Unified Multiphysics Framework for Industrial Scale Simulations with Moving Geometries

Complex Phenomena Unified Simulation Research Team

Makoto Tsubokura (mtsubo@riken.jp), Keiji Onishi, Rahul Bale, Koji Nishiguchi and WangWeiHsiang Wang

Research and development of a unified multiphysics framework for large scale industrial applications. Focusing on efficient and scalable numerical methods for turbulent flow, fluid-structured interaction, heat transfer, aero-acoustics, chemical reactions and their complex interaction thereof.

Unified Multiphysics Framework

- Multiphysics treated as a unified continuum
- All discretized equations mapped onto a hierarchical Cartesian mesh
- Multiblock hierarchical mesh using Building Cube Method (BCM)
- Immersed boundary methods to capture complex geometries
- Automatic wrapping techniques to handle dirty CAD data

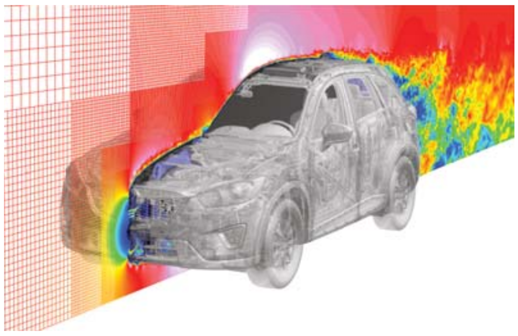


Fig. 1. Example of a BCM mesh.

High Performance Computing

- Each logical subdomain (cube) can be computed independently
- Easy and efficient parallelization, cache/SIMD friendly data structures
- Hybrid MPI+OpenMP approach, with multithreaded halo-exchange

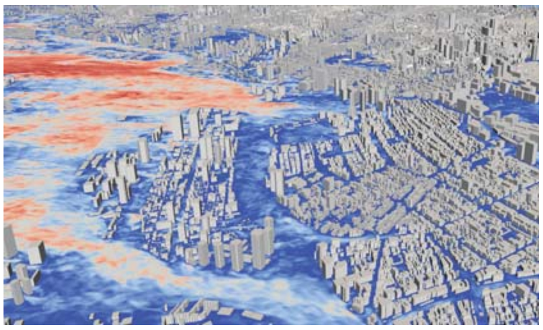
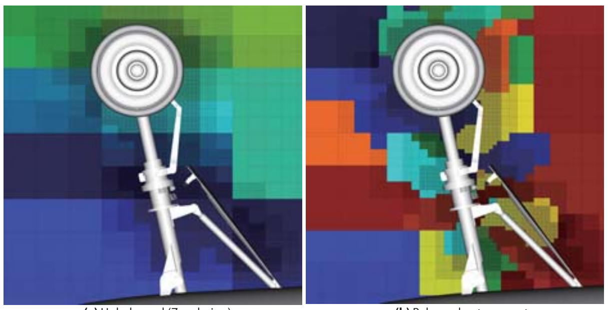


Fig. 2. Simulation of urban boundary layer around Tokyo bay.

Dynamic Load Balancing

- Workload modelled with a weighted dual graph of the Eulerian mesh
- Node weights represent computational cost (fluid, reactions, geometry)
- Edge weights represent communication cost (halo-exchange)
- A priori estimation of redistribution cost and computational gain
- Balanced using an intelligent remapping based load balancer



(a) Unbalanced (Z-ordering).
(b) Balanced wrt. geometry.

Fig. 3. Load balancing wrt. immersed geometry colored by MPI rank.

Lagrangian-Eulerian approach for moving geometries

- Geometry is discretized into Lagrangian points
- Underlying cubes of the BCM are used to partition the points
- Lagrangian points are stored as cube based unordered sets
- Unordered set enables efficient insertion and removal as points move
- Any IBM can easily be coupled with the Lagrangian data structure

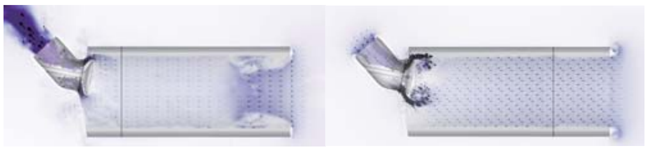


Fig. 4. Piston movement in an internal combustion engine.

