

BEM_ALL用户手册

V1.3

[陈天文](#)

1206342642@qq.com

开发环境 **Windows +Qt+ MinGW**

Copyright © 2022 SJTU-NAOCE. All Rights Reserved.

说明

本手册是对BEM_ALL.exe以及BEM_GUI.exe程序的一个说明文档。

- 2023-03-30 —— V1.3:
 - 添加简要地描述了程序相关输入、输出文件的必要说明
- 2023-03-20 —— V1.3:
 - 添加工具函数部分
- 2023-03-18 —— V1.3:
 - 添加测试案例说明

目录

1. 程序简介

1.1 主要功能

1.1.1 频域求解程序: FDC(Frequency Domain Code)

1.1.2 时域求解程序: TDC(Time Domain Code)

1.1.3 工具函数: 可以通过命令行指定参数实现

1.1.4 其他说明:

1.2 框架结构

2. BEM_ALL.exe 使用帮助

2.1 如何运行BEM_ALL.exe软件

2.1.1 运行环境

2.1.2 查看版本信息

2.1.3 获得使用帮助

2.1.4 生成模板文件

2.1.5 如何运行程序

2.1.5.1 使用CMD窗口

2.1.5.2 使用可视化界面

2.2 输入文件有哪些?

2.2.1 频域计算配置模板文件(default_FD.json)

2.2.1.1 设定算例基本信息

2.2.1.1.1 项目名称

2.2.1.1.2 求解器类型

2.2.1.1.3 项目路径

2.2.1.2 设定计算工况

2.2.1.3 设定计算控制参数

2.2.1.3.1 Lua UDF文件定义

2.2.1.3.2 使用Wall Damping Model

2.2.1.4 设定流场监测点

2.2.1.4.1 从文件中读取

2.2.1.4.2 从UDF中读取检测点

2.2.1.5 设定浮体信息(Bodies)

2.2.1.5.1 浮体类型

2.2.1.5.2 浮体构成

2.2.1.5.3 浮体基础信息

2.2.1.5.4 浮体计算控制参数

2.2.1.5.5 浮体面元模型

- 2.2.1.5.6 浮体静力学模型
- 2.2.1.5.7 浮体质量模型
- 2.2.1.5.8 阻尼盖模型
- 2.2.2 时域计算配置模板文件(default_TD.json)
 - 2.2.2.1 设定时域算例基本信息
 - 2.2.2.2 设定时域计算工况
 - 2.2.2.2.1 入射波浪参数
 - 2.2.2.2.2 风载荷
 - 2.2.2.2.3 流载荷
 - 2.2.2.2.4 其他外力
 - 2.2.2.3 设定时域计算控制参数
 - 2.2.2.3.1 "1.Create Test(*)"
 - 2.2.2.3.2 "2.Output Interval(*)"
 - 2.2.2.3.3 "3.RK4 Solver(*)"
 - 2.2.2.3.4 "4.Lua UDF(*)"
 - 2.2.2.4 设定浮体信息
 - 2.2.2.4.1 浮体名称
 - 2.2.2.4.2 浮体频域数据文件夹
 - 2.2.2.5 设定锚点信息(Anchors)
 - 2.2.2.5.1 锚点Anchor属性
 - 2.2.2.6 设定附加连接结构
- 2.2.3 面元文件
 - 2.2.3.1 GDF面元文件
 - 2.2.3.2 msh格式文件
 - 2.2.3.3 pan格式文件
- 2.2.4 UDF文件
 - 2.2.4.1 阻尼系数分布函数(Damping Lid)
 - 2.2.4.2 壁面阻尼系数分布函数(Wall Damping)
 - 2.2.4.3 壁面阻尼区域定义(Wall Damping Range)
 - 2.2.4.3.1 例1: 壁面阻尼系数分布函数
 - 2.2.4.3.2 例2: 壁面阻尼区域范围定义函数
 - 2.2.4.3.3 例3: 壁面阻尼区域范围定义函数(多个区域取并集)
 - 2.2.4.4 监测点定义(OffBodyPoints)
 - 2.2.4.5 时域外力定义(OtherForce)
 - 2.2.4.6 Lua基础的数学运算符
- 2.2.5 系泊系统配置文件
- 2.3 输出文件有哪些?
 - 2.3.1 log文件

- 2. 3. 1. 1 [Brief Summary Check](#)
 - 2. 3. 1. 2 [其他提示信息](#)
 - 2. 3. 2 [频域结果文件](#)
 - 2. 3. 2. 1 [浮体几何信息](#)
 - 2. 3. 2. 2 [波浪力结果](#)
 - 2. 3. 2. 3 [运动结果](#)
 - 2. 3. 2. 4 [水动力系数结果](#)
 - 2. 3. 2. 5 [空间监测点](#)
 - 2. 3. 3 [时域结果文件](#)
 - 2. 3. 3. 1 [时域计算测试文件](#)
 - 2. 4 [如何使用工具函数](#)
 - 2. 4. 1 [处理面元文件](#)
 - 2. 4. 1. 1 [面元文件格式转换](#)
 - 2. 4. 1. 2 [计算面元文件静力学参数](#)
 - 2. 4. 1. 3 [处理面元文件](#)
 - 2. 4. 2 [使用第三方库](#)
 - 2. 4. 2. 1 [C++编写dll](#)
 - 2. 4. 2. 2 [Fortran编写dll](#)
- ### 3. [BEM_GUI.exe使用帮助](#)
- 3. 1 [软件GUI界面](#)
 - 3. 2 [软件基本操作流程:](#)
 - 3. 3 [特殊操作](#)
 - 3. 3. 1 [利用bat执行多个算例](#)
 - 3. 3. 2 [从文件夹中算例数据](#)
- ## 4. [测试算例](#)
- 4. 1 [收敛性及并行测试](#)
 - 4. 1. 1 [硬件条件](#)
 - 4. 1. 2 [计算模型与工况](#)
 - 4. 1. 3 [收敛性与并行测试结果](#)
 - 4. 2 [频域算例](#)
 - 4. 2. 1 [单浮体频域计算\(无限水深\)](#)
 - 4. 2. 1. 1 [test01: Wigley I \(四边形面元\)](#)
 - 4. 2. 1. 2 [test02: Wigley I \(三角形面元\)](#)
 - 4. 2. 1. 3 [test03: Barge \(驳船模型\)](#)
 - 4. 2. 2 [使用内部盖模型](#)
 - 4. 2. 2. 1 [test04: Barge Lid\(单浮体+内部盖\)](#)
 - 4. 2. 3 [多浮体频域计算](#)
 - 4. 2. 3. 1 [test05: Barge-Barge\(双浮体+监测点\)](#)

- 4.2.3.2 test06: Wigley_Box(双浮体+验证)
- 4.2.4 使用阻尼盖模型
 - 4.2.4.1 test07: Barge_Barge Damping Lid(使用阻尼盖)
 - 4.2.4.2 test08: Barge_Barge Damping Lid(使用阻尼盖+Newtonian)
- 4.2.5 使用壁面阻尼模型
 - 4.2.5.1 test09: Barge_Barge Wall_Damping(使用壁面阻尼模型)
- 4.2.6 特殊算例
 - 4.2.6.1 test10: 方形box (检测浮体附近空间波形)
 - test11: 半潜平台(面元数目 10000+)
 - 4.2.6.2 10000+)
 - 4.2.6.3 test12: N浮体频域水动力计算
 - 4.2.6.4 test13: 半球(有限水深)
 - 4.2.6.5 test14: 风机下半平台
 - 4.2.6.6 test15: 不知道怎么描述
 - 4.2.6.7 test16: 月池共振(添加阻尼盖)
- 4.3 时域算例
 - 4.3.1 test01_TD_Wigley_III(白噪声, 时域计算)
 - 4.3.2 test02_TD_Barge_Barge(双浮体, Cable, Fender, Mooring)
- 5. 工具函数
 - 5.1 BEMRosetta进行面元处理
 - 5.2 MATLAB后处理数据
 - 5.2.1 读取波浪力(ReadEXForce.m)
 - 5.2.2 读取水动力系数(ReadHydroCoef.m)
 - 5.2.3 读取监测点结果(ReadOffBodyPoints.m)
 - 5.2.4 读取面元文件
 - 5.2.4.1 读取.txt面元文件
 - 5.2.4.2 读取.gdf面元文件
 - 5.2.5 读取实数, 复数矩阵文件
 - 5.2.5.1 读取实数(.txt)文件(ReadMat.m)
 - 5.2.5.2 读取复数(.txt)文件(ReadMatC.m)
 - 5.2.6 读取系泊系统文件
 - 5.2.6.1 读取系泊缆绳位置信息(readLineOut.m)
 - 5.2.6.2 读取系泊缆绳受力信息(readLineOutForce.m)
 - 5.2.7 时历数据统计分析
 - 5.2.7.1 计算有义值(Cal_Hs.m)
 - 5.2.7.2 谱分析
 - 5.3 生成自由面监测点
- 6. 参考文献

1. 程序简介

1.1 主要功能

1.1.1 频域求解程序：FDC(Frequency Domain Code)

1. 经过算例测试，程序能够正确完成单体、多体频域计算内容
2. 支持添加 `Internal Lid`、`Damping Lid` 模型用于限制不规则频率与抑制间隙共振
3. 支持添加多个Body，Body的数目不设置上限
 - Body可以设置为Floating、Fixed、Internal Lid、Damping Lid四种类型
4. 支持三角形和四边形面元文件
5. 总面元数量受限于计算机运行内存
 - 例如：8000面元，所需运行内存约 10 G
6. OpenMP多线程并行
 - 例如：默认4线程，可提高3倍计算效率
7. 支持在Lua脚本中自定义函数
8. 添加Wall Damping Model，该模型可用于抑制间隙共振而不需要添加阻尼盖面元
9. 支持自定义WallDampingRang以及OffBodyPoints功能
10. 对于FloatBody与FixeBody类型，单独输出FK力
11. 对于PanelModel可以支持msh格式，pan格式文件

1.1.2 时域求解程序：TDC(Time Domain Code)

1. 基于间接时域法，多浮体时域模拟
2. 支持添加多浮体之间连接结构：连接缆绳，防撞垫
3. 当前版本支持添加系泊系统(集中质量法)
4. 支持添加自定义外力

1.1.3 工具函数：可以通过命令行指定参数实现

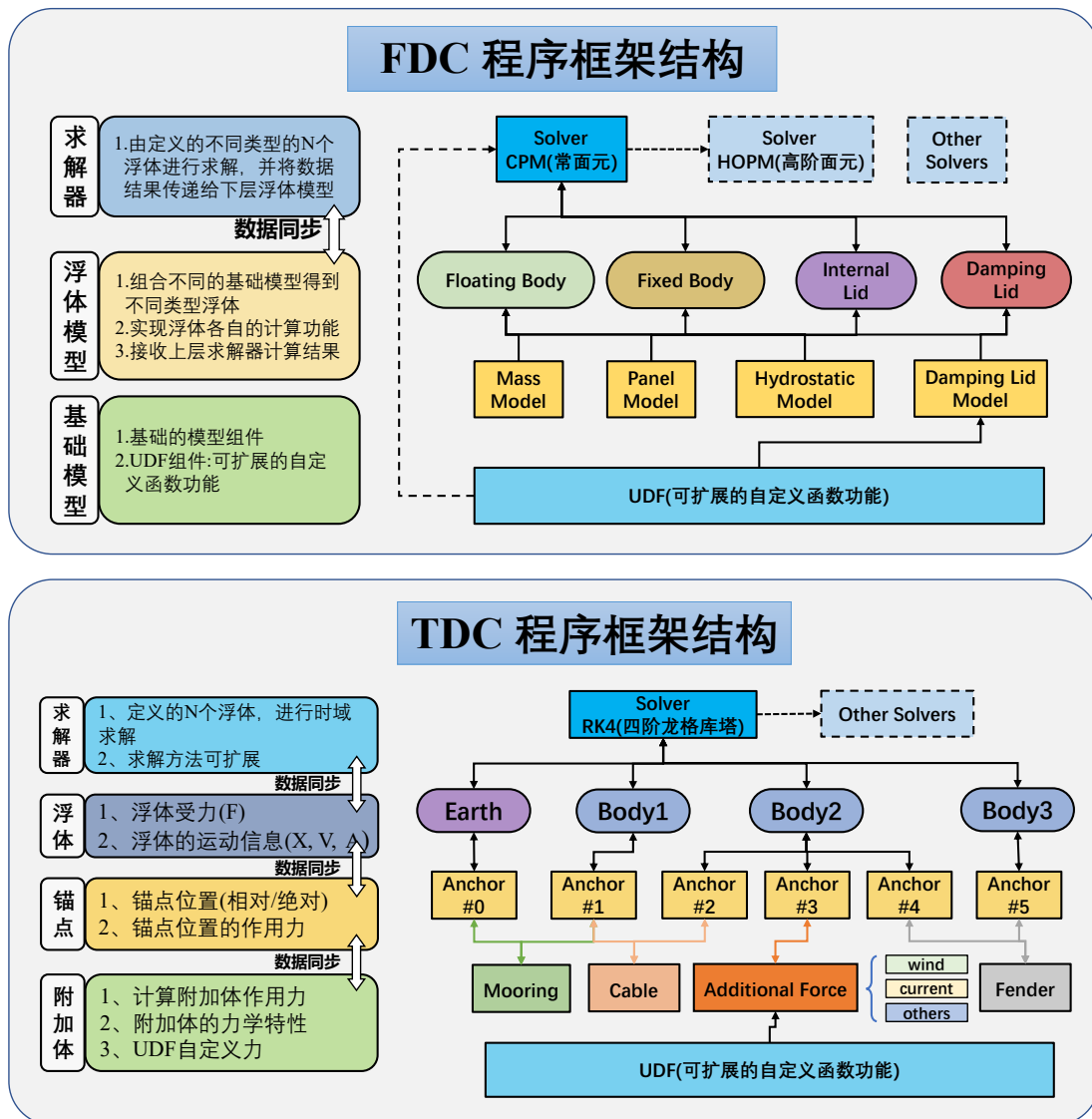
1. 可以单独对单个面元文件(*.gdf, *.msh, *.pan)进行缩放、平移、对称等处理
2. 可以单独对面元文件(*.gdf, *.msh, *.pan)进行法相检查并矫正(还没写完,暂时没用)
3. 可以单独对面元文件(*.gdf, *.msh, *.pan)计算静力学参数
4. 以上执行顺序：`correctNormal` -> `scale` -> `translate` -> `symmetry` -> `calcHydrostaticInfo`
5. 转换面元文件格式 [*.gdf, *.msh, *.pan] -> [*.gdf, *.pan]

1.1.4 其他说明:

1. FDC 至多设置一个Floating Body
2. FDC 当前版本不支持二阶力计算

1.2 框架结构

本程序提供的频域求解模块（**FDC**: **F**requency **D**omain **C**ode）与间接时域模块（**TDC**: **T**ime **D**omain **C**ode）程序，下图分别展示了 **FDC** 与 **TDC** 程序的分层框架结构。清晰的层次结构有助于进行程序后续调试与扩展，本层的模型可以利用下层模型提供的数据与函数接口，实现自身功能，并为上层模型提供函数数据与函数调用接口。



2. BEM_ALL.exe 使用帮助

2.1 如何运行BEM_ALL.exe软件

目前提供两种版本的程序，分别是 `BEM_XXX.exe` 与 `BEM_XXX_OpenBLAS.exe`，两者主要的区别在于后者利用 `OpenBLAS库作为矩阵运算后端` 程序，后者计算效率更高，但并未对其进行全面测试。

2.1.1 运行环境

BEM_ALL.exe软件可在Windows系统上独立运行。

注意：如果需要在Linux系统上运行，则需要添加 `#define LINUX` 宏，重新编译，所使用的第三方库也需要重新编译。

2.1.2 查看版本信息

- 双击程序 (相关的版权信息和必要的说明内容如下图所示)



```

选择 C:\Users\chentianwen\Desktop\Bem_GUI_exe_Simple\example_with_results\BEM_ALL_V2.0.exe
Active code page: 65001

#####
#####          SJTU          #####
#####
#####
Last Compile Time:Mar  8 2023 15:29:53

Press to show the detailsPress any key to continue . . .

#####
##### [ 频域 + 时域 ] 多浮体求解程序 BEM_ALL.exe #####
#####

使用说明: [ BEM_ALL.exe -h ] 获得使用帮助
当前版本: V2.0
联系邮箱: 1206342642@qq.com
版本说明:
频域求解程序: FDC(Frequency Domain Code)
1、经过算例测试, 程序能够正确完成单体、多体频域计算内容
2、支持添加Internal Lid、Damping Lid模型用于限制不规则频率与抑制间隙共振
3、支持添加多个Body, Body的数目不设置上限
   Body可以设置为Floating、Fixed、Internal Lid、Damping Lid四种类型
4、支持三角形和四边形面元文件
5、总面元数量受限于计算机运行内存
   例如: 8000面元, 所需运行内存约 10 G
6、OpenMP多线程并行
   例如: 默认4线程, 可提高3倍计算效率
7、支持在Lua脚本中自定义函数
8、添加Wall Damping Model, 该模型可用于抑制间隙共振而不需要添加阻尼盖面元
9、支持自定义WallDampingRang以及OffBodyPoints功能
10、对于FloatBody与FixeBody类型, 单独输出FK力
11、对于PanelModel可以支持msh格式, pan格式文件

时域求解程序: TDC(Time Domain Code)
1、基于间接时域法, 多浮体时域模拟
2、支持添加多浮体之间连接结构: 连接缆绳, 防撞垫
3、当前版本支持添加系泊系统(集中质量法)
4、支持添加自定义外力
  
```

2.1.3 获得使用帮助

1. 在当前路径打开cmd窗口，或者其他方法自行百度
2. cmd窗口输入 `BEM_ALL_V2.0.exe -h`，显示以下使用帮助

```

C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19045.2604]
(c) Microsoft Corporation。保留所有权利。

C:\Users\chentianwen\Desktop\Bem_GUI_exe_Simple\example_with_results>BEM_ALL_V2.0.exe -h
INFO (2023.03.14 20:23:04) create Log [ default.log ]
INFO (2023.03.14 20:23:04) Current EXE Version: BEM_ALL_V2.0
INFO (2023.03.14 20:23:04)
Example: BEM_ALL.exe -n 8 -j config.json -l my.log

|| INFO (2023.03.14 20:23:04) || USAGE:
BEM_ALL_V2.0.exe [-?|-h|--help] [-n|--nthread <num of threads>] [-j|--json <json input config name>] [-
c|--check] [-e|--example] [-l|--log <generate log File>] [-i|--inputfile <input "*.gdf">] [-o|--outputfil
e <output "*.gdf *.pan">] [--calcHydrostatic] [--correctNormal] [--scale <set the scale factor,default is
1.0>] [--translate <set the translate vector,default is 0,0,0>] [--symmetry <set the symmetry type, 0 =
NULL;1 = AlongX;2 = AlongY;4 = AlongZ;3 = AlongXY,default = 0;>] [--hos] [--moor] [--valid_moor]

Display usage information.

OPTIONS, ARGUMENTS:
-?, -h, --help
-n, --nthread <num of threads>
    Specifies the number of parallel threads, default n=4
-j, --json <json input config name>
    Specifies the name of json config file, defalue j="default.json"
-c, --check
    Only check the calc info
-e, --example
    Generate FD/TD example [ default_FD/TD.json ] file
-l, --log <generate log File>
    Specific log file name, default="default.log"
-i, --inputfile <input "*.gdf">
    input panel file(input "*.gdf")
-o, --outputfile <output "*.gdf *.pan">
    output panel file(output "*.gdf *.pan")
  
```

3. 其中显示了全部的命令行参数，常用的参数如下

参数	含义
<code>-j</code> or <code>--json</code>	指定输入的配置文件(*.json)
<code>-n</code> or <code>--nthread</code>	指定并行线程数(如果不指定，默认为4)
<code>-c</code> or <code>--check</code>	添加此参数，则仅执行检查，不执行计算
<code>-l</code> or <code>--log</code>	指定输出的log文件名称(默认为default.log)
<code>-e</code> or <code>--example</code>	当前路径下，输出模板文件

全部参数

1 USAGE:

```

2   BEM_ALL_V2.0.exe [-?|-h|--help] [-n|--nthread <num of threads>] [-j|--json <json
input config name>] [-c|--check] [-e|--example] [-l|--log <generate log File>] [-i|--
-inputfile <input "*.gdf">] [-o|--outputfile <output "*.gdf *.pan">] [--
calcHydrostatic] [--correctNormal] [--scale <set the scale factor,default is 1.0>]
[--translate <set the translate vector,default is 0,0,0>] [--symmetry <set the
symmetry type, 0 = NULL;1 = AlongX;2 = AlongY;4 = AlongZ;3 = AlongXY;default = 0;>]
[--hos] [--moor] [--valid_moor]

3
4   Display usage information.
5
6   OPTIONS, ARGUMENTS:
7       -?, -h, --help
8       -n, --nthread <num of threads>
9                               Specifies the number of parallel threads, default n=4
10      -j, --json <json input config name>
11                               Specifies the name of json config file, default
j="default.json"
12      -c, --check               Only check the calc info
13      -e, --example             Generate FD/TD example [ default_FD/TD.json ] file
14      -l, --log <generate log File>
15                               Specific log file name, default="default.log"
16      -i, --inputfile <input "*.gdf">
17                               input panel file(input "*.gdf")
18      -o, --outputfile <output "*.gdf *.pan">
19                               output panel file(output "*.gdf,*.pan")
20      --calcHydrostatic         calculate hydrostatic info for panel
21      --correctNormal           correct Normal for the panel
22      --scale <set the scale factor,default is 1.0>
23
24      --translate <set the translate vector,default is 0,0,0>
25
26      --symmetry <set the symmetry type, 0 = NULL;1 = AlongX;2 = AlongY;4 = AlongZ;3 =
AlongXY;default = 0;>
27
28      --hos                     HOS execute( only test )
29      --moor                    Moor Dyn Lib( only test )
30      --valid_moor              Valid for MoorDynLib(single line). Input
File(top_motion.txt,line.cfg)

