

# hydroMex3 Reference Manual

Generated by Doxygen 1.4.6-NO

Fri Jun 9 15:06:50 2006



# Contents

<b>1</b>	<b>hydroMex3 Namespace Index</b>	<b>1</b>
1.1	hydroMex3 Namespace List . . . . .	1
<b>2</b>	<b>hydroMex3 Hierarchical Index</b>	<b>3</b>
2.1	hydroMex3 Class Hierarchy . . . . .	3
<b>3</b>	<b>hydroMex3 Data Structure Index</b>	<b>5</b>
3.1	hydroMex3 Data Structures . . . . .	5
<b>4</b>	<b>hydroMex3 File Index</b>	<b>7</b>
4.1	hydroMex3 File List . . . . .	7
<b>5</b>	<b>hydroMex3 Namespace Documentation</b>	<b>9</b>
5.1	std Namespace Reference . . . . .	9
<b>6</b>	<b>hydroMex3 Data Structure Documentation</b>	<b>11</b>
6.1	datas Struct Reference . . . . .	11
6.2	joy_parameters Struct Reference . . . . .	34
6.3	ODE_data Struct Reference . . . . .	37
6.4	parameters Struct Reference . . . . .	40
6.5	seaDatas Struct Reference . . . . .	41
6.6	SolidObject Struct Reference . . . . .	43
<b>7</b>	<b>hydroMex3 File Documentation</b>	<b>47</b>
7.1	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/CVODEincludes.h File Reference	47
7.2	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.cpp File Reference . . . . .	49
7.3	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.h File Reference . . . . .	53
7.4	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/aero_coeff_- control.cpp File Reference . . . . .	55
7.5	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/aero_coeff_- control.h File Reference . . . . .	57

7.6	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_dyn.cpp File Reference . . . . .	58
7.7	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_dyn.h File Reference . . . . .	60
7.8	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_geo.cpp File Reference . . . . .	61
7.9	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_geo.h File Reference . . . . .	64
7.10	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/compute_state_- dot.cpp File Reference . . . . .	66
7.11	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/compute_state_- dot.h File Reference . . . . .	68
7.12	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/dataFillings.cpp File Reference . . . . .	69
7.13	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/dataFillings.h File Reference . . . . .	72
7.14	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/Datas.h File Refer- ence . . . . .	75
7.15	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/get_mex_- arguments.cpp File Reference . . . . .	77
7.16	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/get_mex_- arguments.h File Reference . . . . .	82
7.17	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/hydro_includes.h File Reference . . . . .	86
7.18	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/return_mex_- arguments.cpp File Reference . . . . .	87
7.19	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/return_mex_- arguments.h File Reference . . . . .	91
7.20	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydroMEX3.cpp File Reference .	93
7.21	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/joystick_PC.cpp File Reference . . . . .	96
7.22	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/joystick_PC.h File Refer- ence . . . . .	99
7.23	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/stdafx.cpp File Reference	101
7.24	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/stdafx.h File Reference . .	102
7.25	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/graphics_main.cpp File Reference . . . . .	103
7.26	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/graphics_relative.h File Reference . . . . .	113
7.27	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/Sea.h File Reference	118
7.28	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/solid_objects.cpp File Reference . . . . .	121

---

7.29	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/solid_objects.h File Reference . . . . .	130
7.30	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/wave_z_- compute_byHand.cpp File Reference . . . . .	136
7.31	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/timing/timing.h File Reference . .	140
7.32	D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/timing/timing_PC.cpp File Refer- ence . . . . .	142



# Chapter 1

## hydroMex3 Namespace Index

### 1.1 hydroMex3 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">std</a> . . . . .	9
-------------------------------	---





# Chapter 2

## hydroMex3 Hierarchical Index

### 2.1 hydroMex3 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

datas . . . . .	11
joy_parameters . . . . .	34
ODE_data . . . . .	37
parameters . . . . .	40
seaDatas . . . . .	41
SolidObject . . . . .	43



# Chapter 3

## hydroMex3 Data Structure Index

### 3.1 hydroMex3 Data Structures

Here are the data structures with brief descriptions:

<a href="#">datas</a> (Datas struct declaration ) . . . . .	11
<a href="#">joy_parameters</a> (Struct for joystick things ) . . . . .	34
<a href="#">ODE_data</a> (Datas relative to the ODE solver ) . . . . .	37
<a href="#">parameters</a> (Parameters struct declaration ) . . . . .	40
<a href="#">seaDatas</a> . . . . .	41
<a href="#">SolidObject</a> (3D object description ) . . . . .	43



# Chapter 4

## hydroMex3 File Index

### 4.1 hydroMex3 File List

Here is a list of all files with brief descriptions:

D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/CVODEincludes.h . . . . .	47
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.cpp . . . . .	49
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.h . . . . .	53
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydroMEX3.cpp . . . . .	93
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/aero_coeff_control.cpp .	55
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/aero_coeff_control.h . . .	57
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_dyn.cpp . . . .	58
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_dyn.h . . . .	60
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_geo.cpp . . . .	61
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/analytics_geo.h . . . .	64
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/compute_state_dot.cpp .	66
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/compute_state_dot.h . . .	68
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/dataFillings.cpp . . . .	69
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/dataFillings.h . . . .	72
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/Datas.h . . . . .	75
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/get_mex_arguments.cpp .	77
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/get_mex_arguments.h . .	82
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/hydro_includes.h . . . .	86
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/return_mex_- arguments.cpp . . . . .	87
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/return_mex_arguments.h	91
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/joystick_PC.cpp . . . . .	96
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/joystick_PC.h . . . . .	99
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/stdafx.cpp . . . . .	101
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/stdafx.h . . . . .	102
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/graphics_main.cpp . . .	103
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/graphics_relative.h . . .	113
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/Sea.h . . . . .	118
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/solid_objects.cpp . . . .	121
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/solid_objects.h . . . .	130
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/wave_z_compute_by- Hand.cpp . . . . .	136
D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/timing/timing.h . . . . .	140

D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/timing/[timing\\_PC.cpp](#) . . . . . 142

## **Chapter 5**

# **hydroMex3 Namespace Documentation**

### **5.1 std Namespace Reference**





## Chapter 6

# hydroMex3 Data Structure Documentation

### 6.1 datas Struct Reference

datas struct declaration

```
#include <Datas.h>
```

#### Data Fields

- double [x\\_foils](#)
- double [tanCal\\_yaw0](#)
- double [tanAngle\\_sail](#)
- double [G\\_shift](#)
- double [alpha\\_foils](#)
- double [Wind\\_angle](#)
- double [z\\_foils](#)
- double [g](#)
- double [Long](#)
- double [tanCal\\_left](#)
- double [tanCal\\_right](#)
- double [tanCal\\_pitch](#)
- double [alpha\\_left](#)
- double [alpha\\_right](#)
- double [y\\_foil\\_left](#)
- double [y\\_foil\\_right](#)
- double [L\\_mast](#)
- double [x\\_sail](#)
- double [y\\_sail](#)
- double [z\\_sail](#)
- double [A\\_sail](#)
- double [Baume](#)
- double [A\\_structure](#)
- double [CD\\_structure](#)

- double [Ix](#)
- double [Iy](#)
- double [Iz](#)
- double [M](#)
- double [E\\_foils](#)
- double [Chord](#)
- double [Chord\\_min](#)
- double [Chord\\_max](#)
- double [epsilon](#)
- double [rho](#)
- double [rho\\_air](#)
- double [E\\_yaw](#)
- double [Env\\_pitch](#)
- double [Chord\\_yaw](#)
- double [Chord\\_pitch](#)
- double [A\\_pitch](#)
- double [x\\_yaw](#)
- double [y\\_yaw](#)
- double [z\\_yaw](#)
- double [x\\_pitch](#)
- double [y\\_pitch](#)
- double [z\\_pitch](#)
- double [Wind](#)
- double [Wind\\_x](#)
- double [Wind\\_y](#)
- double [Env](#)
- double [Lambda0](#)
- double [Wave\\_angle0](#)
- double [Wave\\_amp0](#)
- double [V\\_wave0](#)
- double [Lambda](#) [2]
- double [Wave\\_angle](#) [2]
- double [Wave\\_amp](#) [2]
- double [V\\_wave](#) [2]
- int [N](#)
- double [Lambda1](#)
- double [Lambda2](#)
- double [Wave\\_amp1](#)
- double [Wave\\_amp2](#)
- double [V\\_wave1](#)
- double [V\\_wave2](#)
- double [Wave\\_angle1](#)
- double [Wave\\_angle2](#)
- double [x](#)
- double [y](#)
- double [z](#)
- double [phi](#)
- double [theta](#)
- double [psi](#)
- double [dx](#)

- double [dy](#)
- double [dz](#)
- double [dphi](#)
- double [dtheta](#)
- double [dpsi](#)
- double [x\\_cam](#)
- double [y\\_cam](#)
- double [z\\_cam](#)
- double [dx\\_cam](#)
- double [dy\\_cam](#)
- double [dz\\_cam](#)
- double [tanCal\\_yaw](#)
- double [EE\\_left](#)
- double [EE\\_right](#)
- double [EE\\_yaw](#)
- double [EE\\_left\\_flat](#)
- double [EE\\_right\\_flat](#)
- double [EE\\_yaw\\_flat](#)
- double [Fac\\_update](#)
- double [Equ\\_left](#)
- double [Equ\\_right](#)
- double [Equ\\_yaw](#)
- double [dEqu\\_left](#)
- double [dEqu\\_right](#)
- double [dEqu\\_yaw](#)
- int [k](#)
- double [Tan\\_AoA\\_left](#)
- double [Tan\\_AoA\\_right](#)
- double [Tan\\_AoA\\_pitch](#)
- double [Tan\\_AoA\\_yaw](#)
- double [Tan\\_AoA\\_sail](#)
- double [Force\\_G\\_E1](#)
- double [Force\\_G\\_E2](#)
- double [Force\\_G\\_E3](#)
- double [Torque\\_G\\_E1](#)
- double [Torque\\_G\\_E2](#)
- double [Torque\\_G\\_E3](#)
- double [Force\\_left](#) [3]
- double [Force\\_right](#) [3]
- double [Force\\_pitch](#) [3]
- double [Force\\_yaw](#) [3]
- double [Force\\_sail](#) [3]
- double [Force\\_drag](#) [3]
- double [ddq](#) [6]
- double [t](#)
- double [A\\_left](#)
- double [CD\\_left](#)
- double [CL\\_left](#)
- double [CD\\_right](#)
- double [A\\_right](#)

- double [CL\\_right](#)
- double [CD\\_pitch](#)
- double [CL\\_pitch](#)
- double [A\\_yaw](#)
- double [CD\\_yaw](#)
- double [CL\\_yaw](#)
- double [CD\\_sail](#)
- double [CL\\_sail](#)
- double [Chord\\_right](#)
- double [Chord\\_left](#)
- double [AR\\_left](#)
- double [AR\\_right](#)
- double [AR\\_yaw](#)
- double [AR\\_pitch](#)
- double [AR\\_sail](#)
- double [gamma](#)
- double [dd\\_cam](#) [3]
- double [d\\_cam](#) [3]
- double [L\\_nose](#)
- double [verif](#) [26]
- float [target\\_x](#)
- float [target\\_y](#)
- float [Vslider\\_x](#)
- float [Hslider\\_x](#)
- double [Cal\\_left\\_neutral](#)
- double [Cal\\_right\\_neutral](#)
- double [Cal\\_pitch\\_neutral](#)
- double [Cal\\_yaw\\_neutral](#)
- double [angle\\_girouette](#)
- double [max\\_abs\\_angle\\_baume](#)

### 6.1.1 Detailed Description

datas struct declaration

datas contains all variables related to the model (a lot) plus some other.

Many functions take a pointer \*d to this struct

Definition at line 44 of file Datas.h.

### 6.1.2 Field Documentation

#### 6.1.2.1 double [datas::A\\_left](#)

Definition at line 220 of file Datas.h.

Referenced by [aero\\_coeff\\_control\(\)](#), [forces\\_all\\_compute\(\)](#), and [forces\\_compute\(\)](#).

**6.1.2.2 double `datas::A_pitch`**

Definition at line 101 of file Datas.h.

Referenced by `aero_coeff_control()`, `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.3 double `datas::A_right`**

Definition at line 224 of file Datas.h.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.4 double `datas::A_sail`**

Definition at line 73 of file Datas.h.

Referenced by `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.5 double `datas::A_structure`**

Definition at line 75 of file Datas.h.

Referenced by `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.6 double `datas::A_yaw`**

Definition at line 228 of file Datas.h.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.7 double `datas::alpha_foils`**

Definition at line 50 of file Datas.h.

Referenced by `data_update_param()`.

**6.1.2.8 double `datas::alpha_left`**

Definition at line 62 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.9 double `datas::alpha_right`**

Definition at line 63 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.10 double [datas::angle\\_girouette](#)**

Definition at line 267 of file Datas.h.

Referenced by `data_update_param()`, `draw_graphics()`, and `Girouette_draw()`.

**6.1.2.11 double [datas::AR\\_left](#)**

Definition at line 238 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.12 double [datas::AR\\_pitch](#)**

Definition at line 241 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.13 double [datas::AR\\_right](#)**

Definition at line 239 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.14 double [datas::AR\\_sail](#)**

Definition at line 242 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.15 double [datas::AR\\_yaw](#)**

Definition at line 240 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.16 double [datas::Baume](#)**

Definition at line 74 of file Datas.h.

Referenced by `angles_compute()`, `data_init()`, `forces_all_compute()`, `forces_compute()`, and `Sail_create_-and_draw()`.

**6.1.2.17 double [datas::Cal\\_left\\_neutral](#)**

Definition at line 261 of file Datas.h.

Referenced by `angles_update()`, and `data_update_param()`.

**6.1.2.18 double [datas::Cal\\_pitch\\_neutral](#)**

Definition at line 263 of file Datas.h.

Referenced by `angles_update()`, and `data_update_param()`.

#### 6.1.2.19 double `datas::Cal_right_neutral`

Definition at line 262 of file `Datas.h`.

Referenced by `angles_update()`, and `data_update_param()`.

#### 6.1.2.20 double `datas::Cal_yaw_neutral`

Definition at line 264 of file `Datas.h`.

Referenced by `angles_update()`, and `data_update_param()`.

#### 6.1.2.21 double `datas::CD_left`

Definition at line 221 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

#### 6.1.2.22 double `datas::CD_pitch`

Definition at line 226 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

#### 6.1.2.23 double `datas::CD_right`

Definition at line 223 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

#### 6.1.2.24 double `datas::CD_sail`

Definition at line 231 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

#### 6.1.2.25 double `datas::CD_structure`

Definition at line 76 of file `Datas.h`.

Referenced by `data_init()`, `forces_all_compute()`, and `forces_compute()`.

#### 6.1.2.26 double `datas::CD_yaw`

Definition at line 229 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.27 double `datas::Chord`**

Definition at line 84 of file Datas.h.

Referenced by `aero_coeff_control()`, and `data_init()`.

**6.1.2.28 double `datas::Chord_left`**

Definition at line 237 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.29 double `datas::Chord_max`**

Definition at line 88 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.30 double `datas::Chord_min`**

Definition at line 87 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.31 double `datas::Chord_pitch`**

Definition at line 99 of file Datas.h.

Referenced by `data_init()`.

**6.1.2.32 double `datas::Chord_right`**

Definition at line 236 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.33 double `datas::Chord_yaw`**

Definition at line 98 of file Datas.h.

Referenced by `data_init()`.

**6.1.2.34 double `datas::CL_left`**

Definition at line 222 of file Datas.h.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.



**6.1.2.35 double `datas::CL_pitch`**

Definition at line 227 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.36 double `datas::CL_right`**

Definition at line 225 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.37 double `datas::CL_sail`**

Definition at line 232 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.38 double `datas::CL_yaw`**

Definition at line 230 of file `Datas.h`.

Referenced by `aero_coeff_control()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.39 double `datas::d_cam[3]`**

Definition at line 247 of file `Datas.h`.

**6.1.2.40 double `datas::dd_cam[3]`**

Definition at line 246 of file `Datas.h`.

**6.1.2.41 double `datas::ddq[6]`**

Definition at line 212 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `f()`.

**6.1.2.42 double `datas::dEqu_left`**

Definition at line 182 of file `Datas.h`.

Referenced by `compute_state_dot()`, and `equ_compute()`.

**6.1.2.43 double `datas::dEqu_right`**

Definition at line 183 of file `Datas.h`.

Referenced by `compute_state_dot()`, and `equ_compute()`.

**6.1.2.44 double `datas::dEqu_yaw`**

Definition at line 184 of file `Datas.h`.

Referenced by `compute_state_dot()`, and `equ_compute()`.

**6.1.2.45 double `datas::dphi`**

Definition at line 151 of file `Datas.h`.

Referenced by `acceleration_compute()`, `angles_compute()`, `data_update_state()`, `f()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.46 double `datas::dpsi`**

Definition at line 153 of file `Datas.h`.

Referenced by `acceleration_compute()`, `angles_compute()`, `data_update_state()`, `f()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.47 double `datas::dtheta`**

Definition at line 152 of file `Datas.h`.

Referenced by `acceleration_compute()`, `angles_compute()`, `data_update_state()`, `f()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.48 double `datas::dx`**

Definition at line 147 of file `Datas.h`.

Referenced by `angles_compute()`, `data_update_state()`, `DrawGLScene()`, `f()`, `forces_all_compute()`, `forces_compute()`, `Girouette_draw()`, and `RelativeWind_draw()`.

**6.1.2.49 double `datas::dx_cam`**

Definition at line 159 of file `Datas.h`.

**6.1.2.50 double `datas::dy`**

Definition at line 148 of file `Datas.h`.

Referenced by `angles_compute()`, `data_update_state()`, `DrawGLScene()`, `f()`, `forces_all_compute()`, `forces_compute()`, `Girouette_draw()`, and `RelativeWind_draw()`.

**6.1.2.51 double `datas::dy_cam`**

Definition at line 160 of file `Datas.h`.

**6.1.2.52 double `datas::dz`**

Definition at line 149 of file Datas.h.

Referenced by `angles_compute()`, `data_update_state()`, `f()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.53 double `datas::dz_cam`**

Definition at line 161 of file Datas.h.

**6.1.2.54 double `datas::E_foils`**

Definition at line 83 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.55 double `datas::E_yaw`**

Definition at line 94 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `data_init()`, `data_update_param()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.56 double `datas::EE_left`**

Definition at line 168 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `compute_state_dot()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.57 double `datas::EE_left_flat`**

Definition at line 171 of file Datas.h.

Referenced by `compute_state_dot()`, and `EE_flat_compute()`.

**6.1.2.58 double `datas::EE_right`**

Definition at line 169 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `compute_state_dot()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.59 double `datas::EE_right_flat`**

Definition at line 172 of file Datas.h.

Referenced by `compute_state_dot()`, and `EE_flat_compute()`.

**6.1.2.60 double `datas::EE_yaw`**

Definition at line 170 of file Datas.h.

Referenced by `aero_coeff_control()`, `angles_compute()`, `compute_state_dot()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.61 double `datas::EE_yaw_flat`**

Definition at line 173 of file Datas.h.

Referenced by `compute_state_dot()`, and `EE_flat_compute()`.

**6.1.2.62 double `datas::Env`**

Definition at line 115 of file Datas.h.

Referenced by `data_update_param()`.

**6.1.2.63 double `datas::Env_pitch`**

Definition at line 96 of file Datas.h.

Referenced by `data_init()`.

**6.1.2.64 double `datas::epsilon`**

Definition at line 89 of file Datas.h.

Referenced by `data_init()`.

**6.1.2.65 double `datas::Equ_left`**

Definition at line 179 of file Datas.h.

Referenced by `compute_state_dot()`, and `equ_compute()`.

**6.1.2.66 double `datas::Equ_right`**

Definition at line 180 of file Datas.h.

Referenced by `compute_state_dot()`, and `equ_compute()`.

**6.1.2.67 double `datas::Equ_yaw`**

Definition at line 181 of file Datas.h.

Referenced by `compute_state_dot()`, and `equ_compute()`.

**6.1.2.68 double `datas::Fac_update`**

Definition at line 176 of file Datas.h.

Referenced by `compute_state_dot()`, `data_init()`, and `data_update_param()`.

#### 6.1.2.69 double `datas::Force_drag[3]`

Definition at line 209 of file `Datas.h`.

Referenced by `forces_all_compute()`.

#### 6.1.2.70 double `datas::Force_G_E1`

Definition at line 196 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `forces_compute()`.

#### 6.1.2.71 double `datas::Force_G_E2`

Definition at line 197 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `forces_compute()`.

#### 6.1.2.72 double `datas::Force_G_E3`

Definition at line 198 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `forces_compute()`.

#### 6.1.2.73 double `datas::Force_left[3]`

Definition at line 204 of file `Datas.h`.

Referenced by `forces_all_compute()`.

#### 6.1.2.74 double `datas::Force_pitch[3]`

Definition at line 206 of file `Datas.h`.

Referenced by `forces_all_compute()`.

#### 6.1.2.75 double `datas::Force_right[3]`

Definition at line 205 of file `Datas.h`.

Referenced by `forces_all_compute()`.

#### 6.1.2.76 double `datas::Force_sail[3]`

Definition at line 208 of file `Datas.h`.

Referenced by `forces_all_compute()`.

**6.1.2.77 double `datas::Force_yaw[3]`**

Definition at line 207 of file Datas.h.

Referenced by `forces_all_compute()`.

**6.1.2.78 double `datas::g`**

Definition at line 54 of file Datas.h.

Referenced by `acceleration_compute()`, and `data_init()`.

**6.1.2.79 double `datas::G_shift`**

Definition at line 49 of file Datas.h.

Referenced by `data_update_param()`.

**6.1.2.80 double `datas::gamma`**

Definition at line 243 of file Datas.h.

Referenced by `aero_coeff_control()`.

**6.1.2.81 float `datas::Hslider_x`**

Definition at line 258 of file Datas.h.

Referenced by `angles_update()`, `data_init()`, `draw_graphics()`, and `DrawGLScene()`.

**6.1.2.82 double `datas::Ix`**

Definition at line 78 of file Datas.h.

Referenced by `acceleration_compute()`, and `data_init()`.

**6.1.2.83 double `datas::Iy`**

Definition at line 79 of file Datas.h.

Referenced by `acceleration_compute()`, and `data_init()`.

**6.1.2.84 double `datas::Iz`**

Definition at line 80 of file Datas.h.

Referenced by `acceleration_compute()`, and `data_init()`.

**6.1.2.85 int `datas::k`**

Definition at line 186 of file Datas.h.

**6.1.2.86 double `datas::L_mast`**

Definition at line 68 of file Datas.h.

Referenced by `data_init()`, `data_update_param()`, `RelativeWind_draw()`, and `Sail_create_and_draw()`.

**6.1.2.87 double `datas::L_nose`**

Definition at line 251 of file Datas.h.

Referenced by `data_init()`.

**6.1.2.88 double `datas::Lambda[2]`**

Definition at line 122 of file Datas.h.

Referenced by `data_update_param()`.

**6.1.2.89 double `datas::Lambda0`**

Definition at line 117 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.90 double `datas::Lambda1`**

Definition at line 129 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.91 double `datas::Lambda2`**

Definition at line 130 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.92 double `datas::Long`**

Definition at line 56 of file Datas.h.

Referenced by `Boat_create()`, `data_init()`, and `data_update_param()`.

**6.1.2.93 double `datas::M`**

Definition at line 81 of file Datas.h.

Referenced by `acceleration_compute()`, and `data_init()`.

**6.1.2.94 double `datas::max_abs_angle_baume`**

Definition at line 268 of file Datas.h.

Referenced by `data_update_param()`, and `draw_graphics()`.

**6.1.2.95 int [datas::N](#)**

Definition at line 127 of file Datas.h.

Referenced by `data_update_param()`.

**6.1.2.96 double [datas::phi](#)**

Definition at line 143 of file Datas.h.

Referenced by `acceleration_compute()`, `angles_compute()`, `data_update_state()`, `DrawGLScene()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, `forces_compute()`, and `RelativeWind_draw()`.

**6.1.2.97 double [datas::psi](#)**

Definition at line 145 of file Datas.h.

Referenced by `acceleration_compute()`, `angles_compute()`, `data_update_state()`, `DrawGLScene()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, `forces_compute()`, and `RelativeWind_draw()`.

**6.1.2.98 double [datas::rho](#)**

Definition at line 91 of file Datas.h.

Referenced by `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.99 double [datas::rho\\_air](#)**

Definition at line 92 of file Datas.h.

Referenced by `data_init()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.100 double [datas::t](#)**

Definition at line 219 of file Datas.h.

Referenced by `draw_graphics()`, `equ_compute()`, `mexFunction()`, `solve_ODE()`, and `wave_z_compute()`.

**6.1.2.101 double [datas::Tan\\_AoA\\_left](#)**

Definition at line 189 of file Datas.h.

Referenced by `aero_coeff_control()`, and `angles_compute()`.

**6.1.2.102 double [datas::Tan\\_AoA\\_pitch](#)**

Definition at line 191 of file Datas.h.

Referenced by `aero_coeff_control()`, and `angles_compute()`.



**6.1.2.103 double [datas::Tan\\_AoA\\_right](#)**

Definition at line 190 of file Datas.h.

Referenced by `aero_coeff_control()`, and `angles_compute()`.

**6.1.2.104 double [datas::Tan\\_AoA\\_sail](#)**

Definition at line 193 of file Datas.h.

Referenced by `aero_coeff_control()`, and `angles_compute()`.

**6.1.2.105 double [datas::Tan\\_AoA\\_yaw](#)**

Definition at line 192 of file Datas.h.

Referenced by `aero_coeff_control()`, and `angles_compute()`.

**6.1.2.106 double [datas::tanAngle\\_sail](#)**

Definition at line 48 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `draw_graphics()`, `forces_all_compute()`, `forces_compute()`, `Girouette_draw()`, and `Sail_create_and_draw()`.

**6.1.2.107 double [datas::tanCal\\_left](#)**

Definition at line 58 of file Datas.h.

Referenced by `angles_compute()`, `angles_update()`, `data_init()`, and `data_update_param()`.

**6.1.2.108 double [datas::tanCal\\_pitch](#)**

Definition at line 60 of file Datas.h.

Referenced by `angles_compute()`, `angles_update()`, and `data_init()`.

**6.1.2.109 double [datas::tanCal\\_right](#)**

Definition at line 59 of file Datas.h.

Referenced by `angles_compute()`, `angles_update()`, `data_init()`, and `data_update_param()`.

**6.1.2.110 double [datas::tanCal\\_yaw](#)**

Definition at line 164 of file Datas.h.

Referenced by `angles_compute()`, `angles_update()`, and `data_update_param()`.

**6.1.2.111 double [datas::tanCal\\_yaw0](#)**

Definition at line 47 of file Datas.h.

Referenced by `angles_update()`, and `data_update_param()`.

#### **6.1.2.112** float `datas::target_x`

Definition at line 257 of file `Datas.h`.

Referenced by `angles_update()`, `data_init()`, `draw_graphics()`, and `DrawGLScene()`.

#### **6.1.2.113** float `datas::target_y`

Definition at line 257 of file `Datas.h`.

Referenced by `angles_update()`, `data_init()`, `draw_graphics()`, and `DrawGLScene()`.

#### **6.1.2.114** double `datas::theta`

Definition at line 144 of file `Datas.h`.

Referenced by `acceleration_compute()`, `angles_compute()`, `data_update_state()`, `DrawGLScene()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, `forces_compute()`, and `RelativeWind_draw()`.

#### **6.1.2.115** double `datas::Torque_G_E1`

Definition at line 199 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `forces_compute()`.

#### **6.1.2.116** double `datas::Torque_G_E2`

Definition at line 200 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `forces_compute()`.

#### **6.1.2.117** double `datas::Torque_G_E3`

Definition at line 201 of file `Datas.h`.

Referenced by `acceleration_compute()`, and `forces_compute()`.

#### **6.1.2.118** double `datas::V_wave[2]`

Definition at line 125 of file `Datas.h`.

Referenced by `data_update_param()`.

#### **6.1.2.119** double `datas::V_wave0`

Definition at line 120 of file `Datas.h`.

Referenced by `data_init()`, `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.120 double `datas::V_wave1`**

Definition at line 133 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.121 double `datas::V_wave2`**

Definition at line 134 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.122 double `datas::verif`[26]**

Definition at line 255 of file Datas.h.

**6.1.2.123 float `datas::Vslider_x`**

Definition at line 258 of file Datas.h.

Referenced by `angles_update()`, `data_init()`, `draw_graphics()`, and `DrawGLScene()`.

**6.1.2.124 double `datas::Wave_amp`[2]**

Definition at line 124 of file Datas.h.

Referenced by `data_update_param()`, and `draw_graphics()`.

**6.1.2.125 double `datas::Wave_amp0`**

Definition at line 119 of file Datas.h.

Referenced by `data_init()`, `data_update_param()`, `draw_graphics()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.126 double `datas::Wave_amp1`**

Definition at line 131 of file Datas.h.

