` now.

vecb

- Linear system of equations `Ax = b` …
- Code equivalent `[A][vecx] = [vecb]`
- But `vecb` contains a `NaN`



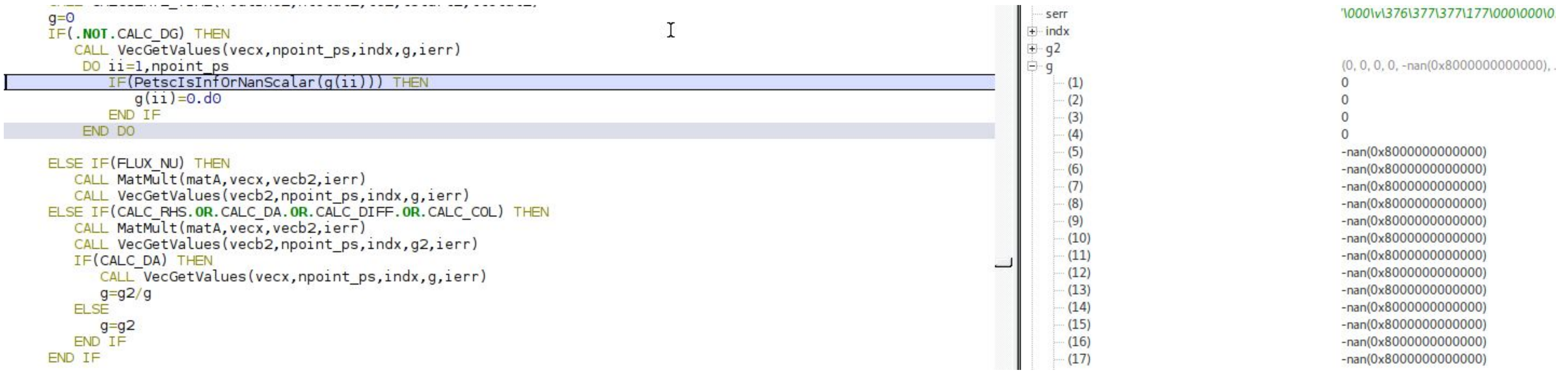
```
CALL VecSetValues(vecb,npoint_ps,indx,c,INSERT_VALUES,ierr)
CALL VecAssemblyBegin(vecb,ierr)
CALL VecAssemblyEnd(vecb,ierr)
!Solve
CALL CPU_TIME(tstart2)


!CALL VecRestoreArrayReadF90(x,xx_v,ierr)

CALL KSPSolve(ksp,vecb,vecx,ierr)
```



```
serr     \001\000\000\000\000\000\000\00...
xx_v     (0, 0.041105771488678021, -nan(0...
  (1)    0
  (2)    0.041105771488678021
  (3)    -nan(0x8000000000000)
  (4)    0.044991712773031232
  (5)    -2.1565856624709245
  (6)    -0.097111904168478161
  (7)    0.047302123011333047
```

- `vecb` has type `Vec` of PETSc
- Convert it to simple array using `VecGetArrayReadF90(vecb,xx_v,ierr)`
  Where `xx_v` is `PetscScalar, pointer :: xx_v(:)`

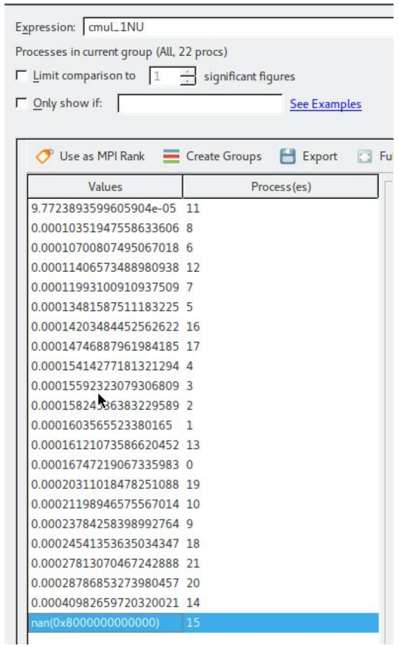
- `Inf` or `NaN` can be checked for a scalar value like: `PetscIsInfOrNanScalar(PetscScalar value)`
- Error propagates to `vecx` etc. when `KSPSolve(ksp,vecb,vecx,ierr)` is called.



```
g=0
IF(.NOT.CALC_DG) THEN
   CALL VecGetValues(vecx,npoint_ps,indx,g,ierr)
    DO ii=1,npoint_ps
       IF(PetscIsInfOrNanScalar(g(ii))) THEN
          g(ii)=0.d0
       END IF
    END DO

ELSE IF(FLUX_NU) THEN
   CALL MatMult(matA,vecx,vecb2,ierr)
   CALL VecGetValues(vecb2,npoint_ps,indx,g,ierr)
ELSE IF(CALC_RHS.OR.CALC_DA.OR.CALC_DIFF.OR.CALC_COL) THEN
   CALL MatMult(matA,vecx,vecb2,ierr)
   CALL VecGetValues(vecb2,npoint_ps,indx,g2,ierr)
   IF(CALC_DA) THEN
      CALL VecGetValues(vecx,npoint_ps,indx,g,ierr)
      g=g2/g
   ELSE
      g=g2
   END IF
END IF
```

```
serr       '\000\v\376\377\377\177\000\000\0
+ indx
+ g2
- g        (0, 0, 0, 0, -nan(0x8000000000000), .
   (1)     0
   (2)     0
   (3)     0
   (4)     0
   (5)     -nan(0x8000000000000)
   (6)     -nan(0x8000000000000)
   (7)     -nan(0x8000000000000)
   (8)     -nan(0x8000000000000)
   (9)     -nan(0x8000000000000)
   (10)    -nan(0x8000000000000)
   (11)    -nan(0x8000000000000)
   (12)    -nan(0x8000000000000)
   (13)    -nan(0x8000000000000)
   (14)    -nan(0x8000000000000)
   (15)    -nan(0x8000000000000)
   (16)    -nan(0x8000000000000)
   (17)    -nan(0x8000000000000)
```

cmul_1NU, D11 after collisionality, D11 after integrate_g



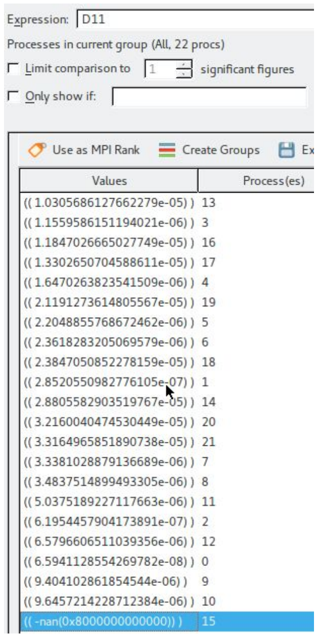
Expression: cmul_1NU

Processes in current group (All, 22 procs)

☐ Limit comparison to 1 significant figures

☐ Only show if: See Examples

Values	Process(es)
9.7723893599605904e-05	11
0.00010351947558633606	8
0.00010700807495067018	6
0.00011406573488980938	12
0.00011993100910937509	7
0.00013481587511183225	5
0.00014203484452562622	16
0.00014746887961984185	17
0.00015414277181321294	4
0.00015592323079306809	3
0.00015824536383229589	2
0.0001603565523380165	1
0.00016121073586620452	13
0.00016747219067335983	0
0.00020311018478251088	19
0.00021198946575567014	10
0.00023784258398992764	9
0.00024541353635034347	18
0.00027813070467242888	21
0.00028786853273980457	20
0.00040982659720320021	14
nan(0x8000000000000)	15



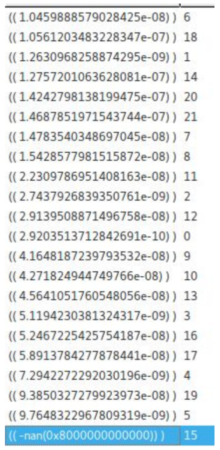
Expression: D11

Processes in current group (All, 22 procs)

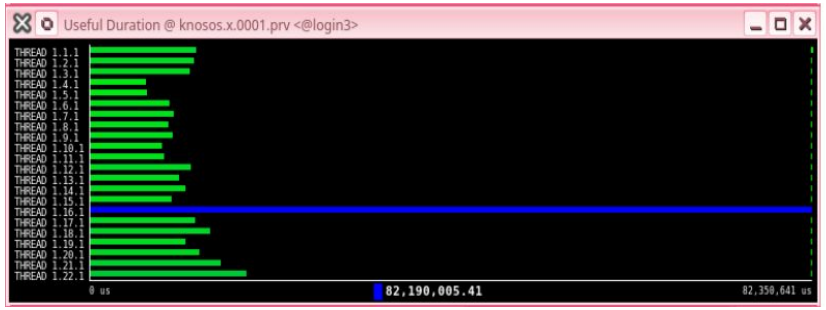
☐ Limit comparison to 1 significant figures

☐ Only show if:

Values	Process(es)
((1.0305686127662279e-05))	13
((1.1559586151194021e-06))	3
((1.1847026665027749e-05))	16
((1.3302650704588611e-05))	17
((1.6470263823541509e-06))	4
((2.1191273614805567e-05))	19
((2.2048855768672462e-06))	5
((2.3618283205069579e-06))	6
((2.3847050852278159e-05))	18
((2.8520550982776105e-07))	1
((2.8805582903519767e-05))	14
((3.2160040474530449e-05))	20
((3.3164965851890738e-05))	21
((3.3381028879136689e-06))	7
((3.4837514899493305e-06))	8
((5.0375189227117663e-06))	11
((6.1954457904173891e-07))	2
((6.5796606511039356e-06))	12
((6.5941128554269782e-08))	0
((9.404102861854544e-06))	9
((9.6457214228712384e-06))	10
((-nan(0x8000000000000)))	15