```

2.1.4 生成模板文件

cmd窗口输入 `BEM_ALL_V2.0.exe -e`，则在当前路径下生成以下文件

```

1   default_FD.json
2   default_TD.json
3   UDF.lua
4   lines.cfg

```

相关文件具体解释可参考：

1. [频域计算配置模板文件\(default_FD.json\)](#)
2. [时域计算配置模板文件\(default_TD.json\)](#)
3. [UDF文件](#)
4. [系泊系统配置文件](#)

2.1.5 如何运行程序

2.1.5.1 使用CMD窗口

在cmd窗口输入以下命令：

```
1 | BEM_ALL_V2.0.exe -n 8 -j test1.json -l test1_folder/test.log
```

以8个线程执行test1.json算例文件，并将显示信息输出至test.log，

注意： test1.json文件中会指定输出文件夹路径，如果使用命令行方式运行，则此文件需要自行创建，如果使用BEM_GUI界面，则不需要自行创建。

2.1.5.2 使用可视化界面

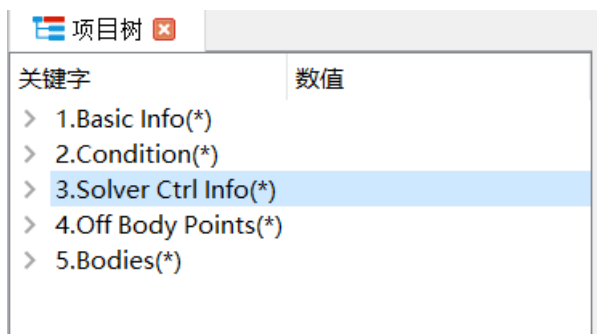
参考 [BEM_GUI.exe使用帮助](#)

2.2 输入文件有哪些？

目前，软件输入文件主要有4类：

1. [算例配置文件\(*.json\)](#)
 - [频域算例配置文件](#)
 - [时域算例配置文件](#)
2. [面元文件\(*.GDF\)](#)
3. [自定义函数文件UDF\(*.lua\)](#)
4. [系泊系统配置文件\(lines\)](#)

配置文件以json格式文件给出，模板文件可以通过指定 `--example` 参数输出模板文件，参考 [生成模板文件](#)。这里我们是用 `BEM_GUI` 中 `项目树窗口` 查看json文件，如下图所示。



注意： 若关键词存在 `(*)` 标记，则表示该关键字下存在必填信息，若不存在，则表示该关键字对应的数值由软件自动生成。

2.2.1 频域计算配置模板文件(default_FD.json)

频域计算的配置文件包含五个部分，同上图。

2.2.1.1 设定算例基本信息

```
1  "1.Basic Info(*)": {
2      "1.Name(*)": "object",
3      "2.Solver Type(*)": {
4          "info": "0=FD_Single;1=FD_MultiBody;2=TD_Single;3=TD_MultiBody",
5          "value": 0
6      },
7      "3.Project Path(*)": "Project/Path"
8  }
```

2.2.1.1.1 项目名称

关键字	类型	数值	说明
"1.Name (*) "	字符串 (string)	"object"	为算例指定一个名称(建议以 Solver_ 为前缀)

注意：部分结果文件将以此为文件名前缀，例如：[offBodyPoints](#)的输出文件便是以此名称为前缀，并且在路径下会额外生成一个以此为名称命名的json文件，其中为当前算例输出的[频域配置文件](#)

2.2.1.1.2 求解器类型

在 "2.Solver Type(*)" -> "value" 关键字下指定不同的求解器类型。

关键字	类型	数值	说明
"2.Solver Type(*) "	整数(int)	0	单浮体频域计算
"2.Solver Type(*) "	整数(int)	1	多浮体频域计算
"2.Solver Type(*) "	整数(int)	2	单浮体时域计算
"2.Solver Type(*) "	整数(int)	3	多浮体时域计算

2.2.1.1.3 项目路径

关键字	类型	数值	说明
"3.Project Path(*)"	字符串(string)	"Project/Path"	指定文件路径

注意：对应文件夹需要自行创建，可以使用相对路径或者绝对路径，路径必须为全英文字符，注意 `\\` 与 `/` 的区别，例如：

```

1  ./case_folder
2  .\\case_folder
3  E:\\FDC\\case_folder
4  E:/FDC/case_folder

```

2.2.1.2 设定计算工况

```

1  "2.Condition(*)": {
2      "1.Rho(*)": 1000.0,
3      "2.Water Depth(*)": {
4          "info": "-1 for infinite water depth",
5          "value": -1.0
6      },
7      "3.Frequency(*)": "1.000000:1.000000:2.000000",
8      "4.Direction(*)": "0.000000:45.000000:90.000000"
9  }

```

关键字	类型	数值	说明
"1.Rho(*)"	浮点数 (double)	1000.0	流体密度，默认为1000 kg/m^2
"2.Water Depth(*)"	浮点数 (double)	-1.0	水深条件，若为无限水深，则设定为-1.0(实际计算时对应为10000m)
"3.Frequency(*)"	字符串 (string)	"1.00:1.00:2.00"	最小值:步长:最大值，单位为 rad/s
"4.Direction(*)"	字符串 (string)	"0.00:45.00:90.00"	最小值:步长:最大值，单位为 deg

可以在log文件中查看相关参数，例如：

```

1  || INFO (2023.03.14 17:33:06) ||
   *****
2  || INFO (2023.03.14 17:33:06) || ***** FDC::Condition::showInfo()
   *****
3  || INFO (2023.03.14 17:33:06) ||
   *****
4  || INFO (2023.03.14 17:33:06) || G =9.81
5  || INFO (2023.03.14 17:33:06) || Rho =1000
6  || INFO (2023.03.14 17:33:06) || Water Depth = 10000
7  || INFO (2023.03.14 17:33:06) ||
8  || INFO (2023.03.14 17:33:06) || Frequency Set(rad/s). count = 2
9  || INFO (2023.03.14 17:33:06) ||      1.000    2.000
10 || INFO (2023.03.14 17:33:06) ||
11 || INFO (2023.03.14 17:33:06) || Direction Set(deg). count = 3
12 || INFO (2023.03.14 17:33:06) ||      0.000    45.000    90.000
13 || INFO (2023.03.14 17:33:06) ||

```

2.2.1.3 设定计算控制参数

```

1  "3.Solver Ctrl Info(*)": {
2      "1.Lua UDF(*)": {
3          "info": "the path to lua script",
4          "value": "UDF.lua"
5      },
6      "2.Wall Damping Model(*)": {
7          "1.UDF Func Name": "",
8          "2.Range": {
9              "info": "[[x_min,y_min,z_min],[x_max,y_max,z_max]]",
10             "value": "[[0.000000,0.000000,0.000000],[0.000000,0.000000,0.000000]]"
11         },
12         "3.Range From UDF": {
13             "UDF Func Name": "",
14             "info": "define the range in UDF;in range \"return true\",else \"return
false\";"
15         }
16     }
17 }

```

2.2.1.3.1 Lua UDF文件定义

关键字	类型	数值	说明
"1.Lua UDF(*)"	字符串 (string)	json	设定 Lua UDF 文件路径，一般放在当前目录下，Lua中写入自定义函数，若是 <code>value</code> 为空字符串，则不使用UDF功能，在 log文件: Brief Summary 中有体现。
"2.Wall Damping Model(*)"	字符串 (string)	json	定义壁面阻尼模型 Wall Damping Model 相关参数

2.2.1.3.2 使用Wall Damping Model

相关算例：[test09: Barge Barge Wall Damping\(使用壁面阻尼模型\)](#)

Wall Damping Model 相关参数如下：

```
1  "2.Wall Damping Model(*)": {
2      "1.UDF Func Name": "",
3      "2.Range": {
4          "info": "[[x_min,y_min,z_min],[x_max,y_max,z_max]]",
5          "value": "[[0.000000,0.000000,0.000000],[0.000000,0.000000,0.000000]]"
6      },
7      "3.Range From UDF": {
8          "UDF Func Name": "",
9          "info": "define the range in UDF;in range \"return true\",else \"return
10         false\";"
11     }
12 }
```

注意：

- 1. Wall Damping Model 当且仅当定义 "1.Lua UDF(*)" 时才会有效，在 [查看log文件信息](#) 中存在提示信息，查看计算中是否使用 Wall Damping Model 。
- 2. 若计算中使用 Wall Damping Model ，则在 [项目文件路径](#) 下生成 Wall_Damping_Panel.gdf 与 Wall_Damping_Panel.txt 文件，用于检查设定壁面阻尼区域。

关键字	类型	说明
"1.UDF Func Name":	字符串 (string)	指定壁面阻尼系数分布函数(UDF_Wall_Damping_mu(x, y, z, omega))，该函数存在于 上一节 中指定的lua文件中)， 注意：函数参数 (x, y, z, omega) 形式固定，函数名 UDF_Wall_Damping_mu 可自定义，如 例1 所示。
"2.Range"	字符串 (string)	定义一个三维空间区域： "[[x_min,y_min,z_min],[x_max,y_max,z_max]]" ，在此区域内的面元将会按照 UDF_Wall_Damping_mu 计算阻尼系数。
"3.Range From UDF"	字符串 (string)	在UDF中定义函数，此方法作为 "2.Range" 的扩充，可以在UDF中编写更复杂的自定义函数， 例2 与 例3 给出了两个函数实例。

2.2.1.4 设定流场监测点

设定流场监测点，可以监测点位置的速度势以及波面升高，对应的配置文件如下所示。

```
1  "4.Off Body Points(*)": {
2      "1.Off Body Points Num": 0,
3      "2.Off Body Points filepath(*)": "",
4      "3.Get Points From UDF": {
5          "1.UDF Name": "",
6          "info": "The program will preferentially load points from Lua. If you don't
7              want to do this, leave [UDF Name] blank "
8      }
9  }
```

关键字	类型	说明
"1.Off Body Points Num"	整数(int)	不需要填写，程序自动生成
"2.Off Body Points filepath(*)"	字符串(string)	从txt文件中读取监测点坐标
"3.Get Points From UDF"	字符串(string)	从UDF中读取监测点坐标

可以在log文件中查看相关参数，如下

```
1  || INFO (2023.03.10 12:31:57) || *****
2  || INFO (2023.03.10 12:31:57) || ***** FDC::OffBodyPoints::showInfo() *****
3  || INFO (2023.03.10 12:31:57) || *****
4  || INFO (2023.03.10 12:31:57) || OffBody Points Num = 21
5  || INFO (2023.03.10 12:31:57) || 1 ( 2.000, 0.000, 0.000)
6  || INFO (2023.03.10 12:31:57) || 2 ( 1.800, 0.000, 0.000)
7  || INFO (2023.03.10 12:31:57) || 3 ( 1.600, 0.000, 0.000)
8  || INFO (2023.03.10 12:31:57) || 4 ( 1.400, 0.000, 0.000)
9  || INFO (2023.03.10 12:31:57) || 5 ( 1.200, 0.000, 0.000)
10 || INFO (2023.03.10 12:31:57) || 6 ( 1.000, 0.000, 0.000)
11 || INFO (2023.03.10 12:31:57) || 7 ( 0.800, 0.000, 0.000)
12 || INFO (2023.03.10 12:31:57) || 8 ( 0.600, 0.000, 0.000)
13 || INFO (2023.03.10 12:31:57) || 9 ( 0.400, 0.000, 0.000)
14 || INFO (2023.03.10 12:31:57) || 10 ( 0.200, 0.000, 0.000)
15 || INFO (2023.03.10 12:31:57) || 11 ( 0.000, 0.000, 0.000)
16 || INFO (2023.03.10 12:31:57) || 12 (-0.200, 0.000, 0.000)
17 || INFO (2023.03.10 12:31:57) || 13 (-0.400, 0.000, 0.000)
18 || INFO (2023.03.10 12:31:57) || 14 (-0.600, 0.000, 0.000)
19 || INFO (2023.03.10 12:31:57) || 15 (-0.800, 0.000, 0.000)
20 || INFO (2023.03.10 12:31:57) || 16 (-1.000, 0.000, 0.000)
21 || INFO (2023.03.10 12:31:57) || 17 (-1.200, 0.000, 0.000)
22 || INFO (2023.03.10 12:31:57) || 18 (-1.400, 0.000, 0.000)
23 || INFO (2023.03.10 12:31:57) || 19 (-1.600, 0.000, 0.000)
24 || INFO (2023.03.10 12:31:57) || 20 (-1.800, 0.000, 0.000)
25 || INFO (2023.03.10 12:31:57) ||
26 || INFO (2023.03.10 12:31:57) || only show the first 20 points
```

可以通过两种方式定义检测点空间坐标：

2.2.1.4.1 从文件中读取

编写如下格式的txt文件

Line	x	y	z
1	21		
2	2.000	0.000	0.000
3	1.800	0.000	0.000
4	1.600	0.000	0.000
5	1.400	0.000	0.000
6	1.200	0.000	0.000
7	1.000	0.000	0.000
8	0.800	0.000	0.000
9	0.600	0.000	0.000
10	0.400	0.000	0.000
11	0.200	0.000	0.000
12	0.000	0.000	0.000
13	-0.200	0.000	0.000
14	-0.400	0.000	0.000
15	-0.600	0.000	0.000
16	-0.800	0.000	0.000
17	-1.000	0.000	0.000
18	-1.200	0.000	0.000
19	-1.400	0.000	0.000
20	-1.600	0.000	0.000
21	-1.800	0.000	0.000
22	-2.000	0.000	0.000

第一行为点数目，后续为空间点坐标 (x y z)，以空格作为分隔符。

可以采用 [工具函数脚本](#)，生成更为复杂的空间监测点。

2.2.1.4.2 从UDF中读取检测点

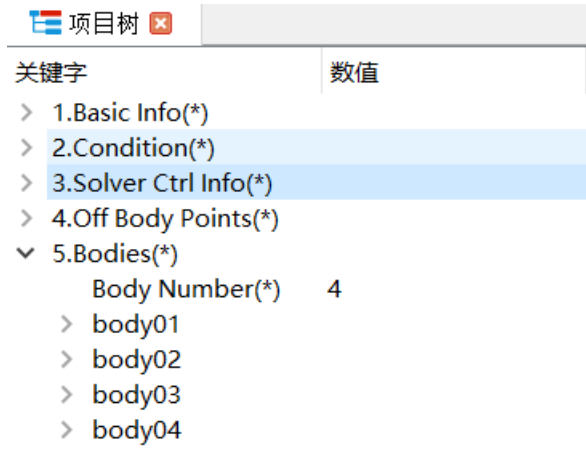
填写 "3.Get Points From UDF" -> "1.UDF Name" 对应的函数名称，例如：[监测点定义 \(OffBodyPoints\)](#)。

注意：

1. 这种方式的优先级高于" [从文件中读取](#) "，"1.UDF Name" 对应的名称为空字符串或者未UDF文件中找到相关函数，则不能由UDF中读取监测点。
2. 在 [log文件: Brief Summary](#) 中有关键字 Using OffBodyPoints UDF or not ，可通过其确定是成功从UDF中读取监测点。

2.2.1.5 设定浮体信息(Bodies)

本程序支持N浮体频域计算，当前程序最大浮体数目设定为99，以下为模板(default_FD.json)中的 "5.Bodies(*)" 。



计算时在CMD窗口，或者在log文件中会显示当前算例计算的浮体相关信息，如下：

```

1  || INFO || *****
2  || INFO || ***** FDC::Solver_FD::showSolverInfo() *****
3  || INFO || *****
4  || INFO || Bodies Info. count = 5
5  || INFO ||      #      Body Name      Panel Num      Body Type
6  || INFO ||      1      Barge1      836      FixedBody
7  || INFO ||      2      Barge2      836      FixedBody
8  || INFO ||      3      Internal_Lid1      200      Internal Lid
9  || INFO ||      4      Internal_Lid2      200      Internal Lid
10 || INFO ||      5      Damping Lid      400      Damping
    Lid(mu=0.000000|Uniform)

```

2.2.1.5.1 浮体类型

首先指定 "Body Number(*)" 指定浮体数目，目前程序中提供四种浮体模型，分别是

- FloatBody：浮体
- FixedBody：固定浮体
- InternalLid：内部盖(用于消除不规则频率)
- DampingLid：阻尼盖(用于抑制过大的间隙共振响应)

各个类型的浮体会输出不同的结果文件。

体类型	波浪力	附加质量	阻尼系数	面元文件	水线面文件	静力学参数文件
FloatBody	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
FixedBody	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
InternalLid				<input checked="" type="checkbox"/>		
DampingLid				<input checked="" type="checkbox"/>		

2.2.1.5.2 浮体构成

参考 [程序框架结构](#)，浮体由基础的模型构成，基础模型包括：

- **"1.Basic Info(*)"**：浮体的基本信息，名称以及类型等
- **"2.Calculate Info(*)"**：
- **"3.Panel Model(*)"**：
- **"4.HydroStatic Model"**：
- **"5.Mass Model(*)"**：
- **"5.Damping Lid Model"**：

以下表格显示了不同类型浮体的不同构成。

体类型	"1.Basic Info(*)"	"2.Calculate Info(*)"	"3.Panel Model(*)"	"4.HydroStatic Model"	"5.Mass Model(*)"	"5.Damping Lid Model"
FloatBody	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
FixedBody	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
InternalLid	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
DampingLid	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>

2.2.1.5.3 浮体基础信息

指定**浮体名称**，与 [浮体类型](#)

```

1  "1.Basic Info(*)": {
2      "1.Name(*)": "FloatBody",
3      "2.Type(*)": {
4          "info": "0=FloatBody;1=FixedBody;2=InternalLid;3=DampingLid",
5          "value": 0
6      }
7  }

```

注意：

1. 浮体相关结果文件均是以浮体名称为前缀，例如：

BodyA 受到的波浪干扰力，则结果文件为：BodyA_EXForce.txt

2. 对于多个浮体而言，存在浮体之间的相关干扰作用，例如：

BodyA 运动对自身水动力系数的影响，则结果文件为：BodyA-

BodyA_AddMass.txt

BodyA 运动对 BodyB 水动力系数的影响，则结果文件为：BodyA-

BodyB_AddMass.txt

2.2.1.5.4 浮体计算控制参数

指定计算曲面积分时的计算精度，对于四边形面元，使用高斯积分进行计算，Gauss N可以指定为 2 或者 4，详细细节可[参考博客](#)；对于三角形面元，使用高斯积分进行计算，Tri Rule建议指定为 7、10 或者 11，详细细节可[参考博客](#)；

```

1  "2.Calculate Info(*)": {
2      "0.info": "Gauss N = 2 or 4, Tri rule = 7,10,11",
3      "1.Gauss N(*)": 4,
4      "2.Tri Rule(*)": 7
5  }

```

2.2.1.5.5 浮体面元模型

定义浮体的面元模型，其中，存在 (*) 号标记项的为必填项目。

```

1  "3.Panel Model(*)": {
2      "1.Panel File Path(*)": "Panel/File/Path/FloatBody.GDF",
3      "2.Patch Num": 0,
4      "3.Panel Type": "PanelType::Quadrangle",
5      "4.Ns": 4,
6      "5.Mesh(*)": {
7          "1.Scale(*)": 1.0,
8          "2.Translate(*)": "[0.000000,0.000000,0.000000]",
9          "3.Rotate(*)": "[0.000000,0.000000,0.000000]",
10         "4.Symmetry(*)": {
11             "info": "0=NULL;1=AlongX;2=AlongY;3=AlongXY;4=AlongZ",
12             "value": 0
13         }
14     },

```

```

15     "6.Coord": "[0.000000,0.000000,0.000000]"
16 }

```

"1.Panel File Path(*)": 指定面元文件路径, 建议使用相对路径, 目前支持生成的 (***.GDF** ***.pan** ***.msh** (GAMBIT)) 文件, 但是面元文件需要提前进行法向矫正。可以利用 [BEMRosetta.exe](#) 软件进行面元前处理操作。所以推荐使用GDF格式面元文件进行计算, 对于 GAMBIT 划分导出的 **msh** 格式面元, 可以使用本程序进行格式转换, 参考 [这一节](#)

