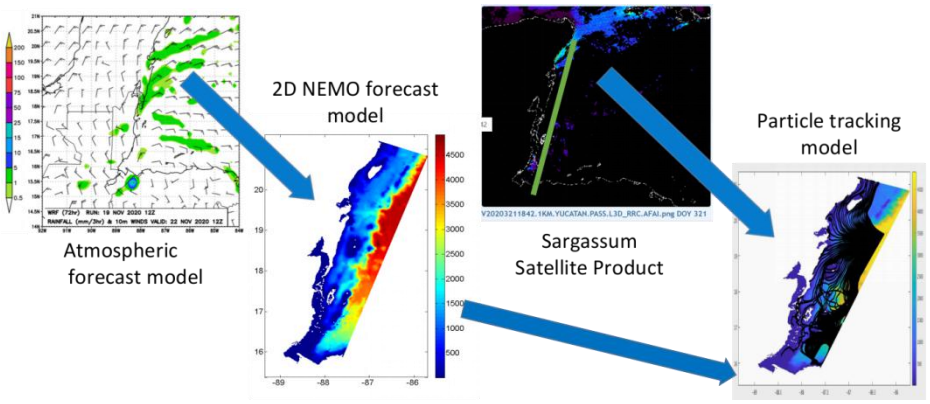
Sargassum Forecast

Sargassum Forecast using NEMO surge and Ocean Parcels.

version 0.1.0



2021 Edition

Running a NEMO surge model and particle tracking module in an operational framework

National Oceanography Centre

TABLE OF CONTENTS

Sargassum Forecasting Documentation

Introduction/Description

This manual details the framework that is used to run an Sargassum Forecasting product, briefly this comprises of:

- a containerised NEMO surge model
- particle tracking python module
- Sargassum satellite product
- python script workers to undertake specific tasks
- java-script process manager to start and stop the worker scripts

These requirements are met by undertaking the following work-flow:

- download the latest atmospheric forcing at defined intervals
- process forcing into NEMO compatible netcdf
- run NEMO surge model using docker/podman container
- watch NEMO model and QA output
- download the latest sargassum locations at defined intervals
- generate seed location list
- run open parcels module that takes seed locations and NEMO current output and returns particle tracks within a netcdf file.
- plots a defined number of tracks on to a map and saves as PNG
- clean up logs, NEMO run directory and old outputs.

This system will start at a set interval using cron (process scheduler), creating new data and outputs to be interpreted by end users. The source workers, (download weather and get sargassum) determine when the process starts as both need to successfully download data to be able to run the model. The other workers are secondary and make regular checks (default 10 minutes) to see if there is new data or if the previous worker has run successfully.

Changelog:

0.1.0: First release

Installation

Sargassum Forecasting has the following requirements:

- containerisation framework: docker
- python package manager: miniconda or anaconda
- nodejs package manager: npm
- cron service running

The framework has been designed and tested on Linux, both Fedora and Ubuntu have been used. Once the requirements are in place the framework can be installed as follows:

Clone the repository (best practise is from user home directory) and switch to correct branch

```
$ git clone https://github.com/ocgabs/BLZ_SURGE
$ git checkout surge-container
```

The repository needs to be renamed due to filename parsing requirements, until then please rename manually as follows:

```
$ mv BLZ_SURGE BLZ-SURGE
```

Navigate to the BLZ-SURGE main directory and build the python environment using conda (select yes if asked to confirm):

```
$ conda env create -f environment.yml
$ conda activate BLZ-SURGE
```

Install the process manager PM2:

```
$ sudo npm install pm2@latest -g
```

A useful monitoring program to install is ctop, while it only workers for docker containers, once installed it allows the monitoring of the container. If using podman this does not work but running “podman events” in the terminal” will provide a feed showing container events e.g. start stop etc. There is also the possibility of using cockpit to provide a website based interface but this is all outside the scope of this manual.

Before first use

Once the python environment and process manager are installed the framework is ready to go, however the system requires some initial setup steps to reflect the new install location, these are:

- Populate INPUTS folder
- Amend configuration files
- Generate weighting files
- set processor options

Populating INPUTS folder

The NEMO model requires a number of input files for it to successfully run. It is not possible to put this in the github repository, so they need to be added after it is cloned. The current requirements for files are:

- bathy_meter.nc
- coordinates.nc
- coordinates.bdy.nc
- domain_cfg.nc
- tidal forcing files.

