# A Sustainable Collaboratory for Coastal Resilience Research

--Shuai Yuan, Steven R. Brandt, Qin Chen, Ling Zhu, and Rion Dooley



# Outline

Introduction

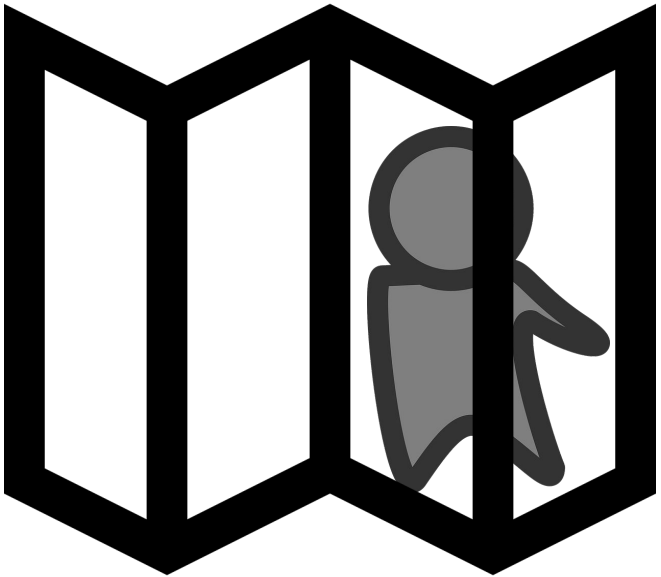
Methodology

Workflow

System

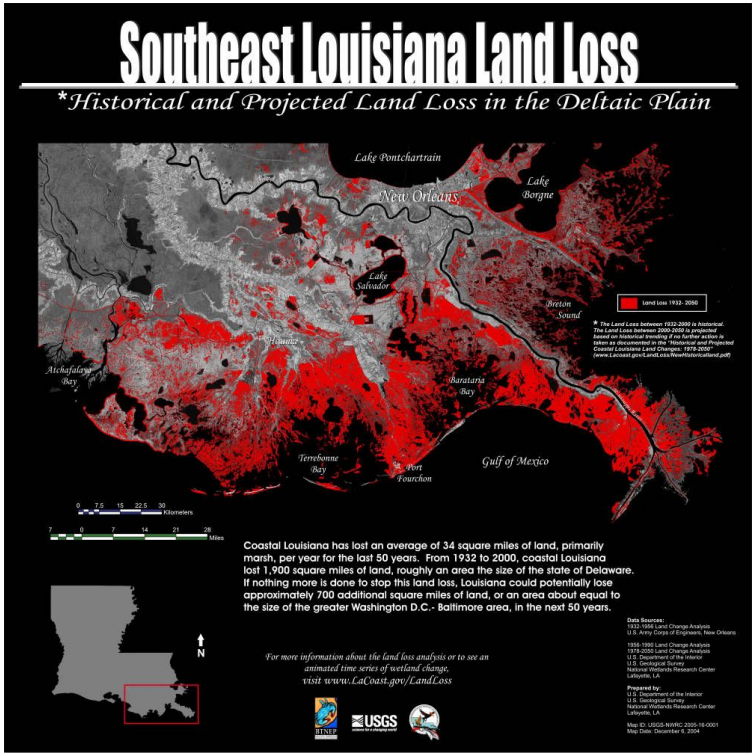
Result

Conclusion



# Introduction

Problem: Communities on modern river deltas are threatened due to land subsidence, global reductions in river sediment, and rising sea levels. It is a grand challenge for earth system science to address these issues.