"5.Mesh(*)": 为节点下可以对面元模型进行各项操作, 包括 **缩放 (Scale: 以坐标原点为基准)**, **平移 (Translate)**, **旋转 (Rotate: 不建议使用, 可能有bug)**, **对称 (Symmetry: 以坐标原点为基准)**, 按照顺序执行。

"6.Coord": 此项参数为局部坐标系原点在大地坐标系下的位置, 不需要填写。

2.2.1.5.6 浮体静力学模型

静力学模型是根据面元模型计算得到的静力学参数 (大地坐标系下), 由上到下分别为

1. 模型空间范围, 即三个坐标值的最大值, 最小值 m
2. 湿表面积 m^2
3. 水线面积 m^2
4. 体积 m^3
5. 浮心坐标 m
6. 稳性半径 m
7. 水线面形心(几何中心, 又称漂心) m

以下取自 [test01](#) 算例中, 程序计算Wigley I模型的静力学参数。

```

1  "4.HydroStatic Model": {
2      "1.Dimensions": {
3          "info": "From (x,y,z) to (x,y,z)",
4          "value": "[-1.000000,-0.150000,-0.125122] to [1.000000,0.150000,0.000000]"
5      },
6      "2.Swet(m^2)": 0.7502960635822891,
7      "3.Aw(m^2)": 0.41480239999999985,
8      "4.Volume(m^3)": 0.04168037133857894,
9      "5.COB(xb,yb,zb)": "[0.000000,0.000000,-0.053496]",
10     "6.BM(rxx,ryy,rzz)": "[0.052090,2.073846,0.000000]",
11     "7.Xf(xf,yf)": "[0.000000,-0.000000]"
12 }

```

注意: 如果结果存在明显错误, 请检查面元法向。

2.2.1.5.7 浮体质量模型

只有 `FloatBody` 才需要添加质量模型，质量模型包含计算浮体运动所需的相关参数，由上到下分别为：

1. 质量 kg
2. 惯性半径 m
3. 重心 m
4. 粘性阻尼系数 $\mu * \dot{X}$
5. 人工弹簧固有周期
6. 开放的自由度， $0 = fixed, 1 = free$

以下取自 [test01](#) 算例中，程序计算Wigley I模型的质量模型。

```

1  "5.Mass Model(*)": {
2      "1.Mass(kg) (*)": 40.9,
3      "2.Inertia Radius(m) (*)": "[0.100000,0.500000,0.500000]",
4      "3.COG(xg,yg,zg) (*)": "[0.000000,0.000000,0.000000]",
5      "4.Viscous_Damping(*)": {
6          "info": "viscous damping for 6 dof,mu*X_dot",
7          "value": "[0.000000,0.000000,0.000000,0.000000,0.000000,0.000000]"
8      },
9      "5.T of Spring(s) (*)": {
10         "info": "Period of Spring for (surge,sway,yaw)",
11         "value": "[100.000000,50.000000,100.000000]"
12     },
13     "6.Degree of Freedom(*)": {
14         "info": "[surge,sway,heave,roll,pitch,yaw];0=fixed;1 for free;",
15         "value": "[1,1,1,1,1,1]"
16     }
17 }

```

注意： 目前程序只支持设定一个浮体。

2.2.1.5.8 阻尼盖模型

相关算例：

- [test07: Barge_Barge Damping Lid\(使用阻尼盖\)](#)
- [test08: Barge_Barge Damping Lid\(使用阻尼盖+Newtonian\)](#)

阻尼盖模型是一种常用的自由面阻尼修正方法，目前被广泛应用于WADAM与Hydrostar中，理论基础详见参考文献。


```

1  "5.Damping Lid Model": {
2      "1.Damping Lid Type(*)": {
3          "info": "0=Uniform;1=NonUniform;2=Newtonian;3=XB_Chen;4=UDF",
4          "value": 0
5      },
6      "2.Damping Lid Coeff(*)": 0,
7      "3.UDF": {
8          "Func Name In Lua(*)": "",
9          "info": "this is the UDF Func Name which should be found in Lua Script"
10     }
11 }

```

对于阻尼盖模型而言，最重要的是定义阻尼系数，不同 **阻尼系数** μ_0 形式如下表所示

阻尼盖类型	对应数值	阻尼形式	说明
Uniform	0	$\frac{\partial \phi}{\partial z} = \frac{(\omega + i\mu)^2}{g} \phi, \mu = \mu_0$	
NonUniform	1	$\frac{\partial \phi}{\partial z} = \frac{(\omega + i\mu)^2}{g} \phi, \mu = \mu_0 f(\omega) g(x)$	仅测试
Newtonian	2	$\frac{\partial \phi}{\partial z} = k(1 + i\varepsilon)^2 \phi, \varepsilon = \mu_0 / \omega$	WADAM
XB_Chen	3	$\frac{\partial \phi}{\partial z} = \frac{(\omega^2 + i\omega\mu)}{g} \phi, \mu = \mu_0$	Hydrostar
UDF	4	$\frac{\partial \phi}{\partial z} = \frac{(\omega + i\mu)^2}{g} \phi, \mu = UDF(x, y, z, \omega)$	自定义

注意：

1. 阻尼系数为公式中对应的 μ_0 ，UDF的优先级最高，**"Func Name In Lua(*)"** 指定为空字符串，则表示不使用UDF，以下给出了一个UDF函数实例。
2. 在log文件中显示相关的阻尼参数

```

1  | 5      Damping Lid      400      Damping
    | Lid(mu=0.000000|Uniform)

```

```

1  -- UDF Function For Damping Lid Model
2  function UDF_Damping_Lid_mu(x, y, omega)
3      -- write code
4      a = 2.45e-1
5      b = 2.8e-1
6      mu = 0.0175 * a * (math.exp(b * omega) - 1)
7      mu = mu * 2
8      return mu
9  end

```

2.2.2 时域计算配置模板文件(default_TD.json)

时域计算采用间接时域法，在上述频域计算结果上进行展开，计算单浮体、多浮体在不规则波浪下的运动时历，同时考虑系泊系统、连接系统等结构。

时域计算的配置文件包含六个部分。



2.2.2.1 设定时域算例基本信息

"1.Basic Info(*)"同[频域算例基本信息](#)

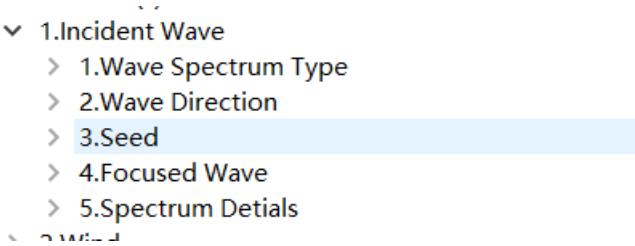
2.2.2.2 设定时域计算工况

设定时域计算工况信息，工况包好四个部分，如下图所示，分别是：

- 1. ["1.Incident Wave"](#)：入射波浪相关参数
- 2. ["2.Wind"](#)：风载荷
- 3. ["3.Current"](#)：流载荷
- 4. ["4.Other Force"](#)：其他外力

2.2.2.2.1 入射波浪参数

入射波浪包含五个部分，如下图所示：



- 入射波浪类型

波浪类型	数值	谱密度函数	说明
------	----	-------	----

波浪类型	数值	谱密度函数	说明
RegularWave	0	输入波高 $H(m)$, 周期 $T(s)$	
Jonswap	1		见参考文献
ITTC	2	$S(\omega) = \frac{173H_{1/3}^2}{T_1^4\omega^5} e^{-691/(T_1^4\omega^4)}$	
P-M	3	$S(\omega) = \frac{8.1 \times 10^{-3} g^2}{\omega^5} e^{-3.11/(H_{1/3}^2\omega^4)}$	
WhiteNoise	4		见参考文献
Gauss	5	$S(f) = \frac{H_s^2}{16} \frac{1}{\sqrt{2\pi}\delta^2} e^{-\frac{(f-f_p)^2}{2\delta^2}}$	
UserWaveEta	6	自定义波浪序列	目前没用

- 指定浪向

浪向与频域结果相关，为频域结果对应浪向的索引

注意：浪向索引从 0 开始，例如：频域结果浪向为 0° , 45° , 90° ，则对应索引为0, 1, 2

```

1  "2.Wave Direction": {
2    "info": "index of wave direction for FreDomain results (interger)",
3    "value": 0
4  }
```

- 指定随机数种子

Seed 为整数，不同的随机数种子对应着不同的随机相位（对于不规则波浪而言）

- 设定聚焦波

指定 **"Is Focused Wave"** 关键字对应数值为 **true**，则设定不规则波浪成分相位，使得其在 **"[x0,t0] (m,s)"** 时间和位置达到最大值。

```

1  "4.Focused Wave": {
2    "Is Focused Wave": false,
3    "Position and Time": {
4      "info": "[x0,t0] (m,s)",
5      "value": "[0.000000,1000.000000]"
6    }
7  }
```

- "5.Spectrum Detials"**

其中包含不同波浪类型对应的参数，自行填写

- ▼ 5.Spectrum Detials
 - > 1.For Regular Wave
 - > 2.For JONSWAP
 - > 3.For ITTC
 - > 4.For PM
 - > 5.For WhiteNoise
 - > 6.For Gauss
 - > 7.User Wave Eta
- ▼ 2.Wind

以ITTC谱为例：以下展示了ITTC谱所需要指定的基本参数。

```

1  "3.For ITTC": {
2      "1.Range": {
3          "info": "frequency range",
4          "value": "[0.000000,0.000000]"
5      },
6      "2.Num": {
7          "info": "Number of frequency",
8          "value": 100
9      },
10     "3.Hs": {
11         "info": "Significat Wave Height(m)",
12         "value": 0.0
13     },
14     "4.T1": {
15         "info": "Character Period(s)",
16         "value": 0.0
17     },
18     "5.Tp": {
19         "info": "Peak Period(s)",
20         "value": 0.0
21     }
22 }
```

注意：这里的波谱分割按照是按照等间距圆频率分割波谱，如果想要按照等间距波数分割，则需要定义 `#define BaseOnK` 宏，再编译。

2.2.2.2.2 风载荷

以下为风载荷作用流程：

```

1  "2.Wind": {
2      "Wind Force Num": 2,
3      "wind01": {
4          "1.ForceType(*)": {
5              "info": "0=NO_Force;1=UDF",
6              "value": 0
7          },
8          "2.OnWhichAnchor": {
9              "info": "specific the anchor name which the wind force on",
10             "value": "anchor_name"
11         },
12         "3.Force Detials": {
13             "1.For UDF": {
14                 "info": "this is the UDF Fun ... be found in Lua Script",
15                 "value": "UDF_Function_Name"

```

```

16         }
17     }
18 },
19 "wind02": {
20     ...

```

□ 外载荷作用流程：

1. 在Lua脚本中定义外力函数

例如：定义了一个名为 `UDF_Other_Force(time, dir)`，`time`为时间，单位为s，返回 (F_x, F_y, F_z) ，大地坐标系下，三个方向外力(可随时间、浪向变化)。

```

1  -- UDF Function For Other Forces
2  function UDF_Other_Force(time, dir)
3      -- write code
4      -- time(s) dir(int index)
5      Fx = 100.*math.sin(0.1*time)
6      Fy = 0
7      Fz = 0
8      return {Fx, Fy, Fz}
9  end

```

2. 在 `"3.Force Detials"` -> `"1.For UDF"` -> `"value"` 中填写函数名称。

3. 在 `"1.ForceType(*)"` 节点下指定 `1`，使用UDF。

4. 在 `"2.OnWhichAnchor"` 节点中指定作用 [锚点](#)

2.2.2.2.3 流载荷

同 [风载荷定义](#)

2.2.2.2.4 其他外力

用于定义额外自己输入的外力

同 [风载荷定义](#)

2.2.2.3 设定时域计算控制参数

控制参数中定义相关的控制参数

```

1  "3.Solver Ctrl Info(*)": {
2      "1.Create Test(*)": true,
3      "2.Output Interval(*)": {
4          "info": "Write File Every [ N ] Step",
5          "value": 500
6      },
7      "3.RK4 Solver(*)": {
8          "1.Time Step(s)": 0.01,
9          "2.Time Max(s)": 50.0,
10         "3.Ramping Time(s)": 10.0
11     },

```

```

12     "4.Lua UDF(*)": {
13         "info": "the path to lua script",
14         "value": "UDF.lua"
15     }
16 }

```

2.2.2.3.1 "1.Create Test(*)"

指定 `"1.Create Test(*)"` 关键字为 `true` 或者 `false`，如果指定为`true`，则在项目路径下生成测试文件，详见：[时域计算测试文件说明](#)

注意：生成测试文件会有点耗时，与频率间隔以及时间步长相关

2.2.2.3.2 "2.Output Interval(*)"

指定 `"2.Output Interval(*)"` 表示写入数据文件的时间步间隔，例如指定步数为 `500`，则表示每隔500个时间步输出数据至文件。

注意：频繁地打开、写入数据文件至文件比较耗时，本程序内部会设置一个暂存容器，容器大小即为 `"2.Output Interval(*)"` 指定的数值，当容器存满数据时再写入至文件。

2.2.2.3.3 "3.RK4 Solver(*)"

时域计算采用4阶数龙格库塔法求解二阶微分方程组，相关理论基础参考博客：[数值计算：四阶龙格-库塔法 for 二阶微分方程](#)

```

1  "3.RK4 Solver(*)": {
2      "1.Time Step(s)": 0.01,
3      "2.Time Max(s)": 50.0,
4      "3.Ramping Time(s)": 10.0
5  }

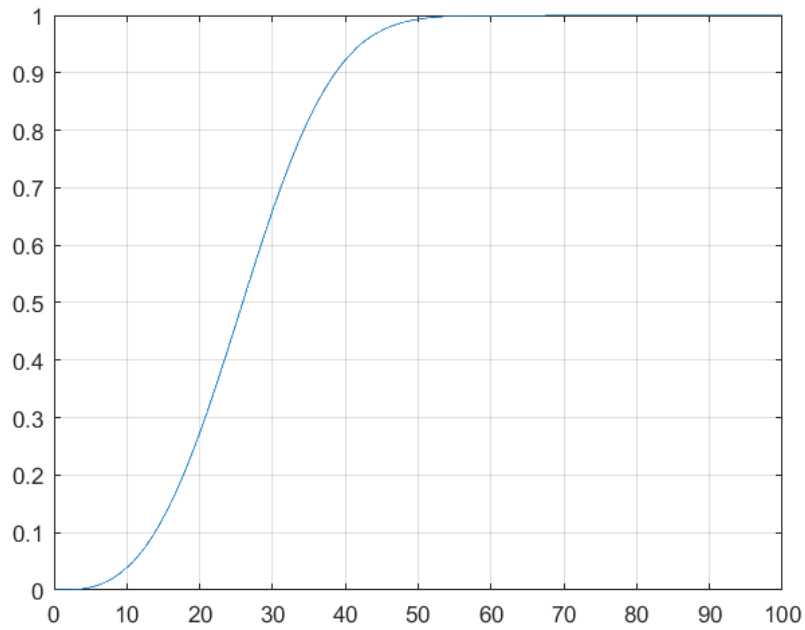
```

本程序中需要设定3个参数，分别是：

- `"1.Time Step(s)"`：时间步大小
- `"2.Time Max(s)"`：最大求解时间
- `"3.Ramping Time(s)"`：缓冲时间，为了避免产生较大的初始运动幅值，设定缓冲函数为以下形式，设定缓冲时间 t_r ，使得在 $[0, t_r]$ 时间内，作用力缓慢增加，使得作用在物体上的外力缓慢增加， $e^{-5} = 0.0067$

$$f_R(t) = 1 - e^{-5\left(\frac{t}{t_r}\right)^3} \quad (1)$$

函数图形如下： $t_r = 50s$



2.2.2.3.4 "4.Lua UDF(*)"

与类似[Lua UDF文件定义](#)。

2.2.2.4 设定浮体信息

浮体信息参数定义如下：

```

1  "4.Bodies(*)": {
2      "Body Number(*)": 3,
3      "body01": {
4          "1.Name": "barge1",
5          "2.Body Folder Name": "body1"
6      },
7      "body02": {
8          "1.Name": "barge2",
9          "2.Body Folder Name": "body2"
10     },
11     "body03": {
12         "1.Name": "barge3",
13         "2.Body Folder Name": "body3"
14     }
15 }

```

时域计算在频域结果基础上展开，因此，需要在 `"4.Bodies(*)"` 节点下定义浮体的详细信息：包括

2.2.2.4.1 浮体名称

为浮体指定一个名称，后续的[锚点定义](#)通过[浮体名称](#)进行关联。

参考算例：[test01_TD_Wigley_III\(白噪声，时域计算\)](#)

2.2.2.4.2 浮体频域数据文件夹

在 **"2.Body Folder Name"** 节点下指定频域结果的文件路径：

注意：控制浮体的自由度，可以在频域结果文件夹下的 ***_HydroStatic.json*** 中修改

```
1  "6.Degree of Freedom(*)": {
2    "info": "[surge,sway,heave,roll,pitch,yaw];0=fixed;1 for free;",
3    "value": "[1,1,1,1,1,1]"
4  }
```

2.2.2.5 设定锚点信息(Anchors)

定义 **"Anchors Number(*)"** 数量。

```
1  "5.Anchors(*)": {
2    "Anchors Number(*)": 3,
3    "anchor01": {
4      "1.Anchor Name(*)": "an1",
5      "2.AnchorType": {
6        "info": "0 = OnEarth(Fixed);1 = OnBody(Floating Body)",
7        "value": 1
8      },
9      "3.Parent(*)": "bargel",
10     "4.Position(*)": {
11       "info": "position on body local coord",
12       "value": "[0.000000,0.000000,0.000000]"
13     }
14   },
15   "anchor02": {
16     ...
17 }
```

2.2.2.5.1 锚点Anchor属性

"1.Anchor Name(*)"

为当前的锚点指定一个名称，后续均需要通过名称进行关联。

"2.AnchorType"

AnchorType指定	数值	坐标系	说明
OnEarth(Fixed)	0	大地坐标系	
OnBody(Floating Body)	1	局部坐标系	表示与浮体坐标系进行关联

"3.Parent(*)"

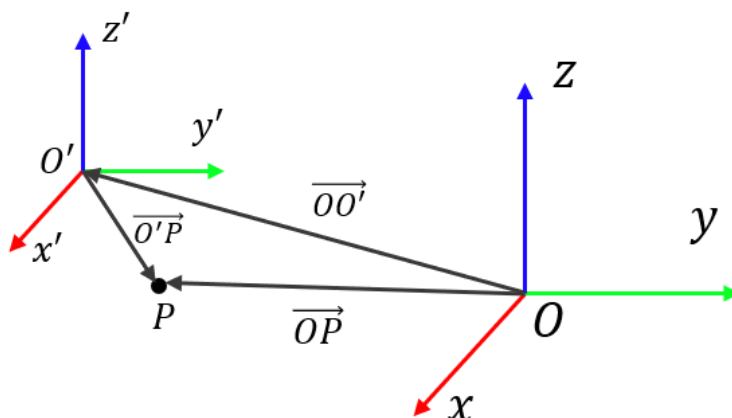
当且仅当指定 "2.AnchorType" 指定为 OnBody 时才会有效, 指定为浮体名称, 浮体名称必须为浮体中对应的名称, 参考: [设定浮体信息](#)

当且仅当指定 "2.AnchorType" 指定为 OnEarth 时, "3.Parent(*)" 建议指定为 "Earth" 虽然没啥用。

"4.Position(*)"

- 如果定义父亲节点为Earth, position为大地坐标系坐标
- 如果定义父亲节点 [浮体名称](#), position为局部坐标

局部坐标系 $O' - x'y'z'$ 和大地坐标系 $O - xyz$ 存在以下关系



物体在 $O' - x'y'z'$ 存在相对位移 $\vec{X}(x, y, z)$, 转动 $\vec{\theta}(\alpha, \beta, \gamma)$ 相对位置关系, 如下:

$$\vec{OP} = \vec{OO'} + \vec{O'P} \quad (2)$$

2.2.2.6 设定附加连接结构

"6.Additional Bodies(*)" 节点下指定附加的连接结构, 包括以下三类:

1. 连接缆绳(Cables)
2. 防撞垫(Fenders)
3. 系泊系统(Moorings)
4. 系泊系统第三方库MoorDynLib

2.2.3 面元文件

2.2.3.1 GDF面元文件

GDF面元格式，GDF格式是WAMIT软件沿用的模型文件格式，只支持四边面元，本程序在其基础上进行修改，使得其支持三角形面元：以下展示了四边形GDF文件开头格式，来源于[test01: Wigley I\(四边形面元\)](#)，在[读取.gdf面元文件](#)给出的读取面元的MATLAB程序

```

1  4 Type=PanelType::Quadrangle
2  1.000000 9.806650
3  0 0
4  640
5  -1.000000 0.000000 -0.125122
6  -1.000000 0.000000 -0.109482
7  -0.950000 0.003668 -0.109863
8  -0.950000 0.000000 -0.125122
9  -1.000000 0.000000 -0.109482
10 -1.000000 0.000000 -0.093842
11 -0.950000 0.006951 -0.094604
12 -0.950000 0.003668 -0.109863