If any of these are missing then the model will likely fail.

Amend configuration files

The configuration files in the config folder will need updating with the relevant paths as by default they are setup for the user “thopri” in the home directory. For simplicity, it is best to clone the repository into the users home directory and then the only change required is to update the user name in the defined paths in the configuration files.

In the ecosystem yml file, every worker entry has an cwd variable, this should be updated to reflect the current user. In the nowcast yaml files most paths will need the user name updating. Easiest way is to search for the default user “thopri” and replace with user on installation system. **NOTE:** the container name is prefixed with the name thopri but this is required to locate the container image on the docker repository so must be kept.

Generating weighting files

NEMO needs weighting files so it can map the atmospheric forcing onto its grid. There is a worker that can generate these files. So the process to generate is as follows, run the download_weather and process_forcing workers manually to generate some input to generate the weightings. Then run the generate_weights worker within the NEMO surge container as the relevant tools to create are stored inside.

```
$ python workers/download_weather.py config/nowcast.yaml
$ python workers/process_forcing.py config/nowcast.yaml -f
```

```
$ docker run --rm -v /path/to/BLZ-SURGE:BLZ-SURGE thopri/nemo-surge-blz:8814 python
/BLZ-SURGE/workers/generate_boundary.py /BLZ-SURGE/config/nowcast.yaml
```

This should run all the steps to generate the weighting files. If successful the weighting files will appear in the weights folder within the main directory.

Changing number of processors

By default the container runs with 7 model processors. This can be changed to suit the architecture the model is running on. E.g. the workstation used for initial testing runs has 10 Cores and with hyper-threading 20 logical processors. Setting the container to use 12 NEMO processors resulted in a 100% utilisation of the system. Some trial and error maybe required for other setups.

The file to set the cores is called run_surge.sh and is located in the RUN_NEMO directory. The line to change is as follows:

```
mpirun -n 12 ./nemo.exe
```

where the number value is the number of NEMO cores to use. Set these values to one appropriate for your system. E.g. for an 4 core/ 8 processor machine, the starting point would be 7 NEMO cores. Different variations may result in quicker run times. **NOTE:** some core counts are not stable, currently the configuration is not stable on 2, 4 or 6 cores. But will run on 1, 3 or 7 cores.

Once these steps are complete the framework should be ready to start.

Usage

At this point the framework is installed and should have all the files needed and be configured correctly. It is important that all commands are run from the conda environment created previously so ensure that the environment is loaded as follows:

```
$ conda activate BLZ-SURGE
```

It can be started as follows (from the BLZ-SURGE directory):

```
$ pm2 start config/ecosystem.yml
```

To monitor the system PM2 has some terminal tools:

```
$ pm2 monit
$ pm2 logs
$ pm2 status
```

Running each of these commands in a separate terminal will give information on the system, monit is an interactive dashboard, logs shows the log files as they are written in real time and status returns the current status of PM2 (which processes are running etc).

The system can be stopped with the following (again from the BLZ-SURGE main directory):

```
$ pm2 stop config/ecosystem.yml
```

Finally the processes can be removed from PM2 by:

```
$ pm2 delete config/ecosystem.yml
```

Individual workers can be stopped, started and deleted in the same way:

```
$ pm2 start download_weather
$ pm2 stop download_weather
$ pm2 reload download_weather
$ pm2 delete download_weather
```

Logs for individual workers can be inspected as follows:

```
$ pm2 logs download_weather
```

This will show both standard output and also errors (stdout and stderr). It is useful for debugging a single process as the main log feed can get overwhelming with the constant writing from each worker to it.

To monitor the exit codes of the workers, the log of the PM2 process can be scrutinized using:

```
$ pm2 logs PM2
```

To see further back as by default only 15 lines for each log are shown the --lines flag can be used. e.g.

```
$ pm2 logs PM2 --lines 50
```

For the last 50 lines of the PM2 logs.