Referenced by `data_update_param()`, `draw_graphics()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.127 double `datas::Wave_amp2`**

Definition at line 132 of file Datas.h.

Referenced by `data_update_param()`, `draw_graphics()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.128 double `datas::Wave_angle`[2]**

Definition at line 123 of file Datas.h.

Referenced by `data_update_param()`.

**6.1.2.129 double [datas::Wave\\_angle0](#)**

Definition at line 118 of file Datas.h.

Referenced by `data_init()`, `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.130 double [datas::Wave\\_angle1](#)**

Definition at line 135 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.131 double [datas::Wave\\_angle2](#)**

Definition at line 136 of file Datas.h.

Referenced by `data_update_param()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.132 double [datas::Wind](#)**

Definition at line 111 of file Datas.h.

Referenced by `data_init()`, `data_update_param()`, `draw_graphics()`, and `DrawGLScene()`.

**6.1.2.133 double [datas::Wind\\_angle](#)**

Definition at line 51 of file Datas.h.

Referenced by `data_update_param()`, and `draw_graphics()`.

**6.1.2.134 double [datas::Wind\\_x](#)**

Definition at line 112 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `draw_graphics()`, `DrawGLScene()`, `forces_all_compute()`, `forces_compute()`, `Girouette_draw()`, and `RelativeWind_draw()`.

**6.1.2.135 double [datas::Wind\\_y](#)**

Definition at line 113 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `draw_graphics()`, `DrawGLScene()`, `forces_all_compute()`, `forces_compute()`, `Girouette_draw()`, and `RelativeWind_draw()`.

**6.1.2.136 double [datas::x](#)**

Definition at line 139 of file Datas.h.

Referenced by `data_update_state()`, `DrawGLScene()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.137 double [datas::x\\_cam](#)**

Definition at line 155 of file Datas.h.

**6.1.2.138 double [datas::x\\_foils](#)**

Definition at line 46 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.139 double [datas::x\\_pitch](#)**

Definition at line 107 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.140 double [datas::x\\_sail](#)**

Definition at line 70 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `forces_all_compute()`, `forces_compute()`, `RelativeWind_draw()`, and `Sail_create_and_draw()`.

**6.1.2.141 double [datas::x\\_yaw](#)**

Definition at line 103 of file Datas.h.

Referenced by `angles_compute()`, `Boat_create()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.142 double [datas::y](#)**

Definition at line 140 of file Datas.h.

Referenced by `data_update_state()`, `DrawGLScene()`, `equ_compute()`, and `wave_z_compute()`.

**6.1.2.143 double [datas::y\\_cam](#)**

Definition at line 156 of file Datas.h.

**6.1.2.144 double [datas::y\\_foil\\_left](#)**

Definition at line 65 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.145 double [datas::y\\_foil\\_right](#)**

Definition at line 66 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

#### **6.1.2.146** double `datas::y_pitch`

Definition at line 108 of file `Datas.h`.

Referenced by `angles_compute()`, `data_update_param()`, `forces_all_compute()`, and `forces_compute()`.

#### **6.1.2.147** double `datas::y_sail`

Definition at line 71 of file `Datas.h`.

Referenced by `angles_compute()`, `data_update_param()`, `forces_all_compute()`, `forces_compute()`, `RelativeWind_draw()`, and `Sail_create_and_draw()`.

#### **6.1.2.148** double `datas::y_yaw`

Definition at line 104 of file `Datas.h`.

Referenced by `angles_compute()`, `Boat_create()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

#### **6.1.2.149** double `datas::z`

Definition at line 141 of file `Datas.h`.

Referenced by `data_update_state()`, `DrawGLScene()`, `EE_flat_compute()`, and `equ_compute()`.

#### **6.1.2.150** double `datas::z_cam`

Definition at line 157 of file `Datas.h`.

#### **6.1.2.151** double `datas::z_foils`

Definition at line 52 of file `Datas.h`.

Referenced by `angles_compute()`, `Boat_create()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, `forces_compute()`, `RelativeWind_draw()`, and `Sail_create_and_draw()`.

#### **6.1.2.152** double `datas::z_pitch`

Definition at line 109 of file `Datas.h`.

Referenced by `angles_compute()`, `data_update_param()`, `forces_all_compute()`, and `forces_compute()`.

#### **6.1.2.153** double `datas::z_sail`

Definition at line 72 of file `Datas.h`.

Referenced by `angles_compute()`, `data_update_param()`, `forces_all_compute()`, and `forces_compute()`.

**6.1.2.154** double [datas::z\\_yaw](#)

Definition at line 105 of file Datas.h.

Referenced by `angles_compute()`, `data_update_param()`, `EE_flat_compute()`, `equ_compute()`, `forces_all_compute()`, and `forces_compute()`.

The documentation for this struct was generated from the following file:

- `D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/Datas.h`

## 6.2 joy\_parameters Struct Reference

Struct for joystick things.

```
#include <Datas.h>
```

### Data Fields

- bool [doX](#)  
*TRUE: use joystick X-axis, FALSE: use keyboard.*
- bool [doY](#)  
*TRUE: use joystick Y-axis, FALSE: use keyboard.*
- bool [doZ](#)  
*TRUE: use joystick Z-axis, FALSE: use keyboard.*
- bool [doR](#)  
*TRUE: use joystick R-axis, FALSE: use keyboard.*
- float [dirX](#)  
*change X axis direction: -1.0f*
- float [dirY](#)  
*change Y axis direction: -1.0f*
- float [dirZ](#)  
*change Z axis direction: -1.0f*
- float [dirR](#)  
*change R axis direction: -1.0f*

### 6.2.1 Detailed Description

Struct for joystick things.

Variables from MATLAB (or default) to indicate how to use the joystick (if any)

Definition at line 306 of file Datas.h.

### 6.2.2 Field Documentation

#### 6.2.2.1 float [joy\\_parameters::dirR](#)

change R axis direction: -1.0f

Definition at line 316 of file Datas.h.

Referenced by `get_joystick_parameters()`, and `pass_joy_parameters()`.



#### 6.2.2.2 float joy\_parameters::dirX

change X axis direction: -1.0f

Definition at line 313 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

#### 6.2.2.3 float joy\_parameters::dirY

change Y axis direction: -1.0f

Definition at line 314 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

#### 6.2.2.4 float joy\_parameters::dirZ

change Z axis direction: -1.0f

Definition at line 315 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

#### 6.2.2.5 bool joy\_parameters::doR

TRUE: use jostick R-axis, FALSE: use keyboard.

Definition at line 311 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

#### 6.2.2.6 bool joy\_parameters::doX

TRUE: use jostick X-axis, FALSE: use keyboard.

Definition at line 308 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

#### 6.2.2.7 bool joy\_parameters::doY

TRUE: use jostick Y-axis, FALSE: use keyboard.

Definition at line 309 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

#### 6.2.2.8 bool joy\_parameters::doZ

TRUE: use jostick Z-axis, FALSE: use keyboard.

Definition at line 310 of file Datas.h.

Referenced by get\_joystick\_parameters(), and pass\_joy\_parameters().

The documentation for this struct was generated from the following file:

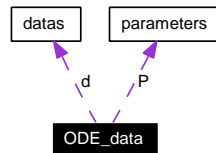
- [D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\\_source/Datas.h](D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/Datas.h)

## 6.3 ODE\_data Struct Reference

Datas relative to the ODE solver.

```
#include <Ddatas.h>
```

Collaboration diagram for ODE\_data:



### Data Fields

- double \* [state](#)
- [parameters](#) \* [P](#)
- [datas](#) \* [d](#)
- double [time\\_param](#)
- double [CumulNumOfSteps](#)
- realtype [reltol](#)
  - < Cumulative number of integration steps
- realtype [t](#)
- realtype [tout](#)
- N\_Vector [y](#)
- N\_Vector [abstol](#)
- void \* [cvmem](#)
- int [flag](#)
- int [flagr](#)
- int [iout](#)
- int [rootsfound](#) [2]

### 6.3.1 Detailed Description

Datas relative to the ODE solver.

Contains data for the solver and pointers to the general structure `datas *d` as well as a pointer to parameters `*P`.

Definition at line 279 of file `Ddatas.h`.

### 6.3.2 Field Documentation

#### 6.3.2.1 N\_Vector [ODE\\_data::abstol](#)

Definition at line 295 of file `Ddatas.h`.

Referenced by `solver_init()`.

**6.3.2.2 double ODE\_data::CumulNumOfSteps**

Definition at line 290 of file Datas.h.

Referenced by solve\_ODE().

**6.3.2.3 void\* ODE\_data::ccode\_mem**

Definition at line 296 of file Datas.h.

Referenced by solve\_ODE(), solver\_free(), and solver\_init().

**6.3.2.4 datas\* ODE\_data::d**

Definition at line 283 of file Datas.h.

Referenced by f(), mexFunction(), and solve\_ODE().

**6.3.2.5 int ODE\_data::flag**

Definition at line 297 of file Datas.h.

Referenced by solve\_ODE().

**6.3.2.6 int ODE\_data::flagr**

Definition at line 297 of file Datas.h.

**6.3.2.7 int ODE\_data::iout**

Definition at line 297 of file Datas.h.

**6.3.2.8 parameters\* ODE\_data::P**

Definition at line 282 of file Datas.h.

Referenced by f(), and mexFunction().

**6.3.2.9 realtype ODE\_data::reltol**

< Cumulative number of integration steps

Definition at line 294 of file Datas.h.

**6.3.2.10 int ODE\_data::rootsfound[2]**

Definition at line 298 of file Datas.h.

**6.3.2.11 double\* [ODE\\_data::state](#)**

Definition at line 281 of file Datas.h.

Referenced by `f()`, `mexFunction()`, and `solver_init()`.

**6.3.2.12 realtype [ODE\\_data::t](#)**

Definition at line 294 of file Datas.h.

**6.3.2.13 double [ODE\\_data::time\\_param](#)**

Definition at line 287 of file Datas.h.

Referenced by `get_time()`, `mexFunction()`, and `solve_ODE()`.

**6.3.2.14 realtype [ODE\\_data::tout](#)**

Definition at line 294 of file Datas.h.

**6.3.2.15 N\_Vector [ODE\\_data::y](#)**

Definition at line 295 of file Datas.h.

Referenced by `solve_ODE()`, `solver_free()`, and `solver_init()`.

The documentation for this struct was generated from the following file:

- `D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro_source/Datas.h`

## 6.4 parameters Struct Reference

parameters struct declaration

```
#include <Datas.h>
```

### Data Fields

- double [reltol](#)
- double [abstol](#) [12]
- double [param](#) [10]

### 6.4.1 Detailed Description

parameters struct declaration

These parameters are passed by MATLAB

Definition at line 21 of file Datas.h.

### 6.4.2 Field Documentation

#### 6.4.2.1 double [parameters::abstol](#)[12]

Definition at line 34 of file Datas.h.

Referenced by `initParameters()`.

#### 6.4.2.2 double [parameters::param](#)[10]

Definition at line 35 of file Datas.h.

Referenced by `data_update_param()`, and `initParameters()`.

#### 6.4.2.3 double [parameters::reltol](#)

Definition at line 33 of file Datas.h.

Referenced by `get_Parameters()`, and `initParameters()`.

The documentation for this struct was generated from the following file:

- D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/[Datas.h](#)

## 6.5 seaDatas Struct Reference

### Data Fields

- float \* [Z](#)
- float \* [R](#)
- float \* [G](#)
- float \* [B](#)
- int [N](#)
- float [L](#)

### 6.5.1 Detailed Description

Definition at line 9 of file `wave_z_compute_byHand.cpp`.

### 6.5.2 Field Documentation

#### 6.5.2.1 float\* [seaDatas::B](#)

Definition at line 14 of file `wave_z_compute_byHand.cpp`.

Referenced by `free_wave_variables()`, and `wave_init()`.

#### 6.5.2.2 float\* [seaDatas::G](#)

Definition at line 13 of file `wave_z_compute_byHand.cpp`.

Referenced by `free_wave_variables()`, and `wave_init()`.

#### 6.5.2.3 float [seaDatas::L](#)

Definition at line 16 of file `wave_z_compute_byHand.cpp`.

Referenced by `Draw_sea()`, `wave_init()`, and `wave_z_compute()`.

#### 6.5.2.4 int [seaDatas::N](#)

Definition at line 15 of file `wave_z_compute_byHand.cpp`.

Referenced by `Draw_sea()`, `wave_init()`, and `wave_z_compute()`.

#### 6.5.2.5 float\* [seaDatas::R](#)

Definition at line 12 of file `wave_z_compute_byHand.cpp`.

Referenced by `free_wave_variables()`, and `wave_init()`.

**6.5.2.6 float\* [seaDatas::z](#)**

Definition at line 11 of file `wave_z_compute_byHand.cpp`.

Referenced by `Draw_sea()`, `free_wave_variables()`, `wave_init()`, and `wave_z_compute()`.

The documentation for this struct was generated from the following file:

- `D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/wave\_z\_compute\_by-Hand.cpp`



## 6.6 SolidObject Struct Reference

3D object description

```
#include <solid_objects.h>
```

### Data Fields

- int [N](#)
- int [M](#)

*matrices size*

- float \*\* [X](#)
- float \*\* [Y](#)
- float \*\* [Z](#)

*vertex coordinates*

- float \*\* [nX](#)
- float \*\* [nY](#)
- float \*\* [nZ](#)

*Vertex normals.*

- float \*\* [R](#)
- float \*\* [G](#)
- float \*\* [B](#)

*vertex color*

### 6.6.1 Detailed Description

3D object description

Structure for the description of an object as a mesh.

N, M: size of the mesh 2D array.

\*\*X, \*\*Y, \*\*Z : vertex coordinates.

\*\*nX, \*\*nY, \*\*nZ : normals to vertexes (for lighting).

\*\*R, \*\*G, \*\*B : vertex color.

Definition at line 28 of file solid\_objects.h.

### 6.6.2 Field Documentation

#### 6.6.2.1 float \*\* [SolidObject::B](#)

vertex color

Definition at line 36 of file solid\_objects.h.

Referenced by [Boat\\_create\(\)](#), [SolidObject\\_create\(\)](#), [SolidObject\\_delete\(\)](#), and [SolidObject\\_draw\(\)](#).

**6.6.2.2 float\*\* SolidObject::G**

Definition at line 36 of file solid\_objects.h.

Referenced by Boat\_create(), SolidObject\_create(), SolidObject\_delete(), and SolidObject\_draw().

**6.6.2.3 int SolidObject::M**

matrices size

Definition at line 30 of file solid\_objects.h.

Referenced by SolidObject\_compute\_normals(), SolidObject\_create(), and SolidObject\_draw().

**6.6.2.4 int SolidObject::N**

Definition at line 30 of file solid\_objects.h.

Referenced by SolidObject\_compute\_normals(), SolidObject\_create(), SolidObject\_delete(), and SolidObject\_draw().

**6.6.2.5 float\*\* SolidObject::nX**

Definition at line 34 of file solid\_objects.h.

Referenced by SolidObject\_compute\_normals(), SolidObject\_create(), SolidObject\_delete(), and SolidObject\_draw().

**6.6.2.6 float\*\* SolidObject::nY**

Definition at line 34 of file solid\_objects.h.

Referenced by SolidObject\_compute\_normals(), SolidObject\_create(), SolidObject\_delete(), and SolidObject\_draw().

**6.6.2.7 float\*\* SolidObject::nZ**

Vertex normals.

Definition at line 34 of file solid\_objects.h.

Referenced by SolidObject\_compute\_normals(), SolidObject\_create(), SolidObject\_delete(), and SolidObject\_draw().

**6.6.2.8 float\*\* SolidObject::R**

Definition at line 36 of file solid\_objects.h.

Referenced by Boat\_create(), SolidObject\_create(), SolidObject\_delete(), and SolidObject\_draw().

**6.6.2.9 float\*\* SolidObject::X**

Definition at line 32 of file solid\_objects.h.

Referenced by `Boat_create()`, `SolidObject_compute_normals()`, `SolidObject_create()`, `SolidObject_delete()`, and `SolidObject_draw()`.

#### 6.6.2.10 `float ** SolidObject::Y`

Definition at line 32 of file `solid_objects.h`.

Referenced by `Boat_create()`, `SolidObject_compute_normals()`, `SolidObject_create()`, `SolidObject_delete()`, and `SolidObject_draw()`.

#### 6.6.2.11 `float ** SolidObject::Z`

vertex coordinates

Definition at line 32 of file `solid_objects.h`.

Referenced by `Boat_create()`, `SolidObject_compute_normals()`, `SolidObject_create()`, `SolidObject_delete()`, and `SolidObject_draw()`.

The documentation for this struct was generated from the following file:

- `D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL_graphics/solid_objects.h`



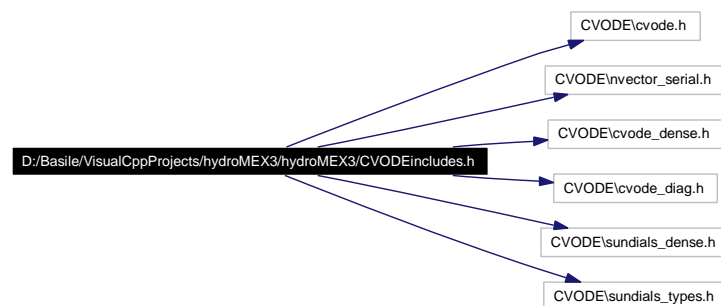
## Chapter 7

# hydroMex3 File Documentation

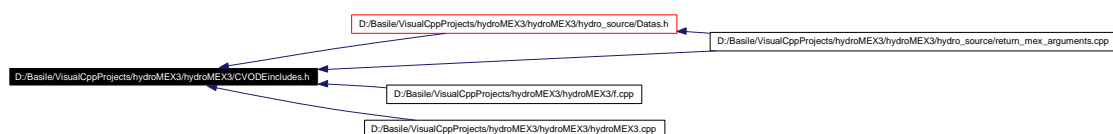
### 7.1 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/CVODEincludes.h File Reference

```
#include "CVODE\cvode.h"
#include "CVODE\nvector_serial.h"
#include "CVODE\cvode_dense.h"
#include "CVODE\cvode_diag.h"
#include "CVODE\sundials_dense.h"
#include "CVODE\sundials_types.h"
```

Include dependency graph for CVODEincludes.h:



This graph shows which files directly or indirectly include this file:



### 7.1.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains all includes for the SUNDIALS CVODE solver. Documentation for CVODE is not included here, see:

<http://www.llnl.gov/casc/sundials/documentation/documentation.html>

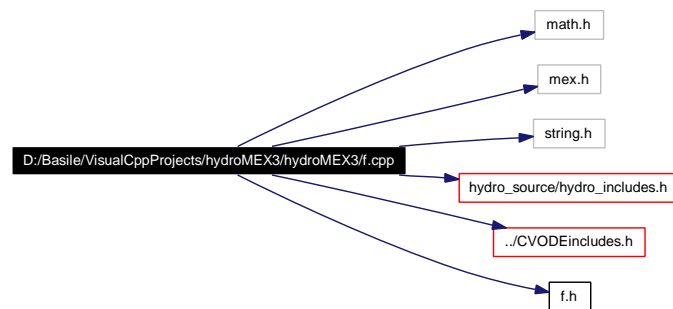
for CVODE documentation

Definition in file [CVODEincludes.h](#).

## 7.2 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "hydro_source/hydro_includes.h"
#include "CVODEincludes.h"
#include "f.h"
```

Include dependency graph for f.cpp:



### Defines

- #define [pi](#) 3.141592653589793
- #define [PI](#) 3.141592653589793
- #define [Ith](#)(v, i) NV\_Ith\_S(v,i-1)
- #define [IJth](#)(A, i, j) DENSE\_ELEM(A,i-1,j-1)
- #define [NEQ](#) 12
- #define [TO](#) RCONST(0.0)

### Functions

- static int [f](#) (realtype t, N\_Vector y, N\_Vector ydot, void \*f\_data)  
*Wrap function for solver f for computing of f in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$ .*
- void [solver\\_init](#) (ODE\_data \*f\_dat)  
*Solver initialisation.*
- void [solve\\_ODE](#) (ODE\_data \*f\_dat, realtype tout)  
*ODE solving.*
- void [solver\\_free](#) (ODE\_data \*f\_dat)  
*Free solver memory.*

## 7.2.1 Detailed Description

### Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains stuff related to evaluation of  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$  and stuff related to the ODE solver (initialisation, solving, closing)

Definition in file [f.cpp](#).

## 7.2.2 Define Documentation

### 7.2.2.1 `#define IJth(A, i, j) DENSE_ELEM(A,i-1,j-1)`

Definition at line 40 of file [f.cpp](#).

### 7.2.2.2 `#define Ith(v, i) NV_Ith_S(v,i-1)`

Definition at line 39 of file [f.cpp](#).

Referenced by [f\(\)](#), [solver\\_init\(\)](#), and [store\\_state\(\)](#).

### 7.2.2.3 `#define NEQ 12`

Definition at line 44 of file [f.cpp](#).

Referenced by [solver\\_init\(\)](#).

### 7.2.2.4 `#define PI 3.141592653589793`

Definition at line 33 of file [f.cpp](#).

### 7.2.2.5 `#define pi 3.141592653589793`

Definition at line 32 of file [f.cpp](#).

Referenced by [aero\\_coeff\\_control\(\)](#), [data\\_init\(\)](#), [data\\_update\\_param\(\)](#), [fillWaveLuts\(\)](#), [wave\\_z\\_compute\(\)](#), [Wcos\(\)](#), and [Wsin\(\)](#).

### 7.2.2.6 `#define T0 RCONST(0.0)`

Definition at line 45 of file [f.cpp](#).

## 7.2.3 Function Documentation

### 7.2.3.1 `static int f(realtype t, N_Vector y, N_Vector ydot, void *f_data)` [static]

Wrap function for solver [f](#) for computing of  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$ .

This function makes the bridge between the Sundials ODE-solver and the evaluation function [compute\\_state\\_dot\(\)](#)



**Parameters:**

- $t$  : time
- $y$  : state ( $\vec{q}$ ) in the datatype needed by the solver
- $ydot$  : evaluation output ( $\frac{\partial \vec{q}}{\partial t}$ )
- $*f\_data,$  : pointer of type void, pointing on the [ODE\\_data](#) type struct containing all the model's data

**Returns:**

- : 0 (no meaning yet <=> no integration error handling yet!!!)

Definition at line 63 of file f.cpp.

References compute\_state\_dot(), ODE\_data::d, data\_update\_state(), datas::ddq, datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, Ith, ODE\_data::P, and ODE\_data::state.

Referenced by Draw\_sea(), DrawGLScene(), fillWaveLuts(), Girouette\_draw(), HSlider\_draw(), InitGL(), joystick\_getXYZR(), one\_normal(), RelativeWind\_draw(), ReSizeGLScene(), Sail\_create\_and\_draw(), Target\_draw(), toc(), VSlider\_draw(), wave\_z\_compute(), and Wsin().

**7.2.3.2 void solve\_ODE ([ODE\\_data](#) \* $f\_dat$ , realtype  $tout$ )**

ODE solving.

Solve the ODE from current time up to  $t_{out}$

**Parameters:**

- $*f\_dat,$  : pointer to structure of type [ODE\\_data](#)
- $tout,$  : output time

Definition at line 233 of file f.cpp.

References ODE\_data::CumulNumOfSteps, ODE\_data::ccode\_mem, ODE\_data::d, ODE\_data::flag, store\_state(), datas::t, ODE\_data::time\_param, and ODE\_data::y.

Referenced by mexFunction().

**7.2.3.3 void solver\_free ([ODE\\_data](#) \* $f\_dat$ )**

Free solver memory.

Free solver memory

**Parameters:**

- $*f\_dat,$  : pointer to structure of type [ODE\\_data](#)

Definition at line 255 of file f.cpp.

References ODE\_data::ccode\_mem, and ODE\_data::y.

Referenced by mexFunction().

**7.2.3.4 void solver\_init ([ODE\\_data](#) \* $f\_dat$ )**

Solver initialisation.

Solver initialisation

Don't forget to call [solver\\_free\(\)](#)!

**Parameters:**

*\*f\_dat,:* pointer to structure of type [ODE\\_data](#)

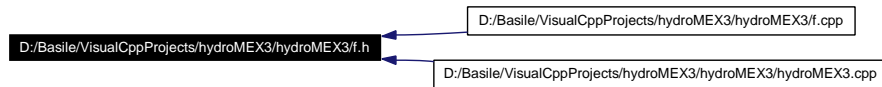
Definition at line 154 of file f.cpp.

References `ODE_data::abstol`, `ODE_data::ccode_mem`, `lth`, `NEQ`, `ODE_data::state`, and `ODE_data::y`.

Referenced by `mexFunction()`.

## 7.3 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- static int **f** (realtpe t, N\_Vector y, N\_Vector ydot, void \*f\_data)
- void **solver\_init** (ODE\_data \*f\_dat)  
*Solver initialisation.*
- void **solve\_ODE** (ODE\_data \*f\_dat, realtype tout)  
*ODE solving.*
- void **solver\_free** (ODE\_data \*f\_dat)  
*Free solver memory.*

### 7.3.1 Detailed Description

#### Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains stuff related to evaluation of  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$  and stuff related to the ODE solver (initialisation, solving, closing)

Definition in file [f.h](#).

### 7.3.2 Function Documentation

**7.3.2.1** static int **f** (realtpe t, N\_Vector y, N\_Vector ydot, void \*f\_data) [static]

**7.3.2.2** void **solve\_ODE** (ODE\_data \*f\_dat, realtype tout)

ODE solving.

Solve the ODE from current time up to  $t_{out}$

#### Parameters:

**\*f\_dat,:** pointer to structure of type [ODE\\_data](#)

**tout,:** output time

Definition at line 233 of file [f.cpp](#).

References [ODE\\_data::CumulNumOfSteps](#), [ODE\\_data::cvmem](#), [ODE\\_data::d](#), [ODE\\_data::flag](#), [store\\_state\(\)](#), [datas::t](#), [ODE\\_data::time\\_param](#), and [ODE\\_data::y](#).

Referenced by [mexFunction\(\)](#).

### 7.3.2.3 void solver\_free (ODE\_data \*f\_dat)

Free solver memory.

Free solver memory

#### Parameters:

*\*f\_dat,:* pointer to structure of type [ODE\\_data](#)

Definition at line 255 of file f.cpp.

References [ODE\\_data::cnode\\_mem](#), and [ODE\\_data::y](#).

Referenced by [mexFunction\(\)](#).

### 7.3.2.4 void solver\_init (ODE\_data \*f\_dat)

Solver initialisation.

Solver initialisation

Don't forget to call [solver\\_free\(\)](#)!

#### Parameters:

*\*f\_dat,:* pointer to structure of type [ODE\\_data](#)

Definition at line 154 of file f.cpp.

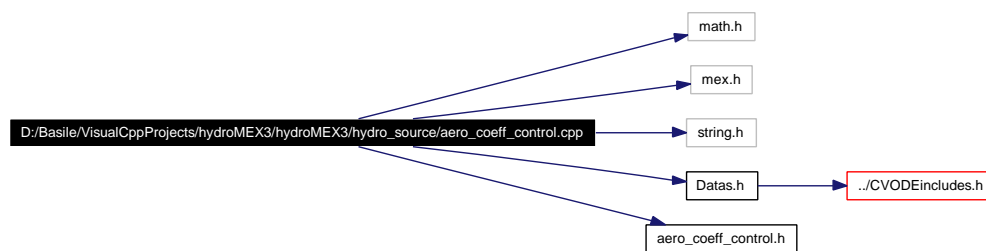
References [ODE\\_data::abstol](#), [ODE\\_data::cnode\\_mem](#), [Ith](#), [NEQ](#), [ODE\\_data::state](#), and [ODE\\_data::y](#).

Referenced by [mexFunction\(\)](#).

## 7.4 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/aero\_coeff\_control.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "Datas.h"
#include "aero_coeff_control.h"
```

Include dependency graph for `aero_coeff_control.cpp`:



### Defines

- `#define pi 3.141592653589793`
- `#define PI 3.141592653589793`

### Functions

- `void aero_coeff_control (datas *d)`  
*Aero/nautical coefficients computing.*

#### 7.4.1 Detailed Description

Author:

`basile.graf@epfl.ch`

This file contains stuff related to Lift and Drag computation...

Definition in file `aero_coeff_control.cpp`.

#### 7.4.2 Define Documentation

##### 7.4.2.1 `#define PI 3.141592653589793`

Definition at line 19 of file `aero_coeff_control.cpp`.

### 7.4.2.2 `#define pi 3.141592653589793`

Definition at line 18 of file `aero_coeff_control.cpp`.

## 7.4.3 Function Documentation

### 7.4.3.1 `void aero_coeff_control (datas * d)`

Aero/nautical coefficients computing.

Computes lift and drag coefficients for all surfaces/bodies.

Polynomial forms are used, polar tables are still to be implemented!!

