

Modernizing OpenRadioss

Modernizing OpenRadioss: A Journey Through Three Decades of Fortran Evolution



Olivier Wienholtz, Director Software Development
olivier@altair.com, co-Author & Speaker

Laurent Berenguer, Lead Software Developer HPC
lberenguer@altair.com, co-Author

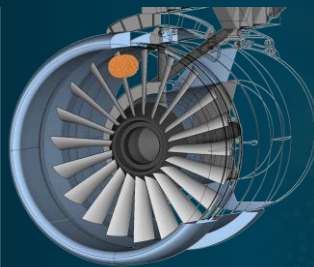
Marian Bulla, Director OpenRadioss Community
bulla@altair.com, co-Author

Altair Radioss – Proven Crash & Impact Simulation Software

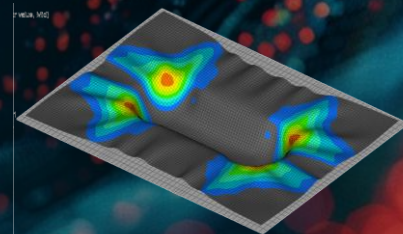
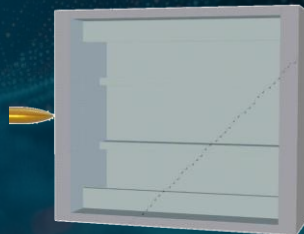
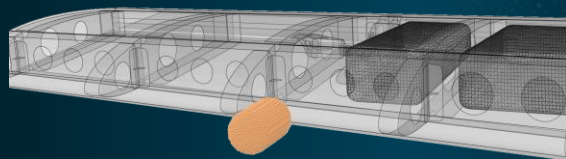
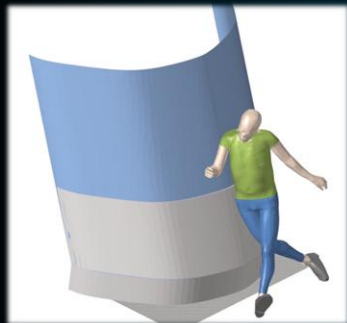
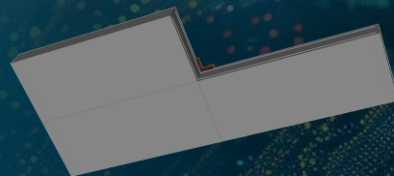
Automotive & Rail



Aero & Defense



Consumer Goods & Manufacturing



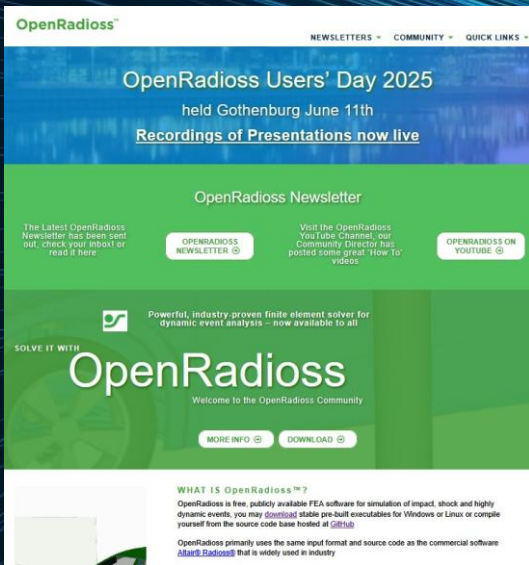
Large Scale Computing and Parallelization

OpenRadioss – The Open-Source Version of Radioss

Democratize explicit dynamics, build active community and accelerate innovation

Introduced 3 years ago

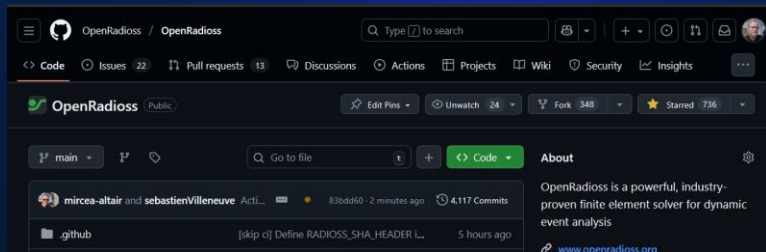
OpenRadioss.org



Fast growing adoption since then

- 16000+ binary downloads
- 4000+ commits / 14 active contributors
- 400+ discussion topics on forum
- Social Media: LinkedIn, YouTube, webinars, ...
- Steering committee / Users Days

<https://github.com/OpenRadioss/OpenRadioss>



R&D Collaborations & Partnerships

Universities



Institutes & Labs



Hardware & Software Providers



OpenRadioss

We continue to use Fortran because

- A lot of working legacy code
- Accessible for our typical contributors: Mechanical Engineer, Matlab background
- Consistent and high performant with only a few guidelines
- Proven reliability and maturity

But C++ contributions are also welcome

Design Constraints

- **Performance:** High Solver efficiency & parallel performance
- **Portability:** ARM64, X86-64, Linux, Windows, Compilers (GCC/Gfortran, ArmFlang, AOCC, Intel OneAPI)
- **Bitwise reproducibility:** Whatever Parallel run modus: MPI x OpenMP configurations
Reduce Optimizations: avoid use of Multiply-Add (FMA) assembly instructions



Modernization Strategy



Modernization ?