It is also sensible to install a log rotate module for PM2 that ensures log files don't get excessively large. This can be installed as follows:

```
$ pm2 install pm2-log-rotate
```

Periodically the PM2 logs will need to be flushed, this process removes all the old logs reclaiming disk space. At the moment there is no automatic method to this. To flush the logs:

```
$ pm2 flush
```

Individual workers can also be run using python command (from the BLZ-SURGE main directory), e.g.

```
$ python workers/download_weather.py config/nowcast.yaml
```

Some worker will require an -f flag, this is a force flag that overrides the workers checking to see if it needs to be run by comparing exit codes of the previous worker and itself. The workers that need the -f flag are:

- process_forcing
- run_nemo
- watch_nemo
- find_seed
- run_parcels
- plot_output

All other workers don't rely on a previous worker output so do not need to check to see if it has successfully completed. **NOTE:** using force flag when previous worker has not ran successfully will result in unexpected behaviour.

When needed docker will pull and run the container but it can be pulled beforehand with this command:

```
$ docker pull thopri/nemo-surge-blz:8814
```

Scheduling

While most of the workers just look for the success of the previous worker and start based on that. However two workers need to be started at a defined interva as they provide the new data required to undertaken a new run. These are the download weather and get sargassum workers, both of these workers will start automatically with the pm2 start ecosystem.yml command but once complete unlike the other workers they do not get restarted. This functionality is implemented using Cron and the setting up of a cronjob. This is a scheduling process within Linux. To add entries the user needs to create a crontab file.

```
$ crontab -e
```

This opens the crontab and the following entry can be added.

```
30 8 * * * /usr/local/bin/pm2 start download_weather > /home/thopri/BLZ-SURGE/logs/cron-weather.log
30 8 * * * /usr/local/bin/pm2 start get_sargassium > /home/thopri/BLZ-SURGE/logs/cron-sar.log
```

This starts the two workers at 8:30 am, saving the terminal output to a log file. If a different time is desired the user can amend the first two values. The first being the minutes and second is the hours, e.g. 30 8 is 8:30 am.

This is essential to starting the framework operationally as without it the system will only run the model once.

To get email notifications, adding the following to the top of the crontab should result in emails being sent showing the worker starting.

```
MAILTO=example@email.com
```

If a cronjob does not start then troubleshooting will need to take place, common issues are the cron itself is not running or the crontab commands having incorrect syntax.

Worker Details

Download Worker:

This worker downloads the latest pressure and wind data from the NOAA GFS hourly model, it uses an grib filter to only download the specific region required. The download consists of hourly grib files, and the worker downloads the latest midnight run. Each hourly file is checked before downloading to ensure all files are present. If the current day forecast is not up then the worker will download the previous days forecast. If there are errors in retrieving the data then this is logged by the process manager. The downloaded data is saved to the grib folder in the main directory.

While there is code to download other daily runs, the model config only supports the midnight run, so the worker is locked to download this run only.

Process forcing Worker:

The downloaded data is a set of hourly grib files, (120 of them) whereas NEMO expects an netcdf file. This worker converts the grib files in the grib folder and saves the converted netcdf file to the netcdf folder in the main director. The worker ensures that the format of the netcdf file is compatible with the NEMO model.

Generate boundary worker:

This is the only worker that is not automatically run by the process manager, it only needs to be run once for each domain to generate the weighting files needed by NEMO to apply forcing data to its boundaries.

It uses the surge container as it also has the NEMO tool-set on board. The command to generate the weighting files is:

```
$ docker run -rm -v /home/username/BLZ-SURGE thopri/nemo-surge-blz:8814 python worker/generate_boundary.py config/nowcast.yaml
```

This will run the weighting tool within the NEMO surge and generate the required weighting netcdf files. This requires the files in namelist folder and also saves the weighting files in the weights folder.

Once this is completed then it will not need to be re run unless the model domain is modified which also requires the container to be rebuilt.

Run NEMO worker:

This worker prepares NEMO to run, this requires changes to the namelist file, such as updating the forcing file names, the start date of the run. What the starting time step is etc. Once complete it then starts the NEMO surge container and exits.

Watch NEMO worker:

Once the container is successfully running, this worker watches it and checks its output for errors. It checks for container start, once it sees the container running it will monitor the time.step file for progress and print that progress to screen or log file. Once the container has stopped running, this worker will undertake the following checks to ensure the run was successful.