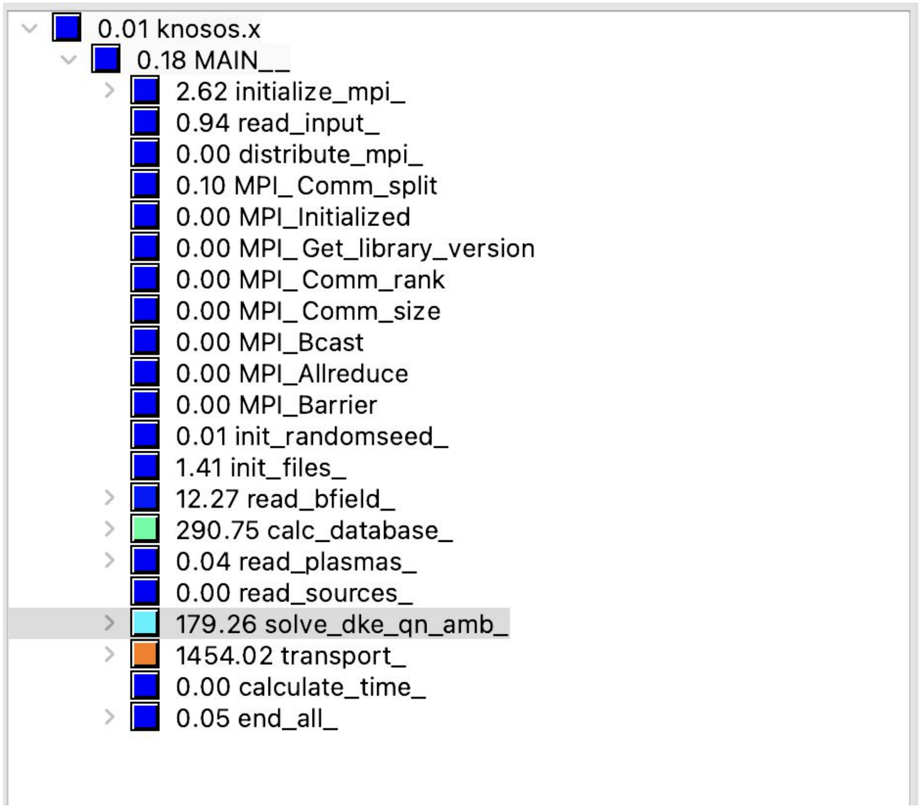


((1.0459888579028425e-08))	6
((1.0561203483228347e-07))	18
((1.2630968258874295e-09))	1
((1.2757201063628081e-07))	14
((1.4242798138199475e-07))	20
((1.4687851971543744e-07))	21
((1.4783540348697045e-08))	7
((1.5428577981515872e-08))	8
((2.2309786951408163e-08))	11
((2.7437926839350761e-09))	2
((2.9139508871496758e-08))	12
((2.9203513712842691e-10))	0
((4.1648187239793532e-08))	9
((4.271824944749766e-08))	10
((4.5641051760548056e-08))	13
((5.1194230381324317e-09))	3
((5.2467225425754187e-08))	16
((5.8913784277878441e-08))	17
((7.2942272292030196e-09))	4
((9.3850327279923973e-08))	19
((9.7648322967809319e-09))	5
((-nan(0x8000000000000)))	15

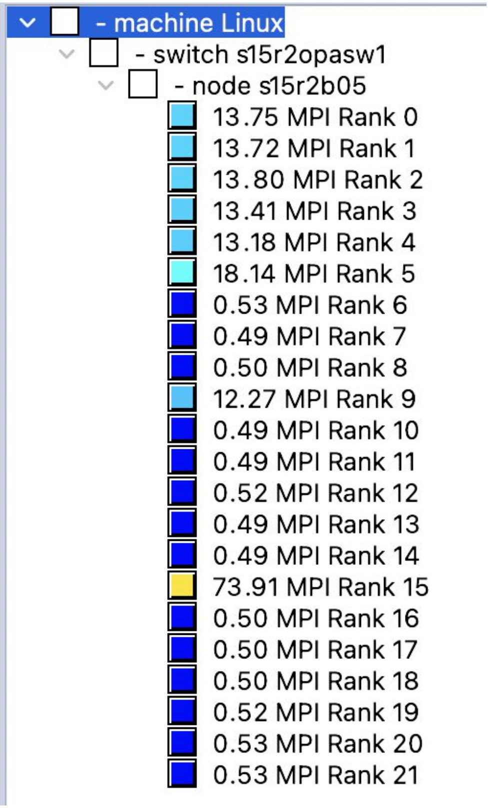


(Courtesy Helena Vela Beltran, Ricard Zarco Badia)

Before replacing `NaN` at `vecb(3)` with random value

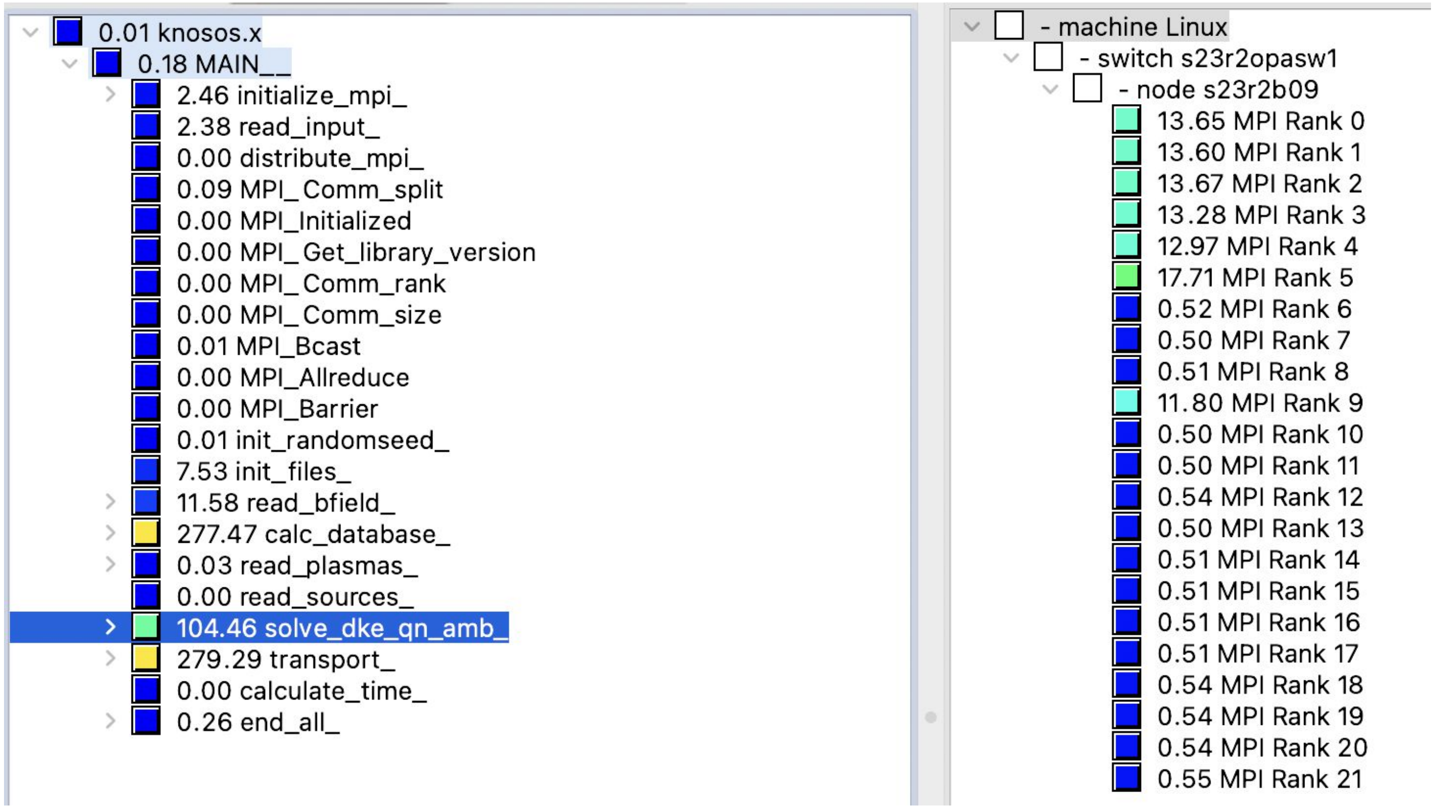


```
0.01 knosos.x
  0.18 MAIN__
    2.62 initialize_mpi_
    0.94 read_input_
    0.00 distribute_mpi_
    0.10 MPI_Comm_split
    0.00 MPI_Initialized
    0.00 MPI_Get_library_version
    0.00 MPI_Comm_rank
    0.00 MPI_Comm_size
    0.00 MPI_Bcast
    0.00 MPI_Allreduce
    0.00 MPI_Barrier
    0.01 init_randomseed_
    1.41 init_files_
    12.27 read_bfield_
    290.75 calc_database_
    0.04 read_plasmas_
    0.00 read_sources_
    179.26 solve_dke_qn_amb_
    1454.02 transport_
    0.00 calculate_time_
    0.05 end_all_
```



```
- machine Linux
  - switch s15r2opasw1
    - node s15r2b05
      13.75 MPI Rank 0
      13.72 MPI Rank 1
      13.80 MPI Rank 2
      13.41 MPI Rank 3
      13.18 MPI Rank 4
      18.14 MPI Rank 5
      0.53 MPI Rank 6
      0.49 MPI Rank 7
      0.50 MPI Rank 8
      12.27 MPI Rank 9
      0.49 MPI Rank 10
      0.49 MPI Rank 11
      0.52 MPI Rank 12
      0.49 MPI Rank 13
      0.49 MPI Rank 14
      73.91 MPI Rank 15
      0.50 MPI Rank 16
      0.50 MPI Rank 17
      0.50 MPI Rank 18
      0.52 MPI Rank 19
      0.53 MPI Rank 20
      0.53 MPI Rank 21
```