Ongoing efforts for more than 30 Years

- **Improve code structure** to adapt parallelism needs
- **Work on main data structures**

Prior to OpenRadioss

- **Makefile** with self-made dependency creation tool (use of Cygwin on Windows)
- **F77 Fixed Format**
- **F77 Common blocks, Some equivalences, Common variables in Modules**
- **OpenMP Threadprivate variables in commons**
- **C preprocessor directives #ifdef** for system specific code

➤ **Need of code modernization**

Modernization strategy: gradual improvements

Data structures

- **Group arrays and variables in user defined types** : reduce argument list in routine call

New files / subroutines : Guidelines from fortran-lang.org (with minor caveats)

- Free-form file format, Fortran 95 + selected recent features
 - C/Fortran interface : `iso_c_bindings`
 - Intrinsic functions etc...
- New subroutines → module
 - Forces explicit interface and compile time errors when argument mismatch
- Limit C preprocessor usage: `#define` → parameter

Modernization strategy: gradual improvements

Legacy files: Start refactoring

- **Equivalences** have been removed.
- **OpenMP Threadprivate** variables in common blocs has been removed
- Remove **COMMON BLOCK : in progress**
 - Common Blocks defined in include files & shared in multiple routines
 - Move them in meaningful types with appropriate arrays
 - Some common variables are parameters : transform them into parameters
- **Rewrite Fortran fixed routines** in Fortran Free format

Executable generation:

Porting to CMake

- **Simplification of build process** : Few 100. lines of CMake wrt. Thousands Makefile
- **Windows native build** with Ninja (no need for Cygwin/MSYS2 or equiv. layer)
- **Automatic dependency checker instead of** handmade dependency checker.
- **Less duplication** because of the variety of compilers
- **fpm (Fortran Package Manager) was not mature enough** (lacks mutli-language capabilities)



```
#
# CMake project file for Radioss Starter
# -----

cmake_minimum_required (VERSION 3.15)
project (Radioss_Engine)

# Windows specific settings
# -----
if (arch STREQUAL "win64")
  set(CMAKE_MSVC_RUNTIME_LIBRARY "MultiThreaded$<<CONFIG:Debug>;Debug")
endif()

# Verbose
# -----
set (CMAKE_VERBOSE_MAKEFILE on)

# Language settings & Compiler Settings
# -----
enable_language (Fortran)
enable_language (C)
enable_language (CXX)
```

Executable generation: Code Quality / Avoiding bugs



Gfortran: Needed for Open sourcing Radioss

- **Use of Gfortran capabilities** with address sanitizer:
 - Find earlier memory related bugs / report to developer is faster
- **GCC plugin for minimal static analysis**
 - argument mismatch on legacy files + other minor checks
- **Several Gfortran warnings** transformed into errors
 - Uninitialized variables : -Werror=maybe-uninitialized

Intel OneAPI

- **Check bounds** and other debug capabilities

Benefits

Code is easier to read and maintain !

- **Lower the entry barrier for new contributors**
 - More modern code style is more appealing for our external (non Altair) contributors
 - Code becomes more readable
 - Reduce routines argument list make reading more comprehensive.
- **Reduce development time:** Less time spent in structural tasks
 - With newly created types : adding members is easier than modifying call tree.
- **Working with LLMs** gives better results
- **Better Code review** by LLM (GitHub CoPilot)
 - Better result with GitHub Co-Pilot review & Pull Request summary
- **Iso_c_bindings:** cleaner and easier to maintain C/C++ interface

Benefits

Less C processor directives and Fortran Free format :

- Compatibility with FORTITUDE

```

starter/source/materials/eos/hm_read_eos_powderburn.F90:46:7: C092 subroutine not contained within (sub)module or program
44 | ||| submodel_mod      ../starter/share/modules1/submodel_mod.F
45 | |||=====
46 | /      subroutine hm_read_eos_powderburn(iout, pm, ipm, unitab, lsubmodel, imideos, eos_struct, npropm, npropmi, mtag, &
47 | |      eos_tag ,ieos)
48 | |-----^ C092
49 | |-----Modules
50 | |-----

```

Improved Code quality on contributions with GCC/Gfortran features

- Compile time error (modules+gcc plugin) / Asan catch on non-regression tests
- Gfortran Warnings moved to errors : Several catches per Week !

