Solving Wave Equation with MPI paralleling

Generated by Doxygen 1.9.1

Chapter 1

ACSE-6 MPI Programming Assignment – Solving the Wave Equation

This piece of coursework was created by Jin Yu (acse-jy220), github handle: https://github.com/acse-jy220

1.1 Compile and run the task

You will need a compiler which supports MPI on your system to compile and run this task.

```
$ mpic++ main.cpp -o main -std=c++11
$ mpiexec -n (number of nodes) ./main
```

1.2 User Handbook

1.2.1 About parameter settings

The initial settings for the problem is located at `parameters.ini`(do not change its name!), it offers a number of changable parameters:

----------—Physical parameters----------—

`length`: the range of the x-coordinate of the region
`width`: the range of the y-coordinate of the region
`imax`: number of grids in the x-coordinate
`jmax`: number of grids in the y-coordinate
`c`: the speed of the wave
`dt`: the numerical time step we use in the forward updating scheme.

----------—Initial condition parameters----------—

for the Initial condition setting, we just use sinusoidal disturbances over a period,
where u_0 = 5.0∗(cos(distance / radius ∗ pi) + 1.0) for some disturbance.
`rs`: array of radius for the disturbances
`xs`: array of the x-coordinate of centers for the disturbances
`ys`: array of the y-coordinate of centers for the disturbances

Note as show in the picture, a column of `rs`, `xs` and `ys` represents the radius, x-coordinate and y-coordinate for **one** specific disturbance,
if you input multiple disturbances, please make sure they are on the correct indexes.
----------—Output parameters----------—
`output`: if turned on, binary files will be created in the `./ouput` folder, if not, just purely run the simulation without any outputs. `t_out` : the output time steps, or say, output images every 't_out' seconds.
`t_end` : the end time of the simulation. (of course the start time is 0)
----------—Boundary types----------—
`boundary_type`: There are three boundary types for choose, they are 'periodic', 'Neumann' and 'Dirichlet'.
periodic boundary: eable wave to passing through boundaries
(i.e the left nerighbour of a grid on the left physical boundary is the grid on the right physical boundary)
Neumann boundary: the divergence of 'u' is zero at the boundary, on square grid it is equal to etting the value on the boundary equal to the neighbouring point.
Dirichlet boundary: 'u' fixed with a value of zero on boundaries
----------—Display options----------—
`display_option`: if turned on, you will see the output information throughout the simulation, like the picture given followed:

if you truned it off, a `std::out` sentence only indicates the running time of the code will be given.

1.2.2 Postprocessing outputs

After you run a successful simulation, the output files would be saved at the `./output` folder in a binary format. (check they are there!)
You could then use python scripts to generate plots and animations from those binary files.
----------—Generate a single shot for a specific time point in the simulation----------—
If you are interested at the image of a single output file and want to view it, just type

```
$ python3 pic.py (number of the output)
```

It would give you a picture shot of the output state you are interested in, like following:

If you want to save this figure, add "-save" flag at the end of the command, like:

```
$ python3 pic.py (number of the output) -save
```

The picture would be save at the `./pic` folder.

----------—Generate an animation for the simulation----------—
If you want to view an animation for the simulation, just type

```
$ python3 animation.py
```

You can also enter any integer after a '-s' flag, indicating a speedup ratio, such as

```
$ python3 animation.py -s 8
```

To make the animation play 8 times faster.

Note that if you do not add the '-s' flag, the speed will just set default as 1.

If you want to save the animation, just add '-save' flag at the end, like

```
$ python3 animation.py -save
```

The animation will be save at `./anime` folder, with name like "anime [length] $*$ [width] grid (Mesh size: [imax] $*$ [jmax]) dt = xxx, boundary_type = xxx.mp4".

I have attached the videos for three_example runs in the `./anime` folder, with intial condition being two splashs of radius 1.12381 and 3.2313, locating at (3.12312, 3.31312) and (9.2313, 9.12312) respectively.

1.2.3 Something important to note

Before you run another simulation, if you are not changing 'mesh size' or 'end_time', then it would be possibly be safe to rerun directly, but it's safer to clear the outputs of the last run before running a new simulation. Here I provide a script to clear the outputs, just use

```
bash ./clear_output.sh
```

to clear the outputs and then change the parameters in `parameters.ini` and run a new simulation.