#### Parameters:

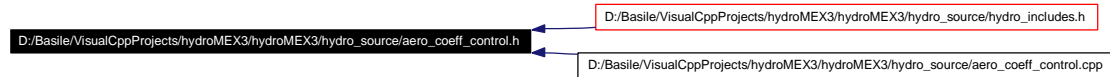
*\*d* : pointer to the `datas` data-structure

Definition at line 23 of file `aero_coeff_control.cpp`.

References `datas::A_left`, `datas::A_pitch`, `datas::A_right`, `datas::A_yaw`, `datas::AR_left`, `datas::AR_pitch`, `datas::AR_right`, `datas::AR_sail`, `datas::AR_yaw`, `datas::CD_left`, `datas::CD_pitch`, `datas::CD_right`, `datas::CD_sail`, `datas::CD_yaw`, `datas::Chord`, `datas::Chord_left`, `datas::Chord_max`, `datas::Chord_min`, `datas::Chord_right`, `datas::CL_left`, `datas::CL_pitch`, `datas::CL_right`, `datas::CL_sail`, `datas::CL_yaw`, `datas::E_foils`, `datas::E_yaw`, `datas::EE_left`, `datas::EE_right`, `datas::EE_yaw`, `datas::gamma`, `pi`, `datas::Tan_AoA_left`, `datas::Tan_AoA_pitch`, `datas::Tan_AoA_right`, `datas::Tan_AoA_sail`, and `datas::Tan_AoA_yaw`.

## 7.5 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/aero\_coeff\_control.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void [aero\\_coeff\\_control](#) ([datas](#) \*d)  
*Aero/nautical coefficients computing.*

#### 7.5.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains stuff related to Lift and Drag computation...

Definition in file [aero\\_coeff\\_control.h](#).

#### 7.5.2 Function Documentation

##### 7.5.2.1 void [aero\\_coeff\\_control](#) ([datas](#) \*d)

Aero/nautical coefficients computing.

Computes lift and drag coefficients for all surfaces/bodies.

Polynomial forms are used, polar tables are still to be implemented!!

**Parameters:**

\*d : pointer to the datas data-structure

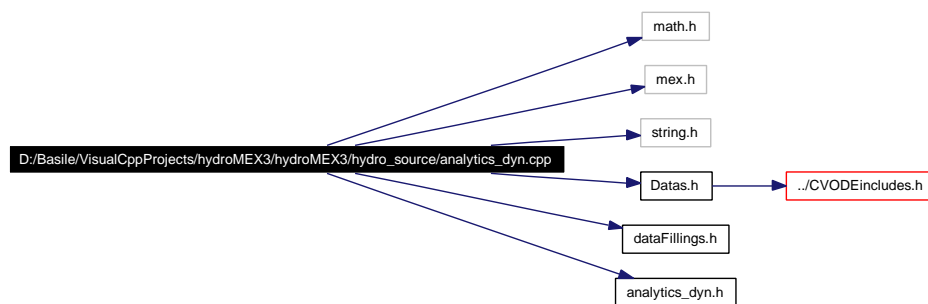
Definition at line 23 of file [aero\\_coeff\\_control.cpp](#).

References [datas::A\\_left](#), [datas::A\\_pitch](#), [datas::A\\_right](#), [datas::A\\_yaw](#), [datas::AR\\_left](#), [datas::AR\\_pitch](#), [datas::AR\\_right](#), [datas::AR\\_sail](#), [datas::AR\\_yaw](#), [datas::CD\\_left](#), [datas::CD\\_pitch](#), [datas::CD\\_right](#), [datas::CD\\_sail](#), [datas::CD\\_yaw](#), [datas::Chord](#), [datas::Chord\\_left](#), [datas::Chord\\_max](#), [datas::Chord\\_min](#), [datas::Chord\\_right](#), [datas::CL\\_left](#), [datas::CL\\_pitch](#), [datas::CL\\_right](#), [datas::CL\\_sail](#), [datas::CL\\_yaw](#), [datas::E\\_foils](#), [datas::E\\_yaw](#), [datas::EE\\_left](#), [datas::EE\\_right](#), [datas::EE\\_yaw](#), [datas::gamma](#), [pi](#), [datas::Tan\\_AoA\\_left](#), [datas::Tan\\_AoA\\_pitch](#), [datas::Tan\\_AoA\\_right](#), [datas::Tan\\_AoA\\_sail](#), and [datas::Tan\\_AoA\\_yaw](#).

## 7.6 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/analytics\_dyn.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "Datas.h"
#include "dataFillings.h"
#include "analytics_dyn.h"
```

Include dependency graph for analytics\_dyn.cpp:



### Defines

- #define [pi](#) 3.141592653589793
- #define [PI](#) 3.141592653589793

### Functions

- void [acceleration\\_compute](#) ([datas](#) \*d)

#### 7.6.1 Define Documentation

##### 7.6.1.1 #define [PI](#) 3.141592653589793

Definition at line 11 of file analytics\_dyn.cpp.

##### 7.6.1.2 #define [pi](#) 3.141592653589793

Definition at line 10 of file analytics\_dyn.cpp.

#### 7.6.2 Function Documentation

##### 7.6.2.1 void [acceleration\\_compute](#) ([datas](#) \*d)

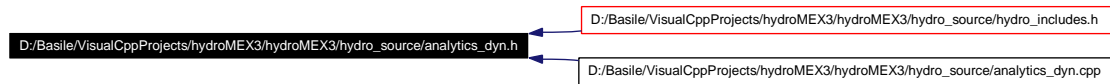
Definition at line 17 of file analytics\_dyn.cpp.



References `datas::ddq`, `datas::dphi`, `datas::dpsi`, `datas::dtheta`, `datas::Force_G_E1`, `datas::Force_G_E2`, `datas::Force_G_E3`, `datas::g`, `datas::Ix`, `datas::Iy`, `datas::Iz`, `datas::M`, `datas::phi`, `datas::psi`, `datas::theta`, `datas::Torque_G_E1`, `datas::Torque_G_E2`, and `datas::Torque_G_E3`.

## 7.7 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/analytics\_dyn.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void [acceleration\\_compute](#) ([datas](#) \*d)

#### 7.7.1 Function Documentation

##### 7.7.1.1 void [acceleration\\_compute](#) ([datas](#) \* d)

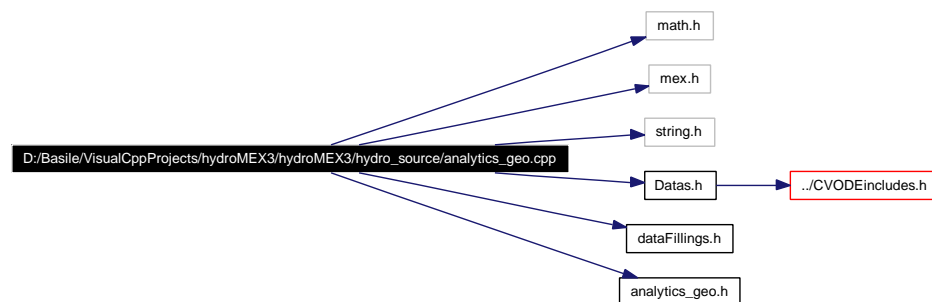
Definition at line 17 of file `analytics_dyn.cpp`.

References `datas::ddq`, `datas::dphi`, `datas::dpsi`, `datas::dtheta`, `datas::Force_G_E1`, `datas::Force_G_E2`, `datas::Force_G_E3`, `datas::g`, `datas::Ix`, `datas::Iy`, `datas::Iz`, `datas::M`, `datas::phi`, `datas::psi`, `datas::theta`, `datas::Torque_G_E1`, `datas::Torque_G_E2`, and `datas::Torque_G_E3`.

## 7.8 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/analytics\_geo.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "Datas.h"
#include "dataFillings.h"
#include "analytics_geo.h"
```

Include dependency graph for analytics\_geo.cpp:



### Defines

- `#define pi 3.141592653589793`
- `#define PI 3.141592653589793`

### Functions

- `void EE_flat_compute (datas *d)`
- `void equ_compute (datas *d)`
- `void angles_compute (datas *d)`
- `void forces_all_compute (datas *d)`
- `void forces_compute (datas *d)`

#### 7.8.1 Define Documentation

##### 7.8.1.1 `#define PI 3.141592653589793`

Definition at line 11 of file `analytics_geo.cpp`.

##### 7.8.1.2 `#define pi 3.141592653589793`

Definition at line 10 of file `analytics_geo.cpp`.

## 7.8.2 Function Documentation

### 7.8.2.1 void angles\_compute (datas \* d)

Definition at line 90 of file analytics\_geo.cpp.

References datas::alpha\_left, datas::alpha\_right, datas::Baume, datas::Chord\_max, datas::Chord\_min, datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, datas::E\_foils, datas::E\_yaw, datas::EE\_left, datas::EE\_right, datas::EE\_yaw, datas::phi, datas::psi, datas::Tan\_AoA\_left, datas::Tan\_AoA\_pitch, datas::Tan\_AoA\_right, datas::Tan\_AoA\_sail, datas::Tan\_AoA\_yaw, datas::tanAngle\_sail, datas::tanCal\_left, datas::tanCal\_pitch, datas::tanCal\_right, datas::tanCal\_yaw, datas::theta, datas::Wind\_x, datas::Wind\_y, datas::x\_foils, datas::x\_pitch, datas::x\_sail, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_pitch, datas::y\_sail, datas::y\_yaw, datas::z\_foils, datas::z\_pitch, datas::z\_sail, and datas::z\_yaw.

### 7.8.2.2 void EE\_flat\_compute (datas \* d)

Definition at line 17 of file analytics\_geo.cpp.

References datas::alpha\_left, datas::alpha\_right, datas::EE\_left\_flat, datas::EE\_right\_flat, datas::EE\_yaw\_flat, datas::phi, datas::psi, datas::theta, datas::x\_foils, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_yaw, datas::z, datas::z\_foils, and datas::z\_yaw.

Referenced by compute\_state\_dot().

### 7.8.2.3 void equ\_compute (datas \* d)

Definition at line 40 of file analytics\_geo.cpp.

References datas::alpha\_left, datas::alpha\_right, datas::dEqu\_left, datas::dEqu\_right, datas::dEqu\_yaw, datas::EE\_left, datas::EE\_right, datas::EE\_yaw, datas::Equ\_left, datas::Equ\_right, datas::Equ\_yaw, datas::Lambda0, datas::Lambda1, datas::Lambda2, datas::phi, datas::psi, datas::t, datas::theta, datas::V\_wave0, datas::V\_wave1, datas::V\_wave2, datas::Wave\_amp0, datas::Wave\_amp1, datas::Wave\_amp2, datas::Wave\_angle0, datas::Wave\_angle1, datas::Wave\_angle2, datas::x, datas::x\_foils, datas::x\_yaw, datas::y, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_yaw, datas::z, datas::z\_foils, and datas::z\_yaw.

Referenced by compute\_state\_dot().

### 7.8.2.4 void forces\_all\_compute (datas \* d)

Definition at line 126 of file analytics\_geo.cpp.

References datas::A\_left, datas::A\_pitch, datas::A\_right, datas::A\_sail, datas::A\_structure, datas::A\_yaw, datas::alpha\_left, datas::alpha\_right, datas::Baume, datas::CD\_left, datas::CD\_pitch, datas::CD\_right, datas::CD\_sail, datas::CD\_structure, datas::CD\_yaw, datas::Chord\_max, datas::Chord\_min, datas::CL\_left, datas::CL\_pitch, datas::CL\_right, datas::CL\_sail, datas::CL\_yaw, datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, datas::E\_foils, datas::E\_yaw, datas::EE\_left, datas::EE\_right, datas::EE\_yaw, datas::Force\_drag, datas::Force\_left, datas::Force\_pitch, datas::Force\_right, datas::Force\_sail, datas::Force\_yaw, datas::phi, datas::psi, datas::rho, datas::rho\_air, datas::tanAngle\_sail, datas::theta, datas::Wind\_x, datas::Wind\_y, datas::x\_foils, datas::x\_pitch, datas::x\_sail, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_pitch, datas::y\_sail, datas::y\_yaw, datas::z\_foils, datas::z\_pitch, datas::z\_sail, and datas::z\_yaw.

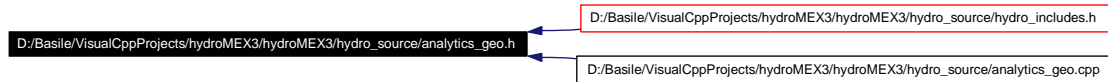
### 7.8.2.5 void forces\_compute (datas \* d)

Definition at line 211 of file analytics\_geo.cpp.

References datas::A\_left, datas::A\_pitch, datas::A\_right, datas::A\_sail, datas::A\_structure, datas::A\_yaw, datas::alpha\_left, datas::alpha\_right, datas::Baume, datas::CD\_left, datas::CD\_pitch, datas::CD\_right, datas::CD\_sail, datas::CD\_structure, datas::CD\_yaw, datas::Chord\_max, datas::Chord\_min, datas::CL\_left, datas::CL\_pitch, datas::CL\_right, datas::CL\_sail, datas::CL\_yaw, datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, datas::E\_foils, datas::E\_yaw, datas::EE\_left, datas::EE\_right, datas::EE\_yaw, datas::Force\_G\_E1, datas::Force\_G\_E2, datas::Force\_G\_E3, datas::phi, datas::psi, datas::rho, datas::rho\_air, datas::tanAngle\_sail, datas::theta, datas::Torque\_G\_E1, datas::Torque\_G\_E2, datas::Torque\_G\_E3, datas::Wind\_x, datas::Wind\_y, datas::x\_foils, datas::x\_pitch, datas::x\_sail, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_pitch, datas::y\_sail, datas::y\_yaw, datas::z\_foils, datas::z\_pitch, datas::z\_sail, and datas::z\_yaw.

## 7.9 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/analytics\_geo.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void [EE\\_flat\\_compute](#) (datas \*d)
- void [equ\\_compute](#) (datas \*d)
- void [angles\\_compute](#) (datas \*d)
- void [forces\\_all\\_compute](#) (datas \*d)
- void [forces\\_compute](#) (datas \*d)

### 7.9.1 Function Documentation

#### 7.9.1.1 void [angles\\_compute](#) (datas \* d)

Definition at line 90 of file analytics\_geo.cpp.

References `datas::alpha_left`, `datas::alpha_right`, `datas::Baume`, `datas::Chord_max`, `datas::Chord_min`, `datas::dphi`, `datas::dpsi`, `datas::dtheta`, `datas::dx`, `datas::dy`, `datas::dz`, `datas::E_foils`, `datas::E_yaw`, `datas::EE_left`, `datas::EE_right`, `datas::EE_yaw`, `datas::phi`, `datas::psi`, `datas::Tan_AoA_left`, `datas::Tan_AoA_pitch`, `datas::Tan_AoA_right`, `datas::Tan_AoA_sail`, `datas::Tan_AoA_yaw`, `datas::tanAngle_sail`, `datas::tanCal_left`, `datas::tanCal_pitch`, `datas::tanCal_right`, `datas::tanCal_yaw`, `datas::theta`, `datas::Wind_x`, `datas::Wind_y`, `datas::x_foils`, `datas::x_pitch`, `datas::x_sail`, `datas::x_yaw`, `datas::y_foil_left`, `datas::y_foil_right`, `datas::y_pitch`, `datas::y_sail`, `datas::y_yaw`, `datas::z_foils`, `datas::z_pitch`, `datas::z_sail`, and `datas::z_yaw`.

#### 7.9.1.2 void [EE\\_flat\\_compute](#) (datas \* d)

Definition at line 17 of file analytics\_geo.cpp.

References `datas::alpha_left`, `datas::alpha_right`, `datas::EE_left_flat`, `datas::EE_right_flat`, `datas::EE_yaw_flat`, `datas::phi`, `datas::psi`, `datas::theta`, `datas::x_foils`, `datas::x_yaw`, `datas::y_foil_left`, `datas::y_foil_right`, `datas::y_yaw`, `datas::z`, `datas::z_foils`, and `datas::z_yaw`.

Referenced by `compute_state_dot()`.

#### 7.9.1.3 void [equ\\_compute](#) (datas \* d)

Definition at line 40 of file analytics\_geo.cpp.

References `datas::alpha_left`, `datas::alpha_right`, `datas::dEqu_left`, `datas::dEqu_right`, `datas::dEqu_yaw`, `datas::EE_left`, `datas::EE_right`, `datas::EE_yaw`, `datas::Equ_left`, `datas::Equ_right`, `datas::Equ_yaw`, `datas::Lambda0`, `datas::Lambda1`, `datas::Lambda2`, `datas::phi`, `datas::psi`, `datas::t`, `datas::theta`, `datas::V_wave0`, `datas::V_wave1`, `datas::V_wave2`, `datas::Wave_amp0`, `datas::Wave_amp1`, `datas::Wave_amp2`,

datas::Wave\_angle0, datas::Wave\_angle1, datas::Wave\_angle2, datas::x, datas::x\_foils, datas::x\_yaw, datas::y, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_yaw, datas::z, datas::z\_foils, and datas::z\_yaw.

Referenced by compute\_state\_dot().

#### 7.9.1.4 void forces\_all\_compute (datas \* d)

Definition at line 126 of file analytics\_geo.cpp.

References datas::A\_left, datas::A\_pitch, datas::A\_right, datas::A\_sail, datas::A\_structure, datas::A\_yaw, datas::alpha\_left, datas::alpha\_right, datas::Baume, datas::CD\_left, datas::CD\_pitch, datas::CD\_right, datas::CD\_sail, datas::CD\_structure, datas::CD\_yaw, datas::Chord\_max, datas::Chord\_min, datas::CL\_left, datas::CL\_pitch, datas::CL\_right, datas::CL\_sail, datas::CL\_yaw, datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, datas::E\_foils, datas::E\_yaw, datas::EE\_left, datas::EE\_right, datas::EE\_yaw, datas::Force\_drag, datas::Force\_left, datas::Force\_pitch, datas::Force\_right, datas::Force\_sail, datas::Force\_yaw, datas::phi, datas::psi, datas::rho, datas::rho\_air, datas::tanAngle\_sail, datas::theta, datas::Wind\_x, datas::Wind\_y, datas::x\_foils, datas::x\_pitch, datas::x\_sail, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_pitch, datas::y\_sail, datas::y\_yaw, datas::z\_foils, datas::z\_pitch, datas::z\_sail, and datas::z\_yaw.

#### 7.9.1.5 void forces\_compute (datas \* d)

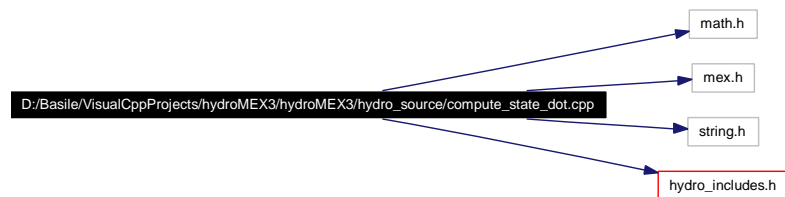
Definition at line 211 of file analytics\_geo.cpp.

References datas::A\_left, datas::A\_pitch, datas::A\_right, datas::A\_sail, datas::A\_structure, datas::A\_yaw, datas::alpha\_left, datas::alpha\_right, datas::Baume, datas::CD\_left, datas::CD\_pitch, datas::CD\_right, datas::CD\_sail, datas::CD\_structure, datas::CD\_yaw, datas::Chord\_max, datas::Chord\_min, datas::CL\_left, datas::CL\_pitch, datas::CL\_right, datas::CL\_sail, datas::CL\_yaw, datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, datas::E\_foils, datas::E\_yaw, datas::EE\_left, datas::EE\_right, datas::EE\_yaw, datas::Force\_G\_E1, datas::Force\_G\_E2, datas::Force\_G\_E3, datas::phi, datas::psi, datas::rho, datas::rho\_air, datas::tanAngle\_sail, datas::theta, datas::Torque\_G\_E1, datas::Torque\_G\_E2, datas::Torque\_G\_E3, datas::Wind\_x, datas::Wind\_y, datas::x\_foils, datas::x\_pitch, datas::x\_sail, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_pitch, datas::y\_sail, datas::y\_yaw, datas::z\_foils, datas::z\_pitch, datas::z\_sail, and datas::z\_yaw.

## 7.10 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/compute\_state\_dot.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "hydro_includes.h"
```

Include dependency graph for compute\_state\_dot.cpp:



### Defines

- `#define pi` 3.141592653589793
- `#define PI` 3.141592653589793

### Functions

- void `compute_state_dot` (struct `datas` \*d, struct `parameters` \*P, double \*state)  
*Evaluates right hand side of partial differential equation.*

#### 7.10.1 Detailed Description

Author:

`basile.graf@epfl.ch`

This file contains stuff related to evaluation of  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$

Definition in file `compute_state_dot.cpp`.

#### 7.10.2 Define Documentation

##### 7.10.2.1 `#define PI` 3.141592653589793

Definition at line 22 of file `compute_state_dot.cpp`.

##### 7.10.2.2 `#define pi` 3.141592653589793

Definition at line 21 of file `compute_state_dot.cpp`.



### 7.10.3 Function Documentation

#### 7.10.3.1 void compute\_state\_dot (struct **datas** \* *d*, struct **parameters** \* *P*, double \* *state*)

Evaluates right hand side of partial differential equation.

This function evaluates  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$

$\vec{q}$  : state, i.e. state

$\vec{Q}$  : generalized applied forces, i.e. computed from \*d and \*P

$\vec{p}$  : lots of parameters , i.e. from \*d and \*P

##### Parameters:

\**d* : pointer to the datas data-structure

\**P* : parameters struct coming from MATLAB

*state*,: state vector (double[12])

Definition at line 37 of file compute\_state\_dot.cpp.

References data\_update\_state(), datas::dEqu\_left, datas::dEqu\_right, datas::dEqu\_yaw, EE\_flat\_compute(), datas::EE\_left, datas::EE\_left\_flat, datas::EE\_right, datas::EE\_right\_flat, datas::EE\_yaw, datas::EE\_yaw\_flat, equ\_compute(), datas::Equ\_left, datas::Equ\_right, datas::Equ\_yaw, and datas::Fac\_update.

Referenced by f().

## 7.11 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/compute\_state\_dot.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void `compute_state_dot` (struct `datas` \*d, struct `parameters` \*P, double \*state)  
*Evaluates right hand side of partial differential equation.*

#### 7.11.1 Detailed Description

**Author:**

`basile.graf@epfl.ch`

This file contains stuff related to evaluation of  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$

Definition in file `compute_state_dot.h`.

#### 7.11.2 Function Documentation

##### 7.11.2.1 void compute\_state\_dot (struct `datas` \*d, struct `parameters` \*P, double \*state)

Evaluates right hand side of partial differential equation.

This function evaluates  $f$  in  $\frac{\partial \vec{q}}{\partial t} = f(\vec{q}, \vec{Q}, \vec{p})$

$\vec{q}$ : state, i.e. state

$\vec{Q}$ : generalized applied forces, i.e. computed from \*d and \*P

$\vec{p}$ : lots of parameters , i.e. from \*d and \*P

**Parameters:**

- \*d : pointer to the datas data-structure
- \*P : parameters struct coming from MATLAB
- state,: state vector (double[12])

Definition at line 37 of file `compute_state_dot.cpp`.

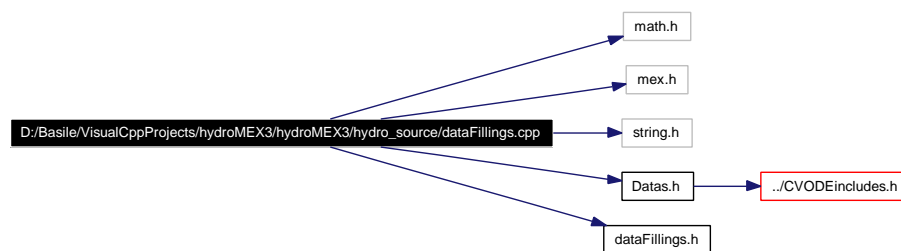
References `data_update_state()`, `datas::dEqu_left`, `datas::dEqu_right`, `datas::dEqu_yaw`, `EE_flat_-compute()`, `datas::EE_left`, `datas::EE_left_flat`, `datas::EE_right`, `datas::EE_right_flat`, `datas::EE_yaw`, `datas::EE_yaw_flat`, `equ_compute()`, `datas::Equ_left`, `datas::Equ_right`, `datas::Equ_yaw`, and `datas::Fac_-update`.

Referenced by `f()`.

## 7.12 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/dataFillings.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "Datas.h"
#include "dataFillings.h"
```

Include dependency graph for dataFillings.cpp:



### Defines

- `#define pi` 3.141592653589793
- `#define PI` 3.141592653589793

### Functions

- void `data_init` (`datas *d`)  
*Initialize constants in the data structure datas.*
- void `data_update_state` (`datas *d`, `double *state`)  
*Update data structure datas from state-vector.*
- void `data_update_param` (`datas *d`, `parameters *P`)  
*Update data structure datas from values from structure \*P.*
- void `angles_update` (`datas *d`)  
*Updates angles from Target/Sliders instrument values.*

### 7.12.1 Detailed Description

Author:

`basile.graf@epfl.ch`

This file contains stuff for updating the `datas` datastructure pointed by `*d`

Definition in file `dataFillings.cpp`.

## 7.12.2 Define Documentation

### 7.12.2.1 #define PI 3.141592653589793

Definition at line 20 of file dataFillings.cpp.

### 7.12.2.2 #define pi 3.141592653589793

Definition at line 19 of file dataFillings.cpp.

## 7.12.3 Function Documentation

### 7.12.3.1 void angles\_update (datas \* d)

Updates angles from Target/Sliders instrument values.

Updates surfaces-angles from Target/Sliders instrument values. Angles should be controlled via these instrument variables so that both angles AND instrument appearance are modified!

#### Parameters:

\**d* : pointer to the datas data-structure

Definition at line 187 of file dataFillings.cpp.

References datas::Cal\_left\_neutral, datas::Cal\_pitch\_neutral, datas::Cal\_right\_neutral, datas::Cal\_yaw\_neutral, datas::Hslider\_x, datas::tanCal\_left, datas::tanCal\_pitch, datas::tanCal\_right, datas::tanCal\_yaw, datas::tanCal\_yaw0, datas::target\_x, datas::target\_y, and datas::Vslider\_x.

Referenced by DrawGLScene().

### 7.12.3.2 void data\_init (datas \* d)

Initialize constants in the data structure datas.

Initialisation for constant values (like acceleration *g*, ...)

#### Parameters:

\**d* : pointer to the datas data-structure

Definition at line 26 of file dataFillings.cpp.

References datas::A\_pitch, datas::A\_sail, datas::A\_structure, datas::Baume, datas::CD\_structure, datas::Chord, datas::Chord\_max, datas::Chord\_min, datas::Chord\_pitch, datas::Chord\_yaw, datas::E\_foils, datas::E\_yaw, datas::Env\_pitch, datas::epsilon, datas::Fac\_update, datas::g, datas::Hslider\_x, datas::Ix, datas::Iy, datas::Iz, datas::L\_mast, datas::L\_nose, datas::Long, datas::M, pi, datas::rho, datas::rho\_air, datas::tanCal\_left, datas::tanCal\_pitch, datas::tanCal\_right, datas::target\_x, datas::target\_y, datas::V\_wave0, datas::Vslider\_x, datas::Wave\_amp0, datas::Wave\_angle0, and datas::Wind.

Referenced by mexFunction().

### 7.12.3.3 void data\_update\_param (datas \* d, parameters \* P)

Update data structure datas from values from structure \*P.

Update data structure datas from values from structure \*P coming from MATLAB

**Parameters:**

\**d* : pointer to the datas data-structure

\**P* : pointer to parameters strucure comming from MATLAB

Definition at line 115 of file dataFillings.cpp.

References datas::alpha\_foils, datas::alpha\_left, datas::alpha\_right, datas::angle\_girouette, datas::Cal\_left\_neutral, datas::Cal\_pitch\_neutral, datas::Cal\_right\_neutral, datas::Cal\_yaw\_neutral, datas::E\_yaw, datas::Env, datas::Fac\_update, datas::G\_shift, datas::L\_mast, datas::Lambda, datas::Lambda0, datas::Lambda1, datas::Lambda2, datas::Long, datas::max\_abs\_angle\_baume, datas::N, parameters::param, pi, datas::tanAngle\_sail, datas::tanCal\_left, datas::tanCal\_right, datas::tanCal\_yaw, datas::tanCal\_yaw0, datas::V\_wave, datas::V\_wave0, datas::V\_wave1, datas::V\_wave2, datas::Wave\_amp, datas::Wave\_amp0, datas::Wave\_amp1, datas::Wave\_amp2, datas::Wave\_angle, datas::Wave\_angle0, datas::Wave\_angle1, datas::Wave\_angle2, datas::Wind, datas::Wind\_angle, datas::Wind\_x, datas::Wind\_y, datas::x\_foils, datas::x\_pitch, datas::x\_sail, datas::x\_yaw, datas::y\_foil\_left, datas::y\_foil\_right, datas::y\_pitch, datas::y\_sail, datas::y\_yaw, datas::z\_foils, datas::z\_pitch, datas::z\_sail, and datas::z\_yaw.