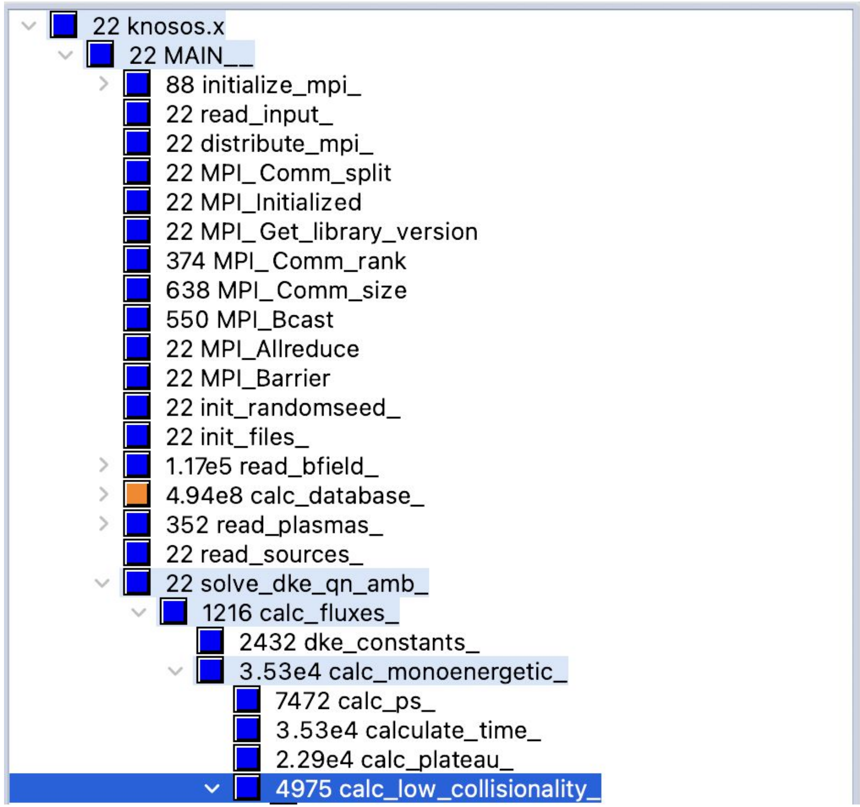
After replacing `NaN` at `vecb(3)` with random value



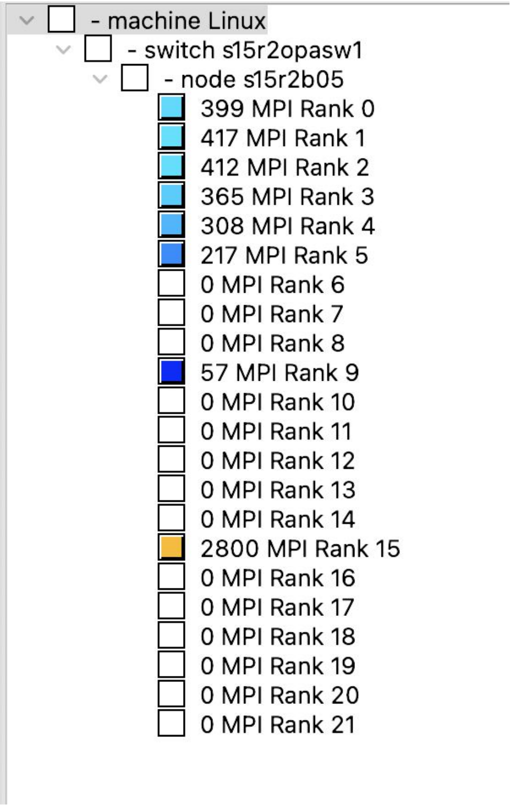
```
0.01 knosos.x
  0.18 MAIN__
    2.46 initialize_mpi_
    2.38 read_input_
    0.00 distribute_mpi_
    0.09 MPI_Comm_split
    0.00 MPI_Initialized
    0.00 MPI_Get_library_version
    0.00 MPI_Comm_rank
    0.00 MPI_Comm_size
    0.01 MPI_Bcast
    0.00 MPI_Allreduce
    0.00 MPI_Barrier
    0.01 init_randomseed_
    7.53 init_files_
    11.58 read_bfield_
    277.47 calc_database_
    0.03 read_plasmas_
    0.00 read_sources_
    104.46 solve_dke_qn_amb_
    279.29 transport_
    0.00 calculate_time_
    0.26 end_all_
```

```
- machine Linux
  - switch s23r2opasw1
    - node s23r2b09
      13.65 MPI Rank 0
      13.60 MPI Rank 1
      13.67 MPI Rank 2
      13.28 MPI Rank 3
      12.97 MPI Rank 4
      17.71 MPI Rank 5
      0.52 MPI Rank 6
      0.50 MPI Rank 7
      0.51 MPI Rank 8
      11.80 MPI Rank 9
      0.50 MPI Rank 10
      0.50 MPI Rank 11
      0.54 MPI Rank 12
      0.50 MPI Rank 13
      0.51 MPI Rank 14
      0.51 MPI Rank 15
      0.51 MPI Rank 16
      0.51 MPI Rank 17
      0.54 MPI Rank 18
      0.54 MPI Rank 19
      0.54 MPI Rank 20
      0.55 MPI Rank 21
```

22 Processes Original calls

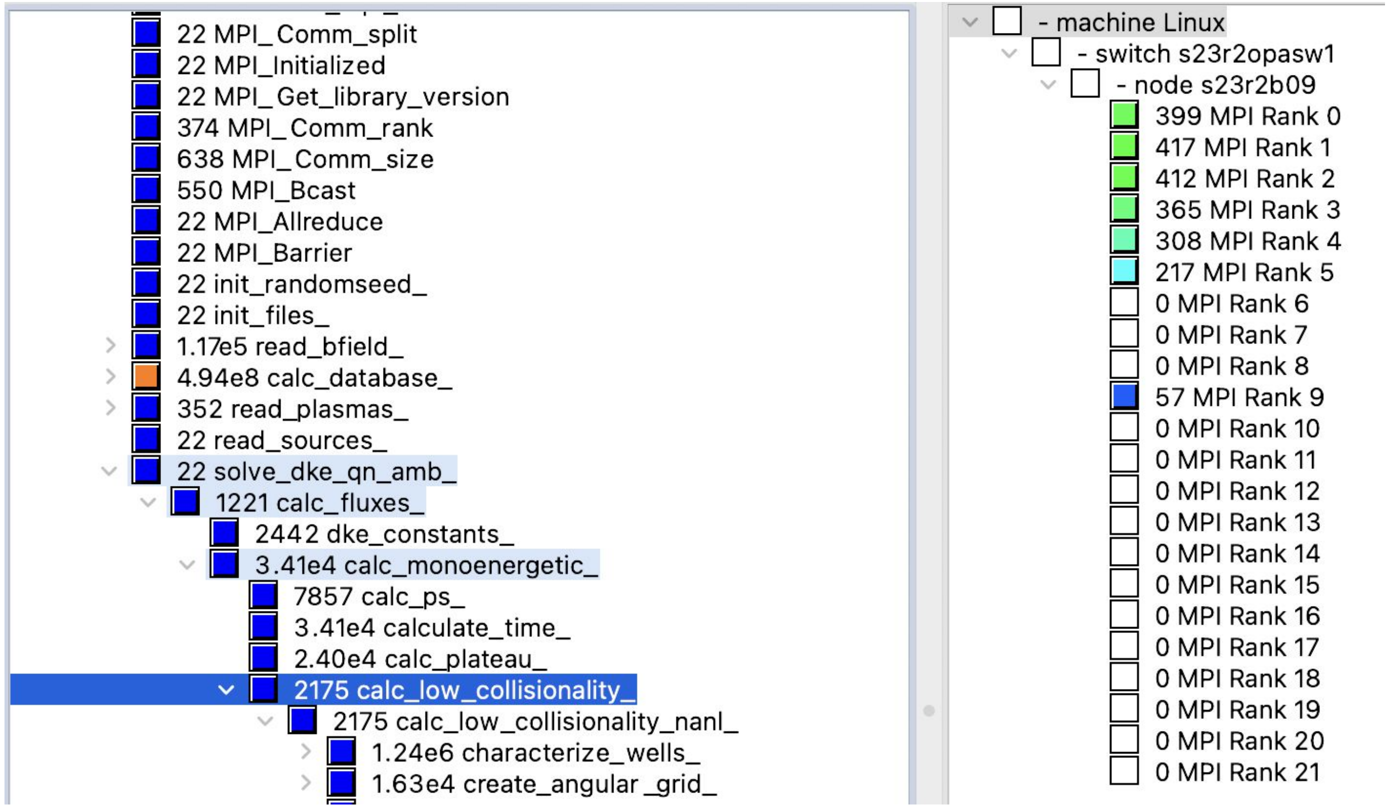


```
22 knosos.x
  22 MAIN__
    88 initialize_mpi_
    22 read_input_
    22 distribute_mpi_
    22 MPI_Comm_split
    22 MPI_Initialized
    22 MPI_Get_library_version
    374 MPI_Comm_rank
    638 MPI_Comm_size
    550 MPI_Bcast
    22 MPI_Allreduce
    22 MPI_Barrier
    22 init_randomseed_
    22 init_files_
    1.17e5 read_bfield_
    4.94e8 calc_database_
    352 read_plasmas_
    22 read_sources_
    22 solve_dke_qn_amb_
      1216 calc_fluxes_
        2432 dke_constants_
        3.53e4 calc_monoenergetic_
          7472 calc_ps_
          3.53e4 calculate_time_
          2.29e4 calc_plateau_
          4975 calc_low_collisionality_
```



```
- machine Linux
  - switch s15r2opasw1
    - node s15r2b05
      399 MPI Rank 0
      417 MPI Rank 1
      412 MPI Rank 2
      365 MPI Rank 3
      308 MPI Rank 4
      217 MPI Rank 5
      0 MPI Rank 6
      0 MPI Rank 7
      0 MPI Rank 8
      57 MPI Rank 9
      0 MPI Rank 10
      0 MPI Rank 11
      0 MPI Rank 12
      0 MPI Rank 13
      0 MPI Rank 14
      2800 MPI Rank 15
      0 MPI Rank 16
      0 MPI Rank 17
      0 MPI Rank 18
      0 MPI Rank 19
      0 MPI Rank 20
      0 MPI Rank 21
```