Many thanks for using this piece of software!!!!

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 animation Namespace Reference

Functions

- def update_grid (int picnum)

Variables

- num_pics = len(glob.glob('./output/∗'))
- parameters = loaddata.read_parameters()
- t_out = float(parameters['t_out'])
- width = int(parameters['width'])
- length = int(parameters['length'])
- imax = int(parameters['imax'])
- jmax = int(parameters['jmax'])
- dt = float(parameters['dt'])
- boundary_type = str(parameters['boundary_type'])
- ele_x = int(length / 5)
- ele_y = int(width / 5)
- ele_i = int(imax / 5)
- ele_j = int(jmax / 5)
- list x_label = [ele_x ∗ i for i in range(6)]
- list y_label = [ele_y ∗ i for i in range(6)]
- list i_loc = [ele_i ∗ i for i in range(6)]
- list j_loc = [ele_j ∗ i for i in range(6)]
- speedratio = int(sys.argv[2])
- fig = plt.figure()
- ax = fig.add_subplot(111)
- fontsize
- text = plt.text(x = 0.95, y = 0.95, s='t = 0 s',fontdict=dict(fontsize=12, color='r',family='monospace',))
- arr = loaddata.load_binary_file(0).reshape((imax, jmax))
- norm = mpl.colors.Normalize(vmin = np.min(arr), vmax = np.max(arr))
- cax = ax.imshow(arr, interpolation='nearest', cmap = 'winter', norm = norm, origin = 'lower')
- cb = fig.colorbar(cax)
- anim = animation.FuncAnimation(fig, update_grid, frames = np.arange(0, num_pics, speedratio), blit=True)
- string ani_name = './anime/%d ∗ %d grid (Mesh size: %d ∗ %d) dt = %f, boundary_type = %s.mp4' % (length, width, imax, jmax, dt, boundary_type)
- fps
- dpi

5.1.1 Function Documentation

5.1.1.1 update_grid()

```
def animation.update_grid (
            int picnum )
```

Here is the call graph for this function:

5.1.2 Variable Documentation

5.1.2.1 ani_name

```
animation.ani_name = './anime/%d * %d grid (Mesh size:  %d * %d) dt = %f, boundary_type = %s.↩
mp4' % (length, width, imax, jmax, dt, boundary_type )
```

5.1.2.2 anim

```
animation.anim = animation.FuncAnimation(fig, update_grid, frames = np.arange(0, num_pics,
speedratio), blit=True)
```

5.1.2.3 arr

```
animation.arr = loaddata.load_binary_file(0).reshape((imax, jmax))
```

5.1.2.4 ax

```
animation.ax = fig.add_subplot(111)
```

5.1.2.5 boundary_type

```
animation.boundary_type = str(parameters['boundary_type'])
```

5.1.2.6 cax

```
animation.cax = ax.imshow(arr, interpolation='nearest', cmap = 'winter', norm = norm, origin =
'lower')
```

5.1.2.7 cb

```
animation.cb = fig.colorbar(cax)
```

5.1.2.8 dpi

```
animation.dpi
```

5.1.2.9 dt

```
animation.dt = float(parameters['dt'])
```

5.1.2.10 ele_i

```
animation.ele_i = int(imax / 5)
```

5.1.2.11 ele_j

```
animation.ele_j = int(jmax / 5)
```

5.1.2.12 ele_x

```
animation.ele_x = int(length / 5)
```

5.1.2.13 ele_y

```
animation.ele_y = int(width / 5)
```

5.1.2.14 fig

```
animation.fig = plt.figure()
```

5.1.2.15 fontsize

```
animation.fontsize
```

5.1.2.16 fps

```
animation.fps
```

5.1.2.17 i_loc

```
list animation.i_loc = [ele_i * i for i in range(6)]
```

5.1.2.18 imax

```
animation.imax = int(parameters['imax'])
```

5.1.2.19 j_loc

```
list animation.j_loc = [ele_j * i for i in range(6)]
```

5.1.2.20 jmax

```
animation.jmax = int(parameters['jmax'])
```

5.1.2.21 length

```
animation.length = int(parameters['length'])
```

5.1.2.22 norm

```
animation.norm = mpl.colors.Normalize(vmin = np.min(arr), vmax = np.max(arr))
```