Applications

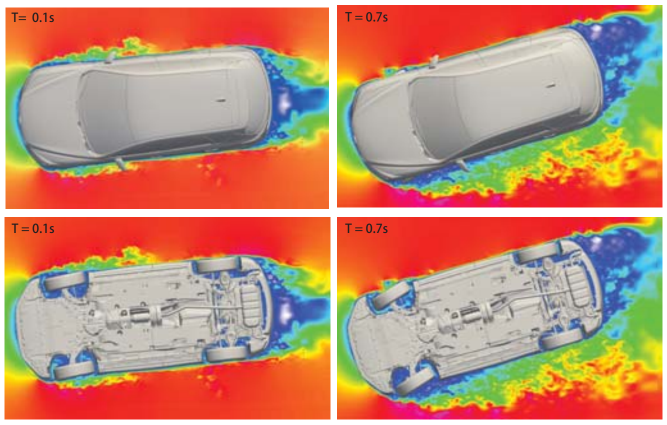


Fig. 5. Simulation of full vehicle turning in a tunnel with uniform flow.

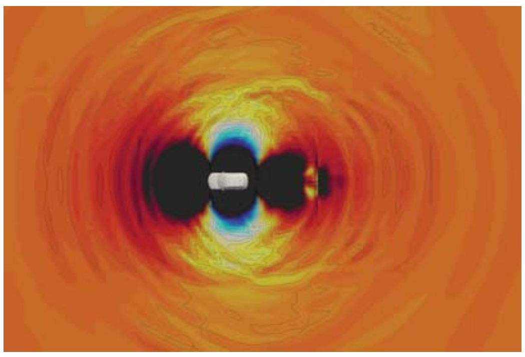


Fig. 6. Pressure propagation during aero-acoustic simulation of vehicle.

Summary

Building Cube Method enables scalable industrial scale simulations with:

- Efficient parallelization using parallel I/O and dynamic load balancing
- Fluid/structure interaction with moving and deforming boundaries
- Treatment of complicated geometry and dirty CAD

References

Onishi, K., et.al. *Use of the Immersed Boundary Method within the Building Cube Method and its Application to Real Vehicle CAD Data*, AIAA-2013-2713, 2013.

Bale, R., et. al. *A Lagrangian-Eulerian Based Immersed Boundary Method For Industrial Scale Simulations with Complex Geometries*, In Proceedings of the 27th ParCFD conference, 2015.

KMATHLIB

High Performance and Scalable Numerical Library for the K Computer

Large-scale Parallel Numerical Computing Technology Research Team, AICS Research Division

KMATHLIB Project

KMATHLIB is a software package including large-scale, highly parallel and high-performance numerical libraries and related software for peta-scale supercomputer systems such as the K computer. KMATHLIB comprises some layers as shown in Fig. 1. The top and the middle layers consist of related software such as a tractable user interface (KMATHLIB API), a data manager and a data structure converter. The numerical libraries on the bottom layer comprise several components such as

- Linear system solver,
- Eigenvalue solver,
- Singular Value Decomposition,
- Fast Fourier Transform,
- Nonlinear equation solver, and
- Other numerical software.

A user can use various libraries via KMATHLIB API. The similar principle is adopted in PETSc[1] and Trilinos[2]. KMATHLIB API's framework provides computer science with advantages such as concise and efficient design of the application code, and high flexibility and maintainability.

The KMATHLIB project promotes research and development of innovative computational technology for the K computer and next generation systems. A part of KMATHLIB has been released in 2016 on our team website.

[1] https://www.mcs.anl.gov/petsc
[2] https://trilinos.org

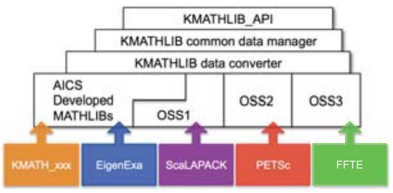


Fig. 1 Conceptual diagram of KMATHLIB

KMATHLIB API

KMATHLIB API is a user interface to write an application code with a unified format without consideration of differences from a usage and a data structure of each of libraries. A sample code is shown in Fig. 2. KMATHLIB API has following features:

1) Flexible data handling by using PETSc,
2) Flexible plug-in,
3) Multi-platform, and
4) High performance and low overhead.