Example: Projected Mississippi River Delta by 2050

# Introduction

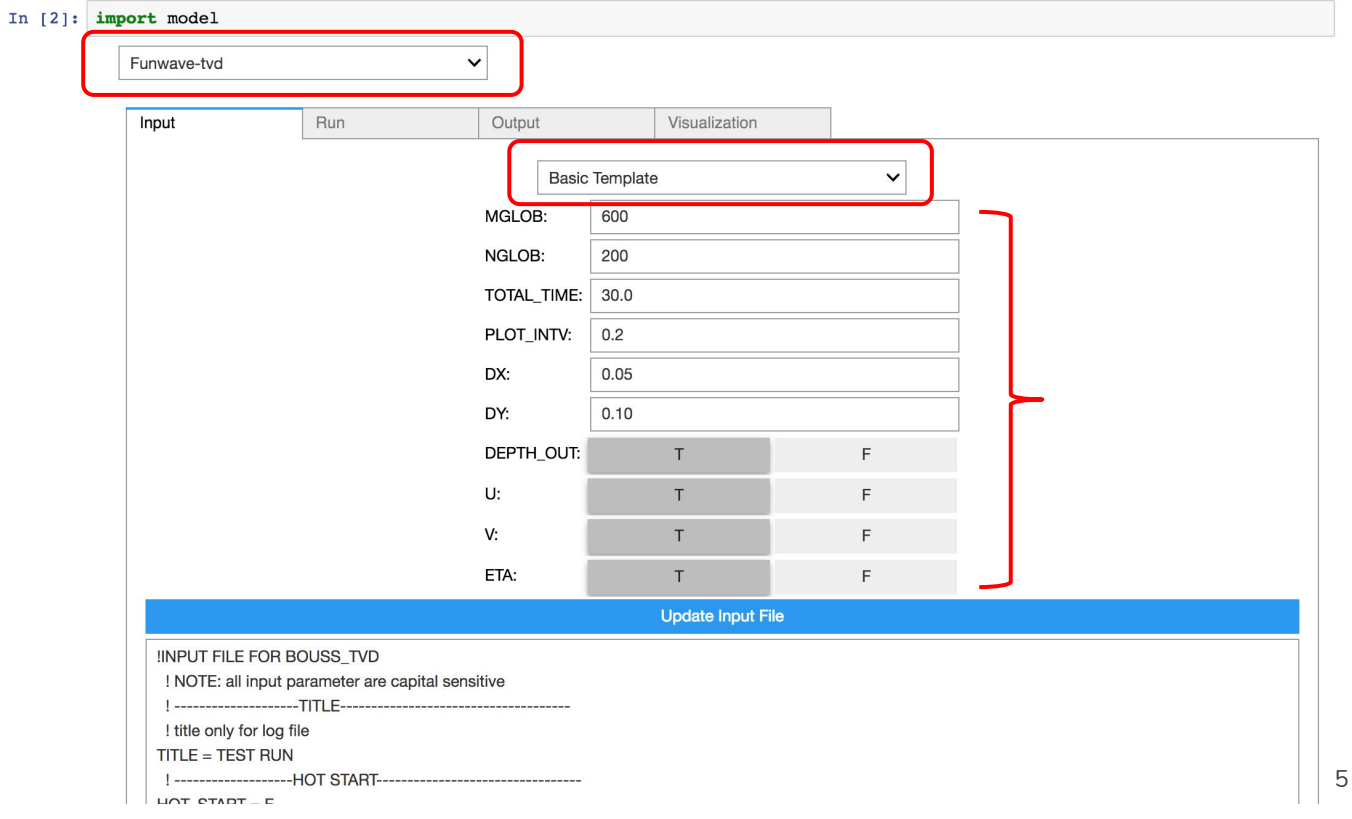
Goal: Create a cloud-ready repository of open-source coastal modeling tools which enable scientists and engineers to more easily use high performance computers (HPC) and to study a variety of physical and ecological processes.

We want our system to be *intuitive*, *repeatable*, *collaborative*

# Introduction

The system is built upon:

1. Jupyter notebooks
2. Agave framework
3. Docker
4. Singularity

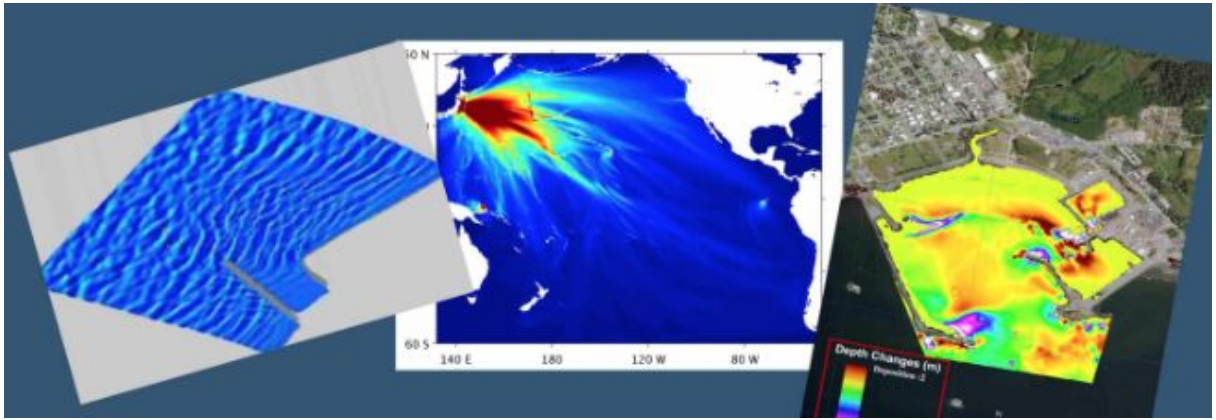


# Introduction

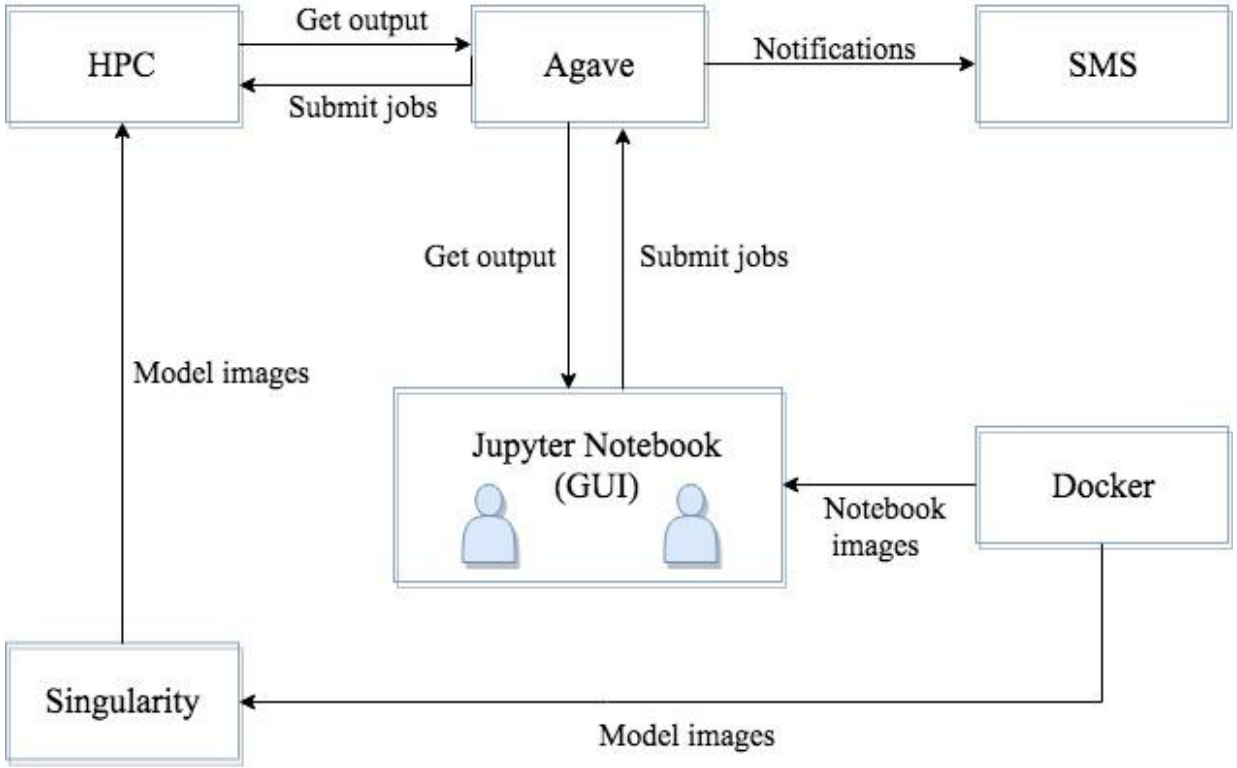
Swan: A third-generation wave model, developed at Delft University of Technology, that computes random, short-crested wind-generated waves in coastal regions and inland waters

# Introduction

Funwave-tvd: A Boussinesq model for simulating nearshore surface-waves, currents and tsunamis from ocean-basin to nearshore scales.



# Workflow


HPC
Get output
Submit jobs
Agave
Notifications
SMS
Get output
Submit jobs
Model images
Jupyter Notebook
(GUI)
Docker
Notebook
images
Singularity
Model images

# Methodology--Docker and Singularity

1. Docker is an open platform allowing developers to build, ship, and run distributed applications in self-contained environments.

   Docker Hub makes it easy to share and discover images.

   Security problem, not available on HPC resources.

2. Singularity makes it easy to use with MPI, and addresses the security concerns.

# Methodology--Docker and Singularity

1. Docker both to construct the images for singularity and to deploy the Jupyter notebooks to use machines and workstations

2. Singularity to deploy and run simulation codes on cloud-enabled resources

# Methodology--Jupyter notebook

1. Jupyter is a tool for interactive computing that is halfway between a GUI and a command-line tool

2. Create a customizable, interactive environment for science discovery and engineering analyses

# Methodology--Agave platform

# Methodology--Agave platform

1. Agave: restful JSON API

   Abstracts a way the details of the job submission process

2. Provides a detailed provenance trail.

3. Fine-grained access controls for sharing jobs and data across machine boundaries.

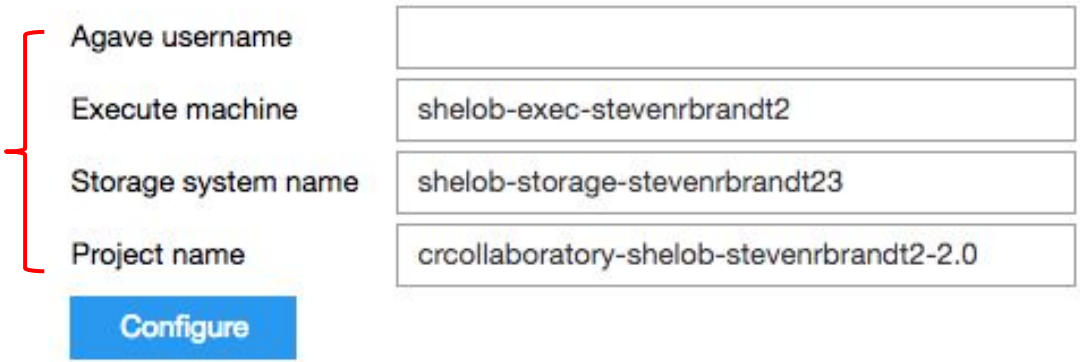
# System--before using

- Configure machine/resource
- Configure user access
- Permissions need to be given