Tests include:

- does results directory exist?
- does the output.abort.nc file exist? (indicates NEMO aborted)
- does the time.step file exist?
- has the expected number of times steps been run?
- Are there E R R O R flags in the NEMO log file ocean.output?
- Does the solver.stat file exist?
- Are there NaN values in solver.stat?

If all these tests pass then the model is considered to have successfully run, and the worker exits.

Get Sargassum worker:

The second worker that acts as an origin point and gets data, this worker downloads the latest sargassum product, this is currently a 7 day composite density plot and is saved to the sargassum folder in the main directory.

Find seed worker:

This worker uses the downloaded sargassum product to generate seed locations based on areas of ‘lots’ of sargassum. The worker saves the locations as CSV files in the sargassum folder.

Run Parcels worker:

This worker takes the checked results from the NEMO run, which are saved in the OUTPUTS folder within the sub-folder netcdf. It also ultilises the locations saved in the sargassum folder. Using the NEMO currents and the seed locations it runs the Ocean Parcels module and outputs a netcdf with tracks for each particle. This is saved in the OUTPUTS folder, within the sub-folder tracks.

Plot Tracks worker

This worker takes the track output netcdf file from the run parcels worker located in the OUTPUTS/tracks folder and plots the tracks onto a map. As there are so many tracks a random subset of tracks is used. (default number is 500). The worker outputs a PNG image in the OUTPUTS folder located in the sub-folder plots.

The extent of the map is determined by the extent of Sargassum locations. This means that the extent is not always the same.

Clean Up Worker

This worker ensures that files are trimmed to ensure that file space requirements are met, this deletes output files older than a set threshold (default 7 days) and also trims the restart files for NEMO when the are no longer needed and any log files created by workers.

Configuration

Sargassum Forecasting has two config files, one to configure the workers (nowcast.yaml) and one to configure the process manager PM2 (ecosystem.yml). These files should not need changing aside from updating the paths to reflect the location of the main BLZ-SURGE directory on the system.

Nowcast YAML

This file contains all the user changeable parameters for the framework, this is mostly paths of where to find inputs or where to put outputs, or model specific info e.g container image name. Each worker configuration is independent from the other, the worker only reads its specific section. While this makes the file modular and changes in one section won't impact on the other workers it also results in a lot of duplication as many workers need the same parameters, e.g. container image name. So if the user changes a parameter they need to ensure all the duplicated parameters are changed as well.

It is planned that future development will change the layout so that these parameters are not repeated. The downside is that workers will be reading in several shared configuration sections so changes to one worker could impact on the other.

Ecosystem YML

This configuration file is used by PM2 to define what processes to start and what parameters they require, e.g. restart delay. While processes can be started on the command line individually, it is much easier and more repeatable to use a configuration file. The user only has to execute one command to start the framework.

Status Codes

Each worker has the same success return code, the table below shows the set of codes that workers can return. The three main codes are 0 for successful working, 1 for workers that rely on a previous workers output, shows that the previous worker has not yet logged a successful result. Finally a status code of 2 denotes that there is no new work to be done, i.e. the previous worker has not yet successfully created new work. The other codes are exit codes for a specific reason why the worker was not able to complete its task. Table 1 below details the codes, reasons and workers it applies too.

Table 1: Current worker exit codes.

Status Code:	Worker:	Reason for Code:
0	All	Worker under took work successfully
1	PF,RN,WN,FS,RP,PT	Previous worker has not yet run successfully
2	PF,RN,GS,RP,PT,	No new work from previous worker
3	DW,GS	Unable to download data
4	GS	Unable to geo transform sargassum image
5	WN	NEMO container not running
6	RP	U and V files do not have same start and end dates
7	CU	NEMO container still running
8	WN	Model Run failed QA checks

Worker Key: DW: Download_weather, PF: process_forcing, GW: generate_weighting, RN: run_nemo, WN: watch_nemo, GS: get_sargassum, FS: find_seed, RP: run_parcels, PT: plot_tracks, CU: clean_up

In addition to exit codes, the framework prints lots of status messages to the screen when issues occur and these will be saved by PM2 logging. If there issues, these messages should give an idea of the problem. They can be read using the PM2 logs command as detailed in the usage section.