5.1.2.23 num_pics

```
animation.num_pics = len(glob.glob('./output/*'))
```

5.1.2.24 parameters

```
animation.parameters = loaddata.read_parameters()
```

5.1.2.25 speedratio

```
int animation.speedratio = int(sys.argv[2])
```

5.1.2.26 t_out

```
animation.t_out = float(parameters['t_out'])
```

5.1.2.27 text

```
animation.text = plt.text(x = 0.95, y = 0.95, s='t = 0 s',fontdict=dict(fontsize=12, color='r',family='monospa
```

5.1.2.28 width

```
animation.width = int(parameters['width'])
```

5.1.2.29 x_label

```
list animation.x_label = [ele_x * i for i in range(6)]
```

5.1.2.30 y_label

```
list animation.y_label = [ele_y * i for i in range(6)]
```

5.2 loaddata Namespace Reference

Functions

- def load_binary_file (int i)
- def load_array (int i)
- def read_parameters ()

5.2.1 Function Documentation

5.2.1.1 load_array()

```
def loaddata.load_array (
            int i )
```

5.2.1.2 load_binary_file()

```
def loaddata.load_binary_file (
            int i )
```

5.2.1.3 read_parameters()

```
def loaddata.read_parameters ( )
```

5.3 pic Namespace Reference

Variables

- parameters = loaddata.read_parameters()
- t_out = float(parameters['t_out'])
- width = int(parameters['width'])
- length = int(parameters['length'])
- imax = int(parameters['imax'])
- jmax = int(parameters['jmax'])
- dt = float(parameters['dt'])
- boundary_type = str(parameters['boundary_type'])
- picnum = int(sys.argv[1])
- data = loaddata.load_binary_file(picnum).reshape((imax, jmax))
- ele_x = int(length / 5)
- ele_y = int(width / 5)
- ele_i = int(imax / 5)
- ele_j = int(jmax / 5)
- list x_label = [ele_x ∗ i for i in range(6)]
- list y_label = [ele_y ∗ i for i in range(6)]
- list i_loc = [ele_i ∗ i for i in range(6)]
- list j_loc = [ele_j ∗ i for i in range(6)]
- t = t_out ∗ picnum
- fig = plt.figure()
- ax = fig.add_subplot(111)
- fontsize
- text = plt.text(x = 0.95, y = 0.95, s='t = %.3f s' % (t),fontdict=dict(fontsize=12, color='r',family='monospace',))
- arr = data
- norm = mpl.colors.Normalize(vmin = np.min(arr), vmax = np.max(arr))
- cax = ax.imshow(arr, interpolation='nearest', cmap = 'winter', norm = norm, origin = 'lower')
- cb = fig.colorbar(cax)
- string pic_name = './pic/%d ∗ %d grid (Mesh size: %d ∗ %d) t = %f, boundary_type = %s.png' % (length, width, imax, jmax, t, boundary_type)
- dpi

5.3.1 Variable Documentation

5.3.1.1 arr

```
pic.arr = data
```

5.3.1.2 ax

```
pic.ax = fig.add_subplot(111)
```

5.3.1.3 boundary_type

```
pic.boundary_type = str(parameters['boundary_type'])
```

5.3.1.4 cax

```
pic.cax = ax.imshow(arr, interpolation='nearest', cmap = 'winter', norm = norm, origin = 'lower')
```

5.3.1.5 cb

```
pic.cb = fig.colorbar(cax)
```

5.3.1.6 data

```
pic.data = loaddata.load_binary_file(picnum).reshape((imax, jmax))
```

5.3.1.7 dpi

```
pic.dpi
```

5.3.1.8 dt

```
pic.dt = float(parameters['dt'])
```

5.3.1.9 ele_i

```
pic.ele_i = int(imax / 5)
```

5.3.1.10 ele_j

```
pic.ele_j = int(jmax / 5)
```

5.3.1.11 ele_x

```
pic.ele_x = int(length / 5)
```

5.3.1.12 ele_y

```
pic.ele_y = int(width / 5)
```

5.3.1.13 fig

```
pic.fig = plt.figure()
```

5.3.1.14 fontsize

```
pic.fontsize
```

5.3.1.15 i_loc

```
list pic.i_loc = [ele_i * i for i in range(6)]
```

5.3.1.16 imax

```
pic.imax = int(parameters['imax'])
```

5.3.1.17 j_loc

```
list pic.j_loc = [ele_j * i for i in range(6)]
```

5.3.1.18 jmax

```
pic.jmax = int(parameters['jmax'])
```

5.3.1.19 length

```
pic.length = int(parameters['length'])
```

5.3.1.20 norm

```
pic.norm = mpl.colors.Normalize(vmin = np.min(arr), vmax = np.max(arr))
```