```

- 第1行：Ns + 空格 + 字符串，Ns 指定为4 (四边形面元)或者3 (三角形面元)，若是不指定Ns，则默认为4 (四边形面元)；
- 第2~3行：为WAMIT使用GDF的固定格式，本程序中无实际意义，但不可省略。
- 第4行：为面元数目
- 后续为面元的各个顶点，以四边形为例，则4个点一组组成一个面元，四个点的顺序决定法向方向

以下展示了四边形GDF文件开头格式，来源于[test02: Wigley I\(三角形面元\)](#)

```

1  3 Type=PanelType::Triangle
2  1.000000 9.806650
3  0 0
4  1154
5  -0.496737 -0.118563 0.000000
6  -0.514314 -0.106908 -0.044663
7  -0.536197 -0.113018 0.000000
8  -0.418618 -0.111470 -0.066330
9  -0.385360 -0.118403 -0.062261
10 -0.382171 -0.099653 -0.092400
11 0.536197 -0.113018 0.000000
12 0.513190 -0.107393 -0.043903
13 0.496737 -0.118563 0.000000

```

2.2.3.2 msh格式文件

不建议使用，可能存在法向不统一的问题，建议使用[GDF格式面元](#)文件，转换方式参考：[面元文件格式转换](#)。

以下为GAMBIT导出的msh文件格式。

其中，

- 行6: 2B8 为十六进制数，表示三维空间点的数目。
- 行21: 2a0 为十六进制数，表示面元的数目。

```

1  (0 "GAMBIT to Fluent File")
2
3  (0 "Dimension:")
4  (2 3)
5
6  (10 (0 1 2B8 1 3))
7  (10 (1 1 2B8 1 3) (
8      4.1662500000e-001    3.0050000000e-001    -1.8500000000e-001
9      -4.1662500000e-001    3.0050000000e-001    -1.8500000000e-001
10     3.1246875000e-001    3.0050000000e-001    -1.8500000000e-001
11     2.0831250000e-001    3.0050000000e-001    -1.8500000000e-001
12     ...
13     ...
14     -4.1662500000e-001    3.1823551231e-001    -1.5140452750e-001
15     -4.1662500000e-001    3.0677016746e-001    -1.6548572283e-001
16     -4.1662500000e-001    2.9279395019e-001    -1.7612677537e-001
17     -4.1662500000e-001    2.7706420302e-001    -1.8275125783e-001
18 ))
19
20 (0 "Faces:")
21 (13(0 1 2a0 0))
22 (13(2 1 2a0 3 0) (
23 4 1e 89 7a 1c 0 0
24 4 89 90 79 7a 0 0
25 4 90 97 78 79 0 0
26 4 97 9e 77 78 0 0
27 ...
28 ...
29 4 49 48 176 16f 0 0
30 4 48 47 17d 176 0 0
31 4 47 46 184 17d 0 0
32 4 46 1d 24 184 0 0
33 ))
34 (0 "Cells:")
35 (12 (0 1 0 0))
36
37 (0 "Zones:")
38 (45 (2 wall wall) ())
39 (45 (4 interior default-interior) ())

```

2.2.3.3 pan格式文件

不建议使用，可能存在法向不统一的问题，建议使用[GDF格式面元](#)文件，转换方式参考：[面元文件格式转换](#)。

1	Total numbers of points and panels			
2	900	953		
3	Points and the conectivities			
4	-105.748808	0.000000	14.000000	
5	105.100000	0.000000	14.000000	
6	-89.533834	0.000000	0.000000	
7	96.657691	0.000000	0.000000	
8	...			
9	...			
10	136	147	100	101
11	147	158	99	100
12	158	169	98	99
13	169	180	97	98

2.2.4 UDF文件

UDF(User Defined Function)借助lua脚本实现自定义函数功能，可以在UDF脚本中自定相关函数，可以通过程序[生成模板文件](#)

UDF文件中可以自定义的函数类型包括以下5类：

1. [阻尼系数分布函数\(Damping Lid\)](#)
2. [壁面阻尼系数分布函数\(Wall Damping\)](#)
3. [壁面阻尼区域定义\(Wall Damping Range\)](#)
4. [监测点定义\(OffBodyPoints\)](#)
5. [时域外力定义\(OtherForce\)](#)

2.2.4.1 阻尼系数分布函数(Damping Lid)

```

1  function UDF_Damping_Lid_mu(x, y, omega)
2      -- write code
3      a = 2.45e-1
4      b = 2.8e-1
5      mu = 0.0175 * a * (math.exp(b * omega) - 1)
6      mu = mu * 2
7      return mu
8  end

```

2.2.4.2 壁面阻尼系数分布函数(Wall Damping)

用于[壁面阻尼模型](#)(Wall Damping Model)定义阻尼系数的空间 (x, y, z) 频域 ω 分布函数。

```

1  -- UDF_Function_For Wall Damping Model
2  A = {0.0014, 0.0020, 0.0032, 0.0042, 0.0054} -- 幅值
3  omega_res = {5.74, 6.54, 7.42, 8.34, 9.24} -- 共振频率
4  function UDF_Wall_Damping_mu(x, y, z, omega)
5      -- write code
6      -- x(m), y(m), z(m), omega(rad/s)
7      mu = 0
8      k = 15
9      for i = 1, 5 do
10         mu = mu + A[i] * math.exp(-k * math.pow(omega - omega_res[i], 2))
11     end
12     return mu * 2
13 end

```

2.2.4.3 壁面阻尼区域定义(Wall Damping Range)

用于 [壁面阻尼模型](#) (Wall Damping Model) 定义阻尼区域范围。

2.2.4.3.1 例1: 壁面阻尼系数分布函数

`UDF_Wall_Damping_mu(x, y, z, omega)`

```

1  -- UDF_Function_For Wall Damping Model
2  function UDF_Wall_Damping_mu(x, y, z, omega)
3      -- write code
4      -- x(m), y(m), z(m), omega(rad/s)
5      mu = 0.1
6      return mu
7  end

```

2.2.4.3.2 例2: 壁面阻尼区域范围定义函数

`isInWallDampingRange(x, y, z)`

```

1  -- 以下给出了判断点是否在区域内的一个示例(三维矩形空间)
2  -- x,y,z为空间点坐标
3  -- 在区域内返回true, 反之返回false
4  function isInWallDampingRange(x, y, z)
5      min = {-100, -0.5, -100} -- 三维坐标最小值
6      max = {100, 0.5, 100} -- 三维坐标最大值
7      if (x >= min[1] and x <= max[1] and y >= min[2] and y <= max[2] and z >=
8          min[3] and z <= max[3]) then
9          return true
10     else
11         return false
12     end
13 end

```

2.2.4.3.3 例3: 壁面阻尼区域范围定义函数(多个区域取并集)

```

1  -- 三维空间1
2  function isInWallDampingRange_1(x, y, z)
3      min = {-100, -0.08+0.45, -100} -- 三维坐标最小值
4      max = {100, 0.08+0.45, 100} -- 三维坐标最大值
5      if (x >= min[1] and x <= max[1] and y >= min[2] and y <= max[2] and z >=
6          min[3] and z <= max[3]) then

```

```

7         return true
8     else
9         return false
10    end
11 end
12 -- 三维空间2
13 function isInWallDampingRange_2(x, y, z)
14     min = {-100, -0.08-0.45, -100} -- 三维坐标最小值
15     max = {100, 0.08-0.45, 100} -- 三维坐标最大值
16     if (x >= min[1] and x <= max[1] and y >= min[2] and y <= max[2] and z >=
17         min[3] and z <= max[3]) then
18         return true
19     else
20         return false
21     end
22 end
23 -- 将三维空间1与三维空间2 并集
24 function isInWallDampingRange(x, y, z)
25     return isInWallDampingRange_1(x, y, z) or isInWallDampingRange_2(x, y, z)
26 end

```

2.2.4.4 监测点定义(OffBodyPoints)

用于 [从UDF中读取OffBodyPoints](#) 定义监测点

```

1  -- 以下给出了在一条线段上均分得到Num个点的例子
2  -- 其中start_point为起始点, end_point为末端点, Num>=2为点数目
3  function getOffBodyPoints()
4      -- write code
5      start_point = {-20, 0, 0}
6      end_point = {20, 0, 0}
7      Num = 10
8      delta = {}
9      for k = 1, 3 do delta[k] = (end_point[k] - start_point[k]) / (Num - 1) end
10     points = {}
11     for i = 1, Num do
12         points[i] = {}
13         for k = 1, 3 do
14             points[i][k] = start_point[k] + delta[k] * (i - 1)
15         end
16     end
17     return points
18 end

```

2.2.4.5 时域外力定义(OtherForce)

用于定义外力, 外力可以为时间 `(time(s))` 与浪向 `(dir)` 的函数,

注意: 浪向为索引值, 参考: [入射波浪参数](#) 说明

```

1  -- UDF Function For Other Forces
2  function UDF_Other_Force(time, dir)
3      -- write code
4      -- time(s) dir(int index)
5      Fx = 0.1
6      Fy = 0.2
7      Fz = 0.3
8      return {Fx, Fy, Fz}
9  end

```

2.2.4.6 Lua基础的数学运算符

```

1  -- ***** Lua Usage
2  -- ***** 以下为编写函数可能需要的运算符与数学函数
3  -- *****
4  -- 1. lua 运算符
5  -- + 加法 A + B
6  -- - 减法 A - B
7  -- * 乘法 A * B
8  -- / 除法 B / A
9  -- % 取余 B % A
10 -- ^ 乘幂 A^2
11 -- - 负号 -A
12 -- // 整除运算符(>=lua5.3) 5//2 输出结果 2
13 -- 2. lua 关系运算符
14 -- == 等于, 检测两个值是否相等, 相等返回 true, 否则返回 false
15 -- ~= 不等于, 检测两个值是否相等, 不相等返回 true, 否则返回 false
16 -- > 大于, 如果左边的值大于右边的值, 返回 true, 否则返回 false
17 -- < 小于, 如果左边的值大于右边的值, 返回 false, 否则返回 true
18 -- >= 大于等于, 如果左边的值大于等于右边的值, 返回 true, 否则返回 false
19 -- <= 小于等于, 如果左边的值小于等于右边的值, 返回 true, 否则返回 false
20 -- 逻辑运算符: 与and 或or 非not
21 -- 3. lua数学函数
22 -- pi 圆周率 math.pi
23 3.1415926535898
24 -- abs 取绝对值 math.abs(-2012) 2012
25 -- ceil 向上取整 math.ceil(9.1) 10
26 -- floor 向下取整 math.floor(9.9) 9
27 -- max 取参数最大值 math.max(2,4,6,8) 8
28 -- min 取参数最小值 math.min(2,4,6,8) 2
29 -- pow 计算x的y次幂 math.pow(2,16) 65536
30 -- sqrt 开平方 math.sqrt(65536) 256
31 -- mod 取模 math.mod(65535,2) 1
32 -- modf 取整数和小数部分 math.modf(20.12) 20
33 0.12
34 -- randomseed 设随机数种子 math.randomseed(os.time())
35 -- random 取随机数 math.random(5,90) 5~90
36 -- rad 角度转弧度 math.rad(180)
37 3.1415926535898
38 -- deg 弧度转角度 math.deg(math.pi) 180
39 -- exp e的x次方 math.exp(4)
40 54.598150033144
41 -- log 计算x的自然对数 math.log(54.598150) 4
42 -- log10 计算10为底, x的对数 math.log10(1000) 3
43 -- frexp 将参数拆成x * (2 ^ y)的形式 math.frexp(160) 0.625 8
44 -- ldexp 计算x * (2 ^ y) math.ldexp(0.625,8) 160
45 -- sin 正弦 math.sin(math.rad(30)) 0.5
46 -- cos 余弦 math.cos(math.rad(60)) 0.5
47 -- tan 反正切 math.tan(math.rad(45)) 1

```

43	-- asin	反正弦	math.deg(math.asin(0.5))	30
44	-- acos	反余弦	math.deg(math.acos(0.5))	60
45	-- atan	反正切	math.deg(math.atan(1))	45

2.2.5 系泊系统配置文件

本程序采用集中质量法计算系泊系统的动态响应，基于开源的计算程序库 [Github链接](#)，同时对该程序库进行验证。目前该程序库作为系泊分析模块已经集成至开源风机模拟软件 OpenFAST。

配置文件，配置文件说明参考官方 [说明文档](#)：

```

1  ----- LINE TYPES -----
2  1 NTypes - number of LineTypes
3  Name      Diam      MassDen      EA      BA/-zeta      Can      Cat      Cdn      Cdt
4  (-)      (m)      (kg/m)      (N)      (N-s/-)      (-)      (-)      (-)      (-)
5  main      0.09      77.7066      384.243E6  -0.8      1.0      0.0      1.6      0.1
6  ----- CONNECTION PROPERTIES -----
7  6 NConnects - number of connections including anchors and fairleads
8  Node      Type      X      Y      Z      M      V      FX      FY      FZ      CdA
9  Ca
10 1          fixed      853.87      0.0      -320.0      0      0      0      0      0      0
11 0
12 2          fixed      -426.94      739.47      -320.0      0      0      0      0      0      0
13 0
14 3          fixed      -426.94      -739.47      -320.0      0      0      0      0      0      0
15 0
16 4          vessel      5.2      0.0      -70.0      0      0      0      0      0      0
17 0
18 5          vessel      -2.6      4.5      -70.0      0      0      0      0      0      0
19 0
20 6          vessel      -2.6      -4.5      -70.0      0      0      0      0      0      0
21 0
22 ----- LINE PROPERTIES -----
23 3 NLines - number of line objects
24 Line      LineType      UnstrLen      NumSegs      NodeAnch      NodeFair      Flags/Outputs
25 1          main          950          40          1          4          p
26 2          main          950          40          2          5          p
27 3          main          950          40          3          6          p
28 ----- SOLVER OPTIONS -----
29 0.001      dtM - time step to use in mooring integration (s)
30 3.0e6      kBot - bottom stiffness (Pa/m)
31 3.0e5      cBot - bottom damping (Pa-s/m)
32 320      WtrDpth - water depth (m)
33 1.0      dtIC - time interval for analyzing convergence during IC gen (s)
34 60.0      TmaxIC - max time for IC gen (s)
35 4.0      CdScaleIC - factor by which to scale drag coefficients during dynamic
36 relaxation (-)
37 0.001      threshIC - threshold for IC convergence (-)
38 ----- OUTPUTS -----
39 FairTen1
40 FairTen2
41 FairTen3
42 AnchTen3
43 L2N4pX
44 ----- need this line -----

```


2.3 输出文件有哪些？

目前，软件输出文件主要包含4类：

- [log文件](#)：显示必要的进度信息，算例信息，以及程序警告和报错
- [频域结果文件](#)：频域计算结果文件
- [时域结果文件](#)

2.3.1 log文件

log文件中包含程序计算时输出的必要信息，是检查计算参数、调试以及查看错误信息的必要依据

2.3.1.1 Brief Summary Check

以下为控制参数的简要总结：

```
1 || INFO (2023.03.14 17:33:06) || *****This is a Brief Summary*****
2 || INFO (2023.03.14 17:33:06) || Using Lua Script or not:0
3 || INFO (2023.03.14 17:33:06) || Using Wall Damping Model or not:0
4 || INFO (2023.03.14 17:33:06) || Using UDF Range or not:0
5 || INFO (2023.03.14 17:33:06) || Using OffBodyPoints UDF or not:0
6 || INFO (2023.03.14 17:33:06) || *****End of the Brief Summary*****
```

关键字	描述	说明
Using Lua Script or not	是否使用Lua脚本	1=Yes; 0=No
Using Wall Damping Model or not	是否使用壁面阻尼模型	1=Yes; 0=No
Using UDF Range or not	是否使用壁面阻尼模型中自定义空间范围	1=Yes; 0=No
Using OffBodyPoints UDF or not	是否使用自定义监测点	1=Yes; 0=No

2.3.1.2 其他提示信息

详见log文件

2.3.2 频域结果文件

以下结果文件，可以采用相关的[MATLAB后处理函数](#)进行数据后处理

BodyName为指定的[浮体名称](#)

2.3.2.1 浮体几何信息

BodyName_**HydroStatic.json**: 浮体的静力学参数

BodyName_**m.GDF**: 浮体面元模型

BodyName_**m.txt**: 面元模型详细信息

数据抬头如下:

```
1 640 4 (Num:Patches Number || Type:PanelType::Quadrangle)
2 #Centroid(x,y,z),#Normal(x,y,z),#ds,#P1(x,y,z),#P2(x,y,z),#P3(x,y,z),#P4(x,y,z)
```

BodyName_**WaterLine.txt**: 水线面

数据抬头如下:

```
1 80 (Water Lines Number)
2 # Node1(x,y,z) Node2(x,y,z)
```

2.3.2.2 波浪力结果

BodyName_**EXForce1st.txt**: 入射力+绕射力

BodyName_**FKForce1st.txt**: 入射力

数据抬头如下:

```
1 100 3 (Num_Fre Num_Dir)
2 Fre Dir real(F1) imag(F1) real(F2) imag(F2) real(F3)
imag(F3) real(F4) imag(F4) real(F5) imag(F5) real(F6) imag(F6)
```

2.3.2.3 运动结果

BodyName_**Motion.txt**: 六自由度运动

数据抬头如下:

```
1 100 3 (Num_Fre Num_Dir)
2 Fre Dir real(X1) imag(X1) real(X2) imag(X2) real(X3)
imag(X3) real(X4) imag(X4) real(X5) imag(X5) real(X6) imag(X6)
```

2.3.2.4 水动力系数结果

BodyName1-BodyName2_**AddMass.txt**: 附加质量, 无因次化方式见数据抬头

```
1 100 4.168037e-02 (Num_Fre | V | Added Mass (A/(rho*V)))
2 Fre A11-A66
```

BodyName1-BodyName2_**Damping.txt**: 阻尼系数, 无因次化方式见数据抬头

```
1 100 4.168037e-02 (Num_Fre | V | Damping Coef (D/(omega*rho*V)))
2 Fre D11-D66
```

注意: BodyName1-BodyName2表示Body1运动对于Body2的水动力系数的影响, 此项主要用与计算时延函数

2.3.2.5 空间监测点

ProjectName_**OffBodyPoints.txt**: 空间点坐标以及对应的位置的速度势

```
1 500 3 21 (Num_Fre Num_dir Num_OffBodyPoints) | (x,y,z) |
  (omega,dir,real(phi),imag(phi),abs(i*phi*omega/g))
2 2.000 0.000 0.000
3 1.800 0.000 0.000
4 1.600 0.000 0.000
5 1.400 0.000 0.000
6 1.200 0.000 0.000
7 1.000 0.000 0.000
8 0.800 0.000 0.000
9 0.600 0.000 0.000
10 0.400 0.000 0.000
11 0.200 0.000 0.000
12 0.000 0.000 0.000
13 -0.200 0.000 0.000
14 -0.400 0.000 0.000
15 -0.600 0.000 0.000
16 -0.800 0.000 0.000
17 -1.000 0.000 0.000
18 -1.200 0.000 0.000
19 -1.400 0.000 0.000
20 -1.600 0.000 0.000
21 -1.800 0.000 0.000
22 -2.000 0.000 0.000
23 2.000000e-02 0.000000e+00 4.204404e-02 -4.904905e+02 9.999806e-01
24 2.000000e-02 0.000000e+00 3.893519e-02 -4.904891e+02 9.999779e-01
25 2.000000e-02 0.000000e+00 3.483940e-02 -4.904867e+02 9.999730e-01
26 2.000000e-02 0.000000e+00 2.964404e-02 -4.904839e+02 9.999672e-01
27 2.000000e-02 0.000000e+00 2.508884e-02 -4.904821e+02 9.999636e-01
28 ...
29 ...
```

2.3.3 时域结果文件

2.3.3.1 时域计算测试文件

2.4 如何使用工具函数

2.4.1 处理面元文件

```

1  -i, --inputfile <input "*.gdf">
2  input panel file(input "*.gdf")
3  -o, --outputfile <output "*.gdf *.pan">
4  output panel file(output "*.gdf,*.pan")
5  --calcHydrostatic      calculate hydrostatic info for panel
6  --correctNormal        correct Normal for the panel
7  --scale <set the scale factor,default is 1.0>
8
9  --translate <set the translate vector,default is 0,0,0>
10
11 --symmetry <set the symmetry type, 0 = NULL;1 = AlongX;2 = AlongY;4 = AlongZ;3 =
    AlongXY;default = 0;>

```

命令行参数	指定参数	说明
<code>-i</code> or <code>--inputfile</code>	<input checked="" type="checkbox"/>	指定输入文件名称(可以为 <code>*.GDF</code> , <code>*.msh</code> , <code>*.pan</code>)
<code>-o</code> or <code>--outputfile</code>	<input checked="" type="checkbox"/>	指定输出文件名称(可以为 <code>*.GDF</code> , <code>*.msh</code> , <code>*.pan</code>), 默认在输入文件名称添加 <code>_Out</code> 后缀
<code>--calcHydrostatic</code>		添加此参数, 则计算静力学参数并输出同名 <code>*.json</code> 文件
<code>--correctNormal</code>		添加此参数, 则进行法向矫正 (有Bug, 暂时没用)
<code>--scale</code>	<input checked="" type="checkbox"/>	对面元文件进行缩放
<code>--translate</code>	<input checked="" type="checkbox"/>	对面元文件进行平移
<code>--symmetry</code>	<input checked="" type="checkbox"/>	对面元文件进行对称 0 = NULL;1 = AlongX;2 = AlongY;4 = AlongZ;3 = AlongXY;