Known Issues

Any known issues with the framework will be entered here.

Advanced Configuration

Changing Container Framework

Changing the container framework, if say the user is using Podman rather than Docker requires the following changes to the configuration file.

container: docker to container: podman
container_name: thopri/nemo-surge-blz:8814 to container_name: docker.io/thopri/nemo-surge-blz:8814

The container variable is located in the run_nemo section and the container name is located in the run_nemo, watch_nemo and clean_up sections. Please see Appendix for highlighted changes.

Appendix

Appendix A – Description of Repository Directory Layout

The BLZ-SURGE repository has the following directories:

- archive: old, deprecated scripts stored here
- BUILD_NEMO: this is where the NEMO container is built
- BUILD_PARCELS: this is where the Ocean Parcels container is built (not currently implemented)
- config: config files such as nowcast and ecosystem are stored here
- grib: downloaded atmospheric grib files are stored here
- INPUTS: NEMO model config files are stored here e.g. bathymeter.nc etc
- logs: log output from cron and containers is stored here
- namelist: template and scripts to create weighting files
- netcdf: forcing netcdf files processed from grib are stored here
    - GFS: Sub folder for source netcdf data
- OUTPUTS: outputs from framework are stored here
    - tracks: Trajectories from ocean parcels are stored here
    - netcdf: NEMO output is stored here
    - plots: plots created are stored here
- RUN_NEMO: the directory that NEMO is run from
- RUN_PARCELS: the directory the ocean parcels is run from
- sargassum: downloaded sargassum images and seed locations are stored here
- weights: generated weights files are stored here
- workers: python worker scripts are stored here.

Many of these folders are empty in the repository as they will contain input data when downloaded or require models or workers to be run before holding anything. If any of these locations is missing then the folders will need creating for the framework to run without errors.