# System--configuration

```
In [1]: import configuration2
```

```
Password or secret: AGAVE_PASSWD
Reading file `AGAVE_PASSWD.txt'
Password or secret: PBTOK
Reading file `PBTOK.txt'
```




```
from agave import *
from ipywidgets import Box, Text, Layout, Label, Button

readpass("AGAVE_PASSWD")
readpass("PBTOK")

item_layout = Layout(
    display = 'flex',
    flex_flow = 'row',
    justify_content = 'flex-start',
    width = '50%'
)

agaveText = Text()
execMachineText = Text(value="shelob-exec-stevenrbrandt2")
machineNameText = Text(value="shelob-storage-stevenrbrandt23")
appText = Text(value="crcollaboratory-shelob-stevenrbrandt2-2.0")
pbtokText = Text()
confBtn = Button(description='Configure', layout= Layout(
    display = 'flex',
    flex_flow = 'row',
    justify_content = 'center',
    width = '100px',
    disabled=False
))

def confBtn_click(a):
    configure2(agaveText.value, execMachineText.value,
               machineNameText.value, appText.value)

confBtn.on_click(confBtn_click)
```

# System--input

```
In [2]: import model
```

Funwave-tvd

Input | Run | Output | Visualization

Basic Template

| | |
|---|---|
| MGLOB: | 600 |
| NGLOB: | 200 |
| TOTAL_TIME: | 30.0 |
| PLOT_INTV: | 0.2 |
| DX: | 0.05 |
| DY: | 0.10 |
| DEPTH_OUT: | T / F |
| U: | T / F |
| V: | T / F |
| ETA: | T / F |

Update Input File

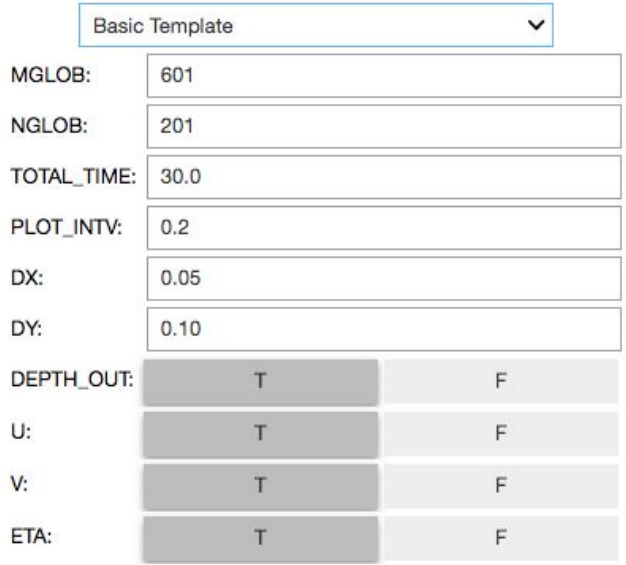
```
!INPUT FILE FOR BOUSS_TVD
  ! NOTE: all input parameter are capital sensitive
  ! --------------------TITLE-------------------------------------
  ! title only for log file
TITLE = TEST RUN
  ! ------------------HOT START---------------------------------
HOT_START = F
```

```
199 ################################## Funwave-tvd Input tab ####################
200
201
202 fwInputdd=Dropdown(options=['Choose Input Template','Basic Template'], value='Choose
203
204
205 parvals = {}
206 inputBox = Box(layout = Layout(flex_flow = 'column'))
207
208 def fw_on_change(change):
209     inputTmp = ''
210     items = []
211     if change['type'] == 'change' and change['name'] == 'value':
212         if(change['new'] == 'Choose Input Template'):
213             items=[]
214             inputBox.children = items
215             return
216         if(change['new'] == 'Basic Template'):
217             cmd("tar -zxvf input_funwave.tgz")
218             inputTmp = 'input_funwave/basic_template.txt'
219
220         with open(inputTmp,"r") as fd:
221             for line in fd.readlines():
222                 g = re.search(r'(\w+)\s*=\s*\${(.*)}',line)
223                 if g:
224                     for match in re.findall(r'(\w+)=("[^"]*"|\'[^\']*|[^,\n]*)',g.gr
225                         if match[0] == 'value':
226                             label = line.split()[0]+'Label'
227                             label = Label(value = line.split()[0].upper()+":")
228                             text = line.split()[0]
229                             text = Text(value=match[1])
230                             box = line.split()[0]+'Box'
231                             box = Box([label, text],layout = Layout(width = '100%',
232                             items.append(box)
233                         if match[0] == 'option':
234                             label = line.split()[0]+'Label'
235                             label = Label(value = line.split()[0].upper()+":")
236                             togBtns = line.split()[0]
237                             togBtns = ToggleButtons(options=['T', 'F'])
238                             box = line.split()[0]+'Box'
239                             box = Box([label, togBtns], layout = Layout(width = '100
240                             items.append(box)
241
242         inputBox.children = items
243
244 fwInputdd.observe(fw_on_change)
245
246 def fwUpInput_btn_clicked(a):
247     inputTmp = ''
248     if(fwInputdd.value == 'Basic Template'):
```