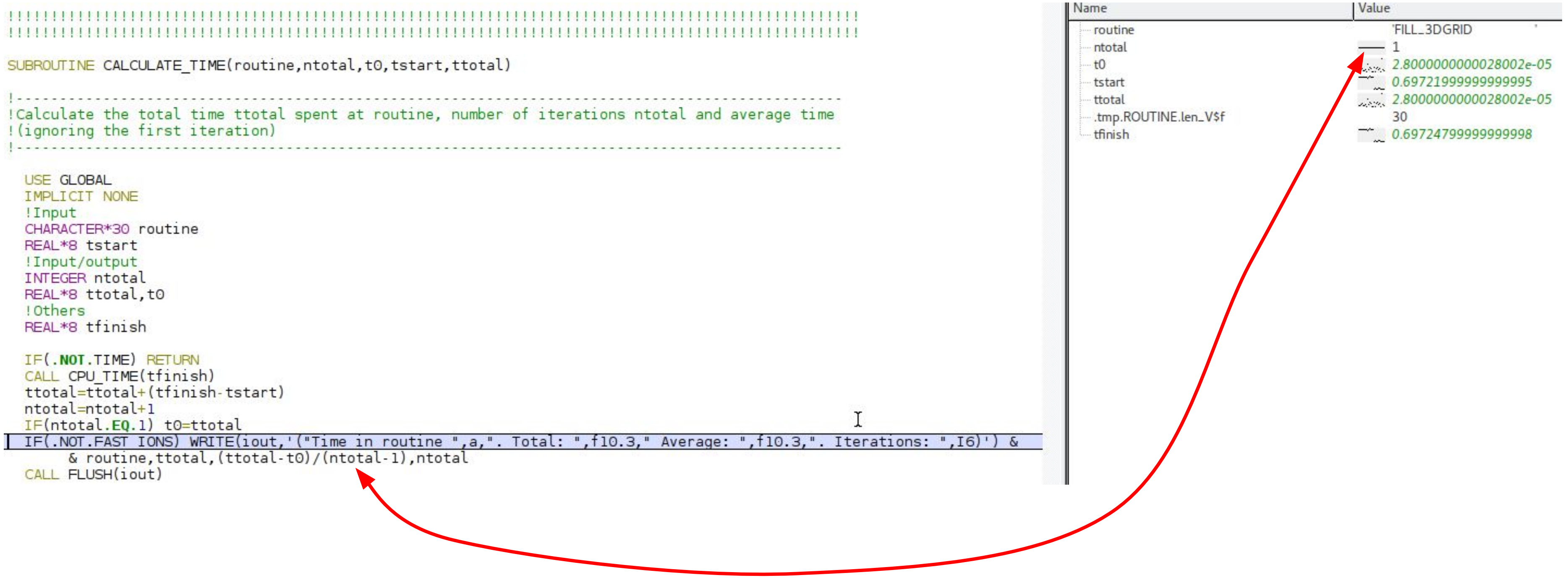
22 Processes corrected calls



```
22 MPI_Comm_split
22 MPI_Initialized
22 MPI_Get_library_version
374 MPI_Comm_rank
638 MPI_Comm_size
550 MPI_Bcast
22 MPI_Allreduce
22 MPI_Barrier
22 init_randomseed_
22 init_files_
1.17e5 read_bfield_
4.94e8 calc_database_
352 read_plasmas_
22 read_sources_
22 solve_dke_qn_amb_
  1221 calc_fluxes_
    2442 dke_constants_
    3.41e4 calc_monoenergetic_
      7857 calc_ps_
      3.41e4 calculate_time_
      2.40e4 calc_plateau_
      2175 calc_low_collisionality_
        2175 calc_low_collisionality_nanl_
          1.24e6 characterize_wells_
          1.63e4 create_angular_grid_
```

```
- machine Linux
  - switch s23r2opasw1
    - node s23r2b09
      399 MPI Rank 0
      417 MPI Rank 1
      412 MPI Rank 2
      365 MPI Rank 3
      308 MPI Rank 4
      217 MPI Rank 5
      0 MPI Rank 6
      0 MPI Rank 7
      0 MPI Rank 8
      57 MPI Rank 9
      0 MPI Rank 10
      0 MPI Rank 11
      0 MPI Rank 12
      0 MPI Rank 13
      0 MPI Rank 14
      0 MPI Rank 15
      0 MPI Rank 16
      0 MPI Rank 17
      0 MPI Rank 18
      0 MPI Rank 19
      0 MPI Rank 20
      0 MPI Rank 21
```

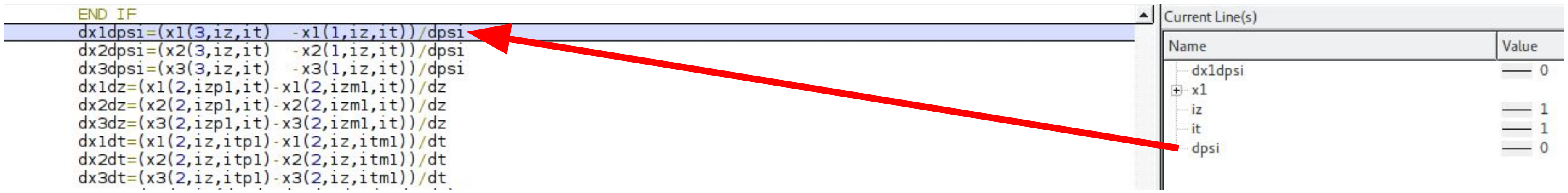
Calculate Time Subroutine - Arithmetic Exception



```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

SUBROUTINE CALCULATE_TIME(routine,ntotal,t0,tstart,ttotal)

!-----------------------------------------------------------------------------------------------
!Calculate the total time ttotal spent at routine, number of iterations ntotal and average time
!(ignoring the first iteration)
!-----------------------------------------------------------------------------------------------

  USE GLOBAL
  IMPLICIT NONE
  !Input
  CHARACTER*30 routine
  REAL*8 tstart
  !Input/output
  INTEGER ntotal
  REAL*8 ttotal,t0
  !Others
  REAL*8 tfinish

  IF(.NOT.TIME) RETURN
  CALL CPU_TIME(tfinish)
  ttotal=ttotal+(tfinish-tstart)
  ntotal=ntotal+1
  IF(ntotal.EQ.1) t0=ttotal
  IF(.NOT.FAST_IONS) WRITE(iout,'("Time in routine ",a,". Total: ",f10.3," Average: ",f10.3,". Iterations: ",I6)') &
       & routine,ttotal,(ttotal-t0)/(ntotal-1),ntotal
  CALL FLUSH(iout)
```

Name	Value
routine	'FILL_3DGRID
ntotal	1
t0	2.8000000000028002e-05
tstart	0.69721999999999995
ttotal	2.8000000000028002e-05
.tmp.ROUTINE.len_V$f	30
tfinish	0.69724799999999998

CHECK_JACSIGN in configuration.f90

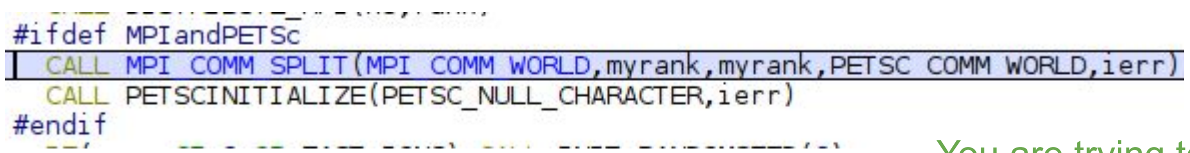


```
    END IF
    dx1dpsi=(x1(3,iz,it)   -x1(1,iz,it))/dpsi
    dx2dpsi=(x2(3,iz,it)   -x2(1,iz,it))/dpsi
    dx3dpsi=(x3(3,iz,it)   -x3(1,iz,it))/dpsi
    dx1dz=(x1(2,izp1,it)-x1(2,izm1,it))/dz
    dx2dz=(x2(2,izp1,it)-x2(2,izm1,it))/dz
    dx3dz=(x3(2,izp1,it)-x3(2,izm1,it))/dz
    dx1dt=(x1(2,iz,itp1)-x1(2,iz,itm1))/dt
    dx2dt=(x2(2,iz,itp1)-x2(2,iz,itm1))/dt
    dx3dt=(x3(2,iz,itp1)-x3(2,iz,itm1))/dt
```