1) Data structure is automatically converted from PETSc's data type to match the library to be called. For example with the sample code, the data structure conversion processing is executed by the KMATH_Solve subroutines in step 1 and 2.

2) Any libraries are available by registering a small amount of their information of the libraries as a plug-in.

3) KMATHLIB API performs on various systems (the K computer, Fujitsu FX10, Fujitsu FX100, A cluster system of x86).

Future work

Future work about KMATHLIB API is as follows.

- An expansion of the number of software libraries by default.
- A compatible upgrade of KMATHLIB API substituting a function of PETSc.
- A development of advanced topics such as numerical reproducibility and fault tolerance, and a support of C, C++ and other programming languages.

```
program abxbxd
 ! Given a real symmetric matrix A and a nonsingular matrix B,
 ! Find a diagonal matrix D and a nonsingular matrix X satisfying ABX=BXD.

 use kmath_lib_mod
 use kmath_plugin_deig_eigenexa_mod
 use kmath_plugin_dlsm_scalapack_mod

 implicit none

#include <finclude/petscsys.h>
#include <finclude/petscvec.h>
#include <finclude/petscmat.h>

 Mat :: A, B, X, Y
 Vec :: D
 PetscInt :: ierr
 integer :: i, j, n, rank
 type(s_context) :: h

 call PetscInitialize(PETSC_NULL_CHARACTER, ierr)
 call MPI_Comm_rank(PETSC_COMM_WORLD,rank,ierr)

 ! Create the matrices and the vector by using PETSc's functions.
 n = 10000
 call MatCreateDense(PETSC_COMM_WORLD, PETSC_DECIDE, &
     PETSC_DECIDE, n, n, PETSC_NULL_SCALAR, A, ierr)
 call MatCreateDense(PETSC_COMM_WORLD, PETSC_DECIDE, &
     PETSC_DECIDE, n, n, PETSC_NULL_SCALAR, B, ierr)
 call MatCreateDense(PETSC_COMM_WORLD, PETSC_DECIDE, &
     PETSC_DECIDE, n, n, PETSC_NULL_SCALAR, X, ierr)
 call MatCreateDense(PETSC_COMM_WORLD, PETSC_DECIDE, &
     PETSC_DECIDE, n, n, PETSC_NULL_SCALAR, Y, ierr)

 call VecCreate(PETSC_COMM_WORLD, D, ierr)
 call VecSetType(D, VECMPI, ierr)
 call VecSetSizes(D, PETSC_DECIDE, n, ierr)

  <Set the elements of A, B>

 call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

 ! Step1: solve the eigenvalue problem AY=YD by using EigenExa.
 call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DEIG_EigenExa, ierr)
 call KMATH_Set_Parameter(h, 'A', A, ierr)
 call KMATH_Set_Parameter(h, 'X', D, ierr)
 call KMATH_Set_Parameter(h, 'Z', Y, ierr)
 call KMATH_Solve(h, ierr)
```

The data structure is converted from PETSc's type to EigenExa's type.

```
 ! Step 2: solve the linear system BX=Y by using ScaLAPACK.
 call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLSM_ScaLAPACK, ierr)
 call KMATH_Set_Parameter(h, 'A', B, ierr)
 call KMATH_Set_Parameter(h, 'B', Y, ierr)
 call KMATH_Set_Parameter(h, 'X', X, ierr)
 call KMATH_Solve(h, ierr)
```

The data structure is converted from EigenExa's type to ScaLAPACK's type via PETSc's type.

```
 call KMATH_Destroy(h, ierr)

 <Post-processing using the results >

 call MatDestroy(A, ierr)
 call MatDestroy(B, ierr)
 call MatDestroy(Y, ierr)
 call MatDestroy(X, ierr)

 call PetscFinalize(ierr)
end program abxbxd
```

Fig. 2 A sample code to solve an eigenvalue problem and a dense linear system by using EigenExa and ScaLAPACK via KMATHLIB API, respectively. Functions of KMATHLIB API and PETSc are written in orange and green.

Our team website: http://www.aics.riken.jp/labs/lpnctrt/index_e.html
For further information, please contact: kmathlib@riken.jp