Appendix B - Example configuration file

```
# System configuration file for Nowcast system

#Download Weather worker configuration
weather:
 download:
   # West Longitude limit
     leftlon: -90
   # East Longitude limit
     rightlon: -82
   # North Latitude limit
     toplat: 22
   # Bottom Latitude limit
     bottomlat: 15
   # varaible 1
     var1: var_MSLET
   # variable 2
     var2: var_UGRD
   # variable 3
     var3: var_VGRD
   # Level to download data (10 m above ground)
     lev1: lev_10_m_above_ground
   # Level to download data (msl)
     lev2: lev_mean_sea_level
   # URL template from which to download NOAA NCEP Grid filter files for GFS model
     url_template: 'https://nomads.ncep.noaa.gov/cgi-
bin/filter_gfs_0p25_1hr.pl?file={file_name}{hours}&{lev1}=on&{lev2}=on&{var1}=on&{var2}=on&{var3}=on&su
bregion=&leftlon={leftlon}&rightlon={rightlon}&toplat={toplat}&bottomlat={bottomlat}&dir=%2{run_time}%2Fa
tmos'
   # Template for GFS model file name
   # **Must be quoted to project {} characters**
     file_template: 'gfs.t{model_run}z.prgb2.0p25.f'
   # Number of hours requested in the forecast Goes upt 120 (I think)
     forecast_hrs: 120
   # Destination directory for downloaded GFS model GRIB data files
     dest_dir: '/home/thopri/BLZ-SURGE/grib/'
     status_dir: /home/thopri/BLZ-SURGE/config/

#Forcing process worker configuration
forcing:
 process:
   #number of forecast hours should be the same as in weather download
     forecast_hrs: 120
   #variable one
     var1: u10
   #variable description for netcdf file
     var1_des: 10 metre U wind component
   #varaible two
     var2: v10
   #variable description for netcdf file
     var2_des: 10 metre V wind component
   #variable three
     var3: msl
```

```
#    #variable description for netcdf file
#       var3_des: Pressure reduced to MSL
#    #Template of GRIB file names
#       file_template: '202011170000_arw_wrfout_d02.grb2'
    #variable description for netcdf file
       var3_des: MSLP (Eta model reduction)
    #Template of GRIB file names
       file_template: 'gfs.t{model_run}z.pgrb2.0p25.f'
    #Destination directory of NetCDF files
       netcdf_dest_dir: '/home/thopri/BLZ-SURGE/netcdf/GFS/'
    #Grib downloaded files location
       grib_dir: '/home/thopri/BLZ-SURGE/grib/'
    #Config name NOTE: do NOT use - in the name as it will break parsing of file names
       netcdf_name: BLZ-SURGE
       config_dir: /home/thopri/BLZ-SURGE/config/
       pm2log: /home/thopri/.pm2/pm2.log

#Generate Boundary worker configuration
generate:
  boundary:
    #name of template bilinear namelist file
       template_1: 'namelist_reshape_bilin_atmos_template'
    #name of template bicubic namelist file
       template_2: 'namelist_reshape_bicubic_atmos_template'
    #namelist file name for bilinear
       file_out_1: 'namelist_reshape_bilin_atmos'
    #namelist file name for bicubic
       file_out_2: 'namelist_reshape_bicubic_atmos'
    #keys to identify populate fields in namelist templates
       input_key_1: 'input_key_1'
       input_key_2: 'input_key_2'
    #destination directories for netcdf, namelist files and output weighted files
       netcdf_dest_dir: '/BLZ-SURGE/netcdf/'
       namelist_dir: '/BLZ-SURGE/namelist/'
       weights_dir: '/BLZ-SURGE/weights/'

#Run Nemo worker configuration
RUN_NEMO:
  # NEMO run duration
       duration: 120  # hours
  # Directory where files (e.g. namelist.time) and symlinks
  # (e,g. iodef.xml) necessary to prepare the NEMO runs are stored
       config_dir: /home/thopri/BLZ-SURGE/RUN_NEMO/
  #weighting file location
       weights_dir: /home/thopri/BLZ-SURGE/weights/
  #netcdf file location
       netcdf_dir: /home/thopri/BLZ-SURGE/netcdf/GFS/
  #location of boundary data for model
       boundary_dest: /home/thopri/BLZ-SURGE/RUN_NEMO/fluxes/
  #location of restart files
       restart_dir: /home/thopri/BLZ-SURGE/RUN_NEMO/Restart_files/
  #name of namelist template file
       namelist_template: 2_namelist_cfg_template
```

```
 #name of populated namelist file for model
      pop_namelist: namelist_cfg
 #name of variables for namelist file
      var1: .*u10
      var2: .*v10
      var3: .*msl
 #Time step of model in seconds.
      time_step : 1 #in seconds (6 mins)
 # Name of the NEMO configuration;
      config_name: BLZ-SURGE
  # Use restart file or not
      restart: false
 # restart interval (same as atmo model frequency)
      restart_int: 24 #hours
 # Name of the model domain coordinates file.
 # It is assumed to be in the EXP00/ directory of the NEMO configuration
 # directory composed from the "NEMO code" and "config name" keys above.
      coordinates: coordinates.nc
## Directory where run results are stored
      results archive: /home/thopri/BLZ-SURGE/RUN_NEMO/
 # Which container framework to use
      container: docker
      container_name: thopri/nemo-surge-blz:8814
      container_mount: /home/thopri/BLZ-SURGE/
      container_dir: BLZ-SURGE
 # Time to start model, starting at midnight fails due to no previous day data
      hour_start: '01' # 1 am
 # where to save container terminal output in log file
      container_log: /home/thopri/BLZ-SURGE/logs/
      status_dir: /home/thopri/BLZ-SURGE/config/
      pm2log: /home/thopri/.pm2/pm2.log

#Watch NEMO worker configuration
WATCH_NEMO:
   #location of model results directory
