Tides

Introduction

This document reports the steps carried out to run `OTPSnc` on Zeus and produce the tides required for the Frontex area.

Requirements

To produce the tides on Zeus through OTPS we will need:

- `makeDat.py` to feed the model with lat/lon/timestep data;
- The model `OTPSnc`

Around these set of tools, a wrapper has been developed in order to split the processing of a large area on 2 deg x 2 deg squares.

Why 2x2?

A performance analysis on the `OTPSnc` model was carried out to determine the best way to split our area.

- OTPSns on a single point takes a few seconds
- Area 1x1: 1 min
- Area 1x2: 3 min
- **Area 2x2: 5 min**
- Area 2x3: 10 min
- Area 3x3: more than 1 hour

Given that, 2x2 is the best trade off between the number of processes spawned and the length of the computation.

The makeDat script

Purpose

It is a python script developed to produce the lat/lon/timestep file needed for `OTPSnc` to run.

Running makeDat

To invoke it on Zeus (user `ov-dev`), the `py38` conda environment can be exploited. On the `ov-dev` account, the `makeDate.py` script can be found in `$HOME/OTPS_tide`.

The required input params are:

- `dx`: zonal step
- `dy`: meridional step
- `startDate`: in the format `YYYYMMDD`
- `endDate`: same as the previous
- `outname`: name of output file
- `boundingBox`: specified as `x0,y0,x1,y1`

Then, to invoke the script:

```
$ python ~/OTPS_tide/makeDat.py --startDate=20211101 --endDate=20211102 --outname=
tides.txt --boundingBox=0,0,100,100 --dx=0.03 --dy=0.03
```

Curiosity...

Before discovering the extreme slowness of `OTPSnc` , running over the whole Frontex domain with a resolution of 0.033 has been hypothesised. Therefore, a test to determine the size of the lat/lon/timestep file varying the size of the domain or the resolution have been carried out:

- Bbox -90,-90,90,90 (world) and step 0.1: 2 GB
- Bbox -90,-90,90,90 (world) and step 0.033: memory allocation error
- Bbox -43,11,43,73 (frontex) and step 0.1: 380 k
- Bbox -43,11,43,73 (frontex) and step 0.033: 900 k

The OTPS model

Compiling the model

The first step is to retrieve the model. In the present use case, the model was copied from `/data/opa/sanifs/static_files/OTPSnc/OTPSnc/` to `/work/opa/ov-dev/OTPS_tide/OTPSnc/`. After that, it was compiled running the script `/users_home/opa/ov-dev/OTPS_tide/compileOTPS.sh`. The resulting binaries can be found under: `/work/opa/ov-dev/OTPS_tide/OTPSnc/bin`

Running the model

Use case #1 — Our "*hello world*"

Let' start from a simple use case: running the model over a single area. First of all, the lat/lon/timestep file must be generated. Let's assume the resulting file called `latlontime`

Then, a setup file must be created following this template found in the documentation:

```
DATA/Model_ES2008           ! 1. tidal model control file
lat_lon_time1               ! 2. latitude/longitude/<time> file
```

```
z                           ! 3. z/U/V/u/v
                            ! 4. tidal constituents to include
AP                          ! 5. AP/RI
geo                         ! 6. oce/geo
1                           ! 7. 1/0 correct for minor constituents
sample.out                  ! 8. output file (ASCII)
```

After that, the model can be run. The model itself is composed by two executables to be run in sequence: `extractHC` and `predict_tide`. The latter invokes the first, so the execution may start by calling `predict_tide`. Done. The output is an ASCII file, in this case called `sample.out`.

Use case #2 — Parallel execution over a wide area

Now that the basic way of running the model has been introduced, the parallel scenario can be presented.

Of course, in order to run `OTPSnc` over the Frontex area, there is the need to:

1. subdivide the area in 2x2 squares (1408 resulting subdomains)
2. create 1408 lat/lon/timestep files
3. create 1408 setup files
4. spawn 1408 instances of the model
5. convert the output files from ASCII to NetCDF
6. merge all the NetCDF files to obtain the final NetCDF file over the whole area.

The final setup

The entry point

All these steps above to run `OTPSnc` on a wide area are performed by the script `invokePredict.sh` placed in the `bin` folder of the `OTPSnc` model and created ad hoc. The only required parameter is the date with the `YYYYMMDD` format, for example:

```
$ sh invokePredict.sh 20211120
```

It will produce an output folder called `YYYYMMDD` (in this case `20211120`). It will also create a working directory called `work_YYYYMMDD` that can be deleted or not, depending

on the settings in `tidesGen.conf` file.

The config file

An example `tideGen.conf` config file is reported here:

```
# clean? 1 = yes, 0 = no
CLEAN=0

# parallelism level
NJOBS=300

# conda environment
CONDA_ENV=/work/opa/${USER}/py38

# paths
SCRIPT_PATH=/work/opa/${USER}/OTPS_tide/OTPSnc/bin
SCRIPT_EXE=${SCRIPT_PATH}/predict_tide
WORK_PATH=/work/opa/${USER}/OTPS_tide/OTPSnc/bin/work_$REQDATE
OUTPUT_PATH=/work/opa/${USER}/OTPS_tide/OTPSnc/bin/$REQDATE
MAKEDAT=$HOME/OTPS_tide/makeDat.py

# modules
MODULES="intel19.5/netcdf/C_4.7.2-F_4.5.2_CXX_4.3.1 curl/7.66.0 intel19.5/19.5.281 int
el19.5/szip/2.1.1 intel19.5/hdf5/1.10.5"

# lat/lon ranges
MIN_LON=-43
MAX_LON=43
MIN_LAT=10
MAX_LAT=72
LAT_STEP=2
LON_STEP=2

# LSF settings
LSF_PROJECT=0496
LSF_QUEUE=p_short
```

From this config file it is possible to configure the number of jobs to spawn simultaneously, the paths of every component and input/output directories as well as the boundaries of the domain.

All the steps in detail

It's time to proceed with a more detailed analysis of the previous steps:

1. The lat/lon/timesteps are produced (of course) using `makeDat.py` that is invoked with `bsub`. The output files follow the nomenclature `frontex_<LAT>_<LON>` where `LAT` and `LON` are those of the bottom-left corner of the subdomain.
2. The setup files follow the nomenclature `setup_frontex_<LAT>_<LON>`. They follow the usual template. The only tidal components to include are `m2,s2,k1,o1,n2`.
3. The *n* model instances will produce the output ASCII files
4. Every ASCII file is then process to:
    a. Fix all the rows related to land points, otherwise it would be impossible to process them
    b. Create a file with all the latitudes and one for the longitudes
    c. Create the output NetCDF file
5. Then all the NetCDF files are merged using python and xarray.
6. Finally the cleaning (if requested through `tideGen.conf`) is performed.

Crontab setup

Every day at 00.00, a script is called. It is named `genTides.sh` and it basically:

- Checks that all the files expected for the current day are available (so all the tides for the time interval [-7, +7]. If something is missing, it starts the model.
- Produces four additional days with respect to the last available
- Deletes all the folders older than DAY-7.

Git repository

The code developed to wrap the `OTPSnc` model is hosted on GitHub:

```
$ git clone https://github.com/fabioviola-cmcc/TidesGeneration
```