2.4.1.1 面元文件格式转换

将GAMBIT导出的 `.msh` 面元文件转换为 `*.pan` 或者 `*.gdf`

```
1 BEM_ALL.exe -i panel.msh -o panel.gdf
2 BEM_ALL.exe -i panel.msh -o panel.pan
```

2.4.1.2 计算面元文件静力学参数

计算gdf面元文件静力学参数

```
1 BEM_ALL.exe -i panel.gdf --calcHydrostatic
```

cmd窗口显示以下输出

```
1 C:\Users\chentianwen\Desktop\example_with_results\ToolFunction>..\BEM_ALL_V2.0.exe -
  i panel.GDF --calcHydrostatic
2 || INFO (2023.03.17 00:16:52) || create Log [ default.log ]
3 || INFO (2023.03.17 00:16:52) || Current EXE Version: BEM_ALL_V2.0
4 || INFO (2023.03.17 00:16:52) || setPoints as PanelType::Quadrangle; Num = 640
5 || INFO (2023.03.17 00:16:52) || PanelModel::update()
6 || INFO (2023.03.17 00:16:52) || PanelModel initialize()
7 || INFO (2023.03.17 00:16:52) || Output config file [panel.GDF.json] successfully
  !!!
8 || INFO (2023.03.17 00:16:52) || output panel file name is not specific, so we
  decide to add the [Out] suffix
9 || INFO (2023.03.17 00:16:52) || Output [panel.GDF_Out] successfully
10 || INFO (2023.03.17 00:16:52) || Finished Operation for [ panel.GDF ]
```

2.4.1.3 处理面元文件

```
1 %% 缩放面元
2 BEM_ALL.exe -i panel.gdf -o panel_sacled.gdf --scale 100.0
3 %% 平移面元
4 BEM_ALL.exe -i panel.gdf -o panel_trans.gdf --translate 100.0,200,300
5 %% 对称面元(关于以Y轴为法向的面对称)
6 BEM_ALL.exe -i panel.gdf -o panel_sym.gdf --symmetry 2
```

2.4.2 使用第三方库

2.4.2.1 C++编写dll

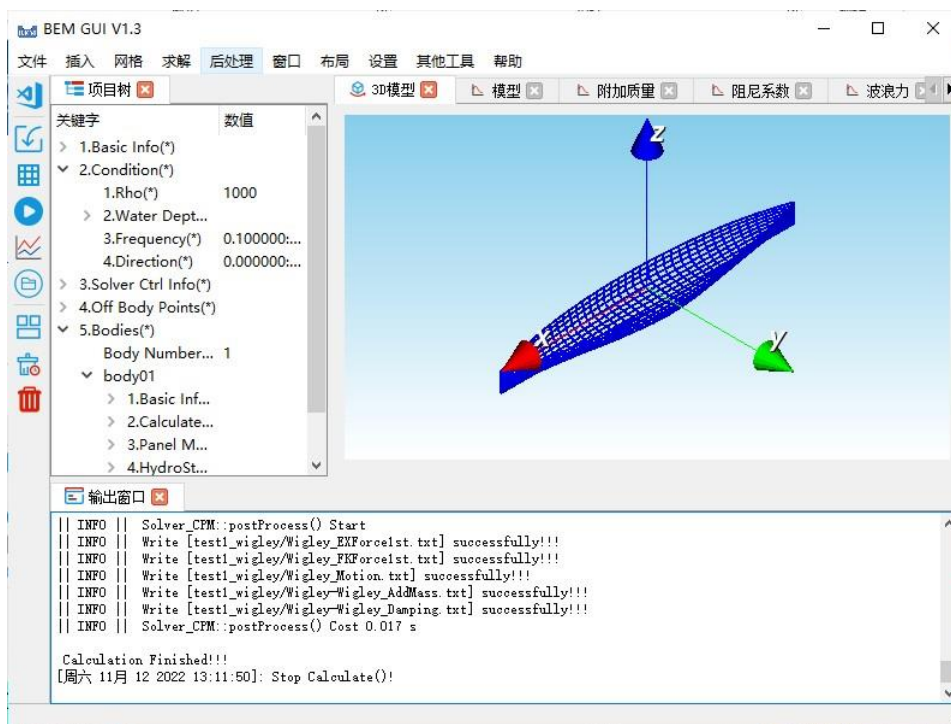
参考博客: [Lua调用C++生成的DLL库](#)

2.4.2.2 Fortran编写dll

建议使用Fortran编写静态库, 再使用C++做接口, 参考上一节 [C++编写dll](#)

3. BEM_GUI.exe使用帮助

3.1 软件GUI界面



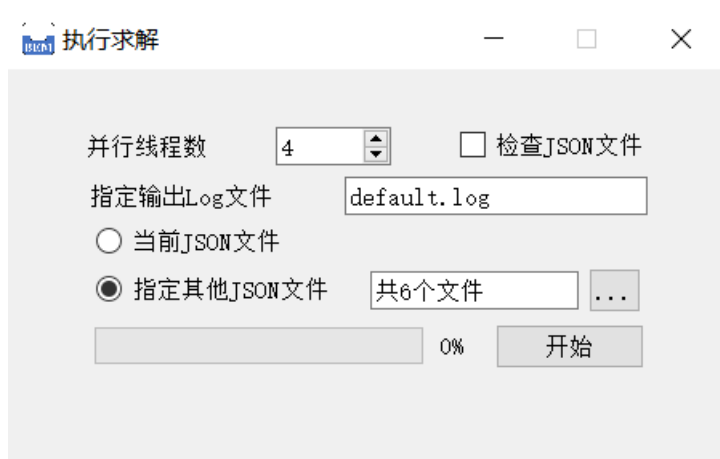
3.2 软件基本操作流程：

软件基本操作流程参见：[B站视频](#)

3.3 特殊操作

3.3.1 利用bat执行多个算例

BEM_GUI一次只能运行一个算例，可以通过此以下方法，批量生成 **bat** 文件(windows平台)，运行 **bat** 脚本按顺序执行多个算例。



选择 **指定其他JSON文件**，多选需要运行的json配置文件，即可生成如下的 **run.bat** 文件

```

1 "C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/solver/BEM_ALL_V2.0.exe" -n 4 -j
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test01_Wigley.json --log
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test01_wigley/default.log
2 "C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/solver/BEM_ALL_V2.0.exe" -n 4 -j
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test01_Wigley_Lid.json --log
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test01_wigley_lid/default.log
3 "C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/solver/BEM_ALL_V2.0.exe" -n 4 -j
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test02_Wigley_tri.json --log
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test02_wigley_tri/default.log
4 "C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/solver/BEM_ALL_V2.0.exe" -n 4 -j
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test03_Barge.json --log
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test03_barge/default.log
5 "C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/solver/BEM_ALL_V2.0.exe" -n 4 -j
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test04_Barge_Lid.json --log
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test04_barge_lid/default.log
6 "C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/solver/BEM_ALL_V2.0.exe" -n 4 -j
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test05_Barge_Barge.json --log
C:/Users/chentianwen/Desktop/Bem_GUI_exe_Simple/example_with_results/FDC
cases/test05_barge_barge/default.log

```

3.3.2 从文件夹中算例数据

菜单栏 -> 后处理 -> 打开数据文件夹

选择文件夹中的带有 项目名称 的json文件，即可进入绘图选项面板，如下图所示



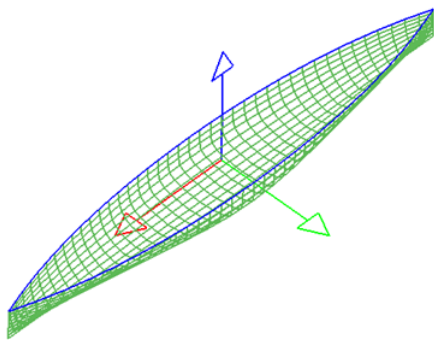
4. 测试算例

4.1 收敛性及并行测试

4.1.1 硬件条件

制造商:	HONOR	
型号:	荣耀 MagicBook Pro	
处理器:	AMD Ryzen 7 4800H with Radeon Graphics	2.90 GHz
已安装的内存(RAM):	16.0 GB (15.4 GB 可用)	

4.1.2 计算模型与工况



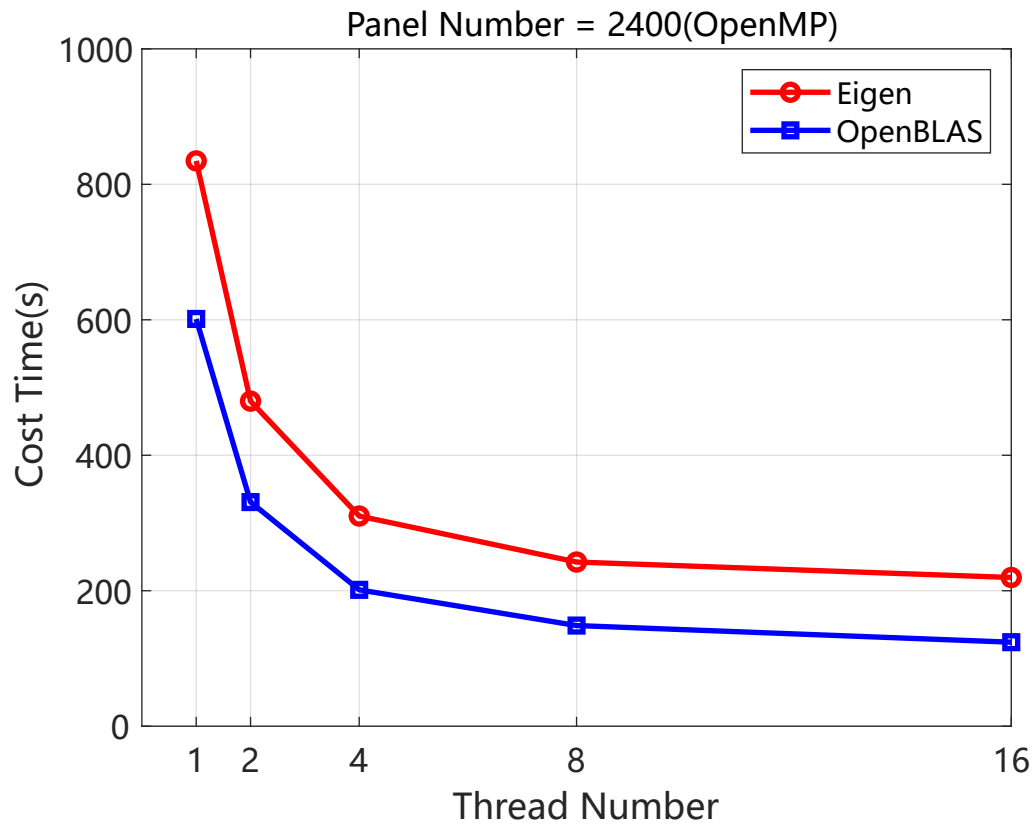
频率	浪向	检测点
0.1:0.1:10 (100)	0:45:90 (3)	0

4.1.3 收敛性与并行测试结果

收敛性测试算例文件夹: [ConvergentTest](#)

并行测试算例文件夹: [Parallel](#) 、 [Parallel_OpenBLAS](#)

面元数目	积分方法	并行线程数	Eigen(s)	OpenBLAS(s)	Eigen 耗时比
160	2×2 <i>Gauss</i>	4	0.967	-	0.003
320	2×2 <i>Gauss</i>	4	3.985	-	0.013
400	2×2 <i>Gauss</i>	4	6.204	-	0.020
640	2×2 <i>Gauss</i>	4	19.017	-	0.061
800	2×2 <i>Gauss</i>	4	29.283	-	0.094
1200	2×2 <i>Gauss</i>	4	67.932	-	0.219
2400	2×2 <i>Gauss</i>	1	834.589	601.239	2.969
2400	2×2 <i>Gauss</i>	2	479.91	330.901	1.547
2400	2×2 <i>Gauss</i>	4	310.262	201.323	1
2400	2×2 <i>Gauss</i>	8	242.376	148.594	0.781
2400	2×2 <i>Gauss</i>	16	219.537	124.019	0.708
4000	2×2 <i>Gauss</i>	4	1035.53	-	3.338



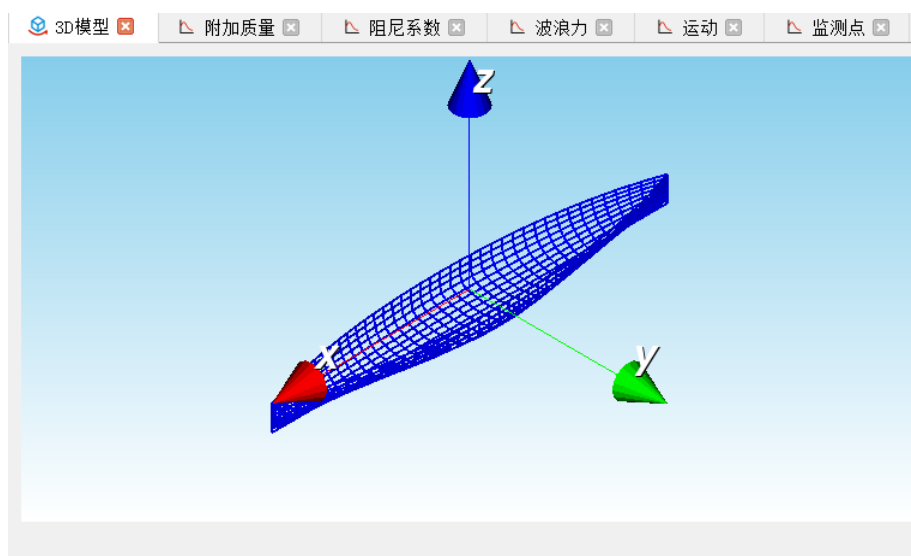
4.2 频域算例

4.2.1 单浮体频域计算(无限水深)

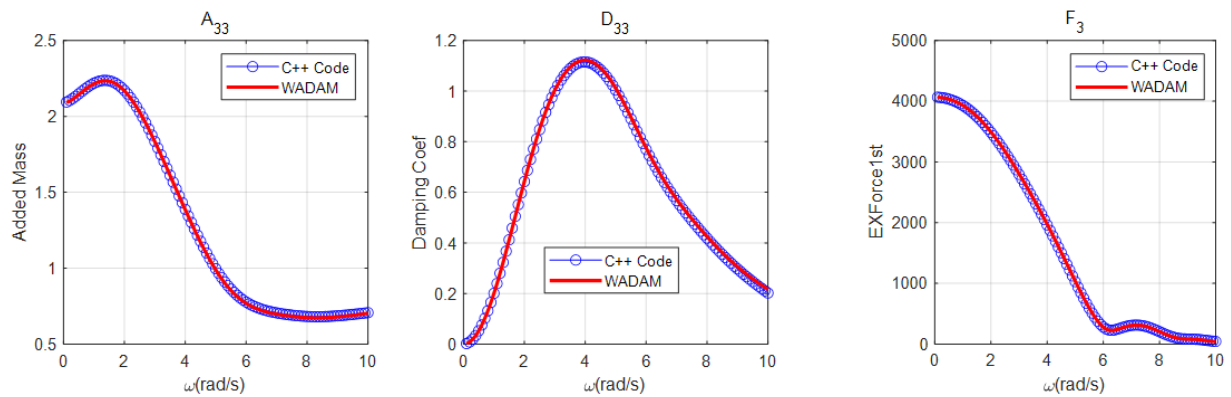
以下为求解在无限水深条件下，单浮体

4.2.1.1 [test01: Wigley I\(四边形面元\)](#)

☐ 计算模型：

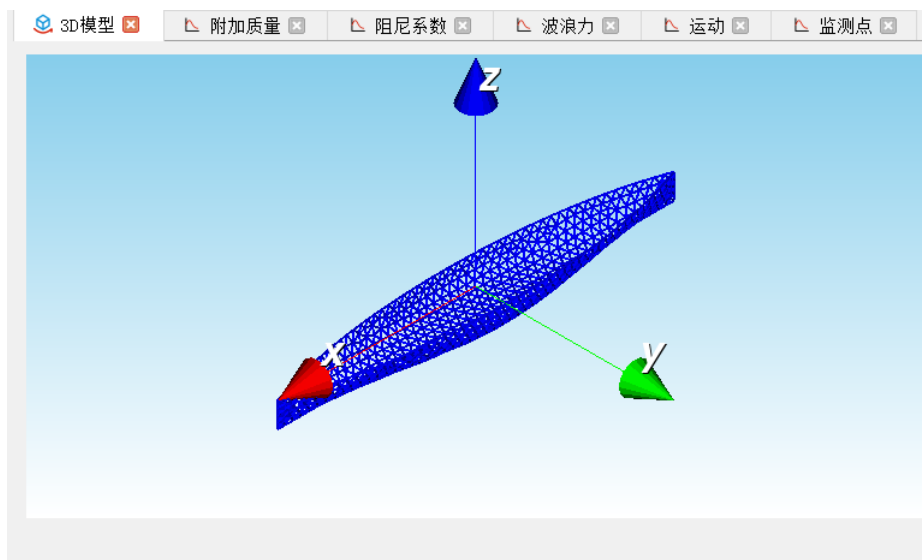


- ☐ 配置文件：test01_Wigley.json
- ☐ 计算内容：针对Wigley I船型(四边形面元)进行单浮体频域计算，最后得到：
 - ☐ 附加质量
 - ☐ 阻尼系数
 - ☐ 入射波浪力结果
 - ☐ 入射波浪力+绕射波浪力结果
- ☐ 计算结果展示：

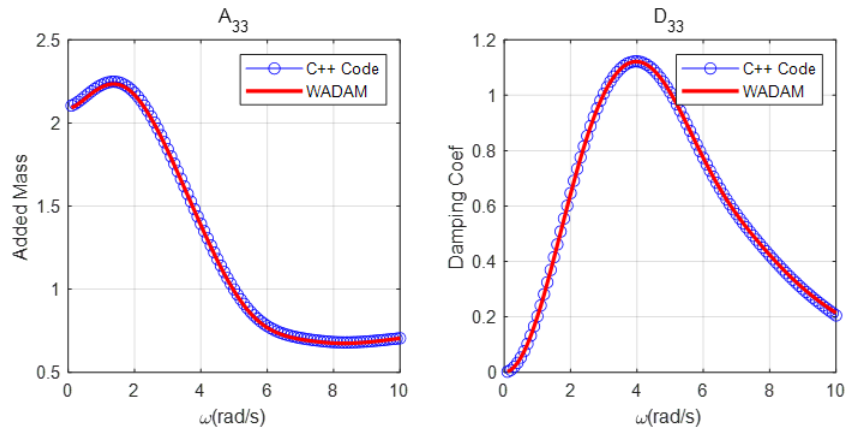


4.2.1.2 test02: Wigley I(三角形面元)

- ☐ 计算模型：

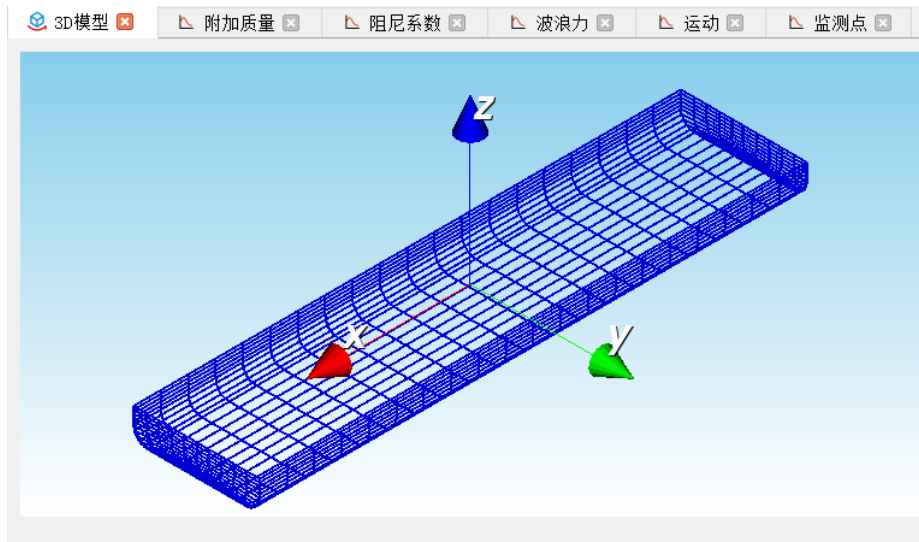


- ☐ 配置文件：test02_Wigley_tri.json
- ☐ 计算内容：针对Wigley I船型(三角形面元)进行频域计算
- ☐ 特点：使用三角形面元文件
- ☐ 计算结果：水动力系数结果与WADAM结果吻合



4.2.1.3 test03: Barge (驳船模型)

