ZZ-HIFU:
Simulator for High-Intensity Focused Ultrasound Therapy

An ultrasound propagation simulation for cancer treatment using high-intensity focused ultrasound (HIFU)

version 1.00,
11 Feb. 2013

Research and Development of the Next-Generation Integrated Simulation of Living Matter
Department of Mechanical Engineering, School of Engineering, the University of Tokyo
Principal developer: Yoichiro Matsumoto (Professor, the University of Tokyo)
Signatured by Kohei Okita (okita.kohei@nihon-u.ac.jp)

Table of Contents

1. Files included in this package 1

2. Compilation and execution 2

   2.1. Compilation 2

   2.2. Execution 2

3. Source code handling 3

   3.1. Introduction 3

   3.2. ZZ-HIFU-K.F source code compiling settings 3

   3.3. Description of ZZ-HIFU-K.F modules 6

   3.4. About physical_properties.F90 9

4. Executing Application Using Specified Input Data 10

   4.1. About input data 10

   4.2. Creating volume data using signed distance function 10

   4.3. Visualizing calculation results 10

5. References 11

1. Files included in this package

ZZ-HIFU-K.F: Source code of the ultrasound propagation simulation
physical_properties.F90: Source code of the module for setting IDs and the physical property values
Makefile: for compiling the source code to install the binary ZZ-HIFU program
mpijob.sh: a shell script to execute jobs
README: this file
COPYING: copyright description

2. Compilation and execution

2.1. Compilation

Compilation of the source code requires modifying the ‘Makefile’ provided in accordance with the computational environment, and is performed by the ‘make’ command.

2.2. Execution

Prior to executing an application, the sample shell script ‘mpijob.sh’ needs to be edited in accordance with the computational environment, including entering jobs in the job scheduler given by the particular computing environment.

3. Source code handling

3.1. Introduction

ZZ-HIFU-K.F performs the simulation of pressure wave propagation in various multicomponent media. The following subsections explain the source code compilation settings, as well as the roles of modules and subroutines. See references 1 and 2 for details of the basic equations and their discretization.

