

ENS DE LYON, UNIVERSITÉ LYON1

RAPPORT DE MASTER 2

---

**Mise en place d'un modèle numérique  
hydrodynamique de l'anse du Cul-de-Loup  
(Normandie, France) Opérateur expédition  
produits finis**

---

*Auteure :* Mme A. BONVALET

*Tuteur :* M. E. POIZOT

# Table des matières

<b>1</b>	<b>INTRODUCTION</b>	<b>2</b>
<b>2</b>	<b>ZONE D'ÉTUDE</b>	<b>3</b>
<b>3</b>	<b>MATÉRIELS ET MÉTHODES</b>	<b>4</b>
3.1	Le code de calcul CROCO . . . . .	4
3.1.1	Présentation générale . . . . .	4
3.1.2	Description du pre-processing . . . . .	4
3.1.3	Post processing . . . . .	6
3.2	Données de terrain - Le projet PROTEC . . . . .	7
<b>4</b>	<b>RÉSULTATS</b>	<b>8</b>
4.1	Le modèle "Anse du Cul-de-Loup" . . . . .	8
4.1.1	Données brutes utilisées . . . . .	8
4.1.2	Paramètres généraux choisis . . . . .	8
4.1.3	Augmentation de la résolution . . . . .	9
4.2	Résultats (sortie de modèle) . . . . .	10
<b>5</b>	<b>DISCUSSION</b>	<b>11</b>
<b>6</b>	<b>CONCLUSION</b>	<b>12</b>
<b>7</b>	<b>MATÉRIELS ET MÉTHODES</b>	<b>13</b>
7.1	Le code de calcul CROCO . . . . .	13
7.1.1	Processus généraux . . . . .	13
7.1.2	Détail des processus du pre-processing . . . . .	18
7.1.3	Détail des processus du modèle . . . . .	24
7.2	Le modèle "Anse du Cul-de-Loup" . . . . .	24
7.3	Mesures de terrain - Le projet PROTEC . . . . .	24

# 1 INTRODUCTION

A faire presque à la fin, mais les grandes idées seront :

- changement climatique, pression anthropique : besoin de gérer l'espace côtier
- modèles numériques, outils pour réaliser divers scénarios pour une bonne gestion
- anse du cul-de-loup (ADCL) : un bon exemple de nécessité de gestion raisonnée
- utilisation du code de calcul CROCO pour réaliser un modèle numériques de ADCL

L'activité de culture d'huître est une source importante de revenus pour certaines régions des côtes françaises (Haut de France, Normandie, Bretagne, Pays de la Loire, Charente-Maritime, Aquitaine). Pour la Normandie, le département de la Manche et ses 670 km de côtes produit plus de 15 000 tonnes d'huîtres par an. Le tourisme est un autre secteur d'activité important de nombreuses communes côtières. Ainsi, avec la prise de conscience de la nécessaire préservation des milieux naturels, de nombreuses zones côtières telles que l'anse du Cul-de-Loup, ont vu des mesures de sauvegarde mises en place, via la création de sites Natura 2000, les classements ZNIEFF<sup>1</sup> (1 et 2) ou encore les directives habitat faune et flore. Malgré ces initiatives de préservation d'un milieu naturel fragile, l'anse du Cul-de-Loup connaît une dégradation de son environnement, avec une disparition progressive de certaine communauté d'algue protégées (la Zostère et Spatine marine), ainsi qu'un envasement de plus en plus important au détriment de la couverture sableuse originale.

---

1. Zones naturelles d'Intérêt Écologique, Faunistique et Floristique

## 2 ZONE D'ÉTUDE

Depuis le XVIII<sup>e</sup> siècle, le Cotentin abrite sur ses côtes des activités conchyliicoles [Kopp2000]. Tout d'abord artisanale, cette culture consistait en un reparquage d'huître plates locales. À partir du XX<sup>e</sup> siècle (début des années 60), la conchyliculture normande connaît une phase d'expansion importante et devient plus industrielle. De nouvelles zones côtières sont investies pour implanter les concessions ostréicoles et des aménagements sont réalisés pour faciliter l'implantation. L'anse du Cul-de-Loup (Normandie, Manche, France, Fig. 1) est un exemple typique de cette évolution au cours du temps (Fig. 2).

D'une superficie avoisinant les 370 ha, l'anse du Cul-de-Loup n'est ouverte que dans sa partie sud sur la baie de Seine. La fermeture complète de l'anse, à l'est, a été réalisée au cours du XVII<sup>e</sup> siècle par la construction d'une route-digue qui rejoint le continent depuis St-Vaast-la-Hougue à l'ex île de la Hougue (Fig. 1). L'ensemble de l'anse découvre plus ou moins en fonction des coefficients de marée (marée semi-diurne), avec un marnage d'environ 5 m en marée de vive eau. Le zéro hydrographique est situé juste à l'entrée de l'anse parallèlement à la ligne côtière. L'altitude maximum dans l'anse ne dépasse pas 3 m au dessus du zéro hydrographique.