Current Line(s)

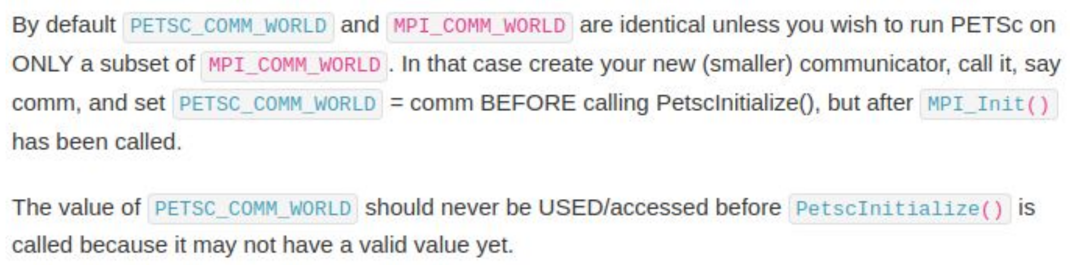
Name	Value
dx1dpsi	0
x1	
iz	1
it	1
dpsi	0

MPI_COMM_SPLIT for PETSC_COMM_WORLD (knosos.f90)



```
#ifdef MPIandPETSc
  CALL MPI_COMM_SPLIT(MPI_COMM_WORLD,myrank,myrank,PETSC_COMM_WORLD,ierr)
  CALL PETSCINITIALIZE(PETSC_NULL_CHARACTER,ierr)
#endif
```

You are trying to put every process in its own communicator = MPI_COMM_SELF



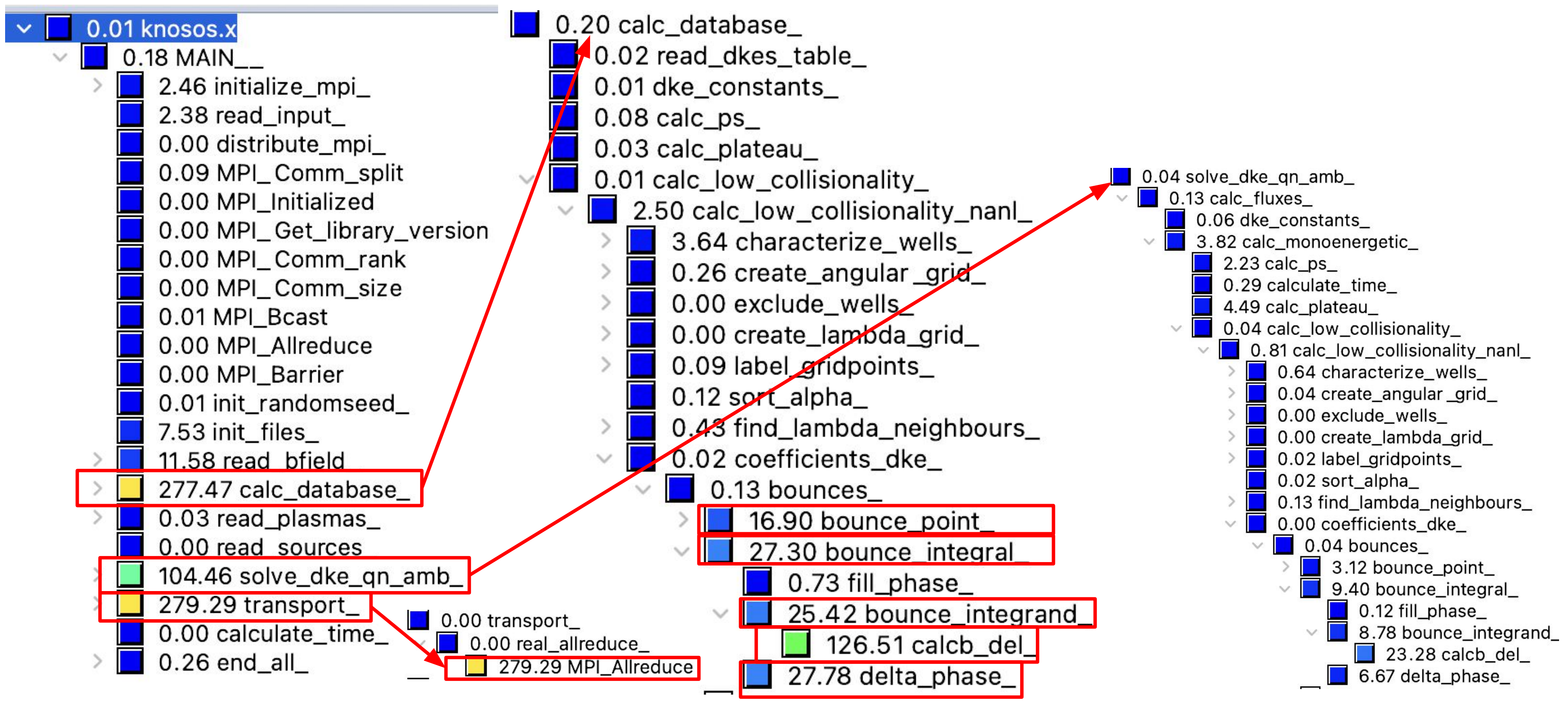
By default `PETSC_COMM_WORLD` and `MPI_COMM_WORLD` are identical unless you wish to run PETSc on ONLY a subset of `MPI_COMM_WORLD`. In that case create your new (smaller) communicator, call it, say comm, and set `PETSC_COMM_WORLD` = comm BEFORE calling PetscInitialize(), but after `MPI_Init()` has been called.

The value of `PETSC_COMM_WORLD` should never be USED/accessed before `PetscInitialize()` is called because it may not have a valid value yet.

Suggestion

- Enable `fp_trap` flag in debug mode (so PETSc can catch floating point exceptions) or enable it from within the program.
- Add a routine that checks for `NaN` and `Inf`, especially the Vectors in PETSc.

Most time consuming subroutines



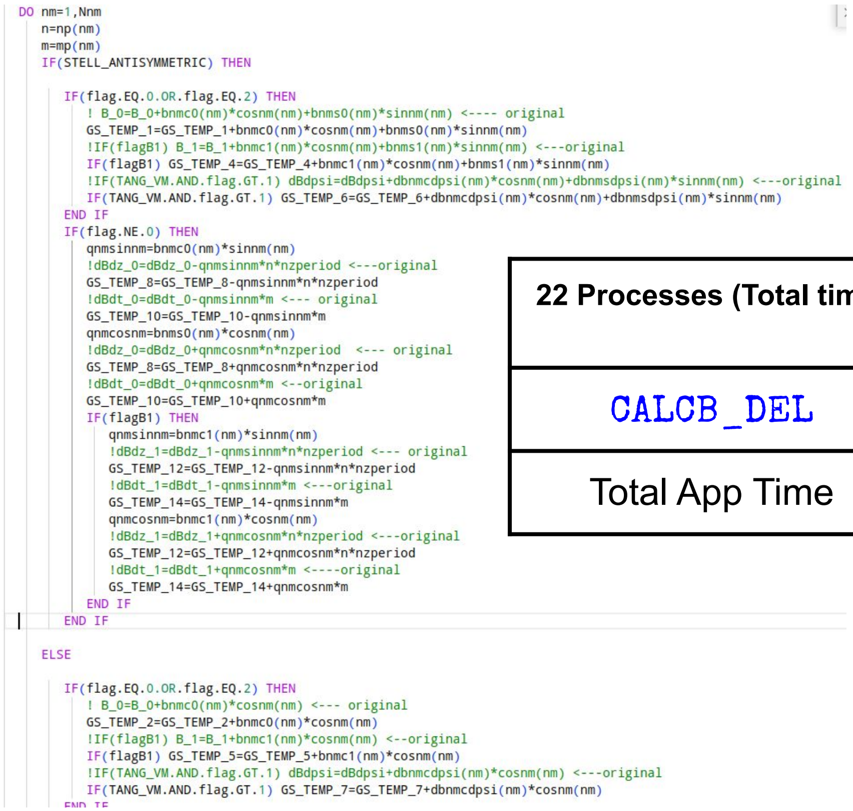
```
0.01 knosos.x
  0.18 MAIN__
    2.46 initialize_mpi_
    2.38 read_input_
    0.00 distribute_mpi_
    0.09 MPI_Comm_split
    0.00 MPI_Initialized
    0.00 MPI_Get_library_version
    0.00 MPI_Comm_rank
    0.00 MPI_Comm_size
    0.01 MPI_Bcast
    0.00 MPI_Allreduce
    0.00 MPI_Barrier
    0.01 init_randomseed_
    7.53 init_files_
    11.58 read_bfield
    277.47 calc_database_
    0.03 read_plasmas_
    0.00 read_sources
    104.46 solve_dke_qn_amb_
    279.29 transport_
    0.00 calculate_time_
    0.26 end_all_
```

```
0.00 transport_
  0.00 real_allreduce_
    279.29 MPI_Allreduce
```

```
0.20 calc_database_
  0.02 read_dkes_table_
  0.01 dke_constants_
  0.08 calc_ps_
  0.03 calc_plateau_
  0.01 calc_low_collisionality_
    2.50 calc_low_collisionality_nanl_
      3.64 characterize_wells_
      0.26 create_angular_grid_
      0.00 exclude_wells_
      0.00 create_lambda_grid_
      0.09 label_gridpoints_
      0.12 sort_alpha_
      0.43 find_lambda_neighbours_
      0.02 coefficients_dke_
        0.13 bounces_
          16.90 bounce_point_
          27.30 bounce_integral_
            0.73 fill_phase_
            25.42 bounce_integrand_
              126.51 calcb_del_
            27.78 delta_phase_
```

```
0.04 solve_dke_qn_amb_
  0.13 calc_fluxes_
    0.06 dke_constants_
    3.82 calc_monoenergetic_
      2.23 calc_ps_
      0.29 calculate_time_
      4.49 calc_plateau_
      0.04 calc_low_collisionality_
        0.81 calc_low_collisionality_nanl_
          0.64 characterize_wells_
          0.04 create_angular_grid_
          0.00 exclude_wells_
          0.00 create_lambda_grid_
          0.02 label_gridpoints_
          0.02 sort_alpha_
          0.13 find_lambda_neighbours_
          0.00 coefficients_dke_
            0.04 bounces_
              3.12 bounce_point_
              9.40 bounce_integral_
                0.12 fill_phase_
                8.78 bounce_integrand_
                  23.28 calcb_del_
                6.67 delta_phase_
```