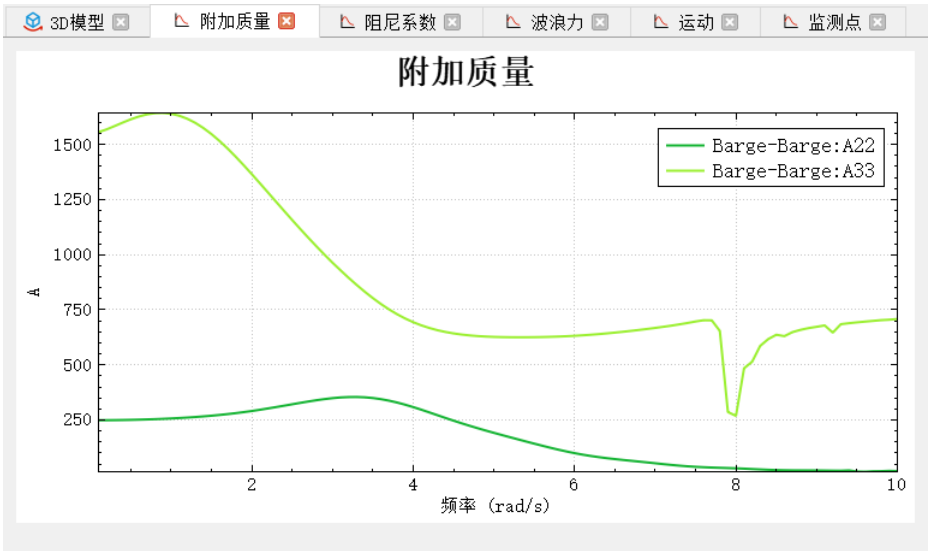
☐ 计算模型



☐ 配置文件：test03_Barge.json

☐ 计算内容：针对Barge船型(四角形面元)进行频域计算

☐ 计算结果展示：下图展示了附加质量计算结果，发现在频率为 8rad/s 出现较大波动，这种现象被称为 **不规则频率** 现象，需要添加内部盖进行修正，相关原理参考文献。



4.2.2 使用内部盖模型

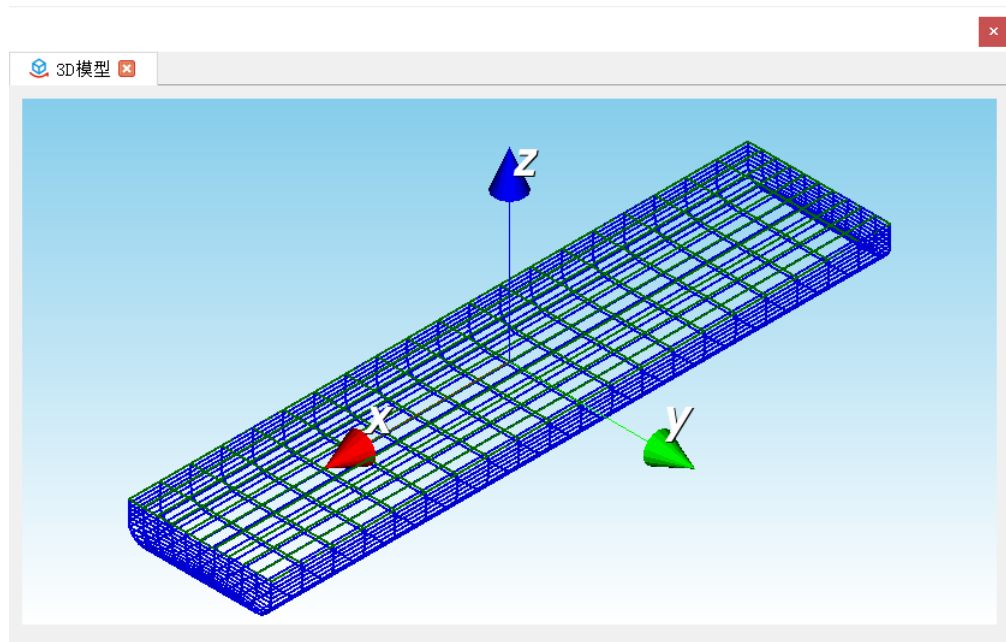
添加内部盖模型需要指定 "body*" 模块中 "1.Basic Info(*)" 中 "2.Type(*)" 节点为 2 (InternalLid)

```
1  "body02": {
2    "1.Basic Info(*)": {
3      "1.Name(*)": "Internal_Lid",
4      "2.Type(*)": {
5        "info": "0=FloatBody;1=FixedBody;2=InternalLid;3=DampingLid",
6        "value": 2
7      }
8    },
9    ...
}
```

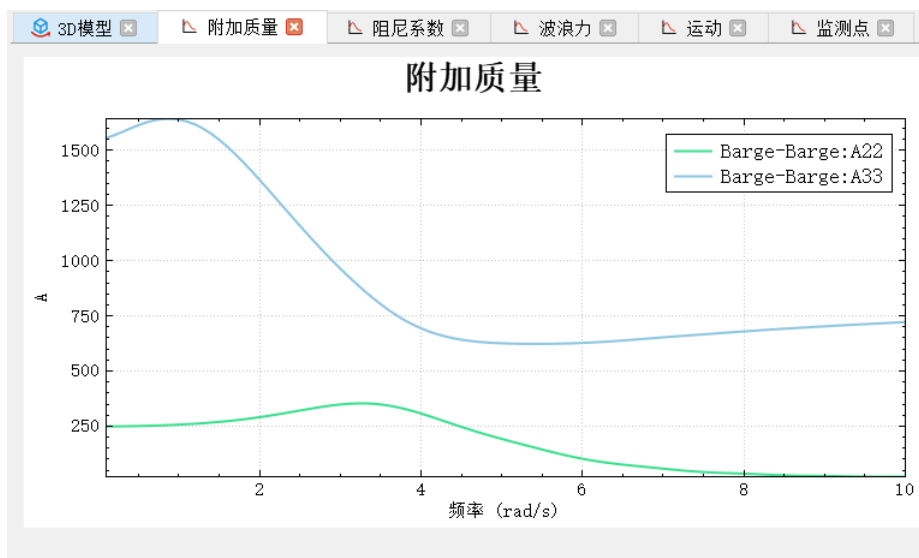
4.2.2.1 test04: Barge Lid(单浮体+内部盖)

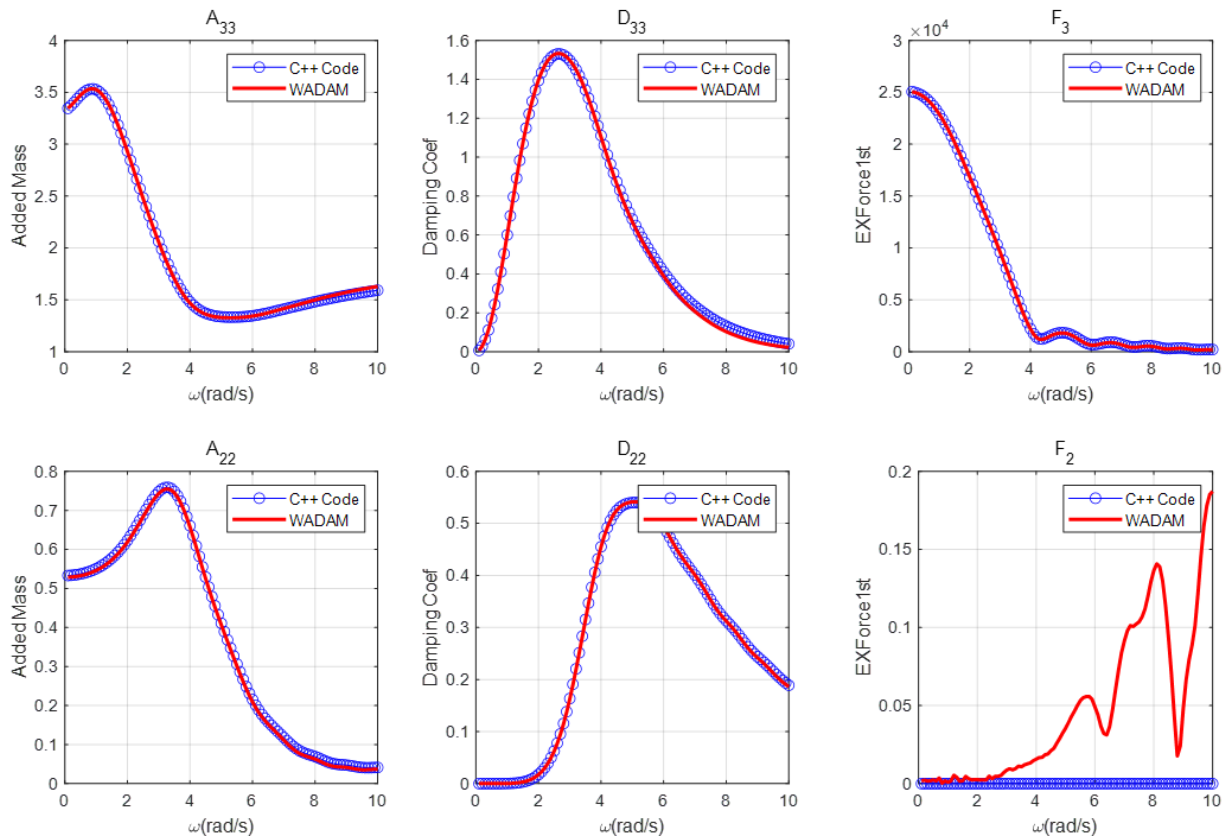
☐ 计算模型

模型	类型
Barge	FloatBody
Lid	Internal Lid



- 配置文件: test04_Barge_Lid.json
- 计算内容: 针对Barge船型(四角形面元)进行频域计算
- 计算结果: 添加内部盖(Internal Lid)能够有效消除不规则频率

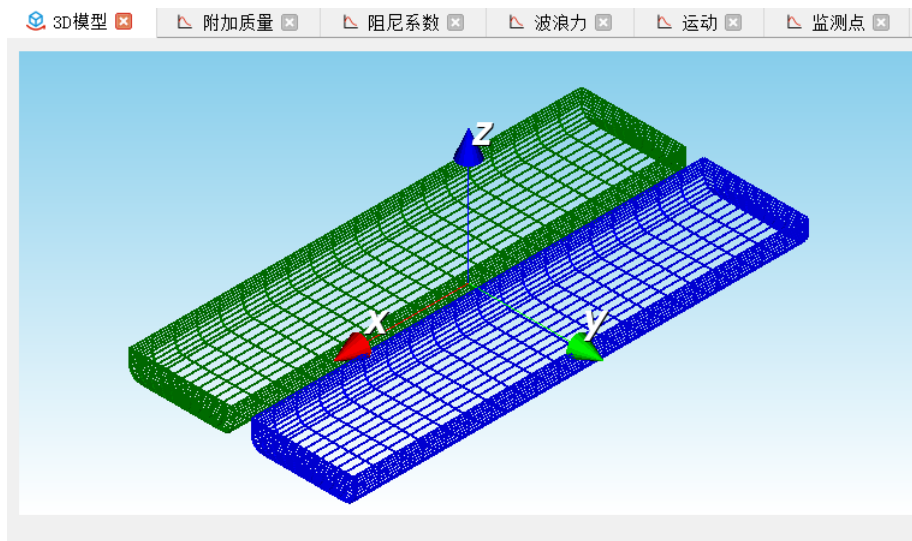




4.2.3 多浮体频域计算

4.2.3.1 [test05: Barge-Barge\(双浮体+监测点\)](#)

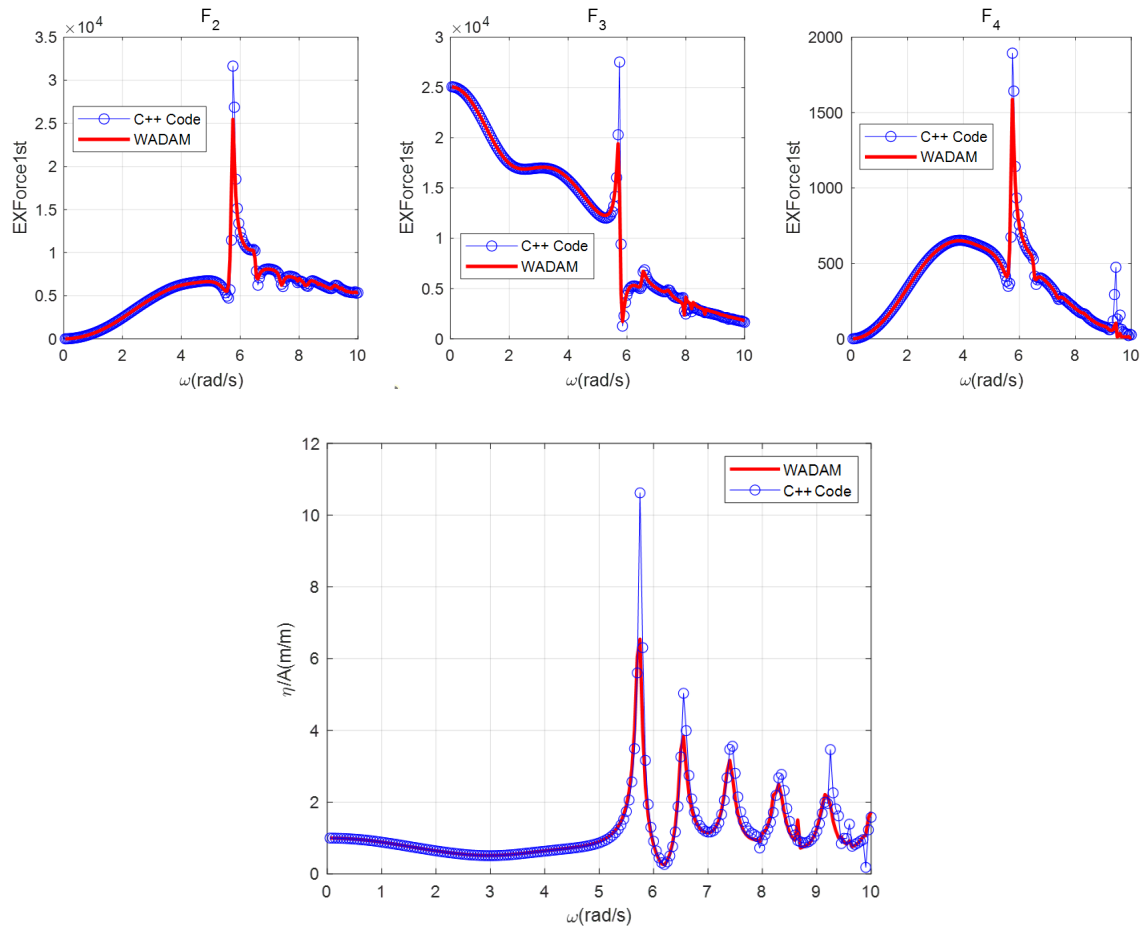
☐ 计算模型



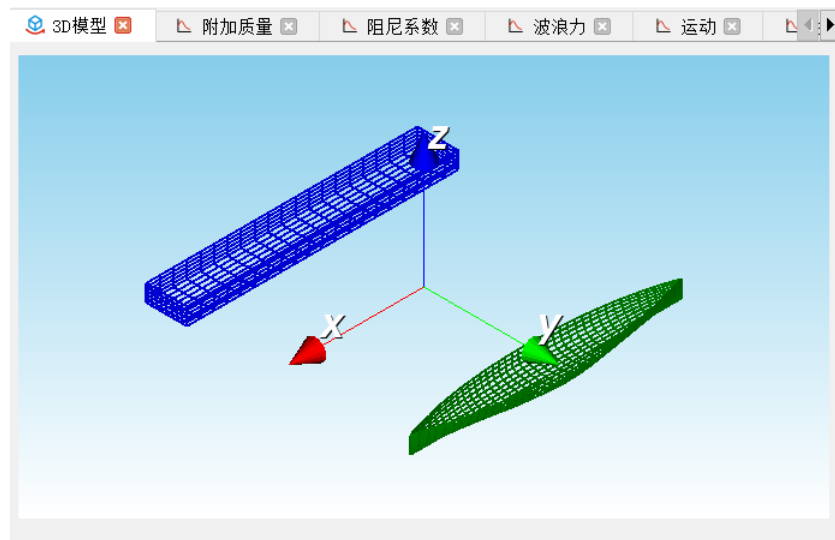
☐ 配置文件: test05_Barge_Barge.json

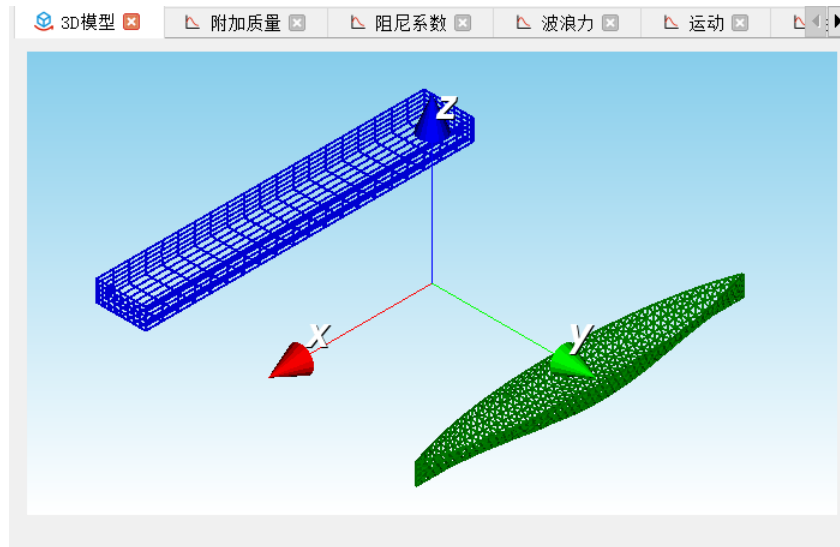
☐ 计算内容: 针对Barge-Barge旁靠模型进行频域计算

- ☐ 特点：没有添加内部盖(Internal_Lid)消除不规则频率
- ☐ 计算结果与 **WADAM** 结果对比



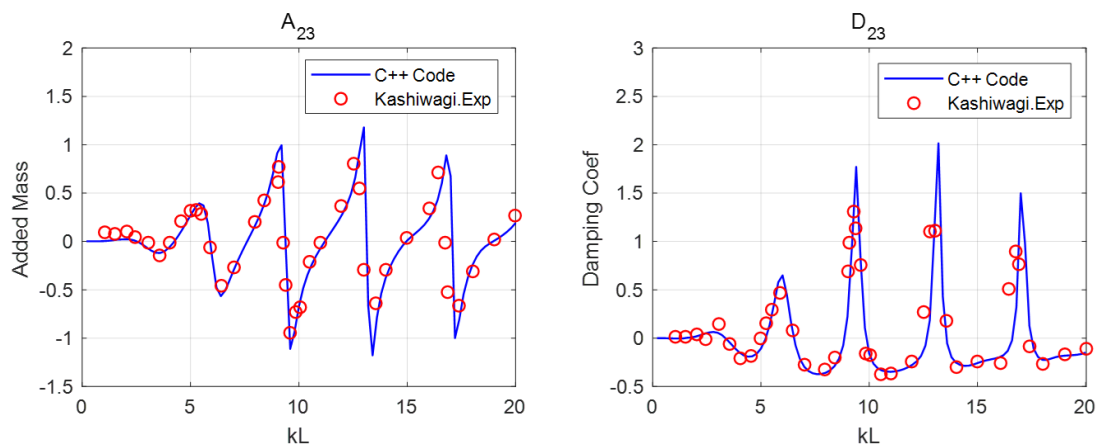
4.2.3.2 [test06: Wigley_Box\(双浮体+验证\)](#)





- ☐ 配置文件: test06_Wigley_Box.json
- ☐ 计算内容: 针对Wigley_Box旁靠模型进行频域计算, 对比2004Kashiwagi试验数据

-
- ☐ 配置文件: test06_Wigley_tri_Box.json
 - ☐ 计算内容: 同test6_Wigley_Box.json, 不同的是, Wigley 船型使用三角形面元, Box模型使用四边形面元
 - ☐ 计算结果:

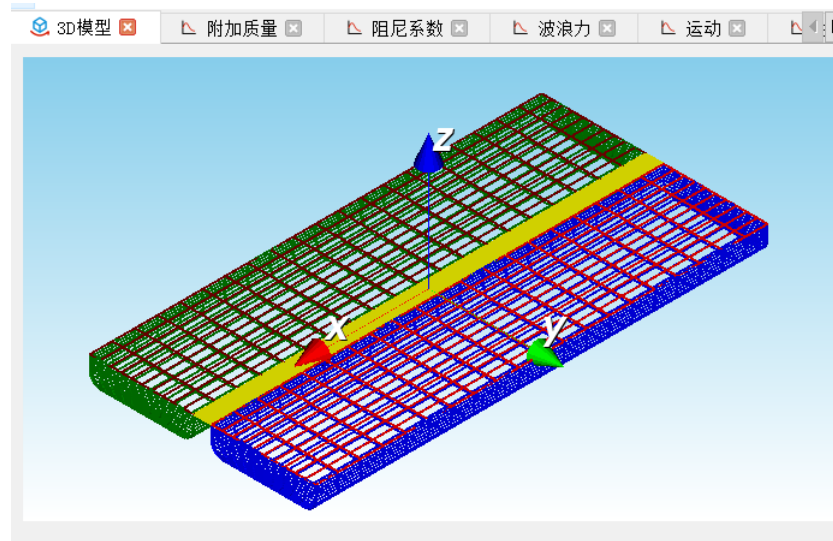


4.2.4 使用阻尼盖模型

添加阻尼盖模型需要指定 "body*" 模块中 "1.Basic Info(*)" 中 "2.Type(*)" 节点为 3(DampingLid), 并且在 "5.Damping Lid Model" 中选择相应的阻尼盖类型以及定义阻尼系数, 可参考 [使用阻尼盖模型](#)。