Referenced by mexFunction().

### 7.12.3.4 void data\_update\_state (datas \* *d*, double \* *state*)

Update data structure datas from state-vector.

Update data structure datas from state-vector (i.e. state or  $\vec{q}$ )

**Parameters:**

\**d* : pointer to the datas data-structure

*state*,: pointer to state vector ( $\vec{q}$ , double[12])

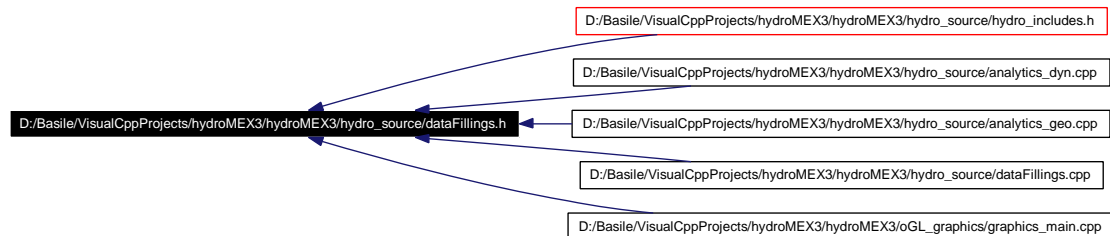
Definition at line 86 of file dataFillings.cpp.

References datas::dphi, datas::dpsi, datas::dtheta, datas::dx, datas::dy, datas::dz, datas::phi, datas::psi, datas::theta, datas::x, datas::y, and datas::z.

Referenced by compute\_state\_dot(), f(), and mexFunction().

## 7.13 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/dataFillings.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void [data\\_init](#) ([datas](#) \*d)  
*Initialize constants in the data strucure datas.*
- void [data\\_update\\_state](#) ([datas](#) \*d, double \*state)  
*Update data structure datas from state-vector.*
- void [data\\_update\\_param](#) ([datas](#) \*d, [parameters](#) \*P)  
*Update data structure datas from values from structure \*P.*
- void [angles\\_update](#) ([datas](#) \*d)  
*Updates angles from Target/Sliders instrument values.*

### 7.13.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains stuff for updating the datas datastructure pointed by \*d

Definition in file [dataFillings.h](#).

### 7.13.2 Function Documentation

#### 7.13.2.1 void [angles\\_update](#) ([datas](#) \*d)

Updates angles from Target/Sliders instrument values.

Updates sufaces-angles from Target/Sliders instrument values. Angles should be controlled via these instrument variables so that both angles AND instrument appearance are modified!

**Parameters:**

\*d : pointer to the datas data-structure

Definition at line 187 of file dataFillings.cpp.

References `datas::Cal_left_neutral`, `datas::Cal_pitch_neutral`, `datas::Cal_right_neutral`, `datas::Cal_yaw_neutral`, `datas::Hslider_x`, `datas::tanCal_left`, `datas::tanCal_pitch`, `datas::tanCal_right`, `datas::tanCal_yaw`, `datas::tanCal_yaw0`, `datas::target_x`, `datas::target_y`, and `datas::Vslider_x`.

Referenced by `DrawGLScene()`.

#### 7.13.2.2 void data\_init (datas \* d)

Initialize constants in the data structure `datas`.

Initialisation for constant values (like acceleration  $g$ , ...)

##### Parameters:

*\*d* : pointer to the `datas` data-structure

Definition at line 26 of file dataFillings.cpp.

References `datas::A_pitch`, `datas::A_sail`, `datas::A_structure`, `datas::Baume`, `datas::CD_structure`, `datas::Chord`, `datas::Chord_max`, `datas::Chord_min`, `datas::Chord_pitch`, `datas::Chord_yaw`, `datas::E_foils`, `datas::E_yaw`, `datas::Env_pitch`, `datas::epsilon`, `datas::Fac_update`, `datas::g`, `datas::Hslider_x`, `datas::Ix`, `datas::Iy`, `datas::Iz`, `datas::L_mast`, `datas::L_nose`, `datas::Long`, `datas::M`, `pi`, `datas::rho`, `datas::rho_air`, `datas::tanCal_left`, `datas::tanCal_pitch`, `datas::tanCal_right`, `datas::target_x`, `datas::target_y`, `datas::V_wave0`, `datas::Vslider_x`, `datas::Wave_amp0`, `datas::Wave_angle0`, and `datas::Wind`.

Referenced by `mexFunction()`.

#### 7.13.2.3 void data\_update\_param (datas \* d, parameters \* P)

Update data structure `datas` from values from structure `*P`.

Update data structure `datas` from values from structure `*P` coming from MATLAB

##### Parameters:

*\*d* : pointer to the `datas` data-structure

*\*P* : pointer to parameters structure coming from MATLAB

Definition at line 115 of file dataFillings.cpp.

References `datas::alpha_foils`, `datas::alpha_left`, `datas::alpha_right`, `datas::angle_girouette`, `datas::Cal_left_neutral`, `datas::Cal_pitch_neutral`, `datas::Cal_right_neutral`, `datas::Cal_yaw_neutral`, `datas::E_yaw`, `datas::Env`, `datas::Fac_update`, `datas::G_shift`, `datas::L_mast`, `datas::Lambda`, `datas::Lambda0`, `datas::Lambda1`, `datas::Lambda2`, `datas::Long`, `datas::max_abs_angle_baume`, `datas::N`, `parameters::param`, `pi`, `datas::tanAngle_sail`, `datas::tanCal_left`, `datas::tanCal_right`, `datas::tanCal_yaw`, `datas::tanCal_yaw0`, `datas::V_wave`, `datas::V_wave0`, `datas::V_wave1`, `datas::V_wave2`, `datas::Wave_amp`, `datas::Wave_amp0`, `datas::Wave_amp1`, `datas::Wave_amp2`, `datas::Wave_angle`, `datas::Wave_angle0`, `datas::Wave_angle1`, `datas::Wave_angle2`, `datas::Wind`, `datas::Wind_angle`, `datas::Wind_x`, `datas::Wind_y`, `datas::x_foils`, `datas::x_pitch`, `datas::x_sail`, `datas::x_yaw`, `datas::y_foil_left`, `datas::y_foil_right`, `datas::y_pitch`, `datas::y_sail`, `datas::y_yaw`, `datas::z_foils`, `datas::z_pitch`, `datas::z_sail`, and `datas::z_yaw`.

Referenced by `mexFunction()`.

#### 7.13.2.4 void data\_update\_state (datas \* d, double \* state)

Update data structure `datas` from state-vector.

Update data structure `datas` from state-vector (i.e. state or  $\vec{q}$ )

**Parameters:**

*\*d* : pointer to the `datas` data-structure

*state,* : pointer to state vector ( $\vec{q}$ , `double[12]`)

Definition at line 86 of file `dataFillings.cpp`.

References `datas::dphi`, `datas::dpsi`, `datas::dtheta`, `datas::dx`, `datas::dy`, `datas::dz`, `datas::phi`, `datas::psi`, `datas::theta`, `datas::x`, `datas::y`, and `datas::z`.

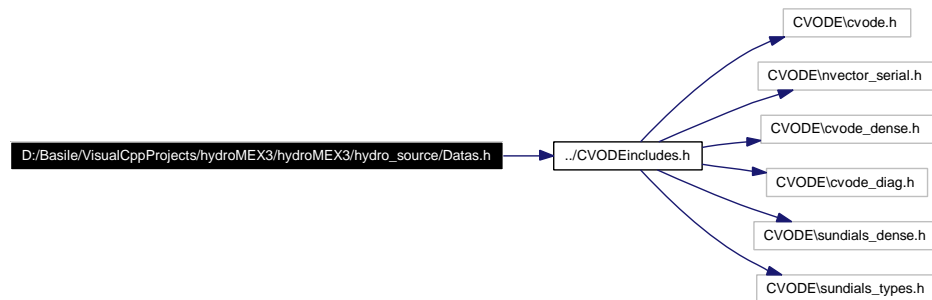
Referenced by `compute_state_dot()`, `f()`, and `mexFunction()`.



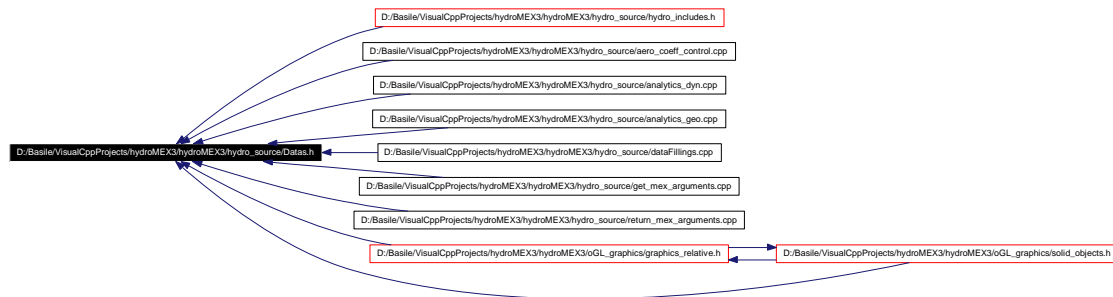
## 7.14 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/Datas.h File Reference

```
#include "../CVODEincludes.h"
```

Include dependency graph for Datas.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [parameters](#)  
*parameters struct declaration*
- struct [datas](#)  
*datas struct declaration*
- struct [ODE\\_data](#)  
*Datas relative to the ODE solver.*
- struct [joy\\_parameters](#)  
*Struct for joystick things.*

#### 7.14.1 Detailed Description

Author:

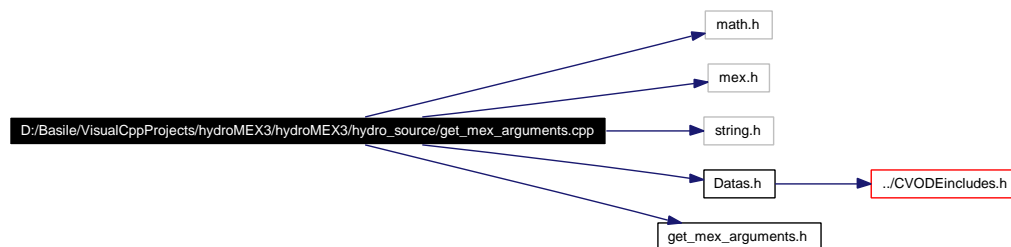
[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains declaration for different data-structures used globally  
Definition in file [Datas.h](#).

## 7.15 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/get\_mex\_arguments.cpp File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
#include "Datas.h"
#include "get_mex_arguments.h"
```

Include dependency graph for get\_mex\_arguments.cpp:



### Defines

- `#define pi` 3.141592653589793
- `#define PI` 3.141592653589793

### Functions

- void `initParameters` (struct `parameters` \*P)  
*Initialize integrator options to default.*
- void `get_Parameters` (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], struct `parameters` \*P)  
*Get integrator options parameters From MATLAB.*
- double \* `get_state` (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])  
*Get the initial ( $t = 0$ ) state vector ( $\vec{q}$ ) from MATLAB.*
- void `delete_state` (double \*state)  
*Free state vector ( $\vec{q}$ ) memory.*
- char \* `get_flag` (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])  
*Get flag from MATLAB (not used).*
- void `get_joystick_parameters` (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], struct `joy_parameters` \*jP)  
*Get joystick options parameters From MATLAB.*
- void `get_time` (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], `ODE_data` \*f\_dat)

*Get time parameter from MATLAB.*

### 7.15.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains functions for getting parameters from MATLAB

Definition in file [get\\_mex\\_arguments.cpp](#).

### 7.15.2 Define Documentation

#### 7.15.2.1 #define PI 3.141592653589793

Definition at line 19 of file [get\\_mex\\_arguments.cpp](#).

#### 7.15.2.2 #define pi 3.141592653589793

Definition at line 18 of file [get\\_mex\\_arguments.cpp](#).

### 7.15.3 Function Documentation

#### 7.15.3.1 void delete\_state (double \* state)

Free state vector ( $\vec{q}$ ) memory.

Free the memory allocated by [get\\_state\(\)](#)

**Parameters:**

*\*state* pointer to the double[12] state array ( $\vec{q}$ )

Definition at line 175 of file [get\\_mex\\_arguments.cpp](#).

Referenced by [mexFunction\(\)](#).

#### 7.15.3.2 char\* get\_flag (int nlhs, mxArray \* plhs[ ], int nrhs, const mxArray \* prhs[ ])

Get flag from MATLAB (not used).

Get the flag parameter from MATLAB (NOT USED)

**Parameters:**

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs[ ]* Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs[ ]* Array of pointers to the right hand side arguments from MATLAB prompt

Definition at line 182 of file [get\\_mex\\_arguments.cpp](#).

**7.15.3.3 void get\_joystick\_parameters (int *nlhs*, mxArray \* *plhs*[], int *nrhs*, const mxArray \* *prhs*[], struct [joy\\_parameters](#) \* *jP*)**

Get joystick options parameters From MATLAB.

Get joystick settings from MATLAB.

In the MATLAB prompt, a struct *jP* can be passed containing some or all of the followings (if not set, default is used):

[jP.dirX](#): Positive direction for axis X

[jP.dirY](#): Positive direction for axis Y

[jP.dirZ](#): Positive direction for axis Z

[jP.dirR](#): Positive direction for axis R

[jP.doX](#): Use joystick control or keyboard for axis X

[jP.doY](#): Use joystick control or keyboard for axis Y

[jP.doZ](#): Use joystick control or keyboard for axis Z

[jP.doR](#): Use joystick control or keyboard for axis R

Definition at line 213 of file `get_mex_arguments.cpp`.

References `joy_parameters::dirR`, `joy_parameters::dirX`, `joy_parameters::dirY`, `joy_parameters::dirZ`, `joy_parameters::doR`, `joy_parameters::doX`, `joy_parameters::doY`, and `joy_parameters::doZ`.

Referenced by `mexFunction()`.

**7.15.3.4 void get\_Parameters (int *nlhs*, mxArray \* *plhs*[], int *nrhs*, const mxArray \* *prhs*[], struct [parameters](#) \* *P*)**

Get integrator options parameters From MATLAB.

Get solver settings from MATLAB.

In the MATLAB prompt, a struct *P* can (must) be passed containing some or all of the followings (if not set, default is used, except for `param`):

`P.abstol`: Absolute tolerance (scalar (all the same) or vector of length 12)

`P.reltol`: Relative tolerance (scalar)

`P.param` : Parameters vector (diverse), MUST be given, see MATLAB model...

**Parameters:**

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs*[] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs*[] Array of pointers to the right hand side arguments from MATLAB prompt

*\*P* pointer to the parameters struct (in C code)

Definition at line 49 of file `get_mex_arguments.cpp`.

References `parameters::reltol`.

Referenced by `mexFunction()`.

### 7.15.3.5 double\* get\_state (int *nlhs*, mxArray \* *plhs*[ ], int *nrhs*, const mxArray \* *prhs*[ ])

Get the initial ( $t = 0$ ) state vector ( $\vec{q}$ ) from MATLAB.

Copy  $\vec{q}$  at  $t = 0$  from MATLAB

#### Parameters:

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs*[ ] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs*[ ] Array of pointers to the right hand side arguments from MATLAB prompt

#### Returns:

pointer to the double[12] state array ( $\vec{q}$ )

Definition at line 159 of file get\_mex\_arguments.cpp.

Referenced by mexFunction().

### 7.15.3.6 void get\_time (int *nlhs*, mxArray \* *plhs*[ ], int *nrhs*, const mxArray \* *prhs*[ ], ODE\_data \* *f\_dat*)

Get time parameter from MATLAB.

Meaning:

0: Simulate indefinitely without returning anything to MATLAB  $t_{end}$ : Simulate from  $t = 0$  to  $t = t_{end}$  and return simulation results to MATLAB

#### Parameters:

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs*[ ] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs*[ ] Array of pointers to the right hand side arguments from MATLAB prompt

*\*f\_dat* ODE solver data struct

Definition at line 293 of file get\_mex\_arguments.cpp.

References ODE\_data::time\_param.

Referenced by mexFunction().

### 7.15.3.7 void initParameters (struct parameters \* *P*)

Initialize integrator options to default.

Initializes P.reltol and P.abstol to default values.

The P.param array is set to zeros (not a valid default for meaningful simulation, just to avoid any crash!!!)

#### Parameters:

*\*P* pointer to the parameters struct

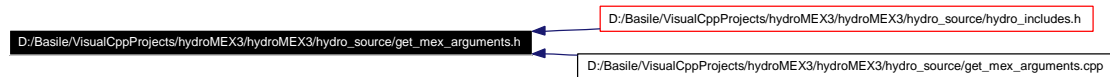
Definition at line 23 of file get\_mex\_arguments.cpp.

References parameters::abstol, parameters::param, PARAMNUM, parameters::reitol, and STATENUM.

Referenced by mexFunction().

## 7.16 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/get\_mex\_arguments.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- #define [PARAMNUM](#) 10
- #define [STATENUM](#) 12

### Functions

- void [initParameters](#) (struct [parameters](#) \*P)  
*Initialize integrator options to default.*
- void [get\\_Parameters](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], struct [parameters](#) \*P)  
*Get integrator options parameters From MATLAB.*
- double \* [get\\_state](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])  
*Get the initial ( $t = 0$ ) state vector ( $\vec{q}$ ) from MATLAB.*
- void [delete\\_state](#) (double \*state)  
*Free state vector ( $\vec{q}$ ) memory.*
- char \* [get\\_flag](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])  
*Get flag from MATLAB (not used).*
- void [get\\_joystick\\_parameters](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], struct [joy\\_parameters](#) \*jP)  
*Get joystick options parameters From MATLAB.*
- void [get\\_time](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], [ODE\\_data](#) \*f\_dat)  
*Get time parameter from MATLAB.*

#### 7.16.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains functions for getting parameters from MATLAB

Definition in file [get\\_mex\\_arguments.h](#).



## 7.16.2 Define Documentation

### 7.16.2.1 #define PARAMNUM 10

Definition at line 13 of file get\_mex\_arguments.h.

Referenced by initParameters().

### 7.16.2.2 #define STATENUM 12

Definition at line 14 of file get\_mex\_arguments.h.

Referenced by initParameters().

## 7.16.3 Function Documentation

### 7.16.3.1 void delete\_state (double \* state)

Free state vector ( $\vec{q}$ ) memory.

Free the memory allocated by [get\\_state\(\)](#)

#### Parameters:

*\*state* pointer to the double[12] state array ( $\vec{q}$ )

Definition at line 175 of file get\_mex\_arguments.cpp.

Referenced by mexFunction().

### 7.16.3.2 char\* get\_flag (int nlhs, mxArray \* plhs[ ], int nrhs, const mxArray \* prhs[ ])

Get flag from MATLAB (not used).

Get the flag parameter from MATLAB (NOT USED)

#### Parameters:

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs[ ]* Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs[ ]* Array of pointers to the right hand side arguments from MATLAB prompt

Definition at line 182 of file get\_mex\_arguments.cpp.

### 7.16.3.3 void get\_joystick\_parameters (int nlhs, mxArray \* plhs[ ], int nrhs, const mxArray \* prhs[ ], struct [joy\\_parameters](#) \* jP)

Get joystick options parameters From MATLAB.

Get joystick settings from MATLAB.

In the MATLAB prompt, a struct jP can be passed containing some or all of the followings (if not set, default is used):

[jP.dirX](#): Positive direction for axis X

**jP.dirY**: Positive direction for axis Y

**jP.dirZ**: Positive direction for axis Z

**jP.dirR**: Positive direction for axis R

**jP.doX**: Use joystick control or keyboard for axis X

**jP.doY**: Use joystick control or keyboard for axis Y

**jP.doZ**: Use joystick control or keyboard for axis Z

**jP.doR**: Use joystick control or keyboard for axis R

Definition at line 213 of file `get_mex_arguments.cpp`.

References `joy_parameters::dirR`, `joy_parameters::dirX`, `joy_parameters::dirY`, `joy_parameters::dirZ`, `joy_parameters::doR`, `joy_parameters::doX`, `joy_parameters::doY`, and `joy_parameters::doZ`.

Referenced by `mexFunction()`.

### 7.16.3.4 `void get_Parameters (int nlhs, mxArray * plhs[ ], int nrhs, const mxArray * prhs[ ], struct parameters * P)`

Get integrator options parameters From MATLAB.

Get solver settings from MATLAB.

In the MATLAB prompt, a struct *P* can (must) be passed containing some or all of the followings (if not set, default is used, except for param):

*P*.abstol: Absolute tolerance (scalar (all the same) or vector of length 12)

*P*.reltol: Relative tolerance (scalar)

*P*.param : Parameters vector (diverse), MUST be given, see MATLAB model...

#### Parameters:

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs*[ ] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs*[ ] Array of pointers to the right hand side arguments from MATLAB prompt

*\*P* pointer to the parameters struct (in C code)

Definition at line 49 of file `get_mex_arguments.cpp`.

References `parameters::reltol`.

Referenced by `mexFunction()`.

### 7.16.3.5 `double* get_state (int nlhs, mxArray * plhs[ ], int nrhs, const mxArray * prhs[ ])`

Get the initial ( $t = 0$ ) state vector ( $\vec{q}$ ) from MATLAB.

Copy  $\vec{q}$  at  $t = 0$  from MATLAB

#### Parameters:

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs*[ ] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs[ ]* Array of pointers to the right hand side arguments from MATLAB prompt

**Returns:**

pointer to the double[12] state array ( $\vec{q}$ )

Definition at line 159 of file get\_mex\_arguments.cpp.

Referenced by mexFunction().

**7.16.3.6 void get\_time (int *nlhs*, mxArray \* *plhs*[ ], int *nrhs*, const mxArray \* *prhs*[ ], ODE\_data \* *f\_dat*)**

Get time parameter from MATLAB.

Meaning:

0: Simulate indefinitely without returning anything to MATLAB  $t_{end}$ : Simulate from  $t = 0$  to  $t = t_{end}$  and return simulation results to MATLAB

**Parameters:**

*nlhs* Number of left hand side arguments in the MATLAB prompt

*\*plhs[ ]* Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

*\*prhs[ ]* Array of pointers to the right hand side arguments from MATLAB prompt

*\*f\_dat* ODE solver data struct

Definition at line 293 of file get\_mex\_arguments.cpp.

References ODE\_data::time\_param.

Referenced by mexFunction().

**7.16.3.7 void initParameters (struct parameters \* *P*)**

Initialize integrator options to default.

Initializes P.reltol and P.abstol to default values.

The P.param array is set to zeros (not a valid default for meaningful simulation, just to avoid any crash!!!)

**Parameters:**

*\*P* pointer to the parameters struct

Definition at line 23 of file get\_mex\_arguments.cpp.

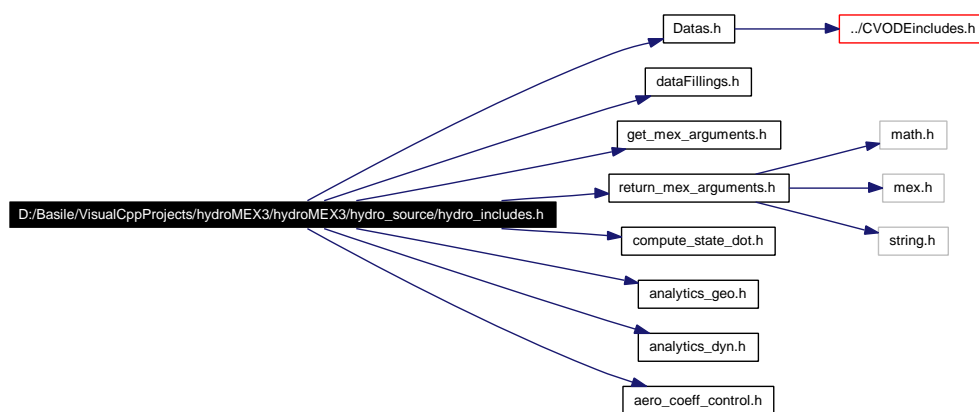
References parameters::abstol, parameters::param, PARAMNUM, parameters::reltol, and STATENUM.

Referenced by mexFunction().

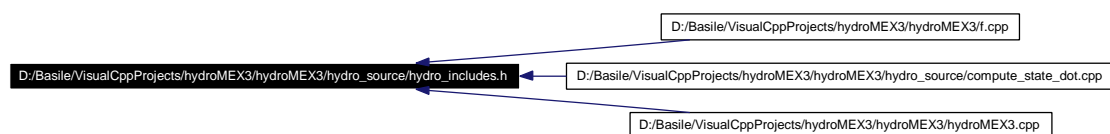
## 7.17 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/hydro\_source/hydro\_includes.h File Reference

```
#include "Datas.h"
#include "dataFillings.h"
#include "get_mex_arguments.h"
#include "return_mex_arguments.h"
#include "compute_state_dot.h"
#include "analytics_geo.h"
#include "analytics_dyn.h"
#include "aero_coeff_control.h"
```

Include dependency graph for hydro\_includes.h:



This graph shows which files directly or indirectly include this file:



### 7.17.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

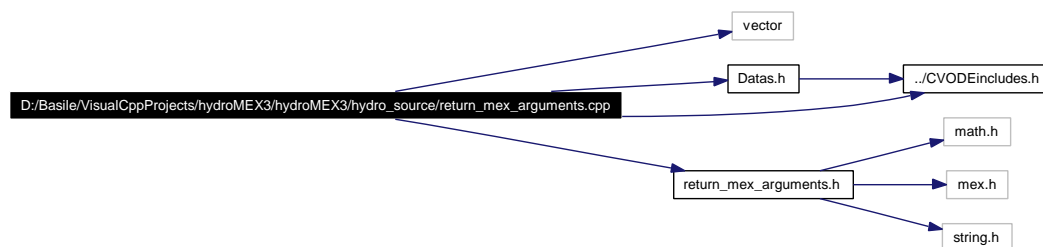
This file contains all includes for the files related to the hydroptere model, i.e files in directory "hydro\_source"

Definition in file [hydro\\_includes.h](#).

## 7.18 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/return\_mex\_arguments.cpp File Reference

```
#include <vector>
#include "Datas.h"
#include "return_mex_arguments.h"
#include "../CVODEIncludes.h"
```

Include dependency graph for return\_mex\_arguments.cpp:



## Namespaces

- namespace `std`

## Defines

- `#define` `pi` 3.141592653589793
- `#define` `PI` 3.141592653589793
- `#define` `Ith(v, i)` NV\_Ith\_S(v,i)

## Functions

- void `store_state` (double time, N\_Vector state\_at\_t)  
*Store state vector  $\vec{q}_i$  and time  $t_i$  at each integration step.*
- void `return_args` (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], ODE\_data \*f\_dat)  
*Return arguments to MATLAB.*

## Variables

- vector< double > `sout00`
- vector< double > `sout01`
- vector< double > `sout02`
- vector< double > `sout03`
- vector< double > `sout04`
- vector< double > `sout05`

- vector< double > [sout06](#)
- vector< double > [sout07](#)
- vector< double > [sout08](#)
- vector< double > [sout09](#)
- vector< double > [sout10](#)
- vector< double > [sout11](#)
- vector< double > [timev](#)

### 7.18.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains functions for returning parameters to MATLAB

Definition in file [return\\_mex\\_arguments.cpp](#).

### 7.18.2 Define Documentation

#### 7.18.2.1 `#define Ith(v, i) NV_Ith_S(v,i)`

Definition at line 24 of file [return\\_mex\\_arguments.cpp](#).

#### 7.18.2.2 `#define PI 3.141592653589793`

Definition at line 21 of file [return\\_mex\\_arguments.cpp](#).

#### 7.18.2.3 `#define pi 3.141592653589793`

Definition at line 20 of file [return\\_mex\\_arguments.cpp](#).

### 7.18.3 Function Documentation

#### 7.18.3.1 `void return_args (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ], ODE\_data *f_dat)`

Return arguments to MATLAB.

Return the simulation results and simulation times (if asked for, see [get\\_time\(\)](#)) and the cumulative number of integration steps performed (generally bigger than number of output points)

**Parameters:**

*nlhs* Number of left hand side arguments in the MATLAB prompt

\**plhs*[ ] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

\**prhs*[ ] Array of pointers to the right hand side arguments from MATLAB prompt

\**f\_dat* ODE solver data struct

## 7.18

D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/return\_mex\_arguments.cpp

### File Reference

89

Definition at line 60 of file return\_mex\_arguments.cpp.

References sout00, sout01, sout02, sout03, sout04, sout05, sout06, sout07, sout08, sout09, sout10, and sout11.

Referenced by mexFunction().

### 7.18.3.2 void store\_state (double time, N\_Vector state\_at\_t)

Store state vector  $\vec{q}_i$  and time  $t_i$  at each integration step.

If the programm has to return simulation results to MATLAB (see [get\\_time\(\)](#)), the state vector  $\vec{q}_i$  and the time  $t_i$  has to be stored at each integration step. They are stored in dynamic C++ vectors of type vector.

#### Parameters:

*time*  $t_i$

*state\_at\_t*  $\vec{q}_i$  in the data format used by the SUNDIALS ODE solver CVODE

Definition at line 43 of file return\_mex\_arguments.cpp.

References Ith, sout00, sout01, sout02, sout03, sout04, sout05, sout06, sout07, sout08, sout09, sout10, sout11, and timev.

Referenced by solve\_ODE().

## 7.18.4 Variable Documentation

### 7.18.4.1 vector<double> sout00

Definition at line 29 of file return\_mex\_arguments.cpp.

Referenced by return\_args(), and store\_state().

### 7.18.4.2 vector<double> sout01

Definition at line 30 of file return\_mex\_arguments.cpp.

Referenced by return\_args(), and store\_state().

### 7.18.4.3 vector<double> sout02

Definition at line 31 of file return\_mex\_arguments.cpp.

Referenced by return\_args(), and store\_state().

### 7.18.4.4 vector<double> sout03

Definition at line 32 of file return\_mex\_arguments.cpp.

Referenced by return\_args(), and store\_state().

### 7.18.4.5 vector<double> sout04

Definition at line 33 of file return\_mex\_arguments.cpp.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.6** `vector<double>` [sout05](#)

Definition at line 34 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.7** `vector<double>` [sout06](#)

Definition at line 35 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.8** `vector<double>` [sout07](#)

Definition at line 36 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.9** `vector<double>` [sout08](#)

Definition at line 37 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.10** `vector<double>` [sout09](#)

Definition at line 38 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.11** `vector<double>` [sout10](#)

Definition at line 39 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.12** `vector<double>` [sout11](#)

Definition at line 40 of file `return_mex_arguments.cpp`.

Referenced by `return_args()`, and `store_state()`.

#### **7.18.4.13** `vector<double>` [timev](#)

Definition at line 41 of file `return_mex_arguments.cpp`.

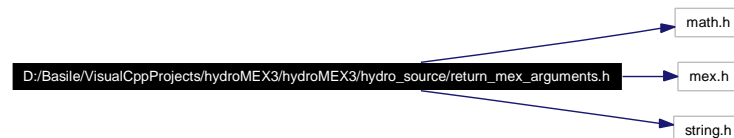
Referenced by `store_state()`.



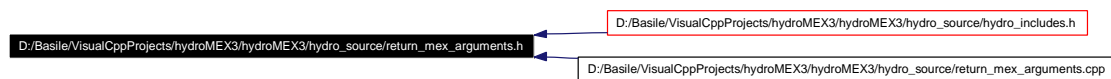
## 7.19 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydro\_source/return\_mex\_arguments.h File Reference