      results_dir: /home/thopri/BLZ-SURGE/RUN_NEMO/
   #duration of model run
      duration: 120 #Hours
   #time step of model
      time_step: 1 #In seconds (7.5 mins)
   #name of NEMO model configuration
      config_name: BLZ-SURGE
   # name of container to watch
      container_name: thopri/nemo-surge-blz:8814
      status_dir: /home/thopri/BLZ-SURGE/config/
      WATCH_INTERVAL: 1 # minutes
      pm2log: /home/thopri/.pm2/pm2.log
      output_dir: /home/thopri/BLZ-SURGE/OUTPUTS/netcdf
      file_parse:  'BLZ-SURGE_1h_*.nc'

#Make Plots NEMO worker configuration
make:
 plots:
```

```
  #location of station list file
    station_dir: /SRC/config/
  #location to save processed image plots both spatial and timeseries
    plots_dir: /SRC/output/plots/
  #location to save images to make gif animation
    animation_plots_dir: /SRC/output/plots/animation_plots/
  #location of output NETCDF directory
    netcdf_dir: /SRC/output/netcdf/
  #location of model outputs
    model_output_dir: /SRC/NEMOGCM/CONFIG/TEST/EXP00/
  #location to save CSV output
    csv_dir: /SRC/output/csv/
  #model time step
    time_step: 1 #In seconds (6 mins)
  #total time of model run (hours)
    total_time: 120
  #interval for spatial plots e.g. every 12 hours
    interval: 10 #timesteps 10 time steps at 6 mins = 1 hour
  #global threshold to check across whole domain (currently not active)
    global_thres: 0.2
  #name of station list file
    station_list: station_list.txt
  #name of NEMO model configuration
    config_name: BLZ-SURGE

#Seeding Open Parcels worker configuration
GET_SAR:
  # URL
    url_template: 'https://optics.marine.usf.edu/subscription/modis/YUCATAN/{year}/comp/'
  # Produce plots? For QA and setup purposes.
    plot: False
  # projection for transforming data or giving it a projection?
    crs: '+proj=longlat'
  # limits of the domain
    north: 22
    south: 15
    east: -82
    west: -90
  # sargassum levels? note values are > or <
    min: 0
    max: 252
  # bathy meter location
    bathy_meter: '/home/thopri/BLZ-SURGE/INPUTS/BLZE12_bathy_meter.nc'
    coordinates: '/home/thopri/BLZ-SURGE/INPUTS/BLZE12_coordinates.nc'
  # directory to store downloaded product
    dest_dir: '/home/thopri/BLZ-SURGE/sargassium/'
  # name of GeoTiff image (processed from downloaded image)
    tiff_name: 'SargassoT7.tif'
  # string to parse correct image, i.e. unique part of filename
    parse_ID: 'DENSITY.png'
    status_dir: /home/thopri/BLZ-SURGE/config/

FIND_SEEDS:
```

```
    # name of GeoTiff image (processed from downloaded image)
    tiff_name: 'SargassoT7.tif'
    # directory to store downloaded product
    dest_dir: '/home/thopri/BLZ-SURGE/sargassium/'
    # Produce plots? For QA and setup purposes.
    plot: False
    bathy_meter: '/home/thopri/BLZ-SURGE/INPUTS/BLZE12_bathy_meter.nc'
    file_parse:  'C*DENSITY.png'
    pm2log: /home/thopri/.pm2/pm2.log
    status_dir: /home/thopri/BLZ-SURGE/config/
    sband_max: 252
    sband_min: 0

RUN_PARCELS:
      lon_csv: '/home/thopri/BLZ-SURGE/sargassium/ilon.csv'
      lat_csv: '/home/thopri/BLZ-SURGE/sargassium/ilat.csv'
      data_path: '/home/thopri/BLZ-SURGE/OUTPUTS/netcdf/'
      ufile_parse: 'BLZ-SURGE_1h_*U.nc'
      vfile_parse: 'BLZ-SURGE_1h_*V.nc'
      file_parse:  'BLZ-SURGE_1h_*.nc'
      grid_file: '/home/thopri/BLZ-SURGE/INPUTS/BLZE12_coordinates.nc'
      out_file: 'nBelize_nemo_sargaso_particlesT2'
      out_dir: '/home/thopri/BLZ-SURGE/OUTPUTS/tracks/'
      status_dir: '/home/thopri/BLZ-SURGE/config/'
      pm2log: /home/thopri/.pm2/pm2.log

PLOT_TRACKS:
      pm2log: /home/thopri/.pm2/pm2.log
      out_dir: '/home/thopri/BLZ-SURGE/OUTPUTS/plots/'
      plot_name_template:
      input_file_template: 'nBelize_nemo_sargaso_particlesT2.nc'
      input_dir: '/home/thopri/BLZ-SURGE/OUTPUTS/tracks/'
      num_particles: 500
      set_extent_N: 0  #how much to increase extent over model domain by in degrees North
      set_extent_S: -1 #how much to increase extent over model domain by in degrees South
      set_extent_E: 0 #how much to increase extent over model domain by in degrees East
      set_extent_W: -1 #how much to increase extent over model domain by in degrees West
#Clean up worker configuration
clean:
    up:
  #location of model run directory
      run_dir: /home/thopri/BLZ-SURGE/RUN_NEMO/
  #location of log files
      forcing_dir: /home/thopri/BLZ-SURGE/netcdf/
  #location of output files
      out_dir: /home/thopri/BLZ-SURGE/OUTPUTS/
  #name of model configuration
      config_name: BLZ-SURGE
  #location of restart files
      restart_dir: /home/thopri/BLZ-SURGE/RUN_NEMO/Restart_files/
  #Runtime model files to delete after successful run
      del_1: layout.dat
      del_2: namelist_cfg
```

```
   del_3: ocean.output
   del_4: output.namelist.dyn
   del_5: solver.stat
   del_6: time.step
#number of days to keep output files
   num_days: 7
# number of days to keep container logs
   log_days: 7
#sub directories for forcing files to trim
   sub_1: GFS/
   sub_2: MBLZ/
#sub directories for log files to trim
   log_dir: /home/thopri/BLZ-SURGE/logs/
# Sargassium dir to trim
   sargassium_dir: /home/thopri/BLZ-SURGE/sargassium/
#subdirectories for processed model output
   out_1: tracks/
   out_2: netcdf/
   out_3: plots/
   container_name: thopri/nemo-surge-blz:8814
```