# System--customize input

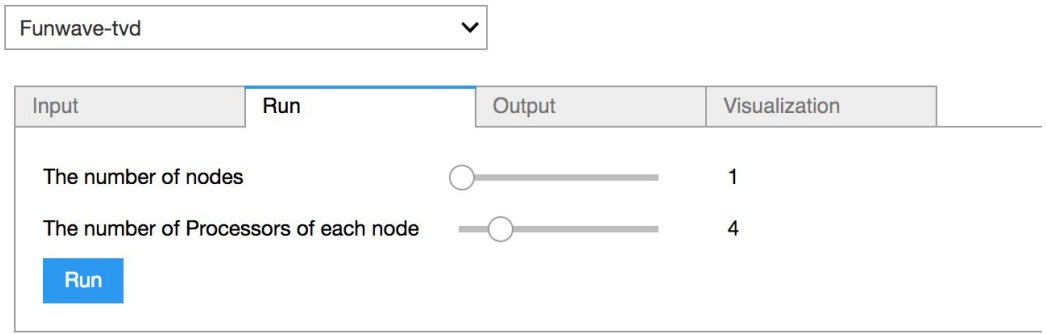
```
  ! ------------------DIMENSION-----------------------------
  ! global grid dimension
Mglob = ${value=601,default=601}
Nglob = ${value=201,default=201}

  ! ----------------- TIME-----------------------------------
  ! time: total computational time/ plot time / screen interval
  ! all in seconds
TOTAL_TIME = ${value=30.0,default=30.0}
PLOT_INTV = ${value=0.2,default=0.2}
PLOT_INTV_STATION = 0.02
SCREEN_INTV = 0.1
HOTSTART_INTV = 360000000000.0
```



# System--run

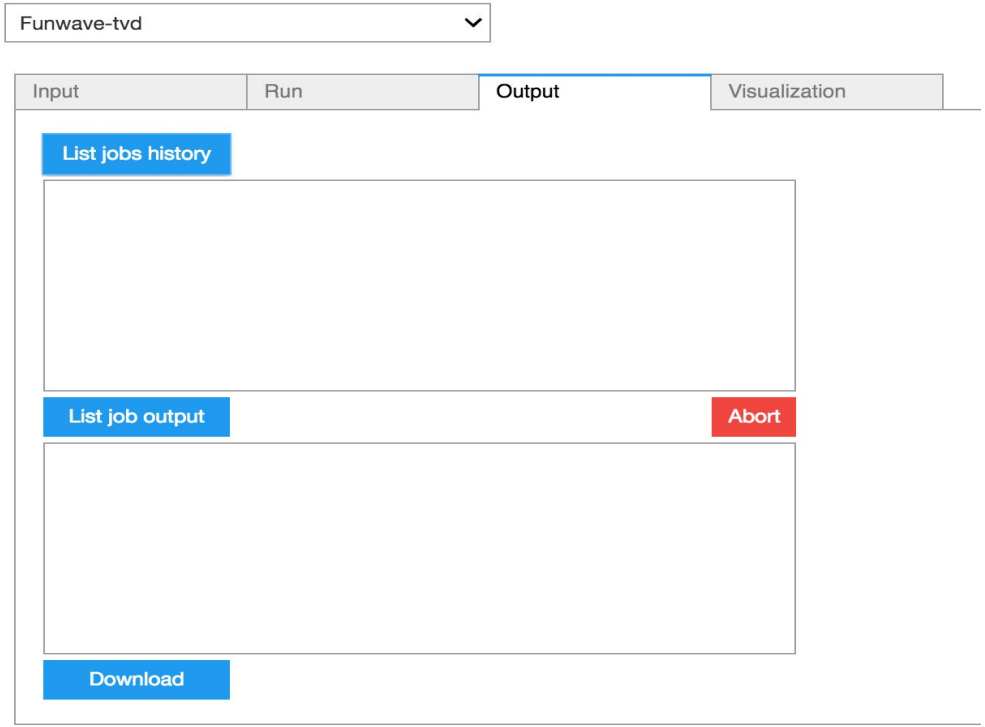
```
In [2]: import model
```




```
285 ################################### Run tab ############################
286 run_item_layout = Layout(
287     display = 'flex',
288     flex_flow = 'row',
289     justify_content = 'flex-start',
290     width = '50%'
291 )
292
293 numnodeSlider = IntSlider(value=0, min=1, max=8, step=1)
294 numprocSlider = IntSlider(value=0, min=1, max=16, step=1)
295
296 runBtn = Button(description='Run', button_style='primary', layout= Layout(width = '5
297
298 run_items = [
299     Box([Label(value='The number of nodes', layout = Layout(width = '350px')), numno
300     Box([Label(value='The number of Processors of each node', layout=Layout(width =
301         layout= run_item_layout),
302     Box([runBtn]),
303 ]
304
305 def modifyFWinput():
306     name_value_pairs = {
307         "PX" : str(numnodeSlider.value),
308         "PY" : str(numprocSlider.value)
309     }
310     with open("input_funwave/input_tmp.txt","r") as tmp:
311         with open("input_funwave/input.txt","w") as inputfile:
312             for line in tmp.readlines():
313                 g = re.match("^(PX|PY)\s*=\s*(\S+)",line)
314                 if g:
315                     name = g.group(1)
316                     if name in name_value_pairs:
317                         inputfile.write(name+" = "+name_value_pairs[name]+"\n")
318                 else:
319                     inputfile.write(line)
320             inputfile.close()
321         tmp.close()
322
323 def runfun_btn_clicked(a):
324     if (modelTitle.value == "Funwave-tvd"):
325         modifyFWinput()
326         cmd("rm -fr input")
327         cmd("mkdir input")
```