3.2. ZZ-HIFU-K.F source code compiling settings

The source code of ZZ-HIFU-K.F provides the optional compilation settings in its header section, allowing the direct editing of source code or the selection of compilation options.

```
! Use MPI
! Specifies whether using MPI or not.
c#define _USE_MPI_
!
! Use buffer send receive
! Specifies whether or not to use buffer in I direction for data transmission and reception.
c#define _BUFFER_SENDRECEIVE_
!
! Specifies whether or not to use buffer in J direction in addition to I direction for data transmission
and reception.
c#define BUFFER_J
!
! Specifies whether or not to use buffer in K direction in addition to I direction for data transmission
and reception.
c#define BUFFER_K

! Specifies whether or not to generate a benchmark test code that requires no data input.
c#define _TEST_
!
! Check results for sample input
! Specifies whether or not to check the result of the executed test code.
c#define _TEST_CHECK_DATA_
!
! The size and division are obtained from command argument
! Specifies whether or not to set the size of calculation domain and its divisions as command line
arguments in test code execution.
c#define _ARGUMENT_

! Number of computational grid
! Specifies the number of grid points in X or I direction.
c#define _NX_ 1400
!
! Specifies the number of grid points in Y or J direction.
c#define _NY_   600
!
! Specifies the number of grid points in Z or K direction.
c#define _NZ_ 1200
!
! Number of grid for PML
! Specifies the number of grid points for the PML.
c#define _NPML_ 20
```

```
! Grid resolution dx = dy = dz
! Width of calculation grid
c#define _PITCH_ .1d-3

! Division of numerical domain
! NPROCS must be _IDIM_ * _JDIM_ * _KDIM_
! Specifies the number of divisions of the domain in I direction.
c#define _IDIM_ 8
!
! Specifies the number of divisions of the domain in J direction.
c#define _JDIM_ 2
!
! Specifies the number of divisions of the domain in K direction.
c#define _KDIM_ 4

! Total time step
! Specifies time step.
c#define _NSTEPMAX_ 10000
!
! Specifies the starting time step to record statistical volume data.
c#define _CHECK_IN_   9000
!
! Data save
! Specifies the time step interval to save three-dimensional data.
c#define _DATA3D_SAVE_ 500
!
! Specifies the time step interval to save two-dimensional data.
c#define _DATA2D_SAVE_ 20000
!
! Specifies the name of a file for restarting calculation in which the restart file is saved.
c#define _RESTART_FILE_NAME_ '("restart-",i3.3,".bin")'
!
! Specifies the restart file to load to begin restarting calculation.
c#define _RESTART_
!
! Specifies whether or not to reset temperature to zero when restarting takes place.
c#define _RECALCULATION_TEMP_

! Specifies whether or not to produce a sound from the sound source set at the target.
c#define _HearingMode_
!
! Specifies the table to store the transducer coordinates that save the time-series pressure data.
c#define _RecordPointsTable_ "VcenterNewID.table"
!
! Specifies whether or not to maintain the time-series pressure data at the specified coordinates.
c#define _RecordPatPiezo_
!
! Specifies whether or not to save the time-series pressure data at the focal point.
c#define _RecordPatFocalPoint_

! Specifies the table for setting phase delays to the transducer arrays.
c#define _PhaseDelayTable_ "PhaseDelayCC.table"

! Position of Focal Point
! Specifies the X coordinate of the focal point.
```

```
c#define _FocalPointX_ 0.d-3
!
! Specifies the Y coordinate of the focal point.
c#define _FocalPointY_ 0.d-3
!
! Specifies the Z coordinate of the focal point.
c#define _FocalPointZ_ 0.d-3

! Using Tait eq. for the equation of state of water
! Applies the Tait calculus to represent the state of material.
c#define _EOS_TAIT_

! CFL number
! Specifies the Courant number.
c#define _CFL_ 0.1d0

! Temperature
! Obtains the time evolution in the temperature field.
c#define _TEMPERATURE_
!
! Time scale for Temperature
! Obtains the timescale ratio between ultrasound propagation and temperature.
c#define _DTAU_ 1d+6

! Denaturation
! Obtains the time evolution in the thermal denaturation.
c#define _DENATURATION_
!
! Increase of acoustic impedance due to denaturation
! Obtains rate of change of sound velocity accompanying the thermal denaturation.
c#define _DENATURATION_CS_ 1.d0

! Viscosity term is solved implicitly
! Specifies the number of loops when the viscosity term is solved implicitly.
c#define _IMPLICIT_VIS_ 3

! Specifies the list of file names containing transducer shape data.
c#define _PIEZO_DATA_LIST_ '("./PiezoList/",i3.3,".list")'
!
! Specifies the list of file names containing partial domain shape data.
c#define _PARTS_DATA_LIST_ '("./PartsList/",i3.3,".list")'
!
! Specifies the volume data of the objects.
c#define _OBJECT_DATA_ '("../Objects/Empty/",i3.3,"-ev.sph")'

! Pressure constant in Tissue with Air properties
! Sets the pressure constant in air domain.
c#define _CONST_PRESSURE_AIR_

! Using array of constants for not binary but mixture model
! Employs arrays as constants.
c#define _ARRAY_CONST_

! No heat conduction in air
! Sets the heat conduction at zero in air domain.
c#define _ZERO_HEATCONDUCTION_AIR_
```

```
! Boundary Condition
! Specifies the boundary conditions for the PML.
c#define _PML_X0_
c#define _PML_X1_
c#define _PML_Y0_
c#define _PML_Y1_
c#define _PML_Z0_
c#define _PML_Z1_
!
! Specifies the symmetric boundary conditions.
c#define _MIRROR_X0_
c#define _MIRROR_X1_
c#define _MIRROR_Y0_
c#define _MIRROR_Y1_
c#define _MIRROR_Z0_
c#define _MIRROR_Z1_
```

3.3. Description of ZZ-HIFU-K.F modules

module Grid
A module to set grid pitch

module FDM
A module to set interpolation and difference coefficients

subroutine FDM_Init
Specifies interpolation and difference coefficients.