Attempt Optimization - 1(a), `CALCB_DEL` in `coefficients.f90`



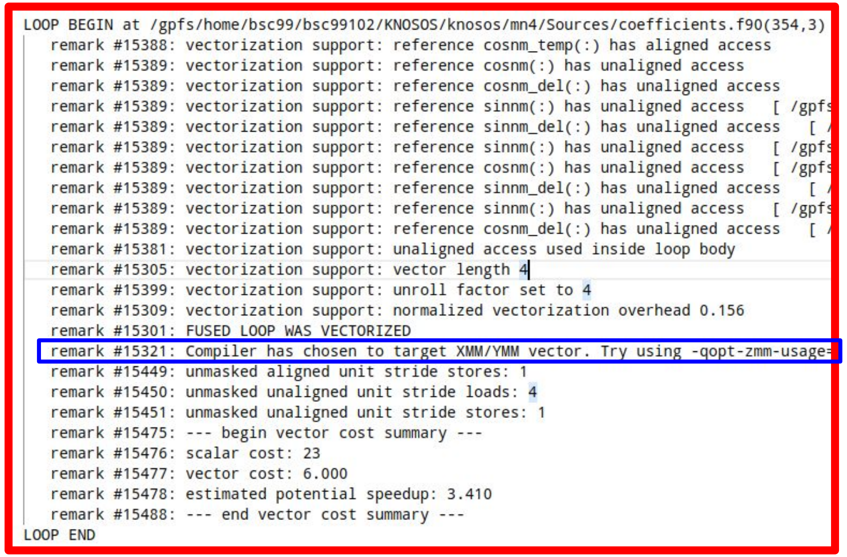
```fortran
DO nm=1,Nnm
  n=np(nm)
  m=mp(nm)
  IF(STELL_ANTISYMMETRIC) THEN

    IF(flag.EQ.0.OR.flag.EQ.2) THEN
      ! B_0=B_0+bnmc0(nm)*cosnm(nm)+bnms0(nm)*sinnm(nm) <---- original
      GS_TEMP_1=GS_TEMP_1+bnmc0(nm)*cosnm(nm)+bnms0(nm)*sinnm(nm)
      !IF(flagB1) B_1=B_1+bnmc1(nm)*cosnm(nm)+bnms1(nm)*sinnm(nm) <---original
      IF(flagB1) GS_TEMP_4=GS_TEMP_4+bnmc1(nm)*cosnm(nm)+bnms1(nm)*sinnm(nm)
      !IF(TANG_VM.AND.flag.GT.1) dBdpsi=dBdpsi+dbnmcdpsi(nm)*cosnm(nm)+dbnmsdpsi(nm)*sinnm(nm) <---original
      IF(TANG_VM.AND.flag.GT.1) GS_TEMP_6=GS_TEMP_6+dbnmcdpsi(nm)*cosnm(nm)+dbnmsdpsi(nm)*sinnm(nm)
    END IF
    IF(flag.NE.0) THEN
      qnmsinnm=bnmc0(nm)*sinnm(nm)
      !dBdz_0=dBdz_0-qnmsinnm*n*nzperiod <---original
      GS_TEMP_8=GS_TEMP_8-qnmsinnm*n*nzperiod
      !dBdt_0=dBdt_0-qnmsinnm*m <--- original
      GS_TEMP_10=GS_TEMP_10-qnmsinnm*m
      qnmcosnm=bnms0(nm)*cosnm(nm)
      !dBdz_0=dBdz_0+qnmcosnm*n*nzperiod  <--- original
      GS_TEMP_8=GS_TEMP_8+qnmcosnm*n*nzperiod
      !dBdt_0=dBdt_0+qnmcosnm*m <--original
      GS_TEMP_10=GS_TEMP_10+qnmcosnm*m
      IF(flagB1) THEN
        qnmsinnm=bnmc1(nm)*sinnm(nm)
        !dBdz_1=dBdz_1-qnmsinnm*n*nzperiod <--- original
        GS_TEMP_12=GS_TEMP_12-qnmsinnm*n*nzperiod
        !dBdt_1=dBdt_1-qnmsinnm*m <---original
        GS_TEMP_14=GS_TEMP_14-qnmsinnm*m
        qnmcosnm=bnmc1(nm)*cosnm(nm)
        !dBdz_1=dBdz_1+qnmcosnm*n*nzperiod <---original
        GS_TEMP_12=GS_TEMP_12+qnmcosnm*n*nzperiod
        !dBdt_1=dBdt_1+qnmcosnm*m <----original
        GS_TEMP_14=GS_TEMP_14+qnmcosnm*m
      END IF
    END IF

  ELSE

    IF(flag.EQ.0.OR.flag.EQ.2) THEN
      ! B_0=B_0+bnmc0(nm)*cosnm(nm) <--- original
      GS_TEMP_2=GS_TEMP_2+bnmc0(nm)*cosnm(nm)
      !IF(flagB1) B_1=B_1+bnmc1(nm)*cosnm(nm) <--original
      IF(flagB1) GS_TEMP_5=GS_TEMP_5+bnmc1(nm)*cosnm(nm)
      !IF(TANG_VM.AND.flag.GT.1) dBdpsi=dBdpsi+dbnmcdpsi(nm)*cosnm(nm) <---original
      IF(TANG_VM.AND.flag.GT.1) GS_TEMP_7=GS_TEMP_7+dbnmcdpsi(nm)*cosnm(nm)
    END IF
```

🚨 Time recorded for *`instance of CALCB_DEL`* taking maximum time

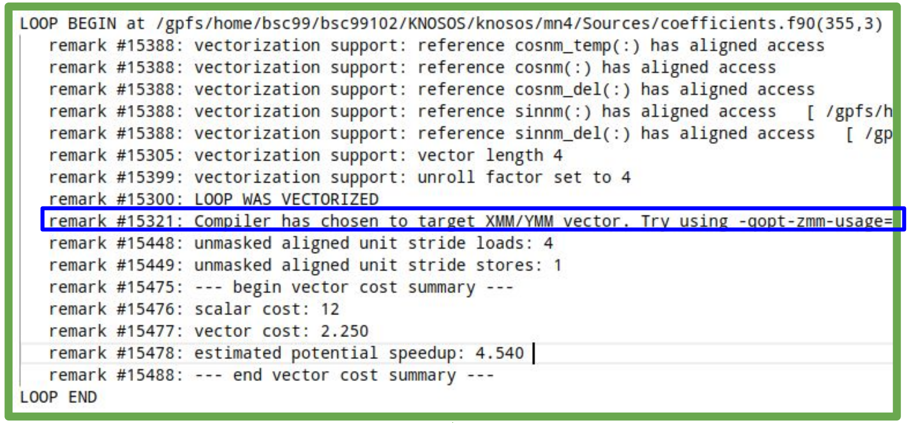
22 Processes (Total time)	Original (NaN)	No NaN (expected behaviour)	Optimized=Vectorized
`CALCB_DEL`	138.31 sec	126.51 sec	47.28 sec
Total App Time	1941.66 sec	685.77	551.22

Speed-up for `CALCB_DEL = 126.51/47.28 =` **2.67x**
Total Speed-up = **685.77/551.22 = 1.24x**
Notes:
(1) Need to check for **correctness**
~~(2) Need to check for **unaligned access.**~~ ✅