4.2.4.1 [test07: Barge_Barge Damping Lid\(使用阻尼盖\)](#)

☐ 计算模型



☐ 配置文件：test07_Barge_Barge_Lid.json

☐ 计算内容：针对Barge-Barge旁靠模型进行频域计算

☐ 特点：添加内部盖(Internal_Lid)消除不规则频率、添加阻尼盖模型(Damping Lid)

☐ Log记录

```

1 *****
2 ***** FDC::Solver_FD::showSolverInfo() *****
3 *****
4 Bodies Info. count = 5
5      #      Body Name      Panel Num      Body Type
6      1      Barge1         836         FixedBody
7      2      Barge2         836         FixedBody
8      3      Internal_Lid1   200         Internal Lid
9      4      Internal_Lid2   200         Internal Lid
10     5      Damping Lid     400         Damping
      Lid(mu=0.000000|Uniform)

```

4.2.4.2 [test08: Barge_Barge Damping Lid\(使用阻尼盖+Newtonian\)](#)

算例设置同 [test07](#)，选择 **Newtonian** 作为阻尼系数频率分布，阻尼形式见 [如何使用阻尼盖模型](#)

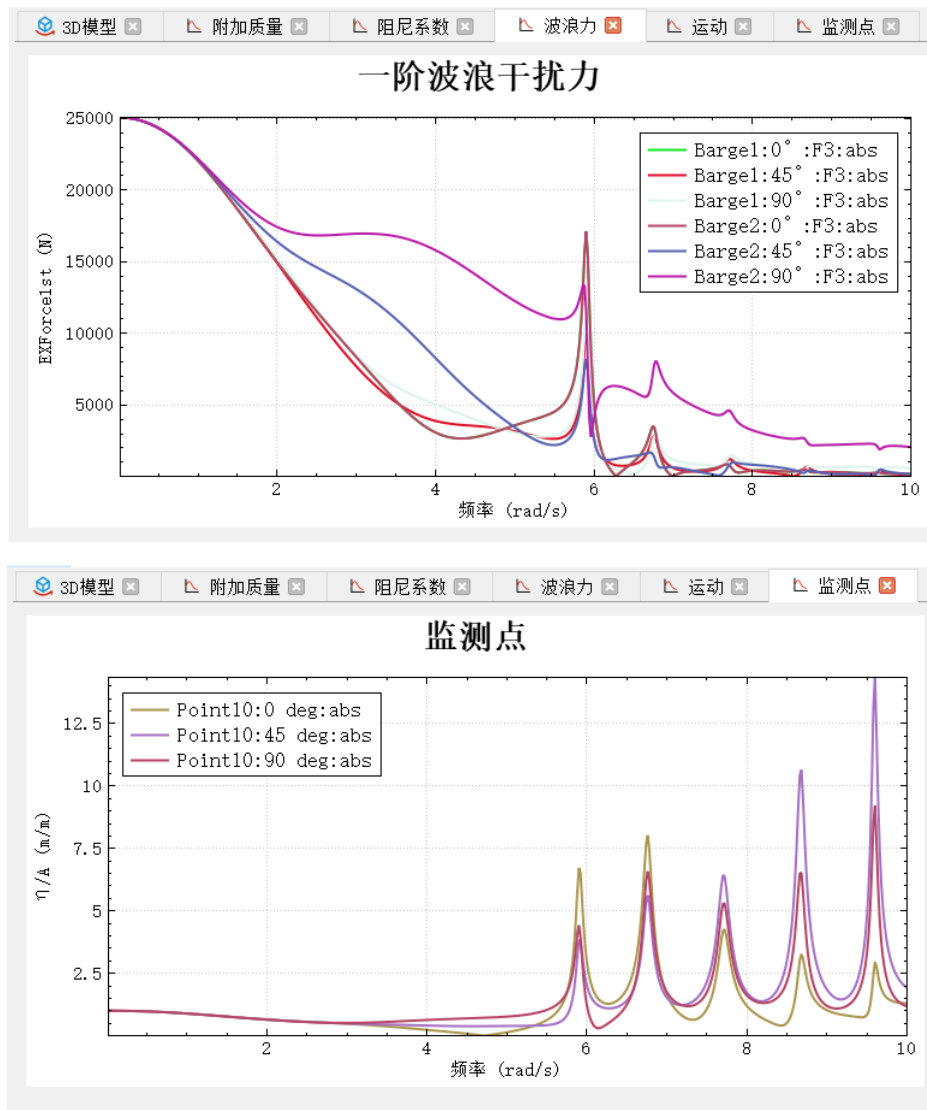
☐ 阻尼盖模型设置如下：

```

1  "5.Damping Lid Model": {
2      "1.Damping Lid Type(*)": {
3          "info": "0=Uniform;1=NonUniform;2=Newtonian;3=XB_Chen;4=UDF",
4          "value": 2
5      },
6      "2.Damping Lid Coeff(*)": 0.003,
7      "3.UDF": {
8          "Func Name In Lua(*)": "",
9          "info": "this is the UDF Func Name which should be found in Lua
10         Script"
11     }
12 }

```

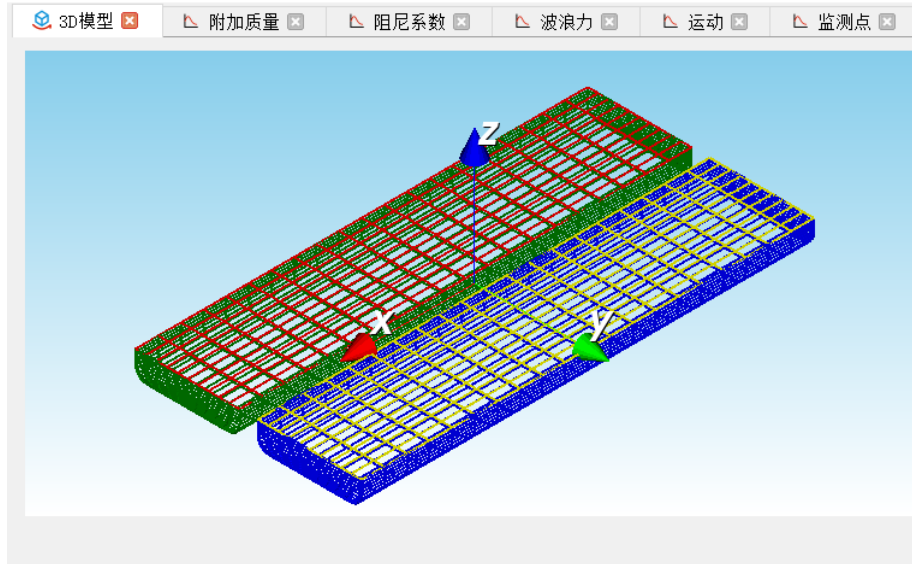
□ 计算结果



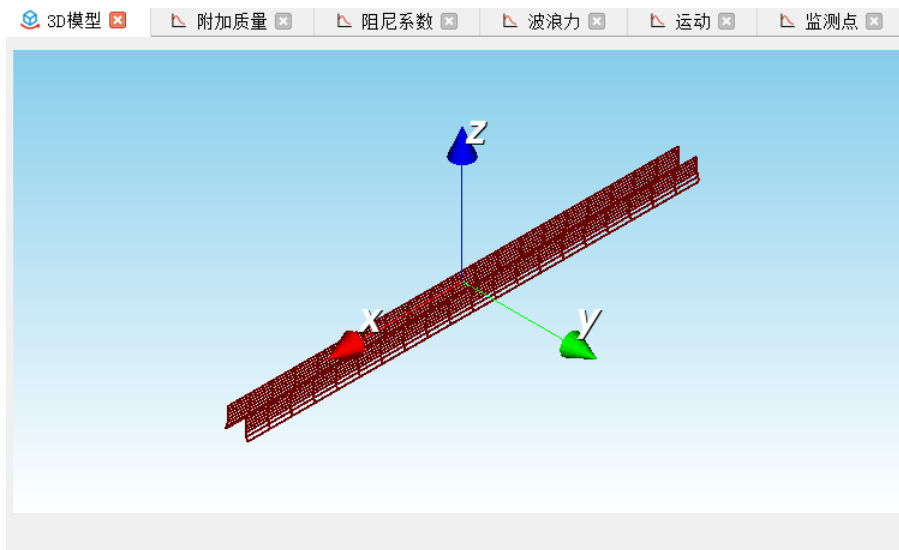
4.2.5 使用壁面阻尼模型

4.2.5.1 [test09: Barge_Barge Wall_Damping](#)(使用壁面阻尼模型)

☐ 计算模型



☐ 其中指定的壁面阻尼区域为:



- ☐ 配置文件：test09_Barge_Barge_Wall_Damping.json
- ☐ 计算内容：针对Barge-Barge旁靠模型进行频域计算
- ☐ 特点：添加内部盖(Internal_Lid)消除不规则频率、使用壁面阻尼模型(Wall Damping)

壁面阻尼模型参数设置如下

```

1  "2.Wall Damping Model(*)": {
2      "1.UDF Func Name": "UDF_Wall_Damping_mu",
3      "2.Range": {
4          "info": "[x_min,y_min,z_min],[x_max,y_max,z_max]",
5          "value": "[[0.000000,0.000000,0.000000],
6              [0.000000,0.000000,0.000000]]"
7      },
8      "3.Range From UDF": {
9          "UDF Func Name": "isInWallDampingRange",
10         "info": "define the range in UDF;in range \"return true\",else
11         \"return false\";"
12     }
13 }

```

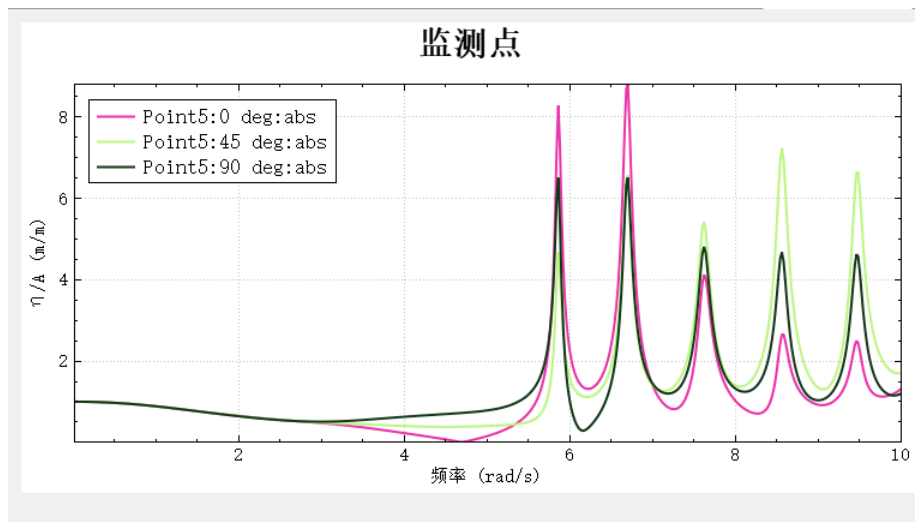
相关的UDF函数如下:

```

1  -- UDF_Function_For Wall Damping Model
2  A = {0.0014, 0.0020, 0.0032, 0.0042, 0.0054} -- 幅值
3  omega_res = {5.74, 6.54, 7.42, 8.34, 9.24} -- 共振频率
4  function UDF_Wall_Damping_mu(x, y, z, omega)
5      -- write code
6      -- x(m),y(m),z(m),omega(rad/s)
7      mu = 0
8      k = 15
9      for i = 1, 5 do
10         mu = mu + A[i] * math.exp(-k * math.pow(omega - omega_res[i], 2))
11     end
12     return mu * 2
13 end
14
15 -- 以下给出了判断点是否在区域内的一个示例(三维矩形空间)
16 -- x,y,z为空间点坐标
17 -- 在区域内返回true, 反之返回false
18 function isInWallDampingRange(x, y, z)
19     min = {-100, -0.08, -100} -- 三维坐标最小值
20     max = {100, 0.08, 100} -- 三维坐标最大值
21     if (x >= min[1] and x <= max[1] and y >= min[2] and y <= max[2] and z
22     >=
23         min[3] and z <= max[3]) then
24         return true
25     else
26         return false
27     end
28 end

```

☐ 计算结果

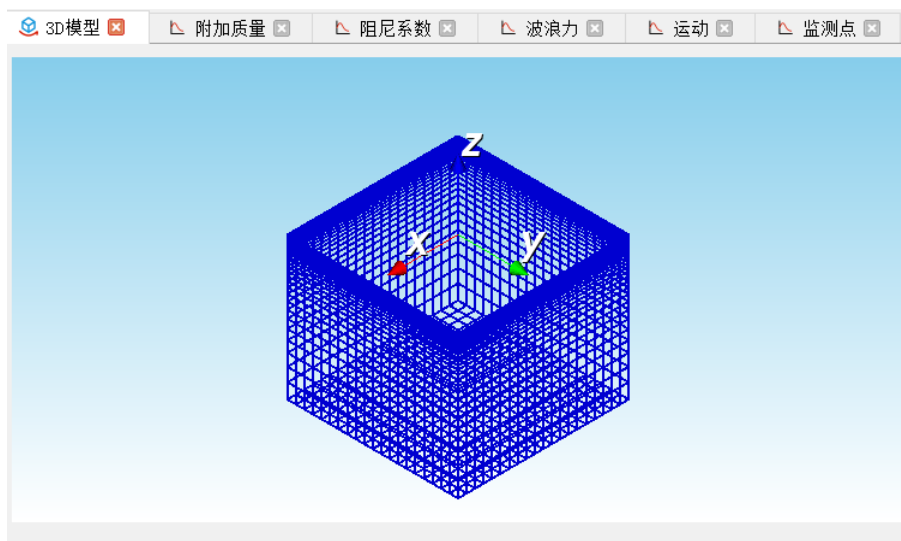


4.2.6 特殊算例

以下为其他类型的算例

4.2.6.1 [test10: 方形box \(检测浮体附近空间波形\)](#)

☐ 计算模型



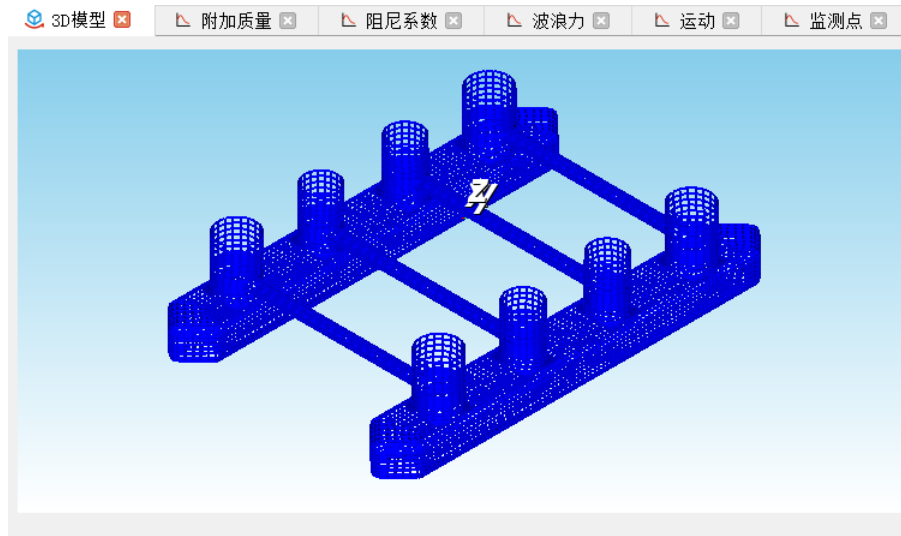
☐ 配置文件：test10_Box.json

☐ 计算内容：针对Box模型进行频域计算，检测近壁面空间波形

☐ 特点：使用UDF定义OffBodyPoints

4.2.6.2 [test11: 半潜平台\(面元数目10000+\)](#)

☐ 计算模型：



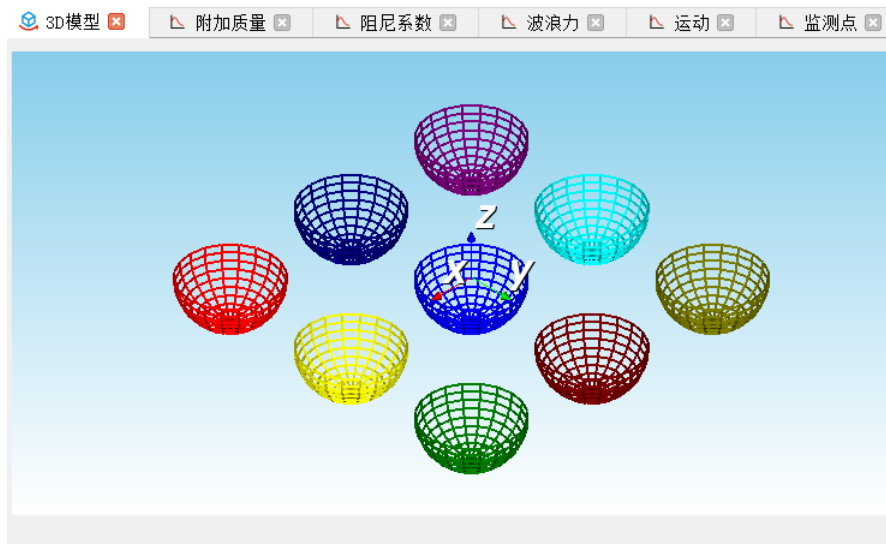
☐ 配置文件：test11_Semisub.json

☐ 计算内容：针对半潜平台模型进行频域计算

☐ 特点：面元数量巨大 $N=10644$

4.2.6.3 [test12: N浮体频域水动力计算](#)

☐ 计算模型：



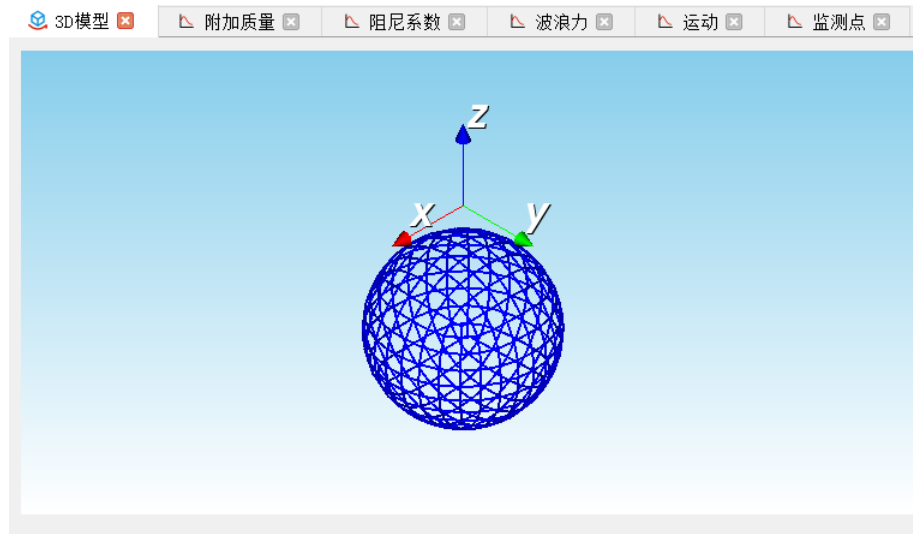
☐ 配置文件：test12_Multi_Sphere.json

☐ 计算内容：针对多浮体浮子水动力性能进行频域计算

☐ 特点：9个浮体

4.2.6.4 [test13: 半球\(有限水深\)](#)

☐ 计算模型：



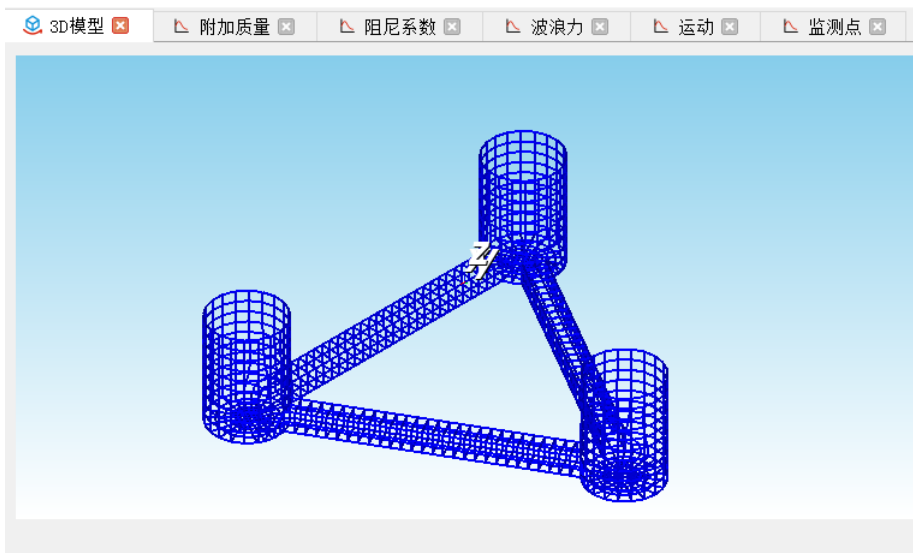
☐ 配置文件：test13_Sub_Sphere_D=1.5A.json

☐ 计算内容：有限水深下，圆球的水动力性能频域计算

☐ 特点：有限水深

4.2.6.5 [test14: 风机下半平台](#)