subroutine FDM_Operator(xc,n,xk,d0k,d1k)
Specifies interpolation and difference coefficients using Lagrange interpolation.
See reference 3 for further details.

module Array_Index
A module to specify index of arrays

subroutine Array_Global_Index_Init(i, j, k)
Specifies index of arrays.

subroutine Array_Index_Init(ista, iend, jsta, jend, ksta, kend)
Specifies index of arrays.

module MyMPI
A module to perform inter-process data communications between divided domains

subroutine MyMPI_init_comm_cart(l, m, n)
See reference 4 for further details.

subroutine MyMPI_init_comm_cart
See reference 4 for further details.

subroutine PARA_RANGE(N1, N2, NPROCS, IRANK, ISTA, IEND)
See reference 4 for further details.

subroutine MyMPI_init_type_vector
See reference 5 for further details.

subroutine MyMPI_Allocate_Buffer_Array
Allocates an array for data transmission and reception using buffer.

subroutine MyMPI_SendReceive_p0(iband, jband, kband, p0, vec)
Data transmission and reception using buffer

subroutine MyMPI_SendReceive_u (iband, jband, kband, u , vec)
Data transmission and reception using buffer

subroutine MyMPI_SendReceive_v (iband, jband, kband, v , vec)
Data transmission and reception using buffer

subroutine MyMPI_SendReceive_w (iband, jband, kband, w , vec)
Data transmission and reception using buffer

subroutine MyMPI_SendReceive_T (iband, jband, kband, T , vec)
Data transmission and reception using buffer

subroutine MyMPI_sendp(iband, jband, kband, p, vec, ireq)
Data transmission and reception using derived data type
See reference 5 for further details.

module Parts
A module for handling partial domain data

subroutine dtloadfp(fname, delta, orig, sp)
Loading SPH data of partial domain

subroutine Parts_Load(fname, nsp, sp)
Loading SPH data of partial domain

subroutine Parts_SDFtoVF(nsp, sp)
Obtains volume fraction converted by the Signed distance function.

module PML
A module for the Perfectly matched layer

subroutine PML_Init
Specifying coefficients used by the Perfectly matched layer

module HIFU3D
A module for simulating ultrasound propagation

subroutine HIFU3D_Allocate
Allocates arrays.

subroutine HIFU3D_Initialize_Array
Initializes arrays.

subroutine HIFU3D_Restart(fname, ifrag, nstep)
Saves and loads restart file.

subroutine HIFU3D_Constant_Mixture_Properties
Specifies constants for calculating physical property values.

subroutine HIFU3D_Update_Mixture_Properties
Updates physical property values.

subroutine HIFU3D_BC_Mirror(dims, sp)
Specifies symmetric boundary conditions.

subroutine HIFU3D_Cal_Viscous
Calculates viscosity terms

subroutine HIFU3D_Cal_U
Updates velocity components in X direction.

subroutine HIFU3D_Update_Boundary_U
Updates symmetric boundary conditions for velocity components in X direction.

subroutine HIFU3D_Cal_V
Updates velocity components in Y direction.

subroutine HIFU3D_Update_Boundary_V
Updates symmetric boundary conditions for velocity components in Y direction.

subroutine HIFU3D_Cal_W
Updates velocity components in Z direction.

subroutine HIFU3D_Update_Boundary_W
Updates symmetric boundary conditions for velocity components in Z direction.

subroutine HIFU3D_Cal_P
Updates pressure.

subroutine HIFU3D_Cal_Boundary_P
Updates non-reflecting boundary conditions for pressure.

subroutine HIFU3D_Update_Boundary_P
Updates symmetric boundary conditions for pressure.

subroutine HIFU3D_Cal_Grad_T
Calculates temperature gradient.

subroutine HIFU3D_Cal_T
Updates temperature.

subroutine HIFU3D_Update_Boundary_T
Updates boundary conditions for temperature.

subroutine HIFU3D_Cal_Dose
Calculates the Thermal dose.

subroutine HIFU3D_Cal_Grad_Phi
Calculates order parameter gradients.

subroutine HIFU3D_Cal_Phi
Updates order parameters.

subroutine HIFU3D_Update_Boundary_Phi
Updates boundary conditions of order parameters.