Attempt Optimization - 2, DELTA_PHASE in coefficients.f90



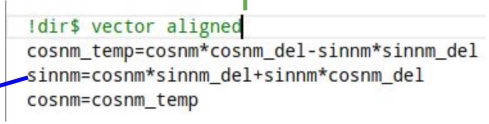
```
LOOP BEGIN at /gpfs/home/bsc99/bsc99102/KNOSOS/knosos/mn4/Sources/coefficients.f90(354,3)
   remark #15388: vectorization support: reference cosnm_temp(:) has aligned access
   remark #15389: vectorization support: reference cosnm(:) has unaligned access
   remark #15389: vectorization support: reference cosnm_del(:) has unaligned access
   remark #15389: vectorization support: reference sinnm(:) has unaligned access   [ /gpfs
   remark #15389: vectorization support: reference sinnm_del(:) has unaligned access   [ /
   remark #15389: vectorization support: reference sinnm(:) has unaligned access   [ /gpfs
   remark #15389: vectorization support: reference cosnm(:) has unaligned access   [ /gpfs
   remark #15389: vectorization support: reference sinnm_del(:) has unaligned access   [ /
   remark #15389: vectorization support: reference sinnm(:) has unaligned access   [ /gpfs
   remark #15389: vectorization support: reference cosnm_del(:) has unaligned access   [ /
   remark #15381: vectorization support: unaligned access used inside loop body
   remark #15305: vectorization support: vector length 4
   remark #15399: vectorization support: unroll factor set to 4
   remark #15309: vectorization support: normalized vectorization overhead 0.156
   remark #15301: FUSED LOOP WAS VECTORIZED
   remark #15321: Compiler has chosen to target XMM/YMM vector. Try using -qopt-zmm-usage=
   remark #15449: unmasked aligned unit stride stores: 1
   remark #15450: unmasked unaligned unit stride loads: 4
   remark #15451: unmasked unaligned unit stride stores: 1
   remark #15475: --- begin vector cost summary ---
   remark #15476: scalar cost: 23
   remark #15477: vector cost: 6.000
   remark #15478: estimated potential speedup: 3.410
   remark #15488: --- end vector cost summary ---
LOOP END
```



```
LOOP BEGIN at /gpfs/home/bsc99/bsc99102/KNOSOS/knosos/mn4/Sources/coefficients.f90(355,3)
   remark #15388: vectorization support: reference cosnm_temp(:) has aligned access
   remark #15388: vectorization support: reference cosnm(:) has aligned access
   remark #15388: vectorization support: reference cosnm_del(:) has aligned access
   remark #15388: vectorization support: reference sinnm(:) has aligned access   [ /gpfs/h
   remark #15388: vectorization support: reference sinnm_del(:) has aligned access   [ /gp
   remark #15305: vectorization support: vector length 4
   remark #15399: vectorization support: unroll factor set to 4
   remark #15300: LOOP WAS VECTORIZED
   remark #15321: Compiler has chosen to target XMM/YMM vector. Try using -qopt-zmm-usage=
   remark #15448: unmasked aligned unit stride loads: 4
   remark #15449: unmasked aligned unit stride stores: 1
   remark #15475: --- begin vector cost summary ---
   remark #15476: scalar cost: 12
   remark #15477: vector cost: 2.250
   remark #15478: estimated potential speedup: 4.540
   remark #15488: --- end vector cost summary ---
LOOP END
```

22 Processes	No NaN	Aligned + zmm
DELTA_PHASE	35.99 sec	29.03 sec

Speed-up = 1.23x



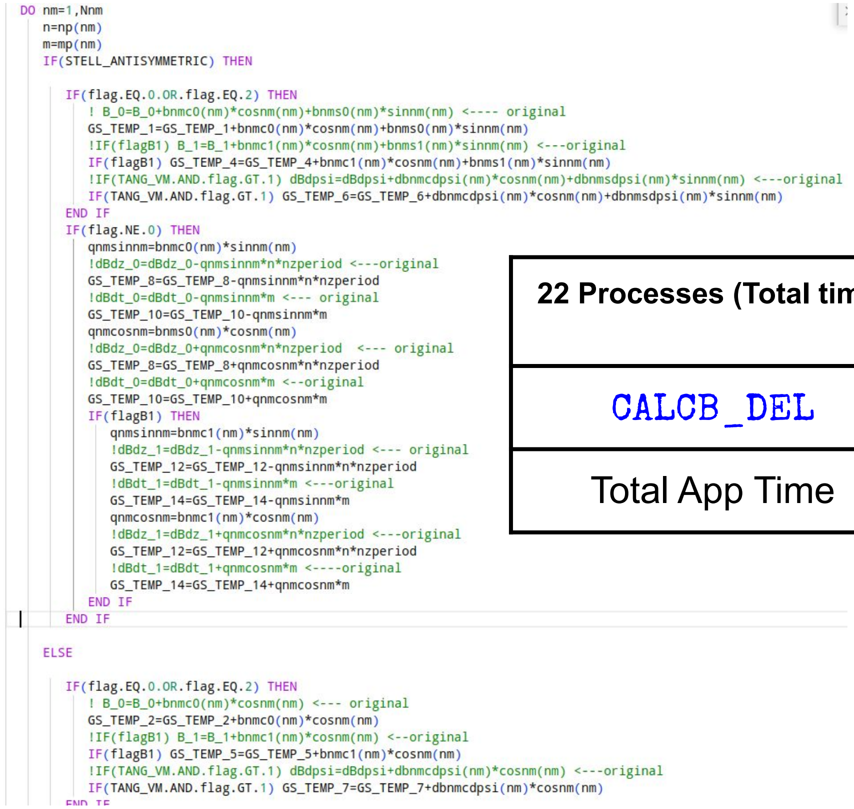
```
!dir$ vector aligned
cosnm_temp=cosnm*cosnm_del-sinnm*sinnm_del
sinnm=cosnm*sinnm_del+sinnm*cosnm_del
cosnm=cosnm_temp
```

This is another loop ! Needs vector aligned separately !

Add at compile time

`-align array64byte -qopt-zmm-usage=high`

Attempt Optimization - 1(b), `CALCB_DEL` in `coefficients.f90`

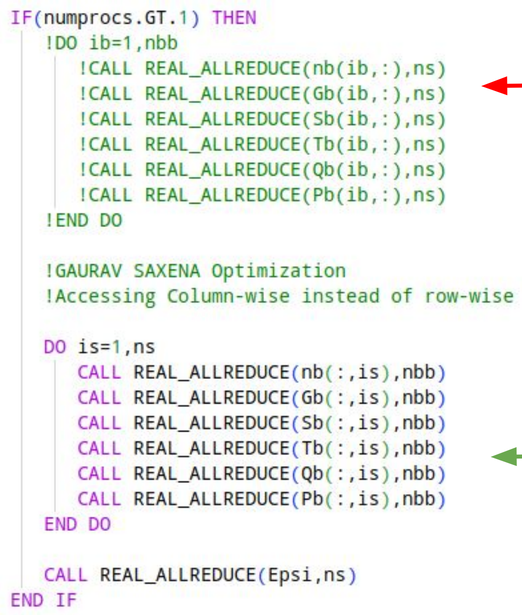


```fortran
DO nm=1,Nnm
  n=np(nm)
  m=mp(nm)
  IF(STELL_ANTISYMMETRIC) THEN

    IF(flag.EQ.0.OR.flag.EQ.2) THEN
      ! B_0=B_0+bnmc0(nm)*cosnm(nm)+bnms0(nm)*sinnm(nm) <---- original
      GS_TEMP_1=GS_TEMP_1+bnmc0(nm)*cosnm(nm)+bnms0(nm)*sinnm(nm)
      !IF(flagB1) B_1=B_1+bnmc1(nm)*cosnm(nm)+bnms1(nm)*sinnm(nm) <---original
      IF(flagB1) GS_TEMP_4=GS_TEMP_4+bnmc1(nm)*cosnm(nm)+bnms1(nm)*sinnm(nm)
      !IF(TANG_VM.AND.flag.GT.1) dBdpsi=dBdpsi+dbnmcdpsi(nm)*cosnm(nm)+dbnmsdpsi(nm)*sinnm(nm) <---original
      IF(TANG_VM.AND.flag.GT.1) GS_TEMP_6=GS_TEMP_6+dbnmcdpsi(nm)*cosnm(nm)+dbnmsdpsi(nm)*sinnm(nm)
    END IF
    IF(flag.NE.0) THEN
      qnmsinnm=bnmc0(nm)*sinnm(nm)
      !dBdz_0=dBdz_0-qnmsinnm*n*nzperiod <---original
      GS_TEMP_8=GS_TEMP_8-qnmsinnm*n*nzperiod
      !dBdt_0=dBdt_0-qnmsinnm*m <--- original
      GS_TEMP_10=GS_TEMP_10-qnmsinnm*m
      qnmcosnm=bnms0(nm)*cosnm(nm)
      !dBdz_0=dBdz_0+qnmcosnm*n*nzperiod  <--- original
      GS_TEMP_8=GS_TEMP_8+qnmcosnm*n*nzperiod
      !dBdt_0=dBdt_0+qnmcosnm*m <--original
      GS_TEMP_10=GS_TEMP_10+qnmcosnm*m
      IF(flagB1) THEN
        qnmsinnm=bnmc1(nm)*sinnm(nm)
        !dBdz_1=dBdz_1-qnmsinnm*n*nzperiod <--- original
        GS_TEMP_12=GS_TEMP_12-qnmsinnm*n*nzperiod
        !dBdt_1=dBdt_1-qnmsinnm*m <---original
        GS_TEMP_14=GS_TEMP_14-qnmsinnm*m
        qnmcosnm=bnmc1(nm)*cosnm(nm)
        !dBdz_1=dBdz_1+qnmcosnm*n*nzperiod <---original
        GS_TEMP_12=GS_TEMP_12+qnmcosnm*n*nzperiod
        !dBdt_1=dBdt_1+qnmcosnm*m <----original
        GS_TEMP_14=GS_TEMP_14+qnmcosnm*m
      END IF
    END IF

  ELSE

    IF(flag.EQ.0.OR.flag.EQ.2) THEN
      ! B_0=B_0+bnmc0(nm)*cosnm(nm) <--- original
      GS_TEMP_2=GS_TEMP_2+bnmc0(nm)*cosnm(nm)
      !IF(flagB1) B_1=B_1+bnmc1(nm)*cosnm(nm) <--original
      IF(flagB1) GS_TEMP_5=GS_TEMP_5+bnmc1(nm)*cosnm(nm)
      !IF(TANG_VM.AND.flag.GT.1) dBdpsi=dBdpsi+dbnmcdpsi(nm)*cosnm(nm) <---original
      IF(TANG_VM.AND.flag.GT.1) GS_TEMP_7=GS_TEMP_7+dbnmcdpsi(nm)*cosnm(nm)
    END IF
```