Examples : 2/3 catches on uninitialized variables

```

Error: 'z14' may be used uninitialized [-Werror=maybe-uninitialized]
/home/lberenguer/OpenRadioss/engine/source/boundary_conditions/bcs_nrf.F90:140:36:
140 |      real(kind=WP) :: X14,Y14,Z14      !< [N1N4] vector

```



Caveats



Restricts usage of Intent(in|out|inout)

- **Easy to forget.**
- **Bug when in a call stack:** INTENT(IN) or OUT and variable is modified
- **Different behavior** depending on compiler (Gfortran/IntelOneAPI)

Legacy Routine 1
INTEGER,INTENT(IN) :: A



Legacy Routine 2
INTEGER :: A



Legacy Routine 2
INTEGER, INTENT(INOUT) :: A
A=A+1

Arrays of user-defined type:

Nice to write, but :

- **Poor** performance: Memory access, Compute loop may not vectorize
- **Memory** overhead
- Opposite is preferred : Types of Arrays

```
type my_type_  
    integer :: var1  
    integer :: var2  
end type my_type_  
...  
type(my_type), dimension(number_options) :: my_typed_array  
  
do i=1, number_options  
    var=my_typed_array(i)%var1 + my_typed_array(i)%var2  
enddo
```

Restricts usage of

Pointers: performance degradation

- **Can be partially mitigated with CONTIGUOUS**
- **Different behavior** depending on compiler (Gfortran/IntelOneAPI)

```
module compute_head_mod
  subroutine compute_head(arrays,n,ibegin)

    integer, intent(in) :: n,ibegin
    type(my_type), intent(inout), TARGET :: arrays
    real(8), dimension(:), pointer, CONTIGUOUS :: acc

    acc => array%in%hierarchy%of%struct
    call compute(acc)

  end subroutine compute_head
end module compute_head_mod
```

MPI: keep #include “mpif.h” instead of use mpi module

- **No module compatibility** between 2 Gfortran versions
Installed OpenMPI on Linux may no be built with used Gfortran compiler
- Encapsulate MPI calls in routines to limit C preprocessor usage
 - Kind of homemade mpi module

Restricts usage of

Polymorphism: One OpenRadioss option uses polymorphism

- **Syntax can be confusing** for our mechanical contributors,
- **Experienced** bugs or unexpected behavior on some compilers: ifort (now deprecated), armflang.

```
class(base), allocatable :: arr(:)
integer :: status
allocate(extended :: arr(3))
select type (assoc => arr)
  type is (extended)
    ! ok
    write(6,*) allocated(arr)
end select
select type (arr)
  type is (extended)
    ! compilation error:
    write(6,*) allocated(arr)
end select
```

```
t1%id = 1
t2%id = 2
ptr => t1
select type (ptr)
  type is (extended)
    ptr => t2
end select
print *, "After: ptr%id =", ptr%id !1
ptr => t1
select type (assoc => ptr)
  type is (extended)
    ptr => t2
end select
print *, "After: ptr%id =", ptr%id !2
```

- **Slow down performance** : Comparison with C++ (C++ de-virtualize & is ~4x faster)

Blog on polymorphism by Steve Lionel

<https://stevelionel.com/drfortran/2020/06/30/doctor-fortran-in-not-my-type>

Issues & forums on Stackoverflow

<https://stackoverflow.com/questions/26068020/error-if-selector-expression-in-select-type-is-not-a-named-variable-associate>

► Conclusion & Wishlist



Conclusion: OpenRadioss will remain Fortran based program !

But Code Modernization is a need for us.

Pragmatic approach: Need drives the refactoring

- **Rewrite/Refactor** legacy code to F95 free format when working on projects
- **Adopt** new feature only when clear benefits

Still a long journey ahead

~ 10 % of our Fortran code base is modernized

Workforce shortage mitigation

- Simple computational kernels, so that non-expert can contribute: “Basic Fortran”
- Open to C++ contribution thanks to iso_c_bindings

Wishlist

Simple generic programming for basic types

- lot of code duplication can be avoided
- Today's workarounds (Parametrized derived types PDTs, Fypp) not satisfactory

ISO C binding : Better String / character handling

ISO C bindings not ideal for Character:

- Silent truncation / garbage
- Array of strings (typically one string per line of a file)

 **ALTAIR** altair.com/Radioss

 altair.com/Radiossopenradioss.org

 github.com/OpenRadioss

 **YouTube** youtube.com/@OpenRadiossCommunity

Thank You !