Other functions

subroutine dtloadS(nstep, fname, delta, orig, dims, sp)
Loads three-dimensional scalar data in SPH format.
See reference 6, "V-Isio manual," for detailed information on format.

subroutine dtsaveS(nstep, fname, delta, orig, dims, sp)
Saves three-dimensional scalar data in SPH format.

subroutine dtsaveS2D(nstep, fname, delta, orig, dims, sp
Saves two-dimensional scalar data in SPH format.

subroutine Input_TestData (omega, rho, cs)
Creates input data for test calculation.

subroutine Check_TestResults
Confirms the result of calculation test.

3.4. About physical_properties.F90

physical_properties.F90 is the source code of a module that relates physical properties of a material to corresponding ID numbers. It requires appropriate modifications for actual calculations. The physical property values of organs are obtained from reference 7.

4. Executing Application Using Specified Input Data

4.1. About input data

To perform simulation of ultrasound propagation in the target material, one of three input data listed below should be used.

* Volume data representing the shape of transducers serving as sound source (using a signed distance function)

* Volume data of target material (ID number or data denoting distribution of a material such as volume fraction)

* Volume data representing the shape of constituents other than two types listed above (using a signed distance function)

The above input data need to be prepared per each divided domain for parallel process computation. The shape data given by signed distance function requires the input data needed to express the shape of the given partial domain. These input data should be provided in SPH format (see reference 8). The ID numbers for the shapes given by the signed distance function are given using two digits before the '.sph' file extension.

In the case of parallel computation, a list of data for each divided domain to be executed by each process helps to load appropriate input data to each divided domain calculation. This is achieved by reading a specified data list at the time of compilation. A data list can be created and defined as an optional compilation setting, as shown below.

```
! List of the names of files holding transducer shape data.
c#define _PIEZO_DATA_LIST_ '("./PiezoList/",i3.3,".list")'
!
! List of the names of files holding shape data of divided domains
c#define _PARTS_DATA_LIST_ '("./PartsList/",i3.3,".list")'
!
! Volume data of the objects
c#define _OBJECT_DATA_ '("../Objects/Empty/",i3.3,"-ev.sph")'
```

4.2. Creating volume data using signed distance function

The V-SDFlib signed distance function (SDF) generation library developed by RIKEN's VCAD System Research Program is helpful in obtaining the volume data using SDF converted from the shape data created by CAD software.

4.3. Visualizing calculation results

The V-Isio, visualizing software also developed by the VCAD System Research Program of RIKEN, is able to visualize SPH format output data. In this example, since the output data represents each divided domain, it is required that those data be merged into single volume data to visualize a whole image.

5. References

[1] K. Okita et. al., Development of High Intensity Focused Ultrasound Simulator for Large Scale Computing, Int. J. Numerical Methods in Fluids, Vol.65, pp.43-66, 2011.

[2] K. Okita et. al., Numerical Simulation of the Tissue Ablation in High Intensity Focused Ultrasound Therapy with Array Transducer, Int. J. Numerical Methods in Fluids, Vol.64, pp.1395-1411, 2010.

[3] T. Kajishima, Numerical Simulation of Turbulent Flow, Yokendo Publishing Co., 1999

[4] Y. Aoyama, Practical MPI Programming, RIKEN Advanced Center for Computing and Communication, 2005.

[5] Benchmark test by R. Himeno, http://accc.riken.jp/2145.htm

[6] V-Sphere, http://vcad-hpsv.riken.jp/jp/release_software/V-Sphere/

[7] International Commission on Radiation Units and Measurements, Tissue Substitutes, Phantoms and Computational Modelling in Medical Ultrasound, ICRU Report 61, 1998; 43-51.

[8] V-SDFlib, http://vcad-hpsv.riken.jp/jp/release_software/VSDFlib/