# System--output

```
In [2]: import model
```




```
361 ############################## Output tab ##############################
362
363 jobListBtn = Button(description='List jobs history', button_style='primary', layo
364
365 jobSelect = Select(layout = Layout(height = '150px', width='100%'))
366
367 jobOutputBtn = Button(description='List job output', button_style='primary', layo
368
369 abortBtn = Button(description='Abort', button_style='danger', layout= Layout(widt
370
371 outputSelect = Select(layout = Layout(height = '150px', width='100%'))
372
373 downloadOpBtn = Button(description='Download', button_style='primary', layout= La
374
375 def jobList_btn_clicked(a):
376     cout = cmd("jobs-list -l 10")
377     out1 = cout["stdout"]
378     jobSelect.options = out1
379
380 jobListBtn.on_click(jobList_btn_clicked)
381
382 def abort_btn_clicked(a):
383     g = re.match(r'^\S+',jobSelect.value)
384     if g:
385         jobid = g.group(0)
386         rcmd = "jobs-stop "+jobid
387         cmd(rcmd)
388
389 abortBtn.on_click(abort_btn_clicked)
390
391
392 def jobOutput_btn_clicked(a):
393     g = re.match(r'^\S+',jobSelect.value)
```

# System--visualization

```
In [2]: import model
```

Funwave-tvd

Input | Run | Output | Visualization

▾ 1D

Y_in_plots: Choose one

▸ 2D

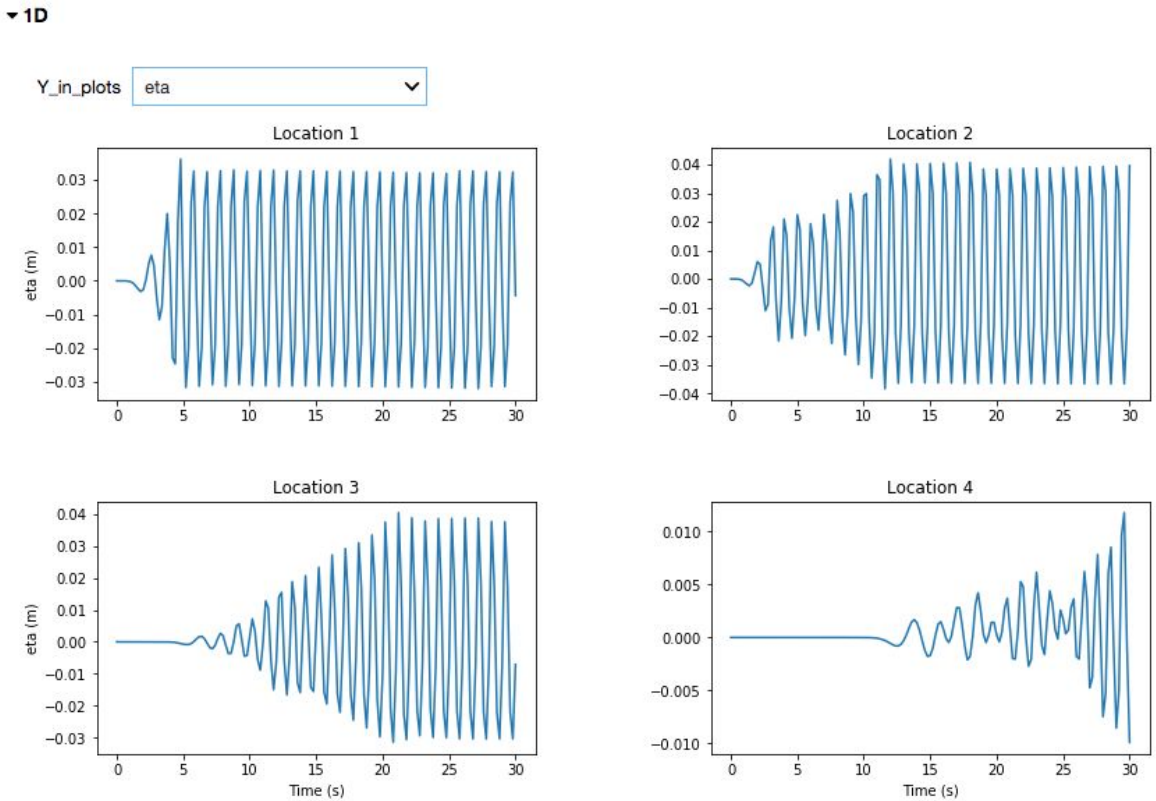
▸ 3D

```
472 ############################ Funwave Visualization tab ###########
473
474
475 fwYoption = Dropdown(options=['Choose one','eta','u','v'])
476 fwplotsInter = widgets.interactive(fwOneD, Y_in_plots = fwYoption)
477 fwoneDBox = Box([fwplotsInter])
478
479
480 depthBtn = Button(description='display',button_style='primary', layout
481 depthOutput = widgets.Output()
482
483
484
485 def depth_Btn_clicked(a):
486     print (fw_para_pairs['Mglob'])
487     print (fw_para_pairs['Nglob'])
488     with depthOutput:
489         display(waterDepth(fw_para_pairs['Mglob'],fw_para_pairs['Nglob
490
491 depthBtn.on_click(depth_Btn_clicked)
492 depthBox = Box([Label(value='Water Depth'),depthBtn],layout = Layout(w
493
494
495 depProfileN = IntSlider(value=0, min=0, max=200)
496 depProfileInter = widgets.interactive(depProfile, N = depProfileN)
497 depProfileBox = Box([Label(value='Depth Profile Snapshot'), depProfile
498
499
500
501 depProfileAnimaRange = FloatRangeSlider(value=[5,7], min=0.0, max=30,
502                                         description='Time period (s):',readou
503 depProfileAnimBtn = Button(description='display',button_style='primary
504 depProfileAnimOutput = widgets.Output()
505 def depProfileAnim_Btn_clicked(a):
506     anim = depProfileWithEta(depProfileAnimaRange.value[0], depProfile
```