☐ 计算模型：



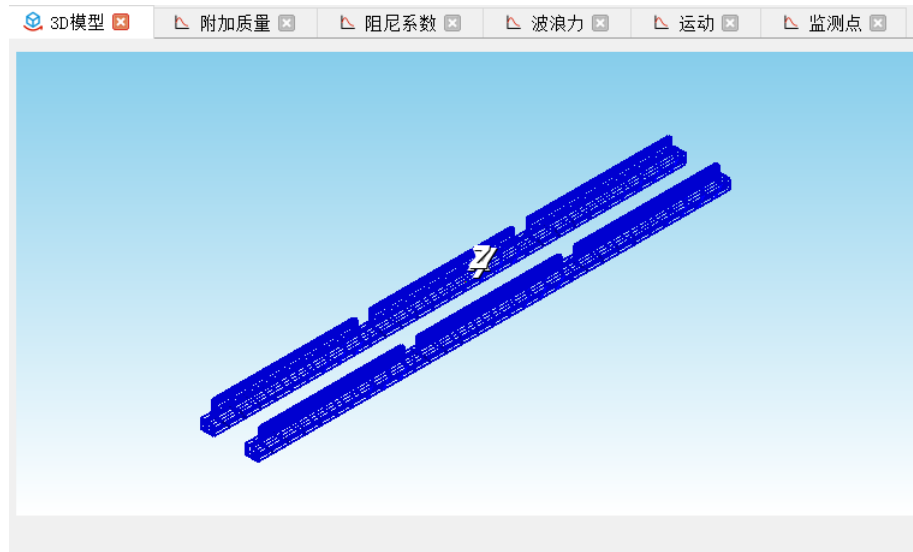
☐ 配置文件：test14_Turbine.json

☐ 计算内容：风机下浮体频域计算

☐ 特点：有限水深

4.2.6.6 [test15: 不知道怎么描述](#)

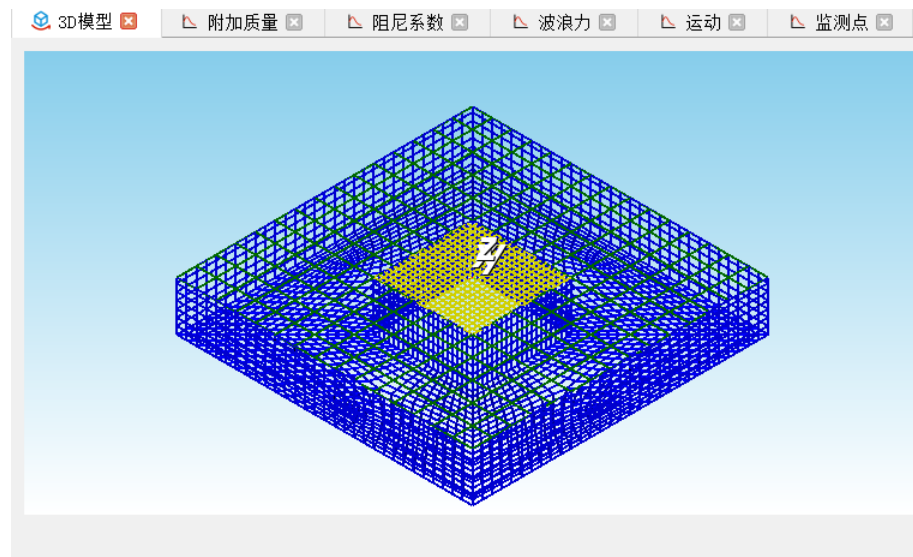
☐ 计算模型



- ☐ 配置文件: test15_Trash.json
- ☐ 计算内容: 多浮体阻挡波浪的效果
- ☐ 特点: 没有什么特点

4.2.6.7 [test16: 月池共振\(添加阻尼盖\)](#)

☐ 计算模型



- ☐ 配置文件: test16_Moon_Pool.json
- ☐ 计算内容: 月池中心波面检测
- ☐ 特点: 内部盖、阻尼盖、月池共振

4.3 时域算例

4.3.1 [test01_TD_Wigley_III](#)(白噪声，时域计算)

4.3.2 [test02_TD_Barge_Barge](#)(双浮体，Cable, Fender, Mooring)

5. 工具函数

5.1 BEMRosetta进行面元处理

BEMRosetta是一个很有用的GDF面元可视化处理工具，可以对面元文件进行：

- 平移、旋转、缩放等几何操作
- 显示并统一面元法向
- 划分网格、内部盖网格
- 以及其他NB的功能

该软件在Github上已开源，[BEMRosetta仓库链接](#)

5.2 MATLAB后处理数据

5.2.1 读取波浪力(**ReadEXForce.m**)

```

1  function out = ReadEXForce(filename)
2  % 读取波浪力
3  fp=fopen(filename,'r');
4  line=fgetl(fp);
5  str=strsplit(line,{' '});
6  N_fre=str2double(str{1});
7  N_dir=str2double(str{2});
8  line=fgetl(fp);
9  C=textscan(fp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f");
10 fclose(fp);
11 out.fre=C{1,1}(1:N_dir:end,:);
12 out.dir=C{1,2}(1:1:N_dir,:);
13 out.EXForce=cell(N_dir,6);
14 for k=1:1:6
15     EXF(:,k)=complex(C{1,2*k+1},C{1,2*k+2});
16 end
17 for i=1:1:N_dir
18     for k=1:1:6
19         out.EXForce{i,k}=EXF(i:N_dir:end,k);
20     end
21 end
22 end

```

5.2.2 读取水动力系数(**ReadHydroCoef.m**)

读取附加质量、阻尼系数结果

```

1 function out = ReadHydroCoef(filename)
2 % 读取水动力系数
3 fp=fopen(filename, 'r');
4 C=textscan(fp, "%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f", 'headerlines', 2);
5 fclose(fp);
6 out.fre=C{1,1};
7 out.AD=cell(6,6);
8 for i=1:1:6
9     for j=1:1:6
10         out.AD{i,j}=C{1, (i-1)*6+j+1};
11     end
12 end
13 end

```

5.2.3 读取监测点结果(ReadOffBodyPoints.m)

```

1 function out= ReadOffBodyPoints(filename)
2 %读取监测点数据
3 fp=fopen(filename, 'r');
4 line=fgetl(fp);
5 str=strsplit(line, {' '});
6
7 N_fre=str2double(str{1});
8 N_dir=str2double(str{2});
9 N_op =str2double(str{3});
10
11 C=textscan(fp, "%f %f %f", N_op);
12 C1=textscan(fp, "%f %f %f %f %f");
13 fclose(fp);
14 out.Points=[C{1,1}, C{1,2}, C{1,3}];
15 out.Eta=cell(1, N_dir);
16 out.Phi=cell(1, N_dir);
17
18 Fre=C1{1,1};
19 Dir=C1{1,2};
20 Phi=complex(C1{1,3}, C1{1,4});
21 Eta=C1{1,5};
22
23 out.Fre=Fre(1:N_dir*N_op:end, :);
24 out.Dir=sort(Dir(1:N_fre*N_op:end, :));
25
26 Phi=reshape(Phi, N_op, []);
27 Eta=reshape(Eta, N_op, []);
28 for i=1:1:N_dir
29     out.Phi{1,i}=Phi(:, i:N_dir:end).';
30     out.Eta{1,i}=Eta(:, i:N_dir:end).';
31 end
32 for i=1:1:N_dir
33     out.eta{1,i}=complex(0,1)/9.81*out.Phi{1,i}.*repmat(out.Fre, 1, N_op);
34 end
35 end

```

5.2.4 读取面元文件

5.2.4.1 读取.txt面元文件

```

1 function TxT = ReadPanelTxT(filename)
2 % 读取面元txt文件
3 tic
4 fp=fopen(filename,'r');
5 line=fgetl(fp);
6 str=strsplit(line,{' '});
7 Num=str2double(str{1});
8 Type=str2double(str{2});
9 line=fgetl(fp);
10 if Type==4
11     C=textscan(fp,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f');
12 else
13     C=textscan(fp,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f');
14 end
15 fclose(fp);
16 TxT.Centroid=[C{1,1},C{1,2},C{1,3}];
17 TxT.Normal=[C{1,4},C{1,5},C{1,6}];
18 TxT.ds=[C{1,7}];
19 TxT.P1=[C{1,8},C{1,9},C{1,10}];
20 TxT.P2=[C{1,11},C{1,12},C{1,13}];
21 TxT.P3=[C{1,14},C{1,15},C{1,16}];
22 if Type==4
23     TxT.P4=[C{1,17},C{1,18},C{1,19}];
24     for i=1:1:Num
25         TxT.Patches(4*i-3:4*i,:)=[TxT.P1(i,:);TxT.P2(i,:);TxT.P3(i,:);TxT.P4(i,:)];
26     end
27 else
28     for i=1:1:Num
29         TxT.Patches(3*i-2:3*i,:)=[TxT.P1(i,:);TxT.P2(i,:);TxT.P3(i,:)];
30     end
31 end
32 TxT.Type=Type;
33 TxT.Num=Num;
34 toc
35 end

```

5.2.4.2 读取.gdf面元文件

```

1 function [out] = ReadGDF(filename)
2 %READGDF 读取GDF文件,并计算相关参数
3 fp=fopen(filename,'rt');
4 C=textscan(fp,'%f %f %f ','headerlines',4);
5 out.x=C{1,1};out.y=C{1,2};out.z=C{1,3};
6 fclose(fp);
7
8 out.Num=length(out.x)/4;
9 for i=1:1:out.Num*4
10     index=ceil(i/4)*4;
11     Point1=[out.x(index-3),out.y(index-3),out.z(index-3)];
12     Point2=[out.x(index-2),out.y(index-2),out.z(index-2)];
13     Point3=[out.x(index-1),out.y(index-1),out.z(index-1)];
14     Point4=[out.x(index),out.y(index),out.z(index)];
15     out.Patches(index-3:index,:)=[Point1;Point2;Point3;Point4];
16 end
17 out.ds=zeros(out.Num,1);

```

```

18 out.Centroid=zeros(out.Num,3);
19 out.Normal=zeros(out.Num,3);
20 for i=1:1:out.Num
21     P1=out.Patches(4*i-3,:);
22     P2=out.Patches(4*i-2,:);
23     P3=out.Patches(4*i-1,:);
24     P4=out.Patches(4*i-0,:);
25     n1=P1-P3;n2=P2-P4;
26     sintheta=sqrt(1-(n1*n2'/(norm(n1)*norm(n2)))^2);
27     out.ds(i)=0.5*(norm(n1)*norm(n2)*sintheta);
28     out.Centroid(i,:)=(P1+P2+P3+P4)/4;
29     out.Normal(i,:)=cross(n1,n2)/(norm(cross(n1,n2)));
30 end
31 end

```

5.2.5 读取实数，复数矩阵文件

5.2.5.1 读取实数(.txt)文件(ReadMat.m)

```

1 function mat = ReadMat(filename,M)
2 tic
3 fp=fopen(filename,'r');
4 line=fgetl(fp);
5 str=strsplit(line',{' '});
6 C=textscan(fp,'%f,%f');
7 fclose(fp);
8 x=reshape(C{1,1}.'. ',M,[]).';
9 fx=reshape(C{1,2}.'. ',M,[]).';
10 toc
11 mat.x=x;
12 mat.fx=fx;
13 end

```

5.2.5.2 读取复数(.txt)文件(ReadMatC.m)

```

1 function mat = ReadMatC(filename)
2 tic
3 fp=fopen(filename,'r');
4 line=fgetl(fp);
5 str=strsplit(line',{' '});
6 M=str2double(str{1});
7 N=str2double(str{2});
8 C=textscan(fp,'%f,%f');
9 fclose(fp);
10 mat=complex(reshape(C{1,1},N,M),reshape(C{1,2},N,M));
11 mat=mat.';
12 toc
13 end

```

5.2.6 读取系泊系统文件

5.2.6.1 读取系泊缆绳位置信息(readLineOut.m)

```

1  function out = readLineOut(filename)
2  % 读取系泊out文件
3  fp=fopen(filename);
4  count=1;
5  line=fgetl(fp); % 从文件读行
6  label=strsplit(line, '\t');
7  label(end)=[];
8  Nnode=(length(label)-1)/3;
9  dim=[];
10 for i=1:length(label)
11     dim=[dim, '%f '];
12 end
13 C=textscan(fp, dim, 'headerlines', 2);
14 fclose(fp);
15 out.t=C{1,1};
16 Nt=length(out.t);
17 out.Node=cell(Nt, 1);
18 tmp=zeros(Nnode, 3);
19 for tt=1:Nt
20     for k=1:Nnode
21         tmp(k, :)=[C{1, 3*k-1}(tt), C{1, 3*k}(tt), C{1, 3*k+1}(tt)];
22     end
23     out.Node{tt, 1}=tmp;
24 end
25 fprintf("readlineout [ %s ]\n", filename);
26 end

```

5.2.6.2 读取系泊缆绳受力信息(readLineOutForce.m)

```

1  function out = readLineOutForce(filename)
2  % 读取out文件
3  fp=fopen(filename);
4  line=fgetl(fp); % 从文件读行
5  label=strsplit(line, '\t');
6  label(end)=[];
7  NN=length(label);
8  dim=[];
9  for i=1:length(label)
10     dim=[dim, '%f '];
11 end
12 C=textscan(fp, dim, 'headerlines', 1);
13 fclose(fp);
14 out.t=C{1,1};
15 out.fx=[];
16 for i=2:NN
17     out.fx(:, i-1)=C{1, i};
18 end
19 fprintf("readlineoutforce [ %s ]\n", filename);
20 end

```

5.2.7 时历数据统计分析

5.2.7.1 计算有义值(Cal_Hs.m)

```

1 function Hs = Cal_Hs(Eta)
2 %CAL_HS 计算有义波高
3 N=length(Eta);
4 Pset=UpCrossZeroPoints(Eta);
5 N_P=length(Pset);
6 H=[];
7 for k=1:1:N_P-1
8     eta_tmp=Eta(Pset(k):Pset(k+1));
9     H=[H;max(eta_tmp)-min(eta_tmp)];
10 end
11 %降序排序
12 H=sort(H,'descend');
13 %最大三分之一平均值
14 Num=floor(length(H)/3);
15 Hs=mean(H(1:Num));
16 function Pset=UpCrossZeroPoints(Eta)
17     %计算上跨零点
18     Pset=[];
19     for i=1:1:N-1
20         if(Eta(i)<=0&&Eta(i+1)>=0)
21             Pset=[Pset;i];
22         end
23     end
24 end
25 end

```

5.2.7.2 谱分析

```

1 function [omega,res] = getWelch(data,dt,N)
2     % 功能: FFT加窗处理
3     % 输入: data 数据段
4     %         dt 时间间隔
5     %         N 每段处理数据长度N
6     fs=1/dt;
7     % 每段数据之间的重叠区域宽度为: N*1/2
8     % [p,f] = pwelch(data,ones(N,1),N/2,N,fs);
9     [p,f] = pwelch(data,hamming(N),round(N*0.8),N,fs);
10    omega = f*2*pi;
11    res = p/2/pi;
12 end

```

```

1 function out = myFFT(time,data)
2 % 功能: FFT
3 % 输入: time: 时间
4 %         data: 数据
5 % 输出: out.Fre:频率 rad
6 %         out.Ampti:幅值
7 %         out.Phase:相位
8     N=length(data);
9     fft_Data=fft(data);
10    Ampti = abs(fft_Data) / (N)*2;
11    Phase = atan2(imag(fft_Data),real(fft_Data));
12    dt=time(2)-time(1);
13    Fre=(0:1:(length(time)-1))/(N*dt)*2*pi;
14    out.Fre=Fre;

```



```

15     out.Ampti=Ampti;
16     out.Phase=Phase;
17 end

```

5.3 生成自由面监测点

```

1  clc;clear;
2  folderpath='FolderWaterLines';
3  WL1=getWaterLine([folderpath,'\','Barge1_WaterLine.txt']);
4  WL2=getWaterLine([folderpath,'\','Barge2_WaterLine.txt']);
5  WL3=getWaterLine([folderpath,'\','Barge3_WaterLine.txt']);
6  %%
7  WaterLines={WL1,WL2,WL3};
8  Xrange=[-3:0.2:-2,-2:0.1:2,2:0.2:3];
9  Yrange=[-5:0.2:-2,-2:0.1:2,2:0.2:5];
10 Xrange=unique(Xrange);
11 Yrange=unique(Yrange);
12 outfile='Three_barges';
13 ofPoints = genFsPoints(Xrange,Yrange,WaterLines,outfile);
14
15 %% 相关函数
16 function [out] = getWaterLine(filename)
17 fp=fopen(filename,'rt');
18 line=fgetl(fp);
19 str=strsplit(line,{' '});
20 N=str2double(str{1});
21 line=fgetl(fp);
22 C=textscan(fp,"%d,%f,%f,%f,%f,%f,%f");
23 out.Node1=[C{1,2},C{1,3},C{1,4}];
24 out.Node2=[C{1,5},C{1,6},C{1,7}];
25 out.Vec=out.Node2-out.Node1;
26 fclose(fp);
27 end
28
29 function ofPoints = genFsPoints(Xrange,Yrange,WaterLines,outfile)
30 %GENFSPPOINTS 输入WaterLine文件，划分自由面网格空间点
31 %   Xrange: X轴范围
32 %   Yrange: Y轴范围
33 %   WaterLines: 水线面文件：注：单一的WL文件为封闭的单连通区域
34 %   outfile: 输出至文件
35 if isempty(WaterLines)
36     warning('WaterLines is empty!!!');
37     return;
38 end
39 % 划分网格
40 [X,Y]=meshgrid(Xrange,Yrange);
41 Grid.X=X';
42 Grid.Y=Y';
43 for i=1:length(Xrange)
44     for j=1:length(Yrange)
45         pos=[Grid.X(i,j),Grid.Y(i,j),0];
46         for k=1:length(WaterLines)
47             if(IsInside(WaterLines{k},pos))
48                 Grid.X(i,j)=nan;
49                 Grid.Y(i,j)=nan;
50             end
51         end
52     end
53 end
54 fprintf('划分网格成功! \n');

```

```

55 fprintf('显示绘图! \n');
56 figure(996)
57 plot(Grid.X,Grid.Y,'ro');hold on
58 xlabel("X");ylabel("Y");axis equal;
59
60 % grid
61 ofPoints=[];
62 for i=1:1:length(Xrange)
63     for j=1:1:length(Yrange)
64         if(~isnan(Grid.X(i,j)))
65             ofPoints=[ofPoints; [Grid.X(i,j),Grid.Y(i,j),0]];
66         end
67     end
68 end
69
70 eval(['save ',outfilename,num2str(size(ofPoints,1),'_%.d'),' Grid']);
71 fprintf('保存mat网格数据! \n');
72 genPointFile([outfilename,num2str(size(ofPoints,1),'_%.d.txt')],ofPoints);
73 fprintf('保存txt网格数据! \n');
74
75 end
76
77 function re=IsInside(WL,Pos)
78 % 判断点Pos是否在封闭曲线WL内
79 N=size(WL.Node1,1);
80 re_sum=zeros(1,3);
81 for i=1:1:N
82     N1=WL.Node1(i,:);
83     N2=WL.Node2(i,:);
84     center=(N1+N2)/2;
85     r=center-Pos;
86     vec=N2-N1;
87     tmp=cross(vec,r);
88     tmp=tmp/norm(tmp);
89     N1p=N1-Pos;
90     N2p=N2-Pos;
91     cosTheta=dot(N1p,N2p)/(norm(N1p)*norm(N2p));
92     theta=acos(cosTheta);
93     re_sum=re_sum+tmp*theta;
94 end
95
96 if norm(re_sum)<1.e-3
97     re=false;
98 else
99     re=true;
100 end
101 % 位于边界
102 if isnan(norm(re_sum))
103     re=true;
104 end
105 end
106
107 function genPointFile(filename,points)
108 % 导出至文件
109 N=size(points,1);
110 fp=fopen(filename,'w');
111 fprintf(fp,'%.d\n',N);
112 fprintf(fp,'%10f %10f %10f\n',points');
113 fclose(fp);
114 end

```

6. 参考文献

- [1] 陈天文. 旁靠外输系统水动力性能数值模拟研究[D].上海交通大学,2023.