Appendix C - Example ecosystem file

#Ecosystem config file for PM2 to run the components of the NEMO Sargassium Operational Forecast System

apps:

```
 - script       : ./workers/download_weather.py
   name         : 'download_weather'
   cwd          : '/home/thopri/BLZ-SURGE'
   args         : '-- config/nowcast.yaml'
   watch        : false
   autorestart    : false
   log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

 - script       : ./workers/process_forcing.py
   name         : 'process_forcing'
   cwd          : '/home/thopri/BLZ-SURGE'
   args         : '-- config/nowcast.yaml'
   watch        : false
   restart_delay  : 600000
   log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

 - script       : ./workers/run_nemo.py
   name         : 'run_nemo'
   cwd          : '/home/thopri/BLZ-SURGE'
   args         : '-- config/nowcast.yaml'
   watch        : false
   restart_delay  : 600000
   log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

 - script       : ./workers/watch_nemo.py
   name         : 'watch_nemo'
   cwd          : '/home/thopri/BLZ-SURGE'
   args         : '-- config/nowcast.yaml'
   watch        : false
   restart_delay  : 600000
   log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

 - script       : ./workers/get_sargassium.py
   name         : 'get_sargassium'
   cwd          : '/home/thopri/BLZ-SURGE'
   args         : '-- config/nowcast.yaml'
   watch        : false
   autorestart    : false
   log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'
```

```
- script        : ./workers/find_seed.py
  name          : 'find_seed'
  cwd           : '/home/thopri/BLZ-SURGE'
  args          : '-- config/nowcast.yaml'
  watch         : false
  restart_delay   : 600000
  log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

- script        : ./workers/v5Belize_Parcels.py
  name          : 'run_parcels'
  cwd           : '/home/thopri/BLZ-SURGE'
  args          : '-- config/nowcast.yaml'
  watch         : false
  restart_delay   : 600000
  log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

- script        : ./workers/PlotParcelsOutput.py
  name          : 'plot_tracks'
  cwd           : '/home/thopri/BLZ-SURGE'
  args          : '-- config/nowcast.yaml'
  watch         : false
  restart_delay   : 600000
  log_date_format : 'YYYY-MM-DD HH:mm:ss.SSS'

- script: ./workers/clean_up.py
  name: 'clean_up'
  cwd: '/home/thopri/BLZ-SURGE'
  args          : '-- config/nowcast.yaml'
  watch: false
  restart_delay: 86400000
  log_date_format: 'YYYY-MM-DD HH:mm:ss.SSS'
```