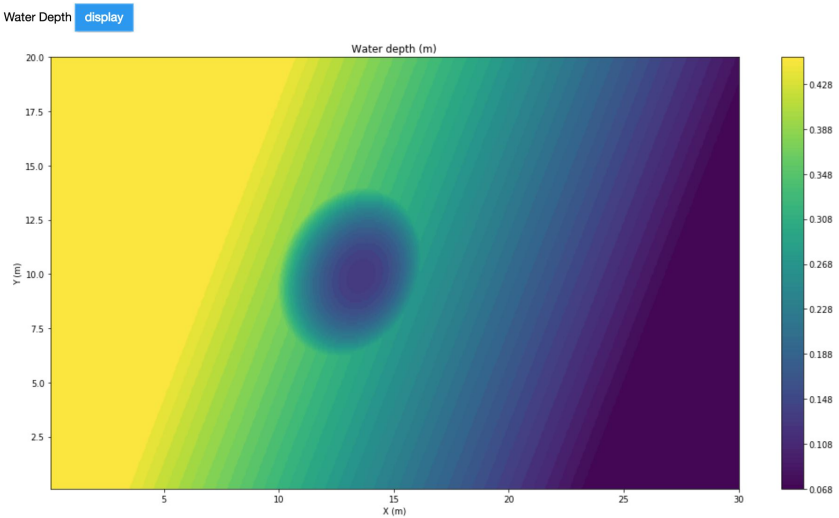
# System--Future Work

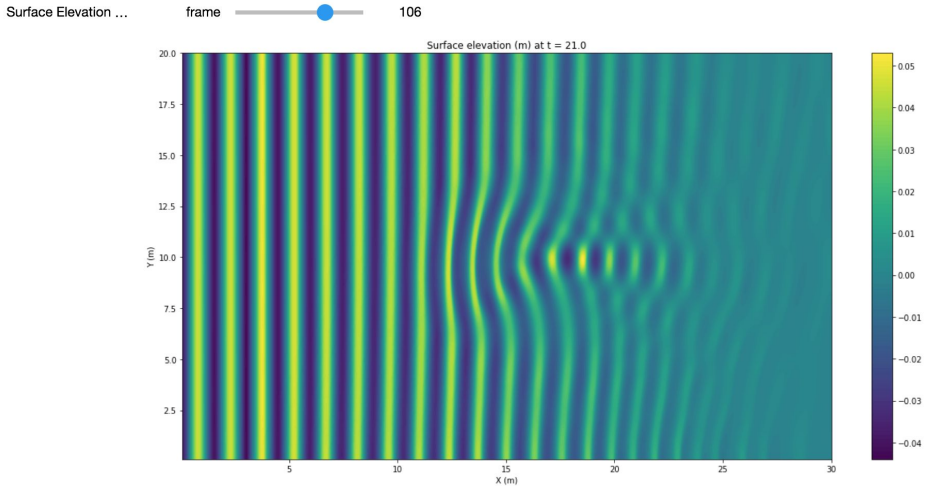
- Update OS, MPI, and source code for Funwave-tvd or Swan once a month, automatically (in progress).
- Intend to add more models
- …...