22 Processes (Total time)	Original (NaN)	No NaN (expected behaviour)	Optimized=Vectorized + aligned + zmm
`CALCB_DEL`	138.31 sec	126.51 sec	35.16 sec
Total App Time	1941.66 sec	685.77	514.02

Speed-up for `CALCB_DEL` = **126.51/35.16 = 3.59x**
Total Speed-up = **685.77/514.02 = 1.33x**
Notes:
(1) Need to check for **correctness**
~~(2) Need to check for **unaligned access.**~~ ✅

Attempt Optimization - 3, MPI_Allreduce()



```
IF(numprocs.GT.1) THEN
   !DO ib=1,nbb
      !CALL REAL_ALLREDUCE(nb(ib,:),ns)
      !CALL REAL_ALLREDUCE(Gb(ib,:),ns)
      !CALL REAL_ALLREDUCE(Sb(ib,:),ns)
      !CALL REAL_ALLREDUCE(Tb(ib,:),ns)
      !CALL REAL_ALLREDUCE(Qb(ib,:),ns)
      !CALL REAL_ALLREDUCE(Pb(ib,:),ns)
   !END DO

   !GAURAV SAXENA Optimization
   !Accessing Column-wise instead of row-wise

   DO is=1,ns
      CALL REAL_ALLREDUCE(nb(:,is),nbb)
      CALL REAL_ALLREDUCE(Gb(:,is),nbb)
      CALL REAL_ALLREDUCE(Sb(:,is),nbb)
      CALL REAL_ALLREDUCE(Tb(:,is),nbb)
      CALL REAL_ALLREDUCE(Qb(:,is),nbb)
      CALL REAL_ALLREDUCE(Pb(:,is),nbb)
   END DO

   CALL REAL_ALLREDUCE(Epsi,ns)
END IF
```

Accessing Matrix by rows but Fortran stores data in column-major order

O(mn) cache-misses

Note:

(1) Demonstrate cache-misses reduction.

(2) Real benefit AFTER load balance problem solved.

Accessing Matrix by columns as Fortran stores data in column-major order

(1) Reduces cache-misses

(2) Useful when problem becomes more load balanced.

O(n) cache-misses

Questions

1. Are `Inf` values acceptable ? i.e. Are you intentionally allowing `Inf` values ?
2. How is a surface being characterized ? (How do you define a surface ? See Q12 also)
3. CAN a surface be divided among multiple MPI processes ? WHAT is divided ?
4. WILL there be a dependency between MPI processes that have sub-parts of a surface ?
5. Will you be moving to a Finite Difference (FDM) scheme ? (Or Have you already moved to a FDM ?)
6. Is `KSP(...)` being called multiply for each time step ? If yes then how is `vecb` being constructed ? (From the manual: drift-kinetic equation solved `N+1` times for each species but `LU` factorization done just once for each `v`)
7. IF using FDM, how will `Ax=b` be solved ? (Direct solver in PETSc)
8. What is the maximum number of processes that have been used in KNOSOS ?
9. Each process outputs a separate file. With 1000's of processes will it not be a problem ?
10. Is there any User Documentation ? Yes at: (a) https://raw.githubusercontent.com/joseluisvelasco/KNOSOS/master/MANUAL/KNOSOSManual.pdf (b) Paper at: https://arxiv.org/pdf/1908.11615.pdf

10. Is the domain a structured or unstructured one ? (Manual describes building a grid around α and λ, using centered and non-centered finite difference with second order accuracy, direct solver based on LU factorization from PETSc used).

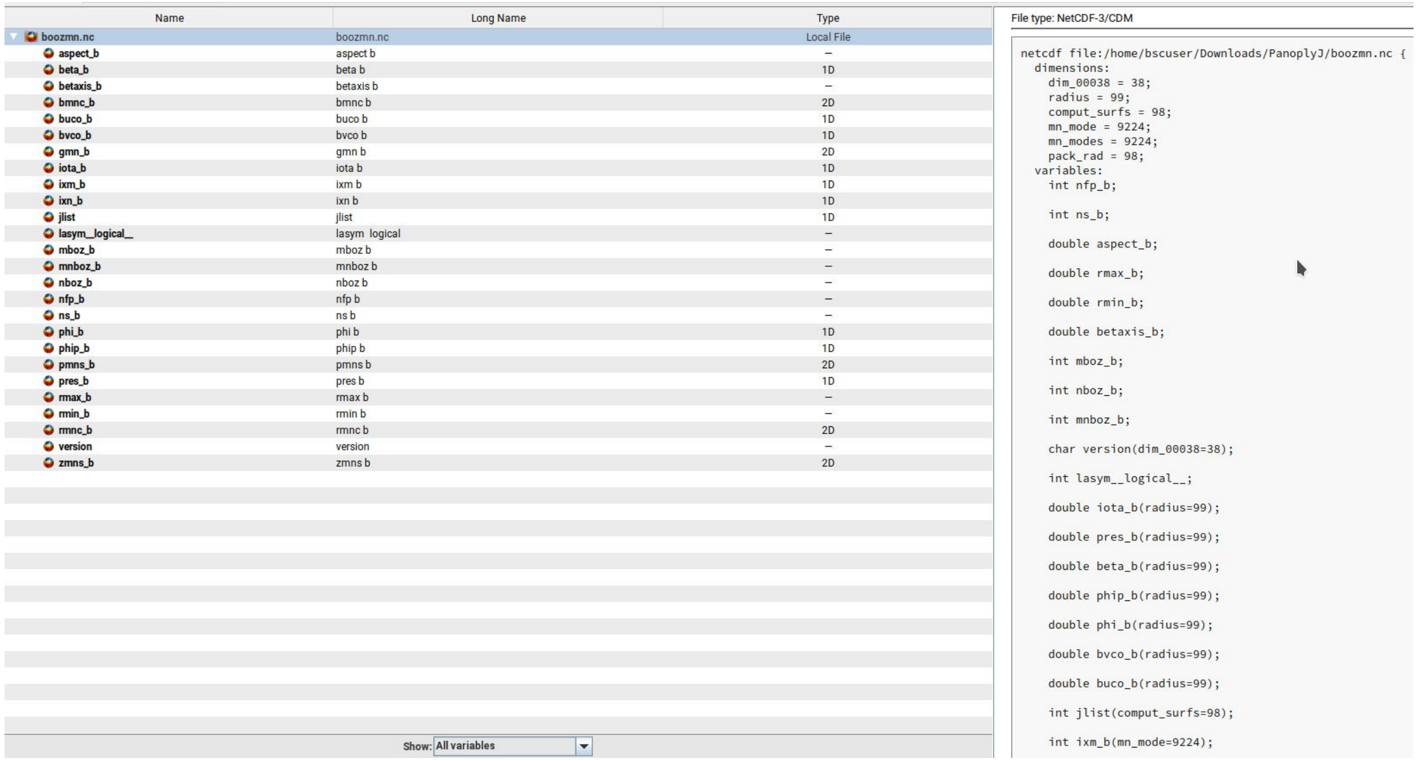
11. If the number of species is nbb and the number of surfaces is ns, then is the total number of surfaces nbb * ns ? i.e. are there ns surfaces per species ? (Allocation of nb(nbb,ns,nerr) kind of indicates this ?)

12. In the file, input.surfaces what does the array S=0.01,0.04,0.09,0.16,0.25,0.36,0.49,0.64,0.81,0.95 indicate ? (How does it lead to the creation of a surface ? Do the coordinates come from boozer.txt ?)

13. rank(is,ierr)=irank what is the use of ierr ? (Number of times calc is repeated ?)

14. What is the relation between boozer.txt and boozmn.nc ?

boozermn.nc - NetCDF file visualized with Panoply



Name	Long Name	Type
boozmn.nc	boozmn.nc	Local File
aspect_b	aspect b	–
beta_b	beta b	1D
betaxis_b	betaxis b	–
bmnc_b	bmnc b	2D
buco_b	buco b	1D
bvco_b	bvco b	1D
gmn_b	gmn b	2D
iota_b	iota b	1D
ixm_b	ixm b	1D
ixn_b	ixn b	1D
jlist	jlist	1D
lasym__logical__	lasym logical	–
mboz_b	mboz b	–
mnboz_b	mnboz b	–
nboz_b	nboz b	–
nfp_b	nfp b	–
ns_b	ns b	–
phi_b	phi b	1D
phip_b	phip b	1D
pmns_b	pmns b	2D
pres_b	pres b	1D
rmax_b	rmax b	–
rmin_b	rmin b	–
rmnc_b	rmnc b	2D
version	version	–
zmns_b	zmns b	2D

Show: All variables

File type: NetCDF-3/CDM

```
netcdf file:/home/bscuser/Downloads/PanoplyJ/boozmn.nc {
  dimensions:
    dim_00038 = 38;
    radius = 99;
    comput_surfs = 98;
    mn_mode = 9224;
    mn_modes = 9224;
    pack_rad = 98;
  variables:
    int nfp_b;

    int ns_b;

    double aspect_b;

    double rmax_b;

    double rmin_b;

    double betaxis_b;

    int mboz_b;

    int nboz_b;

    int mnboz_b;

    char version(dim_00038=38);

    int lasym__logical__;

    double iota_b(radius=99);

    double pres_b(radius=99);

    double beta_b(radius=99);

    double phip_b(radius=99);

    double phi_b(radius=99);

    double bvco_b(radius=99);

    double buco_b(radius=99);

    int jlist(comput_surfs=98);

    int ixm_b(mn_mode=9224);
```

15. Do all processes read the *same values* from the boozermn.nc file ?