5.3.1.21 parameters

```
pic.parameters = loaddata.read_parameters()
```

5.3.1.22 pic_name

```
pic.pic_name = './pic/%d * %d grid (Mesh size:  %d * %d) t = %f, boundary_type = %s.png' %
(length, width, imax, jmax, t, boundary_type)
```

5.3.1.23 picnum

```
int pic.picnum = int(sys.argv[1])
```

5.3.1.24 t

```
pic.t = t_out * picnum
```

5.3.1.25 t_out

```
pic.t_out = float(parameters['t_out'])
```

5.3.1.26 text

```
pic.text = plt.text(x = 0.95, y = 0.95, s='t = %.3f s' % (t),fontdict=dict(fontsize=12, color='r',family='monc
```

5.3.1.27 width

```
pic.width = int(parameters['width'])
```

5.3.1.28 x_label

```
list pic.x_label = [ele_x * i for i in range(6)]
```

5.3.1.29 y_label

```
list pic.y_label = [ele_y * i for i in range(6)]
```

Chapter 6

Class Documentation

6.1 read_parameters Class Reference

`#include <read_parameters.h>`

Public Member Functions

- read_parameters ()
- ~read_parameters ()
- string getValue (const string &name)
- void set_splashes ()
- void print_parameters ()

Public Attributes

- ifstream * infile
- double length
- double width
- int imax
- int jmax
- double c
- double dt
- string boundary_type
- bool display_option
- bool output
- double t_end
- double t_out
- double * rs
- double * xs
- double * ys
- int num_splash

6.1.1 Constructor & Destructor Documentation

6.1.1.1 read_parameters()

```
read_parameters::read_parameters ( )
```

Here is the call graph for this function:

6.2 wave_data Class Reference

```
#include <wave.h>
```

Collaboration diagram for wave_data:

Public Member Functions

- wave_data ()
- ~wave_data ()
- void build_OutputType ()
- void setting_paramters ()
- void allocate_grid ()
- void set_initial_condition ()
- void MPI_grid_to_file ()
- void set_Neumman_boundary_u ()
- void set_Neumman_boundary_new_u ()
- void set_Neumman_boundary_old_u ()
- void find_neighbour_for_other_processor ()
- void build_communication_type ()
- void do_communication ()
- void do_iteration ()
- void run_simulation ()

Public Attributes

- double * u
- double * old_u
- double * new_u
- read_parameters parameters
- int imax
- int jmax
- double t_max
- double t = 0.0
- double t_out = 0.0
- double dt_out
- double dt
- double y_max
- double x_max
- double dx
- double dy
- double c
- bool display
- bool output

- int n
- int intdiv
- int mod
- int start_index
- int end_index
- int * x_index
- int * y_index
- int line_feeds
- int line_start
- int line_end
- int n_line
- int isolated_num
- int non_isolated_num
- int * block_lengths
- vector< int > index_connect_up_edge
- vector< int > index_connect_down_edge
- vector< int > index_connect_left_edge
- vector< int > index_connect_right_edge
- int up_edge_length
- int down_edge_length
- int left_edge_length
- int right_edge_length
- double * up_boundaries
- double * down_boundaries
- double * left_boundaries
- double * right_boundaries
- int * send_length
- double ** receive_from_neighbour
- vector< int > index_isolated
- vector< int > index_not_isolated
- bool left_up_corner
- bool right_up_corner
- bool left_down_corner
- bool right_down_corner

6.2.1 Constructor & Destructor Documentation

6.2.1.1 wave_data()

```
wave_data::wave_data ( )
```

6.2.1.2 ~wave_data()

```
wave_data::~wave_data ( )
```

6.2.2 Member Function Documentation

6.2.2.1 allocate_grid()

```
void wave_data::allocate_grid ( )
```

6.2.2.2 build_communication_type()

```
void wave_data::build_communication_type ( )
```

6.2.2.3 build_OutputType()

```
void wave_data::build_OutputType ( )
```

6.2.2.4 do_communication()

```
void wave_data::do_communication ( )
```

6.2.2.5 do_iteration()

```
void wave_data::do_iteration ( )
```

6.2.2.6 find_neighbour_for_other_processor()

```
void wave_data::find_neighbour_for_other_processor ( )
```

Here is the call graph for this function:

6.2.2.7 MPI_grid_to_file()

```
void wave_data::MPI_grid_to_file ( )
```

6.2.2.8 run_simulation()

```
void wave_data::run_simulation ( )
```

6.2.2.9 set_initial_condition()

```
void wave_data::set_initial_condition ( )
```

6.2.2.10 set_Neumman_boundary_new_u()

```
void wave_data::set_Neumman_boundary_new_u ( )
```

6.2.2.11 set_Neumman_boundary_old_u()

```
void wave_data::set_Neumman_boundary_old_u ( )
```

6.2.2.12 set_Neumman_boundary_u()

```
void wave_data::set_Neumman_boundary_u ( )
```

6.2.2.13 setting_paramters()

```
void wave_data::setting_paramters ( )
```

6.2.3 Member Data Documentation

6.2.3.1 block_lengths

```
int* wave_data::block_lengths
```

6.2.3.2 c

```
double wave_data::c
```

6.2.3.3 display

```
bool wave_data::display
```

6.2.3.4 down_boundaries

```
double* wave_data::down_boundaries
```

6.2.3.5 down_edge_length

```
int wave_data::down_edge_length
```

6.2.3.6 dt

```
double wave_data::dt
```

6.2.3.7 dt_out

```
double wave_data::dt_out
```

6.2.3.8 dx

```
double wave_data::dx
```

6.2.3.9 dy

```
double wave_data::dy
```

6.2.3.10 end_index

```
int wave_data::end_index
```

6.2.3.11 imax

```
int wave_data::imax
```

6.2.3.12 index_connect_down_edge

```
vector<int> wave_data::index_connect_down_edge
```

6.2.3.13 index_connect_left_edge

```
vector<int> wave_data::index_connect_left_edge
```

6.2.3.14 index_connect_right_edge

```
vector<int> wave_data::index_connect_right_edge
```

6.2.3.15 index_connect_up_edge

```
vector<int> wave_data::index_connect_up_edge
```

6.2.3.16 index_isolated

```
vector<int> wave_data::index_isolated
```

6.2.3.17 index_not_isolated

```
vector<int> wave_data::index_not_isolated
```

6.2.3.18 intdiv

```
int wave_data::intdiv
```

6.2.3.19 isolated_num

```
int wave_data::isolated_num
```

6.2.3.20 jmax

```
int wave_data::jmax
```

6.2.3.21 left_boundaries

```
double* wave_data::left_boundaries
```

6.2.3.22 left_down_corner

```
bool wave_data::left_down_corner
```

6.2.3.23 left_edge_length

```
int wave_data::left_edge_length
```

6.2.3.24 left_up_corner

```
bool wave_data::left_up_corner
```

6.2.3.25 line_end

```
int wave_data::line_end
```

6.2.3.26 line_feeds

```
int wave_data::line_feeds
```

6.2.3.27 line_start

```
int wave_data::line_start
```

6.2.3.28 mod

```
int wave_data::mod
```

6.2.3.29 n

```
int wave_data::n
```

6.2.3.30 n_line

```
int wave_data::n_line
```

6.2.3.31 new_u

```
double* wave_data::new_u
```

6.2.3.32 non_isolated_num

```
int wave_data::non_isolated_num
```

6.2.3.33 old_u

```
double* wave_data::old_u
```

6.2.3.34 output

```
bool wave_data::output
```

6.2.3.35 parameters

```
read_parameters wave_data::parameters
```

6.2.3.36 receive_from_neighbour

```
double** wave_data::receive_from_neighbour
```

6.2.3.37 right_boundaries

```
double* wave_data::right_boundaries
```

6.2.3.38 right_down_corner

```
bool wave_data::right_down_corner
```

6.2.3.39 right_edge_length

```
int wave_data::right_edge_length
```

6.2.3.40 right_up_corner

```
bool wave_data::right_up_corner
```

6.2.3.41 send_length

```
int* wave_data::send_length
```

6.2.3.42 start_index

```
int wave_data::start_index
```

6.2.3.43 t

```
double wave_data::t = 0.0
```

6.2.3.44 t_max

```
double wave_data::t_max
```

6.2.3.45 t_out

```
double wave_data::t_out = 0.0
```

6.2.3.46 u

```
double* wave_data::u
```

6.2.3.47 up_boundaries

```
double* wave_data::up_boundaries
```

6.2.3.48 up_edge_length

```
int wave_data::up_edge_length
```

6.2.3.49 x_index

```
int* wave_data::x_index
```

6.2.3.50 x_max

```
double wave_data::x_max
```

6.2.3.51 y_index

```
int* wave_data::y_index
```

6.2.3.52 y_max

```
double wave_data::y_max
```

The documentation for this class was generated from the following files:

- wave.h
- wave.cpp

Chapter 7

File Documentation

7.1 animation.py File Reference

Namespaces

- animation

Functions

- def animation.update_grid (int picnum)

Variables

- animation.num_pics = len(glob.glob('./output/∗'))
- animation.parameters = loaddata.read_parameters()
- animation.t_out = float(parameters['t_out'])
- animation.width = int(parameters['width'])
- animation.length = int(parameters['length'])
- animation.imax = int(parameters['imax'])
- animation.jmax = int(parameters['jmax'])
- animation.dt = float(parameters['dt'])
- animation.boundary_type = str(parameters['boundary_type'])
- animation.ele_x = int(length / 5)
- animation.ele_y = int(width / 5)
- animation.ele_i = int(imax / 5)
- animation.ele_j = int(jmax / 5)
- list animation.x_label = [ele_x ∗ i for i in range(6)]
- list animation.y_label = [ele_y ∗ i for i in range(6)]
- list animation.i_loc = [ele_i ∗ i for i in range(6)]
- list animation.j_loc = [ele_j ∗ i for i in range(6)]
- animation.speedratio = int(sys.argv[2])
- animation.fig = plt.figure()
- animation.ax = fig.add_subplot(111)
- animation.fontsize
- animation.text = plt.text(x = 0.95, y = 0.95, s='t = 0 s',fontdict=dict(fontsize=12, color='r',family='monospace',))
- animation.arr = loaddata.load_binary_file(0).reshape((imax, jmax))

- animation.norm = mpl.colors.Normalize(vmin = np.min(arr), vmax = np.max(arr))
- animation.cax = ax.imshow(arr, interpolation='nearest', cmap = 'winter', norm = norm, origin = 'lower')
- animation.cb = fig.colorbar(cax)
- animation.anim = animation.FuncAnimation(fig, update_grid, frames = np.arange(0, num_pics, speedratio), blit=True)
- string animation.ani_name = './anime/%d ∗ %d grid (Mesh size: %d ∗ %d) dt = %f, boundary_type = %s.mp4' % (length, width, imax, jmax, dt, boundary_type)
- animation.fps
- animation.dpi

7.2 loaddata.py File Reference

Namespaces

- loaddata

Functions

- def loaddata.load_binary_file (int i)
- def loaddata.load_array (int i)
- def loaddata.read_parameters ()

7.3 main.cpp File Reference

```
#include "wave.cpp"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

7.3.1 Function Documentation

7.3.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Here is the call graph for this function:

7.4 pic.py File Reference

Namespaces

- pic

Variables

- pic.parameters = loaddata.read_parameters()
- pic.t_out = float(parameters['t_out'])
- pic.width = int(parameters['width'])
- pic.length = int(parameters['length'])
- pic.imax = int(parameters['imax'])
- pic.jmax = int(parameters['jmax'])
- pic.dt = float(parameters['dt'])
- pic.boundary_type = str(parameters['boundary_type'])
- pic.picnum = int(sys.argv[1])
- pic.data = loaddata.load_binary_file(picnum).reshape((imax, jmax))
- pic.ele_x = int(length / 5)
- pic.ele_y = int(width / 5)
- pic.ele_i = int(imax / 5)
- pic.ele_j = int(jmax / 5)
- list pic.x_label = [ele_x ∗ i for i in range(6)]
- list pic.y_label = [ele_y ∗ i for i in range(6)]
- list pic.i_loc = [ele_i ∗ i for i in range(6)]
- list pic.j_loc = [ele_j ∗ i for i in range(6)]
- pic.t = t_out ∗ picnum
- pic.fig = plt.figure()
- pic.ax = fig.add_subplot(111)
- pic.fontsize
- pic.text = plt.text(x = 0.95, y = 0.95, s='t = %.3f s' % (t),fontdict=dict(fontsize=12, color='r',family='monospace',))
- pic.arr = data
- pic.norm = mpl.colors.Normalize(vmin = np.min(arr), vmax = np.max(arr))
- pic.cax = ax.imshow(arr, interpolation='nearest', cmap = 'winter', norm = norm, origin = 'lower')
- pic.cb = fig.colorbar(cax)
- string pic.pic_name = './pic/%d ∗ %d grid (Mesh size: %d ∗ %d) t = %f, boundary_type = %s.png' % (length, width, imax, jmax, t, boundary_type)
- pic.dpi

7.5 read_parameters.cpp File Reference

```
#include "read_parameters.h"
```

Include dependency graph for read_parameters.cpp: This graph shows which files directly or indirectly include this file:

Macros

- #define COMMENT_CHAR '-'

Functions

- bool IsCommentChar (char c)
- void deleteSpace (string &str)
- void TrimSpaces (string &str)
- void TrimMarks (string &str)
- vector< string > extractArray (string &str)

7.5.1 Macro Definition Documentation

7.5.1.1 COMMENT_CHAR

```
#define COMMENT_CHAR '-'
```

7.5.2 Function Documentation

7.5.2.1 deleteSpace()

```
void deleteSpace (
            string & str )
```

7.5.2.2 extractArray()

```
vector<string> extractArray (
            string & str )
```

Here is the call graph for this function:

7.5.2.3 IsCommentChar()

```
bool IsCommentChar (
            char c )
```

7.5.2.4 TrimMarks()

```
void TrimMarks (
            string & str )
```

7.5.2.5 TrimSpaces()

```
void TrimSpaces (
            string & str )
```

Here is the call graph for this function:

7.6 read_parameters.h File Reference

```
#include <ctype.h>
#include <fstream>
#include <iostream>
#include <vector>
```

Include dependency graph for read_parameters.h: This graph shows which files directly or indirectly include this file:

Classes

- class read_parameters

7.7 README.md File Reference

7.8 wave.cpp File Reference

```
#include <mpi.h>
#include <iostream>
#include <sstream>
#include <fstream>
#include <chrono>
#include <cstdlib>
#include <time.h>
#include <vector>
#include <cmath>
#include "wave.h"
#include "read_parameters.cpp"
```

Include dependency graph for wave.cpp: This graph shows which files directly or indirectly include this file:

Functions

- int find_processor (int a)

Variables

- read_parameters parameters
- int out_cnt = 0
- int it = 0
- int num
- int idx
- chrono::high_resolution_clock::time_point start
- chrono::high_resolution_clock::time_point finish
- std::chrono::duration< double > elapsed
- int id
- int p
- int tag_num = 1
- const double zero = 0
- int * start_indexs
- int * num_allocates
- int ** receive_index
- MPI_Datatype * Send_to_neighbour
- MPI_Datatype * Send_to_neighbour2
- MPI_Datatype * Send_to_neighbour3
- MPI_Aint ** Send_to_neighbour_displacements
- MPI_Aint ** Send_to_neighbour_displacements2
- MPI_Aint ** Send_to_neighbour_displacements3
- MPI_Aint ** Send_to_neighbour_address
- MPI_Aint ** Send_to_neighbour_address2
- MPI_Aint ** Send_to_neighbour_address3
- MPI_Request * requests
- int cnt
- vector< int > * index_right_for_processor
- vector< int > * index_left_for_processor
- vector< int > * index_up_for_processor
- vector< int > * index_down_for_processor
- double ** left_1
- double ** left_2
- double ** left_3
- double ** right_1
- double ** right_2
- double ** right_3
- double ** up_1
- double ** up_2
- double ** up_3
- double ** down_1
- double ** down_2
- double ** down_3
- MPI_Datatype UpEdge
- MPI_Datatype OutputType
- MPI_Datatype DownEdge
- MPI_Datatype OutputType2
- MPI_Datatype OutputType3
- MPI_Aint add_start

7.8.1 Function Documentation

7.8.1.1 find_processor()