```
#include "math.h"
#include "mex.h"
#include "string.h"
```

Include dependency graph for return\_mex\_arguments.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void [store\\_state](#) (double time, N\_Vector state\_at\_t)  
*Store state vector  $\vec{q}_i$  and time  $t_i$  at each integration step.*
- void [return\\_args](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ], [ODE\\_data](#) \*f\_dat)  
*Return arguments to MATLAB.*

### 7.19.1 Detailed Description

Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains functions for returning parameters to MATLAB

Definition in file [return\\_mex\\_arguments.h](#).

### 7.19.2 Function Documentation

#### 7.19.2.1 void [return\\_args](#) (int *nlhs*, mxArray \**plhs*[ ], int *nrhs*, const mxArray \**prhs*[ ], [ODE\\_data](#) \**f\_dat*)

Return arguments to MATLAB.

Return the simulation results and simulation times (if asked for, see [get\\_time\(\)](#)) and the cumulative number of integration steps performed (generally bigger than number of output points)

**Parameters:**

- nlhs* Number of left hand side arguments in the MATLAB prompt
- \*plhs[]* Array of pointers to the left hand side arguments from MATLAB prompt
- nrhs* Number of right hand side arguments in the MATLAB prompt
- \*prhs[]* Array of pointers to the right hand side arguments from MATLAB prompt
- \*f\_dat* ODE solver data struct

Definition at line 60 of file `return_mex_arguments.cpp`.

References `sout00`, `sout01`, `sout02`, `sout03`, `sout04`, `sout05`, `sout06`, `sout07`, `sout08`, `sout09`, `sout10`, and `sout11`.

Referenced by `mexFunction()`.

**7.19.2.2 void store\_state (double *time*, N\_Vector *state\_at\_t*)**

Store state vector  $\vec{q}_i$  and time  $t_i$  at each integration step.

If the program has to return simulation results to MATLAB (see [get\\_time\(\)](#)), the state vector  $\vec{q}_i$  and the time  $t_i$  has to be stored at each integration step. They are stored in dynamic C++ vectors of type `vector`.

**Parameters:**

- time*  $t_i$
- state\_at\_t*  $\vec{q}_i$  in the data format used by the SUNDIALS ODE solver CVODE

Definition at line 43 of file `return_mex_arguments.cpp`.

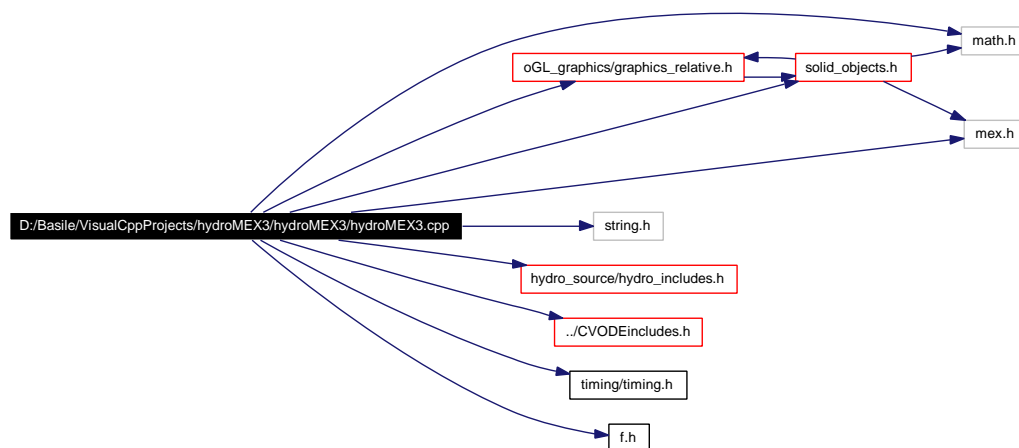
References `Ith`, `sout00`, `sout01`, `sout02`, `sout03`, `sout04`, `sout05`, `sout06`, `sout07`, `sout08`, `sout09`, `sout10`, `sout11`, and `timev`.

Referenced by `solve_ODE()`.

## 7.20 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/hydroMEX3.cpp File Reference

```
#include "oGL_graphics/graphics_relative.h"
#include "oGL_graphics/solid_objects.h"
#include "math.h"
#include "mex.h"
#include "string.h"
#include "hydro_source/hydro_includes.h"
#include "CVODEincludes.h"
#include "timing/timing.h"
#include "f.h"
```

Include dependency graph for hydroMEX3.cpp:



### Defines

- #define [MAXFLAGLEN](#) 100
- #define [pi](#) 3.141592653589793
- #define [PI](#) 3.141592653589793

### Functions

- void [mexFunction](#) (int nlhs, mxArray \*plhs[ ], int nrhs, const mxArray \*prhs[ ])
 

*Main function (mex function).*

#### 7.20.1 Detailed Description

Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This is the main project file. It contains interface to MATLAB and calls all the other routines.

Definition in file [hydroMEX3.cpp](#).

## 7.20.2 Define Documentation

### 7.20.2.1 #define MAXFLAGLEN 100

Definition at line 37 of file hydroMEX3.cpp.

### 7.20.2.2 #define PI 3.141592653589793

Definition at line 41 of file hydroMEX3.cpp.

### 7.20.2.3 #define pi 3.141592653589793

Definition at line 40 of file hydroMEX3.cpp.

## 7.20.3 Function Documentation

### 7.20.3.1 void mexFunction (int *nlhs*, mxArray \**plhs*[ ], int *nrhs*, const mxArray \**prhs*[ ])

Main function (mex function).

This is the main function. It receives and returns pointers to the arguments to MATLAB.

General initialisations and closing are done here. The main program loop is also in this function.

#### Parameters:

*nlhs* Number of left hand side arguments in the MATLAB prompt

\**plhs*[ ] Array of pointers to the left hand side arguments from MATLAB prompt

*nrhs* Number of right hand side arguments in the MATLAB prompt

\**prhs*[ ] Array of pointers to the right hand side arguments from MATLAB prompt

#### < IMPORTANT VARIABLES:

< parameters Par; parameters structure (from MATLAB)

< parameters \*P; pointer to Par, will be passed to many functions

< [joy\\_parameters](#) jPar; joystick parameters structure (from MATLAB)

< [joy\\_parameters](#) \*jP; pointer to jPar, will be passed to other functions

< datas dat; model data structure containing all model (physics) informations

< pointer to dat, will be passed to many many functions

< [ODE\\_data](#) f\_dats; solver data structure, contains pointer to the other structures and some solver relative informations

< [ODE\\_data](#) \*f\_dat; pointer to f\_dats, will be passed to solver relative functions

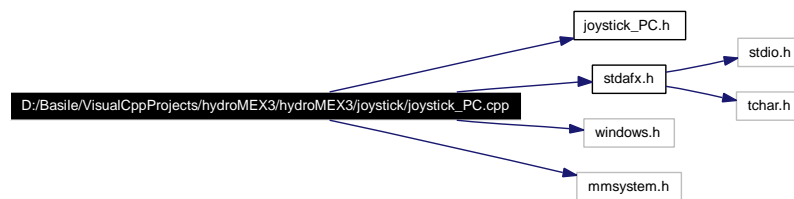
Definition at line 56 of file hydroMEX3.cpp.

References Boat\_create(), Boat\_delete(), ODE\_data::d, data\_init(), data\_update\_param(), data\_update\_state(), delete\_state(), draw\_graphics(), draw\_graphics\_init(), draw\_graphics\_kill(), free\_wave\_variables(), get\_joystick\_parameters(), get\_Parameters(), get\_state(), get\_time(), initParameters(), ODE\_data::P, pass\_joy\_parameters(), return\_args(), solve\_ODE(), solver\_free(), solver\_init(), ODE\_data::state, datas::t, tic(), ODE\_data::time\_param, toc(), and wave\_init().

## 7.21 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/joystick\_PC.cpp File Reference

```
#include "joystick_PC.h"
#include "stdafx.h"
#include <windows.h>
#include <mmsystem.h>
```

Include dependency graph for joystick\_PC.cpp:



### Functions

- int [joystick\\_init](#) ()  
*Initializes joystick stuff.*
- void [joystick\\_getXYZR](#) (float \*X, float \*Y, float \*Z, float \*R, bool [doX](#), bool [doY](#), bool [doZ](#), bool [doR](#))  
*Get joystick position under MS Windows.*
- void [joystick\\_close](#) ()  
*Close joystick under MS Windoze.*

### Variables

- JOYINFOEX [ActualPos](#)
- DWORD [dwBoutons](#)
- UINT [uMax](#) [2]
- UINT [uMin](#) [2]
- UINT [uPoolPeriod](#)
- JOYCAPS [InfosCaps](#)
- BOOL [bStop](#)
- POINT [Curseur](#)

#### 7.21.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains functions related to the joystick under MS Windoze

Definition in file [joystick\\_PC.cpp](#).

## 7.21.2 Function Documentation

### 7.21.2.1 void joystick\_close ()

Close joystick under MS Windoze.

No need to close anything under Windows => this function does nothing...

Definition at line 89 of file joystick\_PC.cpp.

Referenced by draw\_graphics\_kill().

### 7.21.2.2 void joystick\_getXYZR (float \* *X*, float \* *Y*, float \* *Z*, float \* *R*, bool *doX*, bool *doY*, bool *doZ*, bool *doR*)

Get joystick position under MS Windows.

Get the joystick positions for the axes specified. Values should be between -1.0 and 1.0

#### Parameters:

\**X* pointer to the X axis variable

\**Y* pointer to the Y axis variable

\**Z* pointer to the Z axis variable

\**R* pointer to the R axis variable

*doX* modify X value only if true

*doY* modify Y value only if true

*doZ* modify Z value only if true

*doR* modify R value only if true

Definition at line 68 of file joystick\_PC.cpp.

References ActualPos, and f().

Referenced by DrawGLScene().

### 7.21.2.3 int joystick\_init ()

Initializes joystick stuff.

Use Windows API to initialize joystick (if any connected)

Definition at line 36 of file joystick\_PC.cpp.

References ActualPos, dwBoutons, InfosCaps, and uPoolPeriod.

Referenced by draw\_graphics\_init().

## 7.21.3 Variable Documentation

### 7.21.3.1 JOYINFOEX ActualPos

Definition at line 27 of file joystick\_PC.cpp.

Referenced by joystick\_getXYZR(), and joystick\_init().

**7.21.3.2   BOOL   [bStop](#)**

Definition at line 32 of file joystick\_PC.cpp.

**7.21.3.3   POINT   [Curseur](#)**

Definition at line 33 of file joystick\_PC.cpp.

**7.21.3.4   DWORD   [dwBoutons](#)**

Definition at line 28 of file joystick\_PC.cpp.

Referenced by joystick\_init().

**7.21.3.5   JOYCAPS   [InfosCaps](#)**

Definition at line 31 of file joystick\_PC.cpp.

Referenced by joystick\_init().

**7.21.3.6   UINT   [uMax\[2\]](#)**

Definition at line 29 of file joystick\_PC.cpp.

**7.21.3.7   UINT   [uMin\[2\]](#)**

Definition at line 29 of file joystick\_PC.cpp.

**7.21.3.8   UINT   [uPoolPeriod](#)**

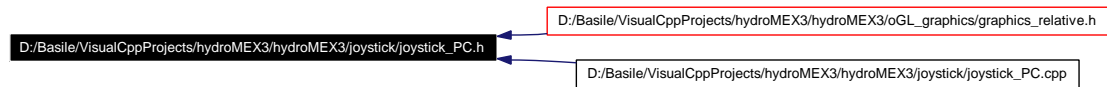
Definition at line 30 of file joystick\_PC.cpp.

Referenced by joystick\_init().



## 7.22 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/joystick\_PC.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- `int joystick_init ()`  
*Initializes joystick stuff.*
- `void joystick_getXYZR (float *X, float *Y, float *Z, float *R, bool doX, bool doY, bool doZ, bool doR)`  
*Get joystick position under MS Windows.*
- `void joystick_close ()`  
*Close joystick under MS Windoze.*

### 7.22.1 Detailed Description

**Author:**

`basile.graf@epfl.ch`

This file contains functions related to the joystick under MS Windoze

Definition in file `joystick_PC.h`.

### 7.22.2 Function Documentation

#### 7.22.2.1 void joystick\_close ()

Close joystick under MS Windoze.

No need to close anything under Windows => this function does nothing...

Definition at line 89 of file `joystick_PC.cpp`.

Referenced by `draw_graphics_kill()`.

#### 7.22.2.2 void joystick\_getXYZR (float \* X, float \* Y, float \* Z, float \* R, bool doX, bool doY, bool doZ, bool doR)

Get joystick position under MS Windows.

Get the joystick positions for the axes specified. Values should be between -1.0 and 1.0

**Parameters:**

\*X pointer to the X axis variable

*\*Y* pointer to the Y axis variable  
*\*Z* pointer to the Z axis variable  
*\*R* pointer to the R axis variable  
*doX* modify X value only if true  
*doY* modify Y value only if true  
*doZ* modify Z value only if true  
*doR* modify R value only if true

Definition at line 68 of file joystick\_PC.cpp.

References ActualPos, and f().

Referenced by DrawGLScene().

### 7.22.2.3 int joystick\_init ()

Initializes joystick stuff.

Use Windows API to initialize joystick (if any connected)

Definition at line 36 of file joystick\_PC.cpp.

References ActualPos, dwBoutons, InfosCaps, and uPoolPeriod.

Referenced by draw\_graphics\_init().

## 7.23 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/joystick/stdafx.cpp File Reference

```
#include "stdafx.h"
```

Include dependency graph for stdafx.cpp:



## 7.24 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/joystick/stdafx.h File Reference

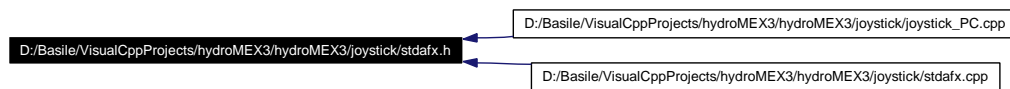
```
#include <stdio.h>
```

```
#include <tchar.h>
```

Include dependency graph for stdafx.h:



This graph shows which files directly or indirectly include this file:



### Defines

- `#define` [WIN32\\_LEAN\\_AND\\_MEAN](#)

#### 7.24.1 Define Documentation

##### 7.24.1.1 `#define` WIN32\_LEAN\_AND\_MEAN

Definition at line 9 of file `stdafx.h`.

## 7.25 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL\_graphics/graphics\_main.cpp File Reference

```
#include "graphics_relative.h"
#include "../hydro_source/dataFillings.h"
```

Include dependency graph for graphics\_main.cpp:



### Functions

- GLvoid [BuildFont](#) (GLvoid)  
*Create OSD font.*
- GLvoid [KillFont](#) (GLvoid)  
*Delete font list.*
- GLvoid [glPrint](#) (const char \*fmt,...)  
*Custom GL "Print" Routine.*
- GLvoid [ReSizeGLScene](#) (GLsizei width, GLsizei height)
- int [InitGL](#) (GLvoid)  
*Initialize OpenGL stuff.*
- int [DrawGLScene](#) (datas \*d)  
*Draws the scene (openGL).*
- GLvoid [KillGLWindow](#) (GLvoid)
- BOOL [CreateGLWindow](#) (char \*title, int width, int height, int bits, bool fullscreenflag)  
*Sub-function for window closing stuffs.*
- LRESULT CALLBACK [WndProc](#) (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)  
*Callback and keyboard management.*
- int [draw\\_graphics\\_init](#) (datas \*d)  
*General graphic stuff initialisation.*
- bool [draw\\_graphics](#) (datas \*d)  
*General graphic (scene) display and keyboard reading.*
- int [draw\\_graphics\\_kill](#) ()  
*General graphic (scene/window) closing.*
- void [pass\\_joy\\_parameters](#) (joy\_parameters \*jP)  
*Copy joystick parameters from MATLAB struct.*

## Variables

- HGLRC `hRC` = NULL
- HDC `hDC` = NULL
- HWND `hWnd` = NULL
- HINSTANCE `hInstance`
- GLuint `base`
- bool `keys` [256]
- bool `active` = TRUE
- bool `fullscreen` = TRUE
- bool `light`
- GLfloat `viewAngle1` = 0.0  
*horizontal camera view angle*
- GLfloat `viewAngle2` = 0.3  
*vertical camera view angle*
- GLfloat `dviewAngle` = 0.01
- GLfloat `viewDist` = 50.0  
*camera view distance*
- GLfloat `dviewDist` = 0.3
- GLfloat `LightAmbient` [ ] = { 0.35f, 0.35f, 0.5f, 1.0f }
- GLfloat `LightDiffuse` [ ] = { 0.6f, 0.6f, 0.6f, 1.0f }
- GLfloat `LightPosition` [ ] = { -10.0f, 20.0f, 150.0f, 0.0f }
- MSG `msg`
- BOOL `done` = FALSE
- int `joystick_exist`
- bool `doX` = TRUE
- bool `doY` = TRUE
- bool `doZ` = TRUE
- bool `doR` = TRUE
- bool `allowChangeDoX` = TRUE
- bool `allowChangeDoY` = TRUE
- bool `allowChangeDoZ` = TRUE
- bool `allowChangeDoR` = TRUE
- float `dirX` = 1.0f
- float `dirY` = 1.0f
- float `dirZ` = 1.0f
- float `dirR` = 1.0f
- bool `allowChangeDirX` = TRUE
- bool `allowChangeDirY` = TRUE
- bool `allowChangeDirZ` = TRUE
- bool `allowChangeDirR` = TRUE
- float `time_before` = 0.0
- float `time_elapsed`
- float `fps` = 0.0

## 7.25.1 Detailed Description

### Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This is the main graphic code. Window management, (main) OpenGL drawing, ...

Definition in file [graphics\\_main.cpp](#).

## 7.25.2 Function Documentation

### 7.25.2.1 GLvoid BuildFont (GLvoid)

Create OSD font.

Creates font for OSD (On Screen Display), i.e. displaying various informations on the screen... (Windows dependent)

Definition at line 74 of file [graphics\\_main.cpp](#).

References [base](#), and [hDC](#).

Referenced by [InitGL\(\)](#).

### 7.25.2.2 BOOL CreateGLWindow (char \* title, int width, int height, int bits, bool fullscreenflag)

Sub-function for window closing stuffs.

Definition at line 331 of file [graphics\\_main.cpp](#).

References [fullscreen](#), [hDC](#), [hInstance](#), [hRC](#), [hWnd](#), [InitGL\(\)](#), [KillGLWindow\(\)](#), [ReSizeGLScene\(\)](#), and [WndProc\(\)](#).

Referenced by [draw\\_graphics\(\)](#), and [draw\\_graphics\\_init\(\)](#).

### 7.25.2.3 bool draw\_graphics (datas \* d)

General graphic (scene) display and keyboard reading.

A call to this function refreshes the entire scene. This function is to be used in the main computing loop.

It also reads the keyboard for the different keyboard controls...

### Parameters:

*\*d* : pointer to the [datas](#) data-structure

### Returns:

true if all went OK

Definition at line 599 of file [graphics\\_main.cpp](#).

References [active](#), [allowChangeDirR](#), [allowChangeDirX](#), [allowChangeDirY](#), [allowChangeDirZ](#), [allowChangeDoR](#), [allowChangeDoX](#), [allowChangeDoY](#), [allowChangeDoZ](#), [datas::angle\\_girouette](#), [CreateGLWindow\(\)](#), [dirR](#), [dirX](#), [dirY](#), [dirZ](#), [done](#), [doR](#), [doX](#), [doY](#), [doZ](#), [DrawGLScene\(\)](#), [dviewAngle](#), [dviewDist](#), [fullscreen](#), [hDC](#), [datas::Hslider\\_x](#), [keys](#), [KillGLWindow\(\)](#), [datas::max\\_abs\\_angle\\_baume](#), [msg](#), [datas::t](#), [datas::tanAngle\\_sail](#), [datas::target\\_x](#), [datas::target\\_y](#), [time\\_before](#), [time\\_elapsed](#), [viewAngle1](#), [viewAngle2](#), [viewDist](#), [datas::Vslider\\_x](#), [datas::Wave\\_amp](#), [datas::Wave\\_amp0](#), [datas::Wave\\_amp1](#), [datas::Wave\\_amp2](#), [datas::Wind](#), [datas::Wind\\_angle](#), [datas::Wind\\_x](#), and [datas::Wind\\_y](#).

Referenced by `mexFunction()`.

#### 7.25.2.4 `int draw_graphics_init (datas * d)`

General graphic stuff initialisation.

Ask for full-screen or not, creates window and try to initialize the joystick.

##### Parameters:

*\*d* : pointer to the datas data-structure

##### Returns:

1 if window created successfully

Definition at line 568 of file `graphics_main.cpp`.

References `CreateGLWindow()`, `fullscreen`, `joystick_exist`, and `joystick_init()`.

Referenced by `mexFunction()`.

#### 7.25.2.5 `int draw_graphics_kill ()`

General graphic (scene/window) closing.

Closes window and joystick

##### Returns:

`msg.wParam` (MS Windows !)

Definition at line 774 of file `graphics_main.cpp`.

References `done`, `joystick_close()`, `KillGLWindow()`, and `msg`.

Referenced by `mexFunction()`.

#### 7.25.2.6 `int DrawGLScene (datas * d)`

Draws the scene (OpenGL).

Draws the scene by applying all necessary geometrical (spacial) transformations and by calling the different element drawing functions (for the boat, the sea, the instruments...). This is not the highest function for drawing the current scene, see `draw_graphics()`.

Definition at line 189 of file `graphics_main.cpp`.

References `angles_update()`, `Boat_draw()`, `dirR`, `dirX`, `dirY`, `dirZ`, `doR`, `doX`, `doY`, `doZ`, `Draw_sea()`, `datas::dx`, `datas::dy`, `f()`, `fps`, `Girouette_draw()`, `glPrint()`, `HSlider_draw()`, `datas::Hslider_x`, `joystick_exist`, `joystick_getXYZR()`, `LightPosition`, `datas::phi`, `datas::psi`, `RelativeWind_draw()`, `Sail_create_and_draw()`, `Target_draw()`, `datas::target_x`, `datas::target_y`, `datas::theta`, `time_elapsed`, `viewAngle1`, `viewAngle2`, `view-Dist`, `VSlider_draw()`, `datas::Vslider_x`, `wave_z_compute()`, `datas::Wind`, `datas::Wind_x`, `datas::Wind_y`, `datas::x`, `datas::y`, and `datas::z`.

Referenced by `draw_graphics()`.

#### 7.25.2.7 `GLvoid glPrint (const char *fmt, ...)`

Custom GL "Print" Routine.



A function that can be used like fprintf(), but for writing on the OpenGL window

Definition at line 116 of file graphics\_main.cpp.

References base.

Referenced by DrawGLScene().

#### **7.25.2.8 int InitGL (GLvoid)**

Initialize OpenGL stuff.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Definition at line 159 of file graphics\_main.cpp.

References BuildFont(), f(), LightAmbient, LightDiffuse, and LightPosition.

Referenced by CreateGLWindow().

#### **7.25.2.9 GLvoid KillFont (GLvoid)**

Delete font list.

Delete the font list created by [BuildFont\(\)](#)

Definition at line 107 of file graphics\_main.cpp.

References base.

#### **7.25.2.10 GLvoid KillGLWindow (GLvoid)**

Definition at line 287 of file graphics\_main.cpp.

References fullscreen, hDC, hInstance, hRC, and hWnd.

Referenced by CreateGLWindow(), draw\_graphics(), and draw\_graphics\_kill().

#### **7.25.2.11 void pass\_joy\_parameters (joy\_parameters \*jP)**

Copy joystick parameters from MATLAB struct.

Copy the joystick parameters (doX, doY ,doZ ,doR and dirX ,dirY ,diZ ,dirR) from the MATLAB struct \*jP to the local joystick parameters variables.

##### **Parameters:**

*\*jP* : Structure coming from MATLAB or filled with default values defining joystick axes usage and direction.

Definition at line 788 of file graphics\_main.cpp.

References dirR, joy\_parameters::dirR, dirX, joy\_parameters::dirX, dirY, joy\_parameters::dirY, dirZ, joy\_parameters::dirZ, doR, joy\_parameters::doR, doX, joy\_parameters::doX, doY, joy\_parameters::doY, doZ, and joy\_parameters::doZ.

Referenced by mexFunction().

#### 7.25.2.12 GLvoid ReSizeGLScene (GLsizei *width*, GLsizei *height*)

Definition at line 138 of file graphics\_main.cpp.

References f().

Referenced by CreateGLWindow(), and WndProc().

#### 7.25.2.13 LRESULT CALLBACK WndProc (HWND *hWnd*, UINT *uMsg*, WPARAM *wParam*, LPARAM *lParam*)

Callback and keyboard management.

Definition at line 510 of file graphics\_main.cpp.

References active, keys, and ReSizeGLScene().

Referenced by CreateGLWindow().

### 7.25.3 Variable Documentation

#### 7.25.3.1 bool **active** = TRUE

Definition at line 23 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and WndProc().

#### 7.25.3.2 bool **allowChangeDirR** = TRUE

Definition at line 62 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.3 bool **allowChangeDirX** = TRUE

Definition at line 59 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.4 bool **allowChangeDirY** = TRUE

Definition at line 60 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.5 bool **allowChangeDirZ** = TRUE

Definition at line 61 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.6 bool **allowChangeDoR** = TRUE

Definition at line 54 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.7 bool allowChangeDoX = TRUE

Definition at line 51 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.8 bool allowChangeDoY = TRUE

Definition at line 52 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.9 bool allowChangeDoZ = TRUE

Definition at line 53 of file graphics\_main.cpp.

Referenced by draw\_graphics().

#### 7.25.3.10 GLuint base

Definition at line 19 of file graphics\_main.cpp.

Referenced by BuildFont(), glPrint(), and KillFont().

#### 7.25.3.11 float dirR = 1.0f

Definition at line 58 of file graphics\_main.cpp.

Referenced by draw\_graphics(), DrawGLScene(), and pass\_joy\_parameters().

#### 7.25.3.12 float dirX = 1.0f

Definition at line 55 of file graphics\_main.cpp.

Referenced by draw\_graphics(), DrawGLScene(), and pass\_joy\_parameters().

#### 7.25.3.13 float dirY = 1.0f

Definition at line 56 of file graphics\_main.cpp.

Referenced by draw\_graphics(), DrawGLScene(), and pass\_joy\_parameters().

#### 7.25.3.14 float dirZ = 1.0f

Definition at line 57 of file graphics\_main.cpp.

Referenced by draw\_graphics(), DrawGLScene(), and pass\_joy\_parameters().

**7.25.3.15    `bool done = FALSE`**

Definition at line 43 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`, and `draw_graphics_kill()`.

**7.25.3.16    `bool doR = TRUE`**

Definition at line 50 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`, `DrawGLScene()`, and `pass_joy_parameters()`.

**7.25.3.17    `bool doX = TRUE`**

Definition at line 47 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`, `DrawGLScene()`, and `pass_joy_parameters()`.

**7.25.3.18    `bool doY = TRUE`**

Definition at line 48 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`, `DrawGLScene()`, and `pass_joy_parameters()`.

**7.25.3.19    `bool doZ = TRUE`**

Definition at line 49 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`, `DrawGLScene()`, and `pass_joy_parameters()`.

**7.25.3.20    `GLfloat dviewAngle = 0.01`**

Definition at line 32 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`.

**7.25.3.21    `GLfloat dviewDist = 0.3`**

Definition at line 34 of file `graphics_main.cpp`.

Referenced by `draw_graphics()`.

**7.25.3.22    `float fps = 0.0`**

Definition at line 65 of file `graphics_main.cpp`.

Referenced by `DrawGLScene()`.

**7.25.3.23    `bool fullscreen = TRUE`**

Definition at line 24 of file `graphics_main.cpp`.

Referenced by `CreateGLWindow()`, `draw_graphics()`, `draw_graphics_init()`, and `KillGLWindow()`.

**7.25.3.24** **HDC** **hDC** = NULL

Definition at line 15 of file graphics\_main.cpp.

Referenced by BuildFont(), CreateGLWindow(), draw\_graphics(), and KillGLWindow().

**7.25.3.25** **HINSTANCE** **hInstance**

Definition at line 17 of file graphics\_main.cpp.

Referenced by CreateGLWindow(), and KillGLWindow().

**7.25.3.26** **HGLRC** **hRC** = NULL

Definition at line 14 of file graphics\_main.cpp.

Referenced by CreateGLWindow(), and KillGLWindow().

**7.25.3.27** **HWND** **hWnd** = NULL

Definition at line 16 of file graphics\_main.cpp.

Referenced by CreateGLWindow(), and KillGLWindow().

**7.25.3.28** **int** **joystick\_exist**

Definition at line 46 of file graphics\_main.cpp.

Referenced by draw\_graphics\_init(), and DrawGLScene().

**7.25.3.29** **bool** **keys**[256]

Definition at line 22 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and WndProc().

**7.25.3.30** **bool** **light**

Definition at line 26 of file graphics\_main.cpp.

**7.25.3.31** **GLfloat** **LightAmbient**[ ] = { 0.35f, 0.35f, 0.5f, 1.0f }

Definition at line 37 of file graphics\_main.cpp.

Referenced by InitGL().

**7.25.3.32** **GLfloat** **LightDiffuse**[ ] = { 0.6f, 0.6f, 0.6f, 1.0f }

Definition at line 38 of file graphics\_main.cpp.

Referenced by InitGL().

**7.25.3.33 GLfloat [LightPosition](#) [ ] = { -10.0f, 20.0f, 150.0f, 0.0f }**

Definition at line 39 of file graphics\_main.cpp.

Referenced by DrawGLScene(), and InitGL().

**7.25.3.34 MSG [msg](#)**

Definition at line 42 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and draw\_graphics\_kill().

**7.25.3.35 float [time\\_before](#) = 0.0**

Definition at line 63 of file graphics\_main.cpp.

Referenced by draw\_graphics().

**7.25.3.36 float [time\\_elapsed](#)**

Definition at line 64 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and DrawGLScene().

**7.25.3.37 GLfloat [viewAngle1](#) = 0.0**

horizontal camera view angle

Definition at line 30 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and DrawGLScene().

**7.25.3.38 GLfloat [viewAngle2](#) = 0.3**

vertical camera view angle

Definition at line 31 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and DrawGLScene().

**7.25.3.39 GLfloat [viewDist](#) = 50.0**

camera view distance

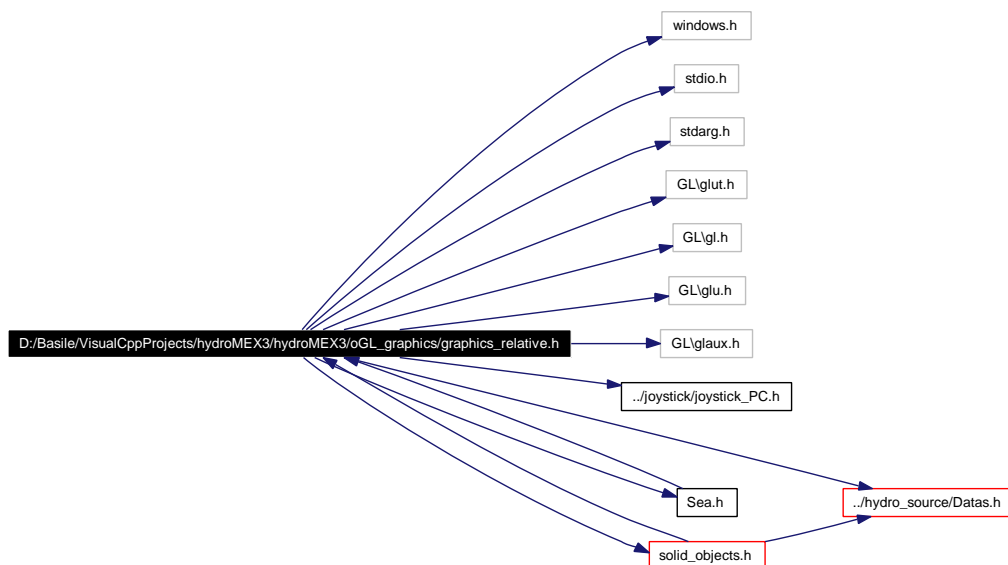
Definition at line 33 of file graphics\_main.cpp.

Referenced by draw\_graphics(), and DrawGLScene().

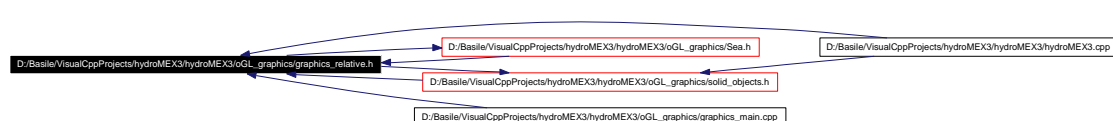
## 7.26 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL\_graphics/graphics\_relative.h File Reference

```
#include <windows.h>
#include <stdio.h>
#include <stdarg.h>
#include <GL\glut.h>
#include <GL\gl.h>
#include <GL\glu.h>
#include <GL\glaux.h>
#include "../joystick/joystick_PC.h"
#include "../hydro_source/Datas.h"
#include "Sea.h"
#include "solid_objects.h"
```

Include dependency graph for graphics\_relative.h:



This graph shows which files directly or indirectly include this file:



## Functions

- LRESULT CALLBACK [WndProc](#) (HWND, UINT, WPARAM, LPARAM)

*Callback and keyboard management.*

- GLvoid [ReSizeGLScene](#) (GLsizei width, GLsizei height)
- int [InitGL](#) (GLvoid)  
*Initialize OpenGL stuff.*
- int [DrawGLScene](#) (datas \*d)  
*Draws the scene (openGL).*
- GLvoid [KillGLWindow](#) (GLvoid)
- BOOL [CreateGLWindow](#) (char \*title, int width, int height, int bits, bool fullscreenflag)  
*Sub-function for window closing stuffs.*
- int [draw\\_graphics\\_init](#) (datas \*d)  
*General graphic stuff initialisation.*
- bool [draw\\_graphics](#) (datas \*d)  
*General graphic (scene) display and keyboard reading.*
- int [draw\\_graphics\\_kill](#) ()  
*General graphic (scene/window) closing.*
- void [pass\\_joy\\_parameters](#) (joy\_parameters \*jP)  
*Copy joystick parameters from MATLAB struct.*

### 7.26.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file contains includes for graphical purposes and declarations for some graphic related functions

Definition in file [graphics\\_relative.h](#).

### 7.26.2 Function Documentation

#### 7.26.2.1 BOOL CreateGLWindow (char \* title, int width, int height, int bits, bool fullscreenflag)

Sub-function for window closing stuffs.

Definition at line 331 of file [graphics\\_main.cpp](#).

References [fullscreen](#), [hDC](#), [hInstance](#), [hRC](#), [hWnd](#), [InitGL\(\)](#), [KillGLWindow\(\)](#), [ReSizeGLScene\(\)](#), and [WndProc\(\)](#).

Referenced by [draw\\_graphics\(\)](#), and [draw\\_graphics\\_init\(\)](#).

#### 7.26.2.2 bool draw\_graphics (datas \* d)

General graphic (scene) display and keyboard reading.

A call to this function refreshes the entire scene. This function is to be used in the main computing loop.



It also reads the keyboard for the different keyboard controls...

**Parameters:**

\**d* : pointer to the datas data-structure

**Returns:**

true if all went OK

Definition at line 599 of file graphics\_main.cpp.

References active, allowChangeDirR, allowChangeDirX, allowChangeDirY, allowChangeDirZ, allowChangeDoR, allowChangeDoX, allowChangeDoY, allowChangeDoZ, datas::angle\_girouette, CreateGLWindow(), dirR, dirX, dirY, dirZ, done, doR, doX, doY, doZ, DrawGLScene(), dviewAngle, dviewDist, fullscreen, hDC, datas::Hslider\_x, keys, KillGLWindow(), datas::max\_abs\_angle\_baume, msg, datas::t, datas::tanAngle\_sail, datas::target\_x, datas::target\_y, time\_before, time\_elapsed, viewAngle1, viewAngle2, viewDist, datas::Vslider\_x, datas::Wave\_amp, datas::Wave\_amp0, datas::Wave\_amp1, datas::Wave\_amp2, datas::Wind, datas::Wind\_angle, datas::Wind\_x, and datas::Wind\_y.

Referenced by mexFunction().

### 7.26.2.3 int draw\_graphics\_init (datas \* d)

General graphic stuff initialisation.

Ask for full-screen or not, creates window and try to initialize the joystick.

**Parameters:**

\**d* : pointer to the datas data-structure

**Returns:**

1 if window created successfully

Definition at line 568 of file graphics\_main.cpp.

References CreateGLWindow(), fullscreen, joystick\_exist, and joystick\_init().

Referenced by mexFunction().

### 7.26.2.4 int draw\_graphics\_kill ()

General graphic (scene/window) closing.

Closes window and joystick

**Returns:**

msg.wParam (MS Windows !)

Definition at line 774 of file graphics\_main.cpp.

References done, joystick\_close(), KillGLWindow(), and msg.

Referenced by mexFunction().

### 7.26.2.5 int DrawGLScene (datas \* d)

Draws the scene (openGL).

Draws the scene by applying all necessary geometrical (spacial) transformations and by calling the different element drawing functions (for the boat, the sea, the instruments...). This is not the highest function for drawing the current scene, see [draw\\_graphics\(\)](#).

Definition at line 189 of file `graphics_main.cpp`.

References `angles_update()`, `Boat_draw()`, `dirR`, `dirX`, `dirY`, `dirZ`, `doR`, `doX`, `doY`, `doZ`, `Draw_sea()`, `datas::dx`, `datas::dy`, `f()`, `fps`, `Girouette_draw()`, `glPrint()`, `HSlider_draw()`, `datas::Hslider_x`, `joystick_exist`, `joystick_getXYZR()`, `LightPosition`, `datas::phi`, `datas::psi`, `RelativeWind_draw()`, `Sail_create_and_draw()`, `Target_draw()`, `datas::target_x`, `datas::target_y`, `datas::theta`, `time_elapsed`, `viewAngle1`, `viewAngle2`, `view-Dist`, `VSlider_draw()`, `datas::Vslider_x`, `wave_z_compute()`, `datas::Wind`, `datas::Wind_x`, `datas::Wind_y`, `datas::x`, `datas::y`, and `datas::z`.

Referenced by `draw_graphics()`.

#### 7.26.2.6 int InitGL (GLvoid)

Initialize OpenGL stuff.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Definition at line 159 of file `graphics_main.cpp`.

References `BuildFont()`, `f()`, `LightAmbient`, `LightDiffuse`, and `LightPosition`.

Referenced by `CreateGLWindow()`.

#### 7.26.2.7 GLvoid KillGLWindow (GLvoid)

Definition at line 287 of file `graphics_main.cpp`.

References `fullscreen`, `hDC`, `hInstance`, `hRC`, and `hWnd`.

Referenced by `CreateGLWindow()`, `draw_graphics()`, and `draw_graphics_kill()`.

#### 7.26.2.8 void pass\_joy\_parameters (joy\_parameters \*jP)

Copy joystick parameters from MATLAB struct.

Copy the joystick parameters (`doX`, `doY`, `doZ`, `doR` and `dirX`, `dirY`, `dirZ`, `dirR`) from the MATLAB struct `*jP` to the local joystick parameters variables.

##### Parameters:

`*jP` : Structure coming from MATLAB or filled with default values defining joystick axes usage and direction.

Definition at line 788 of file `graphics_main.cpp`.

References `joy_parameters::dirR`, `dirR`, `joy_parameters::dirX`, `dirX`, `joy_parameters::dirY`, `dirY`, `joy_parameters::dirZ`, `dirZ`, `joy_parameters::doR`, `doR`, `joy_parameters::doX`, `doX`, `joy_parameters::doY`, `doY`, `joy_parameters::doZ`, and `doZ`.

Referenced by `mexFunction()`.

#### 7.26.2.9 GLvoid ReSizeGLScene (GLsizei width, GLsizei height)

Definition at line 138 of file `graphics_main.cpp`.

References f().

Referenced by CreateGLWindow(), and WndProc().

#### **7.26.2.10 LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM)**

Callback and keyboard management.

Definition at line 510 of file graphics\_main.cpp.

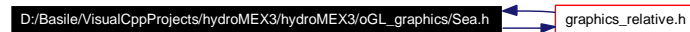
References active, keys, and ReSizeGLScene().

Referenced by CreateGLWindow().

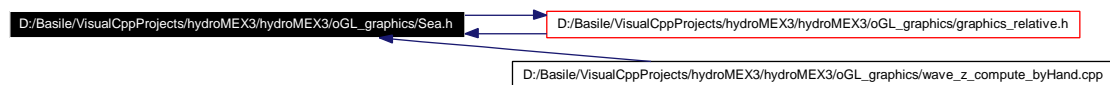
## 7.27 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL\_graphics/Sea.h File Reference

```
#include "graphics_relative.h"
```

Include dependency graph for Sea.h:



This graph shows which files directly or indirectly include this file:



### Functions

- void [wave\\_init](#) ()  
*Initialize and allocate memory for sea data.*
- void [wave\\_z\\_compute](#) (datas \*d)  
*Compute sea-mesh.*
- void [free\\_wave\\_variables](#) ()  
*Free sea data memory.*
- void [Draw\\_sea](#) ()  
*Draws the sea.*
- void [fillWaveLuts](#) ()  
*fill sine and cosine look up tables*

### 7.27.1 Detailed Description

**Author:**

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

Contain declaration for functions defined in [wave\\_z\\_compute\\_byHand.cpp](#)

!!!! Wave-evaluation function in [wave\\_z\\_compute\\_byHand.cpp](#) is not automaticaly updated by MATLAB code generation !!!!!

Definition in file [Sea.h](#).

### 7.27.2 Function Documentation

#### 7.27.2.1 void Draw\_sea ()

Draws the sea.

Once initialization with [wave\\_init\(\)](#) is done, and after each call of [wave\\_z\\_compute\(\)](#), this function can be called to actually draw the sea. The code of this function contains OpenGL code.

Definition at line 159 of file `wave_z_compute_byHand.cpp`.

References `f()`, `seaDatas::L`, `seaDatas::N`, `sd`, and `seaDatas::z`.

Referenced by `DrawGLScene()`.

#### **7.27.2.2 void fillWaveLuts ()**

fill sine and cosine look up tables

Definition at line 40 of file `wave_z_compute_byHand.cpp`.

References `cosLUT`, `f()`, `LUTLEN`, `LUTLENF`, `pi`, and `sinLUT`.

Referenced by `wave_init()`.

#### **7.27.2.3 void free\_wave\_variables ()**

Free sea data memory.

Free memory allocated by [wave\\_init\(\)](#).

Definition at line 144 of file `wave_z_compute_byHand.cpp`.

References `seaDatas::B`, `seaDatas::G`, `seaDatas::R`, `sd`, and `seaDatas::z`.

Referenced by `mexFunction()`.

#### **7.27.2.4 void wave\_init ()**

Initialize and allocate memory for sea data.

Don't forget to call [free\\_wave\\_variables\(\)](#)

Definition at line 82 of file `wave_z_compute_byHand.cpp`.

References `seaDatas::B`, `fillWaveLuts()`, `seaDatas::G`, `seaDatas::L`, `seaDatas::N`, `seaDatas::R`, `sd`, and `seaDatas::z`.

Referenced by `mexFunction()`.

#### **7.27.2.5 void wave\_z\_compute (datas \* d)**

Compute sea-mesh.

Fill the sea-mesh containing wave height on a  $N \times N$  mesh representing a sea square of  $L \times L$  meters using LUT functions (faster)

##### **Parameters:**

*\*d* : pointer to the `datas` data-structure

Definition at line 109 of file `wave_z_compute_byHand.cpp`.

References `f()`, `seaDatas::L`, `datas::Lambda0`, `datas::Lambda1`, `datas::Lambda2`, `seaDatas::N`, `pi`, `sd`, `datas::t`, `datas::V_wave0`, `datas::V_wave1`, `datas::V_wave2`, `datas::Wave_amp0`, `datas::Wave_amp1`, `datas::Wave_amp2`, `datas::Wave_angle0`, `datas::Wave_angle1`, `datas::Wave_angle2`, `Wcos()`, `Wsin()`, `datas::x`, `datas::y`, and `seaDatas::z`.

Referenced by DrawGLScene().

## 7.28 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL\_graphics/solid\_objects.cpp File Reference

```
#include "solid_objects.h"
```

Include dependency graph for solid\_objects.cpp:



### Functions

- void [SolidObject\\_create](#) ([SolidObject](#) \*so, int N, int M)  
*Creates a solid object.*
- void [SolidObject\\_delete](#) ([SolidObject](#) \*so)  
*Deletes a solid object.*
- void [cross\\_product](#) (float \*va, float \*vb, float \*vc)  
*Computes the cross-product.*
- void [one\\_normal](#) (float \*v1, float \*v2, float \*v3, float \*v4, float \*vn)  
*Computes normal vector for one vertex.*
- void [SolidObject\\_compute\\_normals](#) ([SolidObject](#) \*so)  
*Computes the normals for a [SolidObject](#).*
- void [SolidObject\\_draw](#) ([SolidObject](#) \*so)
- void [Sail\\_create\\_and\\_draw](#) ([datas](#) \*d)  
*Create and draw the sail.*
- void [RelativeWind\\_draw](#) ([datas](#) \*d)  
*Draws relative wind vector.*
- void [Target\\_draw](#) (float x, float y)  
*Draws a "target instrument" on the screen.*
- void [VSlider\\_draw](#) (float x)  
*Draws a "vertical slider instrument" on the screen.*
- void [HSlider\\_draw](#) (float x)  
*Draws a "horizontal slider instrument" on the screen.*
- void [Girouette\\_draw](#) ([datas](#) \*d)  
*Draws a girouette on the screen.*
- void [Boat\\_draw](#) ()  
*Draws the boat.*

- void [Boat\\_delete](#) ()  
*Deletes the boat.*
- void [Boat\\_create](#) (datas \*d)  
*Creates and defines all boat related [SolidObject](#) objects.*

## Variables

- [SolidObject hullStruct](#)  
*central hydroptere hull*
- [SolidObject \\* hull = &hullStruct](#)
- [SolidObject rSwimmStruct](#)  
*right swimmer*
- [SolidObject \\* rSwimm = &rSwimmStruct](#)
- [SolidObject lSwimmStruct](#)  
*left swimmer*
- [SolidObject \\* lSwimm = &lSwimmStruct](#)
- [SolidObject bridgeMainStruct](#)  
*front bridge*
- [SolidObject \\* bridgeMain = &bridgeMainStruct](#)
- [SolidObject bridge2Struct](#)  
*bridge 2*
- [SolidObject \\* bridge2 = &bridge2Struct](#)
- [SolidObject mastStruct](#)  
*mast*
- [SolidObject \\* mast = &mastStruct](#)
- [SolidObject foilRightStruct](#)  
*right foil*
- [SolidObject \\* foirl = &foilRightStruct](#)
- [SolidObject foilLeftStruct](#)  
*left foil*
- [SolidObject \\* foill = &foilLeftStruct](#)
- [SolidObject yawStruct](#)  
*yaw*
- [SolidObject \\* yaw = &yawStruct](#)
- [SolidObject pitchStruct](#)  
*pitch*
- [SolidObject \\* pitch = &pitchStruct](#)



## 7.28.1 Detailed Description

### Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file relates to all graphical objects. [SolidObject](#) is the 3D description of an object like a boat hull. This file contains also functions to draw navigation instruments on the screen.

Definition in file [solid\\_objects.cpp](#).

## 7.28.2 Function Documentation

### 7.28.2.1 void Boat\_create ([datas](#) \* *d*)

Creates and defines all boat related [SolidObject](#) objects.

Create and fill all boat informations (coordinates,...)

All hull coordinates are computed/defined herein. Have a look to the code!

#### Parameters:

*\*d* : pointer to the datas data-structure

Definition at line 538 of file [solid\\_objects.cpp](#).

References [SolidObject::B](#), [bridge2](#), [bridgeMain](#), [foill](#), [foilr](#), [SolidObject::G](#), [hull](#), [datas::Long](#), [lSwimm](#), [mast](#), [pitch](#), [SolidObject::R](#), [rSwimm](#), [SolidObject\\_create\(\)](#), [SolidObject::X](#), [datas::x\\_yaw](#), [SolidObject::Y](#), [datas::y\\_yaw](#), [yaw](#), [SolidObject::Z](#), and [datas::z\\_foils](#).

Referenced by [mexFunction\(\)](#).

### 7.28.2.2 void Boat\_delete ()

Deletes the boat.

Deletes all [SolidObject](#) objects used in drawing the boat using [SolidObject\\_delete\(\)](#)

Definition at line 514 of file [solid\\_objects.cpp](#).

References [bridge2](#), [bridgeMain](#), [foill](#), [foilr](#), [hull](#), [lSwimm](#), [mast](#), [pitch](#), [rSwimm](#), [SolidObject\\_delete\(\)](#), and [yaw](#).

Referenced by [mexFunction\(\)](#).

### 7.28.2.3 void Boat\_draw ()

Draws the boat.

Draws the boat once it has been created and defined by [Boat\\_create\(\)](#). This function uses calls to [SolidObject\\_draw\(\)](#). The code of this function contains OpenGL code.

Definition at line 494 of file [solid\\_objects.cpp](#).

References [bridge2](#), [bridgeMain](#), [foill](#), [foilr](#), [hull](#), [lSwimm](#), [mast](#), [pitch](#), [rSwimm](#), [SolidObject\\_draw\(\)](#), and [yaw](#).

Referenced by [DrawGLScene\(\)](#).

#### 7.28.2.4 void cross\_product (float \* va, float \* vb, float \* vc)

Computes the cross-product.

$$v_c = v_a \times v_b$$

**Parameters:**

- \*va : pointer to float array of 3 elements ( $v_a$ )
- \*vb : pointer to float array of 3 elements ( $v_b$ )
- \*vc : pointer to float array of 3 elements ( $v_c$ )

Definition at line 118 of file solid\_objects.cpp.

Referenced by Girouette\_draw(), one\_normal(), and Sail\_create\_and\_draw().

#### 7.28.2.5 void Girouette\_draw (datas \* d)

Draws a girouette on the screen.

Draws a girouette instrument indicating relative wind direction as an on screen instrument. The code of this function contains OpenGL code.

**Parameters:**

- \*d : pointer to the datas data-structure

Definition at line 402 of file solid\_objects.cpp.

References datas::angle\_girouette, cross\_product(), datas::dx, datas::dy, f(), datas::tanAngle\_sail, datas::Wind\_x, and datas::Wind\_y.

Referenced by DrawGLScene().

#### 7.28.2.6 void HSlider\_draw (float x)

Draws a "horizontal slider instrument" on the screen.

Draws a horizontal slider. The code of this function contains OpenGL code.

**Parameters:**

- x,: slider position,  $-1.0 \leq x \leq 1.0$ , clipped outside this range

Definition at line 372 of file solid\_objects.cpp.

References f().

Referenced by DrawGLScene().

#### 7.28.2.7 void one\_normal (float \* v1, float \* v2, float \* v3, float \* v4, float \* vn)

Computes normal vector for one vertex.

$$v_n = \frac{c}{|c|}$$

$$c = v_1 \times v_2 + v_2 \times v_3 + v_3 \times v_4 + v_4 \times v_1$$

**Parameters:**

- \*v1 : pointer to float array of 3 elements ( $v_1$ )

\***v2** : pointer to float array of 3 elements ( $v_2$ )  
 \***v3** : pointer to float array of 3 elements ( $v_3$ )  
 \***v4** : pointer to float array of 3 elements ( $v_4$ )  
 \***vn** : pointer to float array of 3 elements ( $v_n$ )

Definition at line 125 of file solid\_objects.cpp.

References cross\_product(), and f().

Referenced by SolidObject\_compute\_normals().

#### 7.28.2.8 void RelativeWind\_draw (datas \* d)

Draws relative wind vector.

Draws a relative wind vector at the top of the mast. The code of this function contains OpenGL code.

##### Parameters:

\***d** : pointer to the datas data-structure

Definition at line 292 of file solid\_objects.cpp.

References datas::dx, datas::dy, f(), datas::L\_mast, datas::phi, datas::psi, datas::theta, datas::Wind\_x, datas::Wind\_y, datas::x\_sail, datas::y\_sail, and datas::z\_foils.

Referenced by DrawGLScene().

#### 7.28.2.9 void Sail\_create\_and\_draw (datas \* d)

Create and draw the sail.

The sail is not a [SolidObject](#). It is redefined each time it is drawn (its position relative to the boat can change). The code of this function contains OpenGL code.

##### Parameters:

\***d** : pointer to the datas data-structure

Definition at line 261 of file solid\_objects.cpp.

References datas::Baume, cross\_product(), f(), datas::L\_mast, datas::tanAngle\_sail, datas::x\_sail, datas::y\_sail, and datas::z\_foils.

Referenced by DrawGLScene().

#### 7.28.2.10 void SolidObject\_compute\_normals (SolidObject \* so)

Computes the normals for a [SolidObject](#).

Once all X-, Y- Z- coordinates of a SolidObjects have been defined, the vertex-normal vectors in the [SolidObject](#) can be automatically filled by calling this function.

##### Parameters:

\***so,** pointer to the [SolidObject](#)

Definition at line 155 of file solid\_objects.cpp.

References SolidObject::M, SolidObject::N, SolidObject::nX, SolidObject::nY, SolidObject::nZ, one\_normal(), SolidObject::X, SolidObject::Y, and SolidObject::Z.

**7.28.2.11 void SolidObject\_create (SolidObject \* so, int N, int M)**

Creates a solid object.

Initializes memory for a 3D object

Do not forget to call [SolidObject\\_delete\(\)](#) for each object created once finished!!

**Parameters:**

*\*so* pointer to a [SolidObject](#) to be initialized

*N* length of mesh 2D arrays (coordinates, normals, color)

*M* height of mesh 2D arrays (coordinates, normals, color)

Definition at line 46 of file solid\_objects.cpp.

References [SolidObject::B](#), [SolidObject::G](#), [SolidObject::M](#), [SolidObject::N](#), [SolidObject::nX](#), [SolidObject::nY](#), [SolidObject::nZ](#), [SolidObject::R](#), [SolidObject::X](#), [SolidObject::Y](#), and [SolidObject::Z](#).

Referenced by [Boat\\_create\(\)](#).

**7.28.2.12 void SolidObject\_delete (SolidObject \* so)**

Deletes a solid object.

Free memory used by a 3D object

To be used after every creation with [SolidObject\\_create\(\)](#)

Definition at line 82 of file solid\_objects.cpp.

References [SolidObject::B](#), [SolidObject::G](#), [SolidObject::N](#), [SolidObject::nX](#), [SolidObject::nY](#), [SolidObject::nZ](#), [SolidObject::R](#), [SolidObject::X](#), [SolidObject::Y](#), and [SolidObject::Z](#).

Referenced by [Boat\\_delete\(\)](#).

**7.28.2.13 void SolidObject\_draw (SolidObject \* so)**

Definition at line 228 of file solid\_objects.cpp.

References [SolidObject::B](#), [SolidObject::G](#), [SolidObject::M](#), [SolidObject::N](#), [SolidObject::nX](#), [SolidObject::nY](#), [SolidObject::nZ](#), [SolidObject::R](#), [SolidObject::X](#), [SolidObject::Y](#), and [SolidObject::Z](#).

Referenced by [Boat\\_draw\(\)](#).

**7.28.2.14 void Target\_draw (float x, float y)**

Draws a "target instrument" on the screen.

Draws a circular target with a cross inside. The code of this function contains OpenGL code.

**Parameters:**

*x*,: cross position,  $-1.0 \leq x \leq 1.0$ , clipped outside this range

*y*,: cross position,  $-1.0 \leq y \leq 1.0$ , clipped outside this range

Definition at line 309 of file solid\_objects.cpp.

References [f\(\)](#).

Referenced by [DrawGLScene\(\)](#).

### 7.28.2.15 void VSlider\_draw (float x)

Draws a "vertical slider instrument" on the screen.

Draws a vertical slider. The code of this function contains OpenGL code.

#### Parameters:

$x$ : slider position,  $-1.0 \leq x \leq 1.0$ , clipped outside this range

Definition at line 343 of file solid\_objects.cpp.

References f().

Referenced by DrawGLScene().

## 7.28.3 Variable Documentation

### 7.28.3.1 SolidObject\* bridge2 = &bridge2Struct

Definition at line 27 of file solid\_objects.cpp.

Referenced by Boat\_create(), Boat\_delete(), and Boat\_draw().

### 7.28.3.2 SolidObject bridge2Struct

bridge 2

Definition at line 26 of file solid\_objects.cpp.

### 7.28.3.3 SolidObject\* bridgeMain = &bridgeMainStruct

Definition at line 24 of file solid\_objects.cpp.

Referenced by Boat\_create(), Boat\_delete(), and Boat\_draw().

### 7.28.3.4 SolidObject bridgeMainStruct

front bridge

Definition at line 23 of file solid\_objects.cpp.

### 7.28.3.5 SolidObject\* foill = &foillLeftStruct

Definition at line 36 of file solid\_objects.cpp.

Referenced by Boat\_create(), Boat\_delete(), and Boat\_draw().

### 7.28.3.6 SolidObject foillLeftStruct

left foil

Definition at line 35 of file solid\_objects.cpp.

**7.28.3.7   `SolidObject* foilr = &foilRightStruct`**

Definition at line 33 of file `solid_objects.cpp`.

Referenced by `Boat_create()`, `Boat_delete()`, and `Boat_draw()`.

**7.28.3.8   `SolidObject foilRightStruct`**

right foil

Definition at line 32 of file `solid_objects.cpp`.

**7.28.3.9   `SolidObject* hull = &hullStruct`**

Definition at line 15 of file `solid_objects.cpp`.

Referenced by `Boat_create()`, `Boat_delete()`, and `Boat_draw()`.

**7.28.3.10   `SolidObject hullStruct`**

central hydroptere hull

Definition at line 14 of file `solid_objects.cpp`.

**7.28.3.11   `SolidObject* lSwimm = &lSwimmStruct`**

Definition at line 21 of file `solid_objects.cpp`.

Referenced by `Boat_create()`, `Boat_delete()`, and `Boat_draw()`.

**7.28.3.12   `SolidObject lSwimmStruct`**

left swimmer

Definition at line 20 of file `solid_objects.cpp`.

**7.28.3.13   `SolidObject* mast = &mastStruct`**

Definition at line 30 of file `solid_objects.cpp`.

Referenced by `Boat_create()`, `Boat_delete()`, and `Boat_draw()`.

**7.28.3.14   `SolidObject mastStruct`**

mast

Definition at line 29 of file `solid_objects.cpp`.

**7.28.3.15   `SolidObject* pitch = &pitchStruct`**

Definition at line 42 of file `solid_objects.cpp`.

Referenced by `Boat_create()`, `Boat_delete()`, and `Boat_draw()`.

#### 7.28.3.16 **SolidObject** pitchStruct

pitch

Definition at line 41 of file solid\_objects.cpp.

#### 7.28.3.17 **SolidObject\*** rSwimm = &rSwimmStruct

Definition at line 18 of file solid\_objects.cpp.

Referenced by Boat\_create(), Boat\_delete(), and Boat\_draw().

#### 7.28.3.18 **SolidObject** rSwimmStruct

right swimmer

Definition at line 17 of file solid\_objects.cpp.

#### 7.28.3.19 **SolidObject\*** yaw = &yawStruct

Definition at line 39 of file solid\_objects.cpp.

Referenced by Boat\_create(), Boat\_delete(), and Boat\_draw().

#### 7.28.3.20 **SolidObject** yawStruct

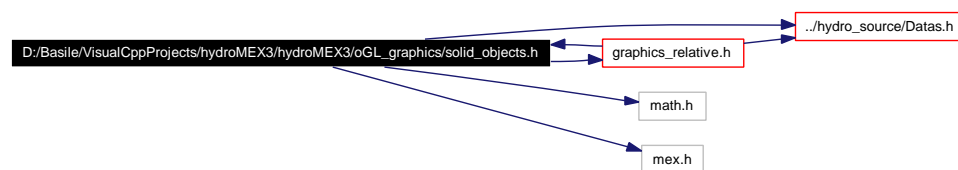
yaw

Definition at line 38 of file solid\_objects.cpp.

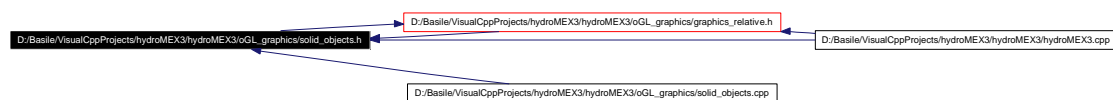
## 7.29 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL\_graphics/solid\_objects.h File Reference

```
#include "../hydro_source/Datas.h"
#include "graphics_relative.h"
#include "math.h"
#include "mex.h"
```

Include dependency graph for solid\_objects.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [SolidObject](#)  
*3D object description*

## Functions

- void [SolidObject\\_create](#) ([SolidObject](#) \*so, int N, int M)  
*Creates a solid object.*
- void [SolidObject\\_delete](#) ([SolidObject](#) \*so)  
*Deletes a solid object.*
- void [cross\\_product](#) (float \*va, float \*vb, float \*vc)  
*Computes the cross-product.*
- void [one\\_normal](#) (float \*v1, float \*v2, float \*v3, float \*v4, float \*vn)  
*Computes normal vector for one vertex.*
- void [Boat\\_delete](#) ()  
*Deletes the boat.*



- void [SolidObject\\_compute\\_normals](#) ([SolidObject](#) \*so)  
*Computes the normals for a [SolidObject](#).*
- void [Boat\\_create](#) ([datas](#) \*d)  
*Creates and defines all boat related [SolidObject](#) objects.*
- void [Sail\\_create\\_and\\_draw](#) ([datas](#) \*d)  
*Create and draw the sail.*
- void [RelativeWind\\_draw](#) ([datas](#) \*d)  
*Draws relative wind vector.*
- void [Target\\_draw](#) (float x, float y)  
*Draws a "target instrument" on the screen.*
- void [VSlider\\_draw](#) (float x)  
*Draws a "vertical slider instrument" on the screen.*
- void [HSlider\\_draw](#) (float x)  
*Draws a "horizontal slider instrument" on the screen.*
- void [Girouette\\_draw](#) ([datas](#) \*d)  
*Draws a girouette on the screen.*
- void [Boat\\_draw](#) ()  
*Draws the boat.*

## 7.29.1 Detailed Description

### Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

This file relates to all graphical objects. [SolidObject](#) is the 3D description of an object like a boat hull. This file contains also functions to draw navigation instruments on the screen.

Definition in file [solid\\_objects.h](#).

## 7.29.2 Function Documentation

### 7.29.2.1 void [Boat\\_create](#) ([datas](#) \* d)

Creates and defines all boat related [SolidObject](#) objects.

Create and fill all boat informations (coordinates,...)

All hull coordinates are computed/defined herein. Have a look to the code!

### Parameters:

\*d : pointer to the datas data-structure

Definition at line 538 of file solid\_objects.cpp.

References SolidObject::B, bridge2, bridgeMain, foill, foilr, SolidObject::G, hull, datas::Long, lSwimm, mast, pitch, SolidObject::R, rSwimm, SolidObject\_create(), SolidObject::X, datas::x\_yaw, SolidObject::Y, datas::y\_yaw, yaw, SolidObject::Z, and datas::z\_foils.

Referenced by mexFunction().

#### 7.29.2.2 void Boat\_delete ()

Deletes the boat.

Deletes all [SolidObject](#) objects used in drawing the boat using [SolidObject\\_delete\(\)](#)

Definition at line 514 of file solid\_objects.cpp.

References bridge2, bridgeMain, foill, foilr, hull, lSwimm, mast, pitch, rSwimm, SolidObject\_delete(), and yaw.

Referenced by mexFunction().

#### 7.29.2.3 void Boat\_draw ()

Draws the boat.

Draws the boat once it has been created and defined by [Boat\\_create\(\)](#). This function uses calls to [SolidObject\\_draw\(\)](#). The code of this function contains OpenGL code.

Definition at line 494 of file solid\_objects.cpp.

References bridge2, bridgeMain, foill, foilr, hull, lSwimm, mast, pitch, rSwimm, SolidObject\_draw(), and yaw.

Referenced by DrawGLScene().

#### 7.29.2.4 void cross\_product (float \* va, float \* vb, float \* vc)

Computes the cross-product.

$$v_c = v_a \times v_b$$

##### Parameters:

\***va** : pointer to float array of 3 elements ( $v_a$ )

\***vb** : pointer to float array of 3 elements ( $v_b$ )

\***vc** : pointer to float array of 3 elements ( $v_c$ )

Definition at line 118 of file solid\_objects.cpp.

Referenced by Girouette\_draw(), one\_normal(), and Sail\_create\_and\_draw().

#### 7.29.2.5 void Girouette\_draw (datas \* d)

Draws a girouette on the screen.

Draws a girouette instrument indicating relative wind direction as an on screen instrument. The code of this function contains OpenGL code.

**Parameters:**

*\*d* : pointer to the datas data-structure

Definition at line 402 of file solid\_objects.cpp.

References datas::angle\_girouette, cross\_product(), datas::dx, datas::dy, f(), datas::tanAngle\_sail, datas::Wind\_x, and datas::Wind\_y.

Referenced by DrawGLScene().

### 7.29.2.6 void HSlider\_draw (float x)

Draws a "horizontal slider instrument" on the screen.

Draws a horizontal slider. The code of this function contains OpenGL code.

**Parameters:**

*x*,: slider position,  $-1.0 \leq x \leq 1.0$ , clipped outside this range

Definition at line 372 of file solid\_objects.cpp.

References f().

Referenced by DrawGLScene().

### 7.29.2.7 void one\_normal (float \* v1, float \* v2, float \* v3, float \* v4, float \* vn)

Computes normal vector for one vertex.

$$v_n = \frac{c}{|c|}$$

$$c = v_1 \times v_2 + v_2 \times v_3 + v_3 \times v_4 + v_4 \times v_1$$

**Parameters:**

*\*v1* : pointer to float array of 3 elements ( $v_1$ )

*\*v2* : pointer to float array of 3 elements ( $v_2$ )

*\*v3* : pointer to float array of 3 elements ( $v_3$ )

*\*v4* : pointer to float array of 3 elements ( $v_4$ )

*\*vn* : pointer to float array of 3 elements ( $v_n$ )

Definition at line 125 of file solid\_objects.cpp.

References cross\_product(), and f().

Referenced by SolidObject\_compute\_normals().

### 7.29.2.8 void RelativeWind\_draw (datas \* d)

Draws relative wind vector.

Draws a relative wind vector at the top of the mast. The code of this function contains OpenGL code.

**Parameters:**

*\*d* : pointer to the datas data-structure

Definition at line 292 of file solid\_objects.cpp.

References `datas::dx`, `datas::dy`, `f()`, `datas::L_mast`, `datas::phi`, `datas::psi`, `datas::theta`, `datas::Wind_x`, `datas::Wind_y`, `datas::x_sail`, `datas::y_sail`, and `datas::z_foils`.

Referenced by `DrawGLScene()`.

#### 7.29.2.9 void Sail\_create\_and\_draw ([datas](#) \* *d*)

Create and draw the sail.

The sail is not a [SolidObject](#). It is redefined each time it is drawn (its position relative to the boat can change). The code of this function contains OpenGL code.

##### Parameters:

*\*d* : pointer to the `datas` data-structure

Definition at line 261 of file solid\_objects.cpp.

References `datas::Baume`, `cross_product()`, `f()`, `datas::L_mast`, `datas::tanAngle_sail`, `datas::x_sail`, `datas::y_sail`, and `datas::z_foils`.

Referenced by `DrawGLScene()`.

#### 7.29.2.10 void SolidObject\_compute\_normals ([SolidObject](#) \* *so*)

Computes the normals for a [SolidObject](#).

Once all X-, Y- Z- coordinates of a `SolidObject`s have been defined, the vertex-normal vectors in the [SolidObject](#) can be automatically filled by calling this function.

##### Parameters:

*\*so,* pointer to the [SolidObject](#)

Definition at line 155 of file solid\_objects.cpp.

References `SolidObject::M`, `SolidObject::N`, `SolidObject::nX`, `SolidObject::nY`, `SolidObject::nZ`, `one_normal()`, `SolidObject::X`, `SolidObject::Y`, and `SolidObject::Z`.

#### 7.29.2.11 void SolidObject\_create ([SolidObject](#) \* *so*, int *N*, int *M*)

Creates a solid object.

Initializes memory for a 3D object

Do not forget to call [SolidObject\\_delete\(\)](#) for each object created once finished!!

##### Parameters:

*\*so* pointer to a [SolidObject](#) to be initialized

*N* length of mesh 2D arrays (coordinates, normals, color)

*M* height of mesh 2D arrays (coordinates, normals, color)

Definition at line 46 of file solid\_objects.cpp.

References `SolidObject::B`, `SolidObject::G`, `SolidObject::M`, `SolidObject::N`, `SolidObject::nX`, `SolidObject::nY`, `SolidObject::nZ`, `SolidObject::R`, `SolidObject::X`, `SolidObject::Y`, and `SolidObject::Z`.

Referenced by `Boat_create()`.

#### 7.29.2.12 void SolidObject\_delete (SolidObject \* so)

Deletes a solid object.

Free memory used by a 3D object

To be used after every creation with [SolidObject\\_create\(\)](#)

Definition at line 82 of file solid\_objects.cpp.

References SolidObject::B, SolidObject::G, SolidObject::N, SolidObject::nX, SolidObject::nY, SolidObject::nZ, SolidObject::R, SolidObject::X, SolidObject::Y, and SolidObject::Z.

Referenced by Boat\_delete().

#### 7.29.2.13 void Target\_draw (float x, float y)

Draws a "target instrument" on the screen.

Draws a circular target with a cross inside. The code of this function contains OpenGL code.

##### Parameters:

**x,** cross position,  $-1.0 \leq x \leq 1.0$ , clipped outside this range

**y,** cross position,  $-1.0 \leq y \leq 1.0$ , clipped outside this range

Definition at line 309 of file solid\_objects.cpp.

References f().

Referenced by DrawGLScene().

#### 7.29.2.14 void VSlider\_draw (float x)

Draws a "vertical slider instrument" on the screen.

Draws a vertical slider. The code of this function contains OpenGL code.

##### Parameters:

**x,** slider position,  $-1.0 \leq x \leq 1.0$ , clipped outside this range

Definition at line 343 of file solid\_objects.cpp.

References f().

Referenced by DrawGLScene().

## 7.30 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/oGL\_graphics/wave\_z\_compute\_byHand.cpp File Reference

```
#include <math.h>
```

```
#include "Sea.h"
```

Include dependency graph for wave\_z\_compute\_byHand.cpp:



### Data Structures

- struct [seaDatas](#)

### Defines

- #define [pi](#) 3.14159265358979
- #define [LUTLEN](#) 64
- #define [LUTLENF](#) 64.0f

### Functions

- void [fillWaveLuts](#) ()  
*fill sine and cosine look up tables*
- \_\_inline float [Wcos](#) (float xarg)  
*Cosine approximation using LUT.*
- \_\_inline float [Wsin](#) (float xarg)  
*Sine approximation using LUT.*
- void [wave\\_init](#) ()  
*Initialize and allocate memory for sea data.*
- void [wave\\_z\\_compute](#) (datas \*d)  
*Compute sea-mesh.*
- void [free\\_wave\\_variables](#) ()  
*Free sea data memory.*
- void [Draw\\_sea](#) ()  
*Draws the sea.*

- float [sinLUT](#) [LUTLEN]  
*Look Up Table for sine function (used for wave drawing).*
- float [cosLUT](#) [LUTLEN]  
*Look Up Table for cosine function (used for wave drawing).*
- [seaDatas](#) [seaDat](#)
- [seaDatas](#) \* [sd](#) = &[seaDat](#)

### 7.30.1 Detailed Description

Author:

[basile.graf@epfl.ch](mailto:basile.graf@epfl.ch)

!!!! Wave-evaluation function in this file is not automaticaly updated by MATLAB code generation !!!!!

Definition in file [wave\\_z\\_compute\\_byHand.cpp](#).

### 7.30.2 Define Documentation

#### 7.30.2.1 #define LUTLEN 64

Definition at line 26 of file [wave\\_z\\_compute\\_byHand.cpp](#).

Referenced by [fillWaveLuts\(\)](#).

#### 7.30.2.2 #define LUTLENF 64.0f

Definition at line 27 of file [wave\\_z\\_compute\\_byHand.cpp](#).

Referenced by [fillWaveLuts\(\)](#), and [Wcos\(\)](#).

#### 7.30.2.3 #define pi 3.14159265358979

Definition at line 25 of file [wave\\_z\\_compute\\_byHand.cpp](#).

### 7.30.3 Function Documentation

#### 7.30.3.1 void Draw\_sea ()

Draws the sea.

Once initialization with [wave\\_init\(\)](#) is done, and after each call of [wave\\_z\\_compute\(\)](#), this function can be called to actually draw the sea. The code of this function contains OpenGL code.

Definition at line 159 of file [wave\\_z\\_compute\\_byHand.cpp](#).

References [f\(\)](#), [seaDatas::L](#), [seaDatas::N](#), [sd](#), and [seaDatas::z](#).

Referenced by [DrawGLScene\(\)](#).

**7.30.3.2 void fillWaveLuts ()**

fill sine and cosine look up tables

Definition at line 40 of file wave\_z\_compute\_byHand.cpp.

References cosLUT, f(), LUTLEN, LUTLENF, pi, and sinLUT.

Referenced by wave\_init().

**7.30.3.3 void free\_wave\_variables ()**

Free sea data memory.

Free memory allocated by [wave\\_init\(\)](#).

Definition at line 144 of file wave\_z\_compute\_byHand.cpp.

References seaDats::B, seaDats::G, seaDats::R, sd, and seaDats::z.

Referenced by mexFunction().

**7.30.3.4 void wave\_init ()**

Initialize and allocate memory for sea data.

Don't forget to call [free\\_wave\\_variables\(\)](#)

Definition at line 82 of file wave\_z\_compute\_byHand.cpp.

References seaDats::B, fillWaveLuts(), seaDats::G, seaDats::L, seaDats::N, seaDats::R, sd, and seaDats::z.

Referenced by mexFunction().

**7.30.3.5 void wave\_z\_compute (datas \* d)**

Compute sea-mesh.

Fill the sea-mesh containing wave height on a  $N \times N$  mesh representing a sea square of  $L \times L$  meters using LUT functions (faster)

**Parameters:**

*\*d* : pointer to the datas data-structure

Definition at line 109 of file wave\_z\_compute\_byHand.cpp.

References f(), seaDats::L, datas::Lambda0, datas::Lambda1, datas::Lambda2, seaDats::N, pi, sd, datas::t, datas::V\_wave0, datas::V\_wave1, datas::V\_wave2, datas::Wave\_amp0, datas::Wave\_amp1, datas::Wave\_amp2, datas::Wave\_angle0, datas::Wave\_angle1, datas::Wave\_angle2, Wcos(), Wsin(), datas::x, datas::y, and seaDats::z.

Referenced by DrawGLScene().

**7.30.3.6 \_\_inline float Wcos (float xarg)**

Cosine approximation using LUT.



**Reference  
Parameters:**

---

*xarg* return result

Definition at line 56 of file wave\_z\_compute\_byHand.cpp.

References cosLUT, LUTLENF, and pi.

Referenced by wave\_z\_compute(), and Wsin().

### 7.30.3.7 `__inline float Wsin (float xarg)`

Sine approximation using LUT.

**Parameters:**

*xarg* return result

Definition at line 72 of file wave\_z\_compute\_byHand.cpp.

References f(), pi, and Wcos().

Referenced by wave\_z\_compute().

## 7.30.4 Variable Documentation

### 7.30.4.1 `float cosLUT[LUTLEN]`

Look Up Table for cosine function (used for wave drawing).

Definition at line 31 of file wave\_z\_compute\_byHand.cpp.

Referenced by fillWaveLuts(), and Wcos().

### 7.30.4.2 `seaDatas* sd = &seaDat`

Definition at line 34 of file wave\_z\_compute\_byHand.cpp.

Referenced by Draw\_sea(), free\_wave\_variables(), wave\_init(), and wave\_z\_compute().

### 7.30.4.3 `seaDatas seaDat`

Definition at line 33 of file wave\_z\_compute\_byHand.cpp.

### 7.30.4.4 `float sinLUT[LUTLEN]`

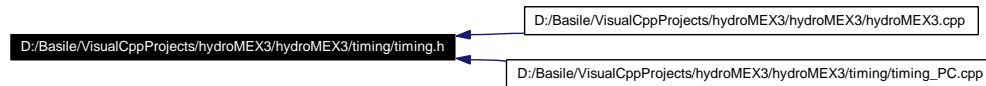
Look Up Table for sine function (used for wave drawing).

Definition at line 30 of file wave\_z\_compute\_byHand.cpp.

Referenced by fillWaveLuts().

## 7.31 D:/Basile/VisualCppProjects/hydroMEX3/hydro-MEX3/timing/timing.h File Reference

This graph shows which files directly or indirectly include this file:



### Functions

- void `tic` ()  
*Timinig function, similar to MATLAB's tic.*
- float `toc` ()  
*Timing function, similar to MATLAB's toc.*

### 7.31.1 Detailed Description

**Author:**

`basile.graf@epfl.ch`

Timing functions for real time simulation

Definition in file `timing.h`.

### 7.31.2 Function Documentation

#### 7.31.2.1 void `tic` ()

Timinig function, similar to MATLAB's tic.

Initializes the timer (for realtime simulation)

Definition at line 15 of file `timing_PC.cpp`.

References offset.

Referenced by `mexFunction()`.

#### 7.31.2.2 float `toc` ()

Timing function, similar to MATLAB's toc.

Returns the time elapsed in seconds since the last call of `tic()`

Resolution is 1 ms

**Returns:**

time in seconds

Definition at line 20 of file timing\_PC.cpp.

References `f()`, and `offset`.

Referenced by `mexFunction()`.

## 7.32 D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/timing/timing\_PC.cpp File Reference

```
#include <windows.h>
```

```
#include "timing.h"
```

Include dependency graph for timing\_PC.cpp:



### Functions

- void `tic` ()  
*Timinig function, similar to MATLAB's tic.*
- float `toc` ()  
*Timing function, similar to MATLAB's toc.*

### Variables

- int `offset`

#### 7.32.1 Detailed Description

**Author:**

`basile.graf@epfl.ch`

Timing functions for real time simulation. MS Windows implementation.

Definition in file `timing_PC.cpp`.

#### 7.32.2 Function Documentation

##### 7.32.2.1 void `tic` ()

Timinig function, similar to MATLAB's tic.

Initializes the timer (for realtime simulation)

Definition at line 15 of file `timing_PC.cpp`.

References `offset`.

Referenced by `mexFunction()`.

### **7.32.2.2 float toc ()**

Timing function, similar to MATLAB's toc.

Returns the time elapsed in seconds since the last call of [tic\(\)](#)

Resolution is 1 ms

#### **Returns:**

time in seconds

Definition at line 20 of file timing\_PC.cpp.

References [f\(\)](#), and [offset](#).

Referenced by [mexFunction\(\)](#).

## **7.32.3 Variable Documentation**

### **7.32.3.1 int [offset](#)**

Definition at line 13 of file timing\_PC.cpp.

Referenced by [tic\(\)](#), and [toc\(\)](#).

# Index

- A\_left
  - datas, [14](#)
- A\_pitch
  - datas, [14](#)
- A\_right
  - datas, [15](#)
- A\_sail
  - datas, [15](#)
- A\_structure
  - datas, [15](#)
- A\_yaw
  - datas, [15](#)
- abstol
  - ODE\_data, [37](#)
  - parameters, [40](#)
- acceleration\_compute
  - analytics\_dyn.cpp, [58](#)
  - analytics\_dyn.h, [60](#)
- active
  - graphics\_main.cpp, [108](#)
- ActualPos
  - joystick\_PC.cpp, [97](#)
- aero\_coeff\_control
  - aero\_coeff\_control.cpp, [56](#)
  - aero\_coeff\_control.h, [57](#)
- aero\_coeff\_control.cpp
  - aero\_coeff\_control, [56](#)
  - PI, [55](#)
  - pi, [55](#)
- aero\_coeff\_control.h
  - aero\_coeff\_control, [57](#)
- allowChangeDirR
  - graphics\_main.cpp, [108](#)
- allowChangeDirX
  - graphics\_main.cpp, [108](#)
- allowChangeDirY
  - graphics\_main.cpp, [108](#)
- allowChangeDirZ
  - graphics\_main.cpp, [108](#)
- allowChangeDoR
  - graphics\_main.cpp, [108](#)
- allowChangeDoX
  - graphics\_main.cpp, [109](#)
- allowChangeDoY
  - graphics\_main.cpp, [109](#)
- allowChangeDoZ
  - graphics\_main.cpp, [109](#)
- alpha\_foils
  - datas, [15](#)
- alpha\_left
  - datas, [15](#)
- alpha\_right
  - datas, [15](#)
- analytics\_dyn.cpp
  - acceleration\_compute, [58](#)
  - PI, [58](#)
  - pi, [58](#)
- analytics\_dyn.h
  - acceleration\_compute, [60](#)
- analytics\_geo.cpp
  - angles\_compute, [62](#)
  - EE\_flat\_compute, [62](#)
  - equ\_compute, [62](#)
  - forces\_all\_compute, [62](#)
  - forces\_compute, [62](#)
  - PI, [61](#)
  - pi, [61](#)
- analytics\_geo.h
  - angles\_compute, [64](#)
  - EE\_flat\_compute, [64](#)
  - equ\_compute, [64](#)
  - forces\_all\_compute, [65](#)
  - forces\_compute, [65](#)
- angle\_girouette
  - datas, [15](#)
- angles\_compute
  - analytics\_geo.cpp, [62](#)
  - analytics\_geo.h, [64](#)
- angles\_update
  - dataFillings.cpp, [70](#)
  - dataFillings.h, [72](#)
- AR\_left
  - datas, [16](#)
- AR\_pitch
  - datas, [16](#)
- AR\_right
  - datas, [16](#)
- AR\_sail
  - datas, [16](#)
- AR\_yaw

- datas, [16](#)
- B
  - seaDatas, [41](#)
  - SolidObject, [43](#)
- base
  - graphics\_main.cpp, [109](#)
- Baume
  - datas, [16](#)
- Boat\_create
  - solid\_objects.cpp, [123](#)
  - solid\_objects.h, [131](#)
- Boat\_delete
  - solid\_objects.cpp, [123](#)
  - solid\_objects.h, [132](#)
- Boat\_draw
  - solid\_objects.cpp, [123](#)
  - solid\_objects.h, [132](#)
- bridge2
  - solid\_objects.cpp, [127](#)
- bridge2Struct
  - solid\_objects.cpp, [127](#)
- bridgeMain
  - solid\_objects.cpp, [127](#)
- bridgeMainStruct
  - solid\_objects.cpp, [127](#)
- bStop
  - joystick\_PC.cpp, [97](#)
- BuildFont
  - graphics\_main.cpp, [105](#)
- Cal\_left\_neutral
  - datas, [16](#)
- Cal\_pitch\_neutral
  - datas, [16](#)
- Cal\_right\_neutral
  - datas, [17](#)
- Cal\_yaw\_neutral
  - datas, [17](#)
- CD\_left
  - datas, [17](#)
- CD\_pitch
  - datas, [17](#)
- CD\_right
  - datas, [17](#)
- CD\_sail
  - datas, [17](#)
- CD\_structure
  - datas, [17](#)
- CD\_yaw
  - datas, [17](#)
- Chord
  - datas, [17](#)
- Chord\_left

- datas, [18](#)
- Chord\_max
  - datas, [18](#)
- Chord\_min
  - datas, [18](#)
- Chord\_pitch
  - datas, [18](#)
- Chord\_right
  - datas, [18](#)
- Chord\_yaw
  - datas, [18](#)
- CL\_left
  - datas, [18](#)
- CL\_pitch
  - datas, [18](#)
- CL\_right
  - datas, [19](#)
- CL\_sail
  - datas, [19](#)
- CL\_yaw
  - datas, [19](#)
- compute\_state\_dot
  - compute\_state\_dot.cpp, [67](#)
  - compute\_state\_dot.h, [68](#)
- compute\_state\_dot.cpp
  - compute\_state\_dot, [67](#)
  - PI, [66](#)
  - pi, [66](#)
- compute\_state\_dot.h
  - compute\_state\_dot, [68](#)
- cosLUT
  - wave\_z\_compute\_byHand.cpp, [139](#)
- CreateGLWindow
  - graphics\_main.cpp, [105](#)
  - graphics\_relative.h, [114](#)
- cross\_product
  - solid\_objects.cpp, [123](#)
  - solid\_objects.h, [132](#)
- CumulNumOfSteps
  - ODE\_data, [37](#)
- Curseur
  - joystick\_PC.cpp, [98](#)
- cvoid\_mem
  - ODE\_data, [38](#)
- d
  - ODE\_data, [38](#)
- D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/CVODEincludes
  - [47](#)
- D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.cpp,
  - [49](#)
- D:/Basile/VisualCppProjects/hydroMEX3/hydroMEX3/f.h,
  - [53](#)





CD\_yaw, 17  
Chord, 17  
Chord\_left, 18  
Chord\_max, 18  
Chord\_min, 18  
Chord\_pitch, 18  
Chord\_right, 18  
Chord\_yaw, 18  
CL\_left, 18  
CL\_pitch, 18  
CL\_right, 19  
CL\_sail, 19  
CL\_yaw, 19  
d\_cam, 19  
dd\_cam, 19  
ddq, 19  
dEqu\_left, 19  
dEqu\_right, 19  
dEqu\_yaw, 19  
dphi, 20  
dpsi, 20  
dtheta, 20  
dx, 20  
dx\_cam, 20  
dy, 20  
dy\_cam, 20  
dz, 20  
dz\_cam, 21  
E\_foils, 21  
E\_yaw, 21  
EE\_left, 21  
EE\_left\_flat, 21  
EE\_right, 21  
EE\_right\_flat, 21  
EE\_yaw, 21  
EE\_yaw\_flat, 22  
Env, 22  
Env\_pitch, 22  
epsilon, 22  
Equ\_left, 22  
Equ\_right, 22  
Equ\_yaw, 22  
Fac\_update, 22  
Force\_drag, 23  
Force\_G\_E1, 23  
Force\_G\_E2, 23  
Force\_G\_E3, 23  
Force\_left, 23  
Force\_pitch, 23  
Force\_right, 23  
Force\_sail, 23  
Force\_yaw, 23  
g, 24  
G\_shift, 24  
gamma, 24  
Hslider\_x, 24  
Ix, 24  
Iy, 24  
Iz, 24  
k, 24  
L\_mast, 24  
L\_nose, 25  
Lambda, 25  
Lambda0, 25  
Lambda1, 25  
Lambda2, 25  
Long, 25  
M, 25  
max\_abs\_angle\_baume, 25  
N, 25  
phi, 26  
psi, 26  
rho, 26  
rho\_air, 26  
t, 26  
Tan\_AoA\_left, 26  
Tan\_AoA\_pitch, 26  
Tan\_AoA\_right, 26  
Tan\_AoA\_sail, 27  
Tan\_AoA\_yaw, 27  
tanAngle\_sail, 27  
tanCal\_left, 27  
tanCal\_pitch, 27  
tanCal\_right, 27  
tanCal\_yaw, 27  
tanCal\_yaw0, 27  
target\_x, 28  
target\_y, 28  
theta, 28  
Torque\_G\_E1, 28  
Torque\_G\_E2, 28  
Torque\_G\_E3, 28  
V\_wave, 28  
V\_wave0, 28  
V\_wave1, 28  
V\_wave2, 29  
verif, 29  
Vslider\_x, 29  
Wave\_amp, 29  
Wave\_amp0, 29  
Wave\_amp1, 29  
Wave\_amp2, 29  
Wave\_angle, 29  
Wave\_angle0, 29  
Wave\_angle1, 30  
Wave\_angle2, 30  
Wind, 30  
Wind\_angle, 30

- Wind\_x, 30
- Wind\_y, 30
- x, 30
- x\_cam, 30
- x\_foils, 31
- x\_pitch, 31
- x\_sail, 31
- x\_yaw, 31
- y, 31
- y\_cam, 31
- y\_foil\_left, 31
- y\_foil\_right, 31
- y\_pitch, 32
- y\_sail, 32
- y\_yaw, 32
- z, 32
- z\_cam, 32
- z\_foils, 32
- z\_pitch, 32
- z\_sail, 32
- z\_yaw, 32
- dd\_cam
  - datas, 19
- ddq
  - datas, 19
- delete\_state
  - get\_mex\_arguments.cpp, 78
  - get\_mex\_arguments.h, 83
- dEqu\_left
  - datas, 19
- dEqu\_right
  - datas, 19
- dEqu\_yaw
  - datas, 19
- dirR
  - graphics\_main.cpp, 109
  - joy\_parameters, 34
- dirX
  - graphics\_main.cpp, 109
  - joy\_parameters, 34
- dirY
  - graphics\_main.cpp, 109
  - joy\_parameters, 35
- dirZ
  - graphics\_main.cpp, 109
  - joy\_parameters, 35
- done
  - graphics\_main.cpp, 109
- doR
  - graphics\_main.cpp, 110
  - joy\_parameters, 35
- doX
  - graphics\_main.cpp, 110
  - joy\_parameters, 35
- doY
  - graphics\_main.cpp, 110
  - joy\_parameters, 35
- doZ
  - graphics\_main.cpp, 110
  - joy\_parameters, 35
- dphi
  - datas, 20
- dpsi
  - datas, 20
- draw\_graphics
  - graphics\_main.cpp, 105
  - graphics\_relative.h, 114
- draw\_graphics\_init
  - graphics\_main.cpp, 106
  - graphics\_relative.h, 115
- draw\_graphics\_kill
  - graphics\_main.cpp, 106
  - graphics\_relative.h, 115
- Draw\_sea
  - Sea.h, 118
  - wave\_z\_compute\_byHand.cpp, 137
- DrawGLScene
  - graphics\_main.cpp, 106
  - graphics\_relative.h, 115
- dtheta
  - datas, 20
- dviewAngle
  - graphics\_main.cpp, 110
- dviewDist
  - graphics\_main.cpp, 110
- dwBoutons
  - joystick\_PC.cpp, 98
- dx
  - datas, 20
- dx\_cam
  - datas, 20
- dy
  - datas, 20
- dy\_cam
  - datas, 20
- dz
  - datas, 20
- dz\_cam
  - datas, 21
- E\_foils
  - datas, 21
- E\_yaw
  - datas, 21
- EE\_flat\_compute
  - analytics\_geo.cpp, 62
  - analytics\_geo.h, 64
- EE\_left

- datas, [21](#)
- EE\_left\_flat
  - datas, [21](#)
- EE\_right
  - datas, [21](#)
- EE\_right\_flat
  - datas, [21](#)
- EE\_yaw
  - datas, [21](#)
- EE\_yaw\_flat
  - datas, [22](#)
- Env
  - datas, [22](#)
- Env\_pitch
  - datas, [22](#)
- epsilon
  - datas, [22](#)
- equ\_compute
  - analytics\_geo.cpp, [62](#)
  - analytics\_geo.h, [64](#)
- Equ\_left
  - datas, [22](#)
- Equ\_right
  - datas, [22](#)
- Equ\_yaw
  - datas, [22](#)
- f
  - f.cpp, [50](#)
  - f.h, [53](#)
- f.cpp
  - f, [50](#)
  - IJth, [50](#)
  - Ith, [50](#)
  - NEQ, [50](#)
  - PI, [50](#)
  - pi, [50](#)
  - solve\_ODE, [51](#)
  - solver\_free, [51](#)
  - solver\_init, [51](#)
  - T0, [50](#)
- f.h
  - f, [53](#)
  - solve\_ODE, [53](#)
  - solver\_free, [53](#)
  - solver\_init, [54](#)
- Fac\_update
  - datas, [22](#)
- fillWaveLuts
  - Sea.h, [119](#)
  - wave\_z\_compute\_byHand.cpp, [137](#)
- flag
  - ODE\_data, [38](#)
- flagr
  - ODE\_data, [38](#)
- foill
  - solid\_objects.cpp, [127](#)
- foilLeftStruct
  - solid\_objects.cpp, [127](#)
- foilr
  - solid\_objects.cpp, [127](#)
- foilRightStruct
  - solid\_objects.cpp, [128](#)
- Force\_drag
  - datas, [23](#)
- Force\_G\_E1
  - datas, [23](#)
- Force\_G\_E2
  - datas, [23](#)
- Force\_G\_E3
  - datas, [23](#)
- Force\_left
  - datas, [23](#)
- Force\_pitch
  - datas, [23](#)
- Force\_right
  - datas, [23](#)
- Force\_sail
  - datas, [23](#)
- Force\_yaw
  - datas, [23](#)
- forces\_all\_compute
  - analytics\_geo.cpp, [62](#)
  - analytics\_geo.h, [65](#)
- forces\_compute
  - analytics\_geo.cpp, [62](#)
  - analytics\_geo.h, [65](#)
- fps
  - graphics\_main.cpp, [110](#)
- free\_wave\_variables
  - Sea.h, [119](#)
  - wave\_z\_compute\_byHand.cpp, [138](#)
- fullscreen
  - graphics\_main.cpp, [110](#)
- G
  - seaDatas, [41](#)
  - SolidObject, [43](#)
- g
  - datas, [24](#)
- G\_shift
  - datas, [24](#)
- gamma
  - datas, [24](#)
- get\_flag
  - get\_mex\_arguments.cpp, [78](#)
  - get\_mex\_arguments.h, [83](#)
- get\_joystick\_parameters

- get\_mex\_arguments.cpp, 78
  - get\_mex\_arguments.h, 83
- get\_mex\_arguments.cpp
  - delete\_state, 78
  - get\_flag, 78
  - get\_joystick\_parameters, 78
  - get\_Parameters, 79
  - get\_state, 79
  - get\_time, 80
  - initParameters, 80
  - PI, 78
  - pi, 78
- get\_mex\_arguments.h
  - delete\_state, 83
  - get\_flag, 83
  - get\_joystick\_parameters, 83
  - get\_Parameters, 84
  - get\_state, 84
  - get\_time, 85
  - initParameters, 85
  - PARAMNUM, 83
  - STATENUM, 83
- get\_Parameters
  - get\_mex\_arguments.cpp, 79
  - get\_mex\_arguments.h, 84
- get\_state
  - get\_mex\_arguments.cpp, 79
  - get\_mex\_arguments.h, 84
- get\_time
  - get\_mex\_arguments.cpp, 80
  - get\_mex\_arguments.h, 85
- Girouette\_draw
  - solid\_objects.cpp, 124
  - solid\_objects.h, 132
- glPrint
  - graphics\_main.cpp, 106
- graphics\_main.cpp
  - active, 108
  - allowChangeDirR, 108
  - allowChangeDirX, 108
  - allowChangeDirY, 108
  - allowChangeDirZ, 108
  - allowChangeDoR, 108
  - allowChangeDoX, 109
  - allowChangeDoY, 109
  - allowChangeDoZ, 109
  - base, 109
  - BuildFont, 105
  - CreateGLWindow, 105
  - dirR, 109
  - dirX, 109
  - dirY, 109
  - dirZ, 109
  - done, 109
  - doR, 110
  - doX, 110
  - doY, 110
  - doZ, 110
  - draw\_graphics, 105
  - draw\_graphics\_init, 106
  - draw\_graphics\_kill, 106
  - DrawGLScene, 106
  - dviewAngle, 110
  - dviewDist, 110
  - fps, 110
  - fullscreen, 110
  - glPrint, 106
  - hDC, 110
  - hInstance, 111
  - hRC, 111
  - hWnd, 111
  - InitGL, 107
  - joystick\_exist, 111
  - keys, 111
  - KillFont, 107
  - KillGLWindow, 107
  - light, 111
  - LightAmbient, 111
  - LightDiffuse, 111
  - LightPosition, 111
  - msg, 112
  - pass\_joy\_parameters, 107
  - ReSizeGLScene, 107
  - time\_before, 112
  - time\_elapsed, 112
  - viewAngle1, 112
  - viewAngle2, 112
  - viewDist, 112
  - WndProc, 108
- graphics\_relative.h
  - CreateGLWindow, 114
  - draw\_graphics, 114
  - draw\_graphics\_init, 115
  - draw\_graphics\_kill, 115
  - DrawGLScene, 115
  - InitGL, 116
  - KillGLWindow, 116
  - pass\_joy\_parameters, 116
  - ReSizeGLScene, 116
  - WndProc, 117
- hDC
  - graphics\_main.cpp, 110
- hInstance
  - graphics\_main.cpp, 111
- hRC
  - graphics\_main.cpp, 111
- HSlider\_draw

- solid\_objects.cpp, 124
- solid\_objects.h, 133
- Hslider\_x
  - datas, 24
- hull
  - solid\_objects.cpp, 128
- hullStruct
  - solid\_objects.cpp, 128
- hWnd
  - graphics\_main.cpp, 111
- hydroMEX3.cpp
  - MAXFLAGLEN, 94
  - mexFunction, 94
  - PI, 94
  - pi, 94
- IJth
  - f.cpp, 50
- InfosCaps
  - joystick\_PC.cpp, 98
- InitGL
  - graphics\_main.cpp, 107
  - graphics\_relative.h, 116
- initParameters
  - get\_mex\_arguments.cpp, 80
  - get\_mex\_arguments.h, 85
- iout
  - ODE\_data, 38
- Ith
  - f.cpp, 50
  - return\_mex\_arguments.cpp, 88
- Ix
  - datas, 24
- Iy
  - datas, 24
- Iz
  - datas, 24
- joy\_parameters, 34
  - dirR, 34
  - dirX, 34
  - dirY, 35
  - dirZ, 35
  - doR, 35
  - doX, 35
  - doY, 35
  - doZ, 35
- joystick\_close
  - joystick\_PC.cpp, 97
  - joystick\_PC.h, 99
- joystick\_exist
  - graphics\_main.cpp, 111
- joystick\_getXYZR
  - joystick\_PC.cpp, 97
  - joystick\_PC.h, 99
- joystick\_init
  - joystick\_PC.cpp, 97
  - joystick\_PC.h, 100
- joystick\_PC.cpp
  - ActualPos, 97
  - bStop, 97
  - Curseur, 98
  - dwBoutons, 98
  - InfosCaps, 98
  - joystick\_close, 97
  - joystick\_getXYZR, 97
  - joystick\_init, 97
  - uMax, 98
  - uMin, 98
  - uPoolPeriod, 98
- joystick\_PC.h
  - joystick\_close, 99
  - joystick\_getXYZR, 99
  - joystick\_init, 100
- k
  - datas, 24
- keys
  - graphics\_main.cpp, 111
- KillFont
  - graphics\_main.cpp, 107
- KillGLWindow
  - graphics\_main.cpp, 107
  - graphics\_relative.h, 116
- L
  - seaDatas, 41
- L\_mast
  - datas, 24
- L\_nose
  - datas, 25
- Lambda
  - datas, 25
- Lambda0
  - datas, 25
- Lambda1
  - datas, 25
- Lambda2
  - datas, 25
- light
  - graphics\_main.cpp, 111
- LightAmbient
  - graphics\_main.cpp, 111
- LightDiffuse
  - graphics\_main.cpp, 111
- LightPosition
  - graphics\_main.cpp, 111
- Long

- datas, 25
- ISwimm
  - solid\_objects.cpp, 128
- ISwimmStruct
  - solid\_objects.cpp, 128
- LUTLEN
  - wave\_z\_compute\_byHand.cpp, 137
- LUTLENF
  - wave\_z\_compute\_byHand.cpp, 137
- M
  - datas, 25
  - SolidObject, 44
- mast
  - solid\_objects.cpp, 128
- mastStruct
  - solid\_objects.cpp, 128
- max\_abs\_angle\_baume
  - datas, 25
- MAXFLAGLEN
  - hydroMEX3.cpp, 94
- mexFunction
  - hydroMEX3.cpp, 94
- msg
  - graphics\_main.cpp, 112
- N
  - datas, 25
  - seaDatas, 41
  - SolidObject, 44
- NEQ
  - f.cpp, 50
- nX
  - SolidObject, 44
- nY
  - SolidObject, 44
- nZ
  - SolidObject, 44
- ODE\_data, 37
  - abstol, 37
  - CumulNumOfSteps, 37
  - cvoid\_mem, 38
  - d, 38
  - flag, 38
  - flagr, 38
  - iout, 38
  - P, 38
  - reitol, 38
  - rootsfound, 38
  - state, 38
  - t, 39
  - time\_param, 39
  - tout, 39

- y, 39
- offset
  - timing\_PC.cpp, 143
- one\_normal
  - solid\_objects.cpp, 124
  - solid\_objects.h, 133
- P
  - ODE\_data, 38
- param
  - parameters, 40
- parameters, 40
  - abstol, 40
  - param, 40
  - reitol, 40
- PARAMNUM
  - get\_mex\_arguments.h, 83
- pass\_joy\_parameters
  - graphics\_main.cpp, 107
  - graphics\_relative.h, 116
- phi
  - datas, 26
- PI
  - aero\_coeff\_control.cpp, 55
  - analytics\_dyn.cpp, 58
  - analytics\_geo.cpp, 61
  - compute\_state\_dot.cpp, 66
  - dataFillings.cpp, 70
  - f.cpp, 50
  - get\_mex\_arguments.cpp, 78
  - hydroMEX3.cpp, 94
  - return\_mex\_arguments.cpp, 88
- pi
  - aero\_coeff\_control.cpp, 55
  - analytics\_dyn.cpp, 58
  - analytics\_geo.cpp, 61
  - compute\_state\_dot.cpp, 66
  - dataFillings.cpp, 70
  - f.cpp, 50
  - get\_mex\_arguments.cpp, 78
  - hydroMEX3.cpp, 94
  - return\_mex\_arguments.cpp, 88
  - wave\_z\_compute\_byHand.cpp, 137
- pitch
  - solid\_objects.cpp, 128
- pitchStruct
  - solid\_objects.cpp, 128
- psi
  - datas, 26
- R
  - seaDatas, 41
  - SolidObject, 44
- RelativeWind\_draw

- solid\_objects.cpp, 125
- solid\_objects.h, 133
- reltol
  - ODE\_data, 38
  - parameters, 40
- ReSizeGLScene
  - graphics\_main.cpp, 107
  - graphics\_relative.h, 116
- return\_args
  - return\_mex\_arguments.cpp, 88
  - return\_mex\_arguments.h, 91
- return\_mex\_arguments.cpp
  - lth, 88
  - PI, 88
  - pi, 88
  - return\_args, 88
  - sout00, 89
  - sout01, 89
  - sout02, 89
  - sout03, 89
  - sout04, 89
  - sout05, 90
  - sout06, 90
  - sout07, 90
  - sout08, 90
  - sout09, 90
  - sout10, 90
  - sout11, 90
  - store\_state, 89
  - timev, 90
- return\_mex\_arguments.h
  - return\_args, 91
  - store\_state, 92
- rho
  - datas, 26
- rho\_air
  - datas, 26
- rootsfound
  - ODE\_data, 38
- rSwimm
  - solid\_objects.cpp, 129
- rSwimmStruct
  - solid\_objects.cpp, 129
- Sail\_create\_and\_draw
  - solid\_objects.cpp, 125
  - solid\_objects.h, 134
- sd
  - wave\_z\_compute\_byHand.cpp, 139
- Sea.h
  - Draw\_sea, 118
  - fillWaveLuts, 119
  - free\_wave\_variables, 119
  - wave\_init, 119
  - wave\_z\_compute, 119
- seaDat
  - wave\_z\_compute\_byHand.cpp, 139
- seaDatas, 41
- seaDatas
  - B, 41
  - G, 41
  - L, 41
  - N, 41
  - R, 41
  - z, 41
- sinLUT
  - wave\_z\_compute\_byHand.cpp, 139
- solid\_objects.cpp
  - Boat\_create, 123
  - Boat\_delete, 123
  - Boat\_draw, 123
  - bridge2, 127
  - bridge2Struct, 127
  - bridgeMain, 127
  - bridgeMainStruct, 127
  - cross\_product, 123
  - foill, 127
  - foilLeftStruct, 127
  - foilr, 127
  - foilRightStruct, 128
  - Girouette\_draw, 124
  - HSlider\_draw, 124
  - hull, 128
  - hullStruct, 128
  - lSwimm, 128
  - lSwimmStruct, 128
  - mast, 128
  - mastStruct, 128
  - one\_normal, 124
  - pitch, 128
  - pitchStruct, 128
  - RelativeWind\_draw, 125
  - rSwimm, 129
  - rSwimmStruct, 129
  - Sail\_create\_and\_draw, 125
  - SolidObject\_compute\_normals, 125
  - SolidObject\_create, 125
  - SolidObject\_delete, 126
  - SolidObject\_draw, 126
  - Target\_draw, 126
  - VSlider\_draw, 126
  - yaw, 129
  - yawStruct, 129
- solid\_objects.h
  - Boat\_create, 131
  - Boat\_delete, 132
  - Boat\_draw, 132
  - cross\_product, 132

- Girouette\_draw, [132](#)
- HSlider\_draw, [133](#)
- one\_normal, [133](#)
- RelativeWind\_draw, [133](#)
- Sail\_create\_and\_draw, [134](#)
- SolidObject\_compute\_normals, [134](#)
- SolidObject\_create, [134](#)
- SolidObject\_delete, [134](#)
- Target\_draw, [135](#)
- VSlider\_draw, [135](#)
- SolidObject, [43](#)
- SolidObject
  - B, [43](#)
  - G, [43](#)
  - M, [44](#)
  - N, [44](#)
  - nX, [44](#)
  - nY, [44](#)
  - nZ, [44](#)
  - R, [44](#)
  - X, [44](#)
  - Y, [45](#)
  - Z, [45](#)
- SolidObject\_compute\_normals
  - solid\_objects.cpp, [125](#)
  - solid\_objects.h, [134](#)
- SolidObject\_create
  - solid\_objects.cpp, [125](#)
  - solid\_objects.h, [134](#)
- SolidObject\_delete
  - solid\_objects.cpp, [126](#)
  - solid\_objects.h, [134](#)
- SolidObject\_draw
  - solid\_objects.cpp, [126](#)
- solve\_ODE
  - f.cpp, [51](#)
  - f.h, [53](#)
- solver\_free
  - f.cpp, [51](#)
  - f.h, [53](#)
- solver\_init
  - f.cpp, [51](#)
  - f.h, [54](#)
- sout00
  - return\_mex\_arguments.cpp, [89](#)
- sout01
  - return\_mex\_arguments.cpp, [89](#)
- sout02
  - return\_mex\_arguments.cpp, [89](#)
- sout03
  - return\_mex\_arguments.cpp, [89](#)
- sout04
  - return\_mex\_arguments.cpp, [89](#)
- sout05
  - return\_mex\_arguments.cpp, [90](#)
- sout06
  - return\_mex\_arguments.cpp, [90](#)
- sout07
  - return\_mex\_arguments.cpp, [90](#)
- sout08
  - return\_mex\_arguments.cpp, [90](#)
- sout09
  - return\_mex\_arguments.cpp, [90](#)
- sout10
  - return\_mex\_arguments.cpp, [90](#)
- sout11
  - return\_mex\_arguments.cpp, [90](#)
- state
  - ODE\_data, [38](#)
- STATENUM
  - get\_mex\_arguments.h, [83](#)
- std, [9](#)
- stdafx.h
  - WIN32\_LEAN\_AND\_MEAN, [102](#)
- store\_state
  - return\_mex\_arguments.cpp, [89](#)
  - return\_mex\_arguments.h, [92](#)
- t
  - datas, [26](#)
  - ODE\_data, [39](#)
- T0
  - f.cpp, [50](#)
- Tan\_AoA\_left
  - datas, [26](#)
- Tan\_AoA\_pitch
  - datas, [26](#)
- Tan\_AoA\_right
  - datas, [26](#)
- Tan\_AoA\_sail
  - datas, [27](#)
- Tan\_AoA\_yaw
  - datas, [27](#)
- tanAngle\_sail
  - datas, [27](#)
- tanCal\_left
  - datas, [27](#)
- tanCal\_pitch
  - datas, [27](#)
- tanCal\_right
  - datas, [27](#)
- tanCal\_yaw
  - datas, [27](#)
- tanCal\_yaw0
  - datas, [27](#)
- Target\_draw
  - solid\_objects.cpp, [126](#)
  - solid\_objects.h, [135](#)



- target\_x
  - datas, 28
- target\_y
  - datas, 28
- theta
  - datas, 28
- tic
  - timing.h, 140
  - timing\_PC.cpp, 142
- time\_before
  - graphics\_main.cpp, 112
- time\_elapsed
  - graphics\_main.cpp, 112
- time\_param
  - ODE\_data, 39
- timev
  - return\_mex\_arguments.cpp, 90
- timing.h
  - tic, 140
  - toc, 140
- timing\_PC.cpp
  - offset, 143
  - tic, 142
  - toc, 142
- toc
  - timing.h, 140
  - timing\_PC.cpp, 142
- Torque\_G\_E1
  - datas, 28
- Torque\_G\_E2
  - datas, 28
- Torque\_G\_E3
  - datas, 28
- tout
  - ODE\_data, 39
- uMax
  - joystick\_PC.cpp, 98
- uMin
  - joystick\_PC.cpp, 98
- uPoolPeriod
  - joystick\_PC.cpp, 98
- V\_wave
  - datas, 28
- V\_wave0
  - datas, 28
- V\_wave1
  - datas, 28
- V\_wave2
  - datas, 29
- verif
  - datas, 29
- viewAngle1
  - graphics\_main.cpp, 112
- viewAngle2
  - graphics\_main.cpp, 112
- viewDist
  - graphics\_main.cpp, 112
- VSlider\_draw
  - solid\_objects.cpp, 126
  - solid\_objects.h, 135
- Vslider\_x
  - datas, 29
- Wave\_amp
  - datas, 29
- Wave\_amp0
  - datas, 29
- Wave\_amp1
  - datas, 29
- Wave\_amp2
  - datas, 29
- Wave\_angle
  - datas, 29
- Wave\_angle0
  - datas, 29
- Wave\_angle1
  - datas, 30
- Wave\_angle2
  - datas, 30
- wave\_init
  - Sea.h, 119
  - wave\_z\_compute\_byHand.cpp, 138
- wave\_z\_compute
  - Sea.h, 119
  - wave\_z\_compute\_byHand.cpp, 138
- wave\_z\_compute\_byHand.cpp
  - cosLUT, 139
  - Draw\_sea, 137
  - fillWaveLuts, 137
  - free\_wave\_variables, 138
  - LUTLEN, 137
  - LUTLENF, 137
  - pi, 137
  - sd, 139
  - seaDat, 139
  - sinLUT, 139
  - wave\_init, 138
  - wave\_z\_compute, 138
  - Wcos, 138
  - Wsin, 139
- Wcos
  - wave\_z\_compute\_byHand.cpp, 138
- WIN32\_LEAN\_AND\_MEAN
  - stdafx.h, 102
- Wind
  - datas, 30

- Wind\_angle
  - datas, [30](#)
- Wind\_x
  - datas, [30](#)
- Wind\_y
  - datas, [30](#)
- WndProc
  - graphics\_main.cpp, [108](#)
  - graphics\_relative.h, [117](#)
- Wsin
  - wave\_z\_compute\_byHand.cpp, [139](#)
- X
  - SolidObject, [44](#)
- x
  - datas, [30](#)
- x\_cam
  - datas, [30](#)
- x\_foils
  - datas, [31](#)
- x\_pitch
  - datas, [31](#)
- x\_sail
  - datas, [31](#)
- x\_yaw
  - datas, [31](#)
- Y
  - SolidObject, [45](#)
- y
  - datas, [31](#)
  - ODE\_data, [39](#)
- y\_cam
  - datas, [31](#)
- y\_foil\_left
  - datas, [31](#)
- y\_foil\_right
  - datas, [31](#)
- y\_pitch
  - datas, [32](#)
- y\_sail
  - datas, [32](#)
- y\_yaw
  - datas, [32](#)
- yaw
  - solid\_objects.cpp, [129](#)
- yawStruct
  - solid\_objects.cpp, [129](#)
- Z
  - SolidObject, [45](#)
- z
  - datas, [32](#)
  - seaDats, [41](#)
- z\_cam
  - datas, [32](#)
- z\_foils
  - datas, [32](#)
- z\_pitch
  - datas, [32](#)
- z\_sail
  - datas, [32](#)
- z\_yaw
  - datas, [32](#)