# Results--1D


1D
Y_in_plots
eta
Location 1
Location 2
Location 3
Location 4
eta (m)
Time (s)

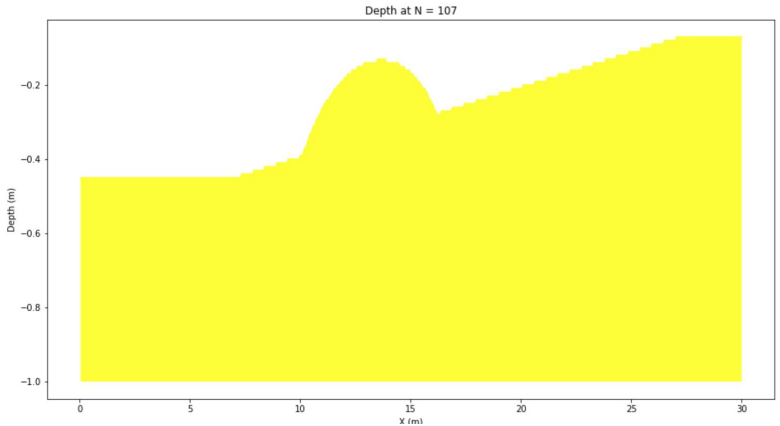
# Results--vertical 2D



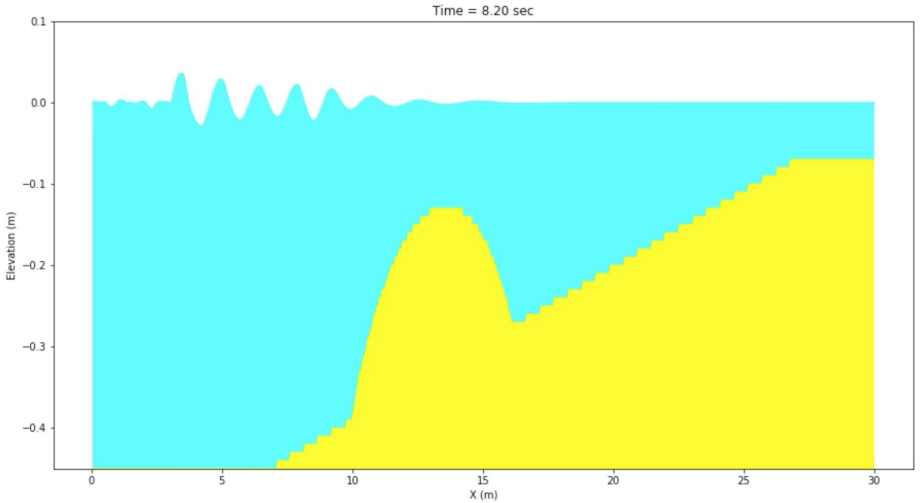



# Results--horizontal 2D

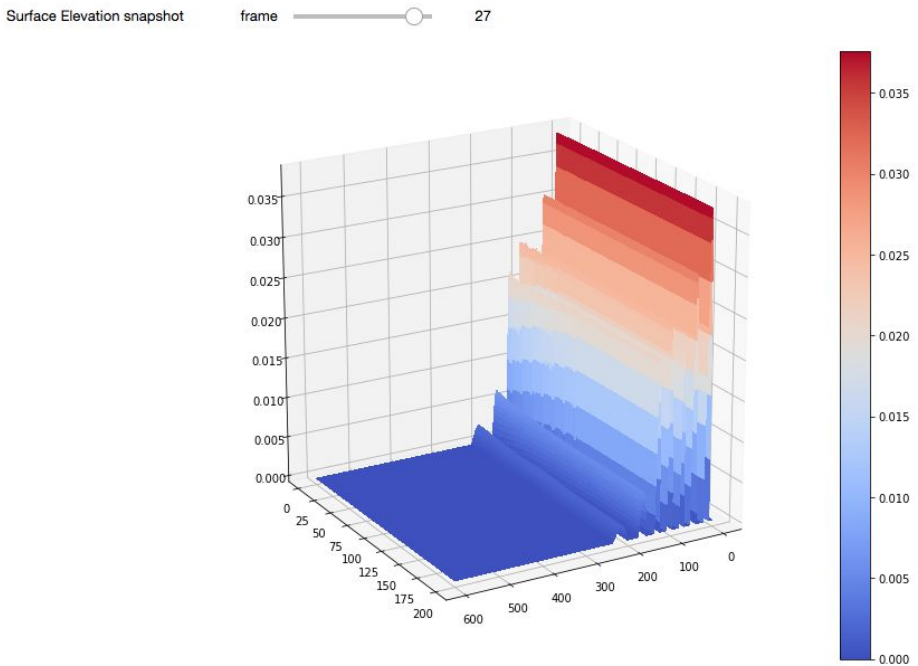
Depth Profile Sna… N 107




Cross-shore profile animation Time perio… 5.00 – 20.40 display



# Results--3D


Surface Elevation snapshot
frame
27
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
25
50
75
100
125
150
175
200
600
500
400
300
200
100
0
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000

# Conclusion

The CMR (Coastal Model Repository)

- is targeting cloud and cloud-like architectures to enable quick deployment of coastal models and their working environment
- serve as a community repository for precompiled open source models that are widely used by coastal researchers

# Thanks!