```
int find_processor (
             int a )
```

7.8.2 Variable Documentation

7.8.2.1 add_start

```
MPI_Aint add_start
```

7.8.2.2 cnt

```
int cnt
```

7.8.2.3 down_1

```
double** down_1
```

7.8.2.4 down_2

```
double** down_2
```

7.8.2.5 down_3

```
double** down_3
```

7.8.2.6 DownEdge

```
MPI_Datatype DownEdge
```

7.8.2.7 elapsed

```
std::chrono::duration<double> elapsed
```

7.8.2.8 finish

```
chrono::high_resolution_clock::time_point finish
```

7.8.2.9 id

```
int id
```

7.8.2.10 idx

```
int idx
```

7.8.2.11 index_down_for_processor

```
vector<int>* index_down_for_processor
```

7.8.2.12 index_left_for_processor

```
vector<int>* index_left_for_processor
```

7.8.2.13 index_right_for_processor

```
vector<int>* index_right_for_processor
```

7.8.2.14 index_up_for_processor

```
vector<int>* index_up_for_processor
```

7.8.2.15 it

```
int it = 0
```

7.8.2.16 left_1

```
double** left_1
```

7.8.2.17 left_2

```
double** left_2
```

7.8.2.18 left_3

```
double** left_3
```

7.8.2.19 num

```
int num
```

7.8.2.20 num_allocates

```
int* num_allocates
```

7.8.2.21 out_cnt

```
int out_cnt = 0
```

7.8.2.22 OutputType

```
MPI_Datatype OutputType
```

7.8.2.23 OutputType2

```
MPI_Datatype OutputType2
```

7.8.2.24 OutputType3

```
MPI_Datatype OutputType3
```

7.8.2.25 p

```
int p
```

7.8.2.26 parameters

```
read_parameters parameters
```

7.8.2.27 receive_index

```
int** receive_index
```

7.8.2.28 requests

```
MPI_Request* requests
```

7.8.2.29 right_1

```
double** right_1
```

7.8.2.30 right_2

```
double** right_2
```

7.8.2.31 right_3

```
double** right_3
```

7.8.2.32 Send_to_neighbour

```
MPI_Datatype* Send_to_neighbour
```

7.8.2.33 Send_to_neighbour2

```
MPI_Datatype* Send_to_neighbour2
```

7.8.2.34 Send_to_neighbour3

```
MPI_Datatype* Send_to_neighbour3
```

7.8.2.35 Send_to_neighbour_address

```
MPI_Aint** Send_to_neighbour_address
```

7.8.2.36 Send_to_neighbour_address2

```
MPI_Aint** Send_to_neighbour_address2
```

7.8.2.37 Send_to_neighbour_address3

```
MPI_Aint** Send_to_neighbour_address3
```

7.8.2.38 Send_to_neighbour_displacements

```
MPI_Aint** Send_to_neighbour_displacements
```

7.8.2.39 Send_to_neighbour_displacements2

```
MPI_Aint** Send_to_neighbour_displacements2
```

7.8.2.40 Send_to_neighbour_displacements3

```
MPI_Aint** Send_to_neighbour_displacements3
```

7.8.2.41 start

```
chrono::high_resolution_clock::time_point start
```

7.8.2.42 start_indexs

```
int* start_indexs
```

7.8.2.43 tag_num

```
int tag_num = 1
```

7.8.2.44 up_1

```
double** up_1
```

7.8.2.45 up_2

```
double** up_2
```

7.8.2.46 up_3

```
double** up_3
```

7.8.2.47 UpEdge

```
MPI_Datatype UpEdge
```

7.8.2.48 zero

```
const double zero = 0
```

7.9 wave.h File Reference

```
#include "read_parameters.h"
```

Include dependency graph for wave.h: This graph shows which files directly or indirectly include this file:

Classes

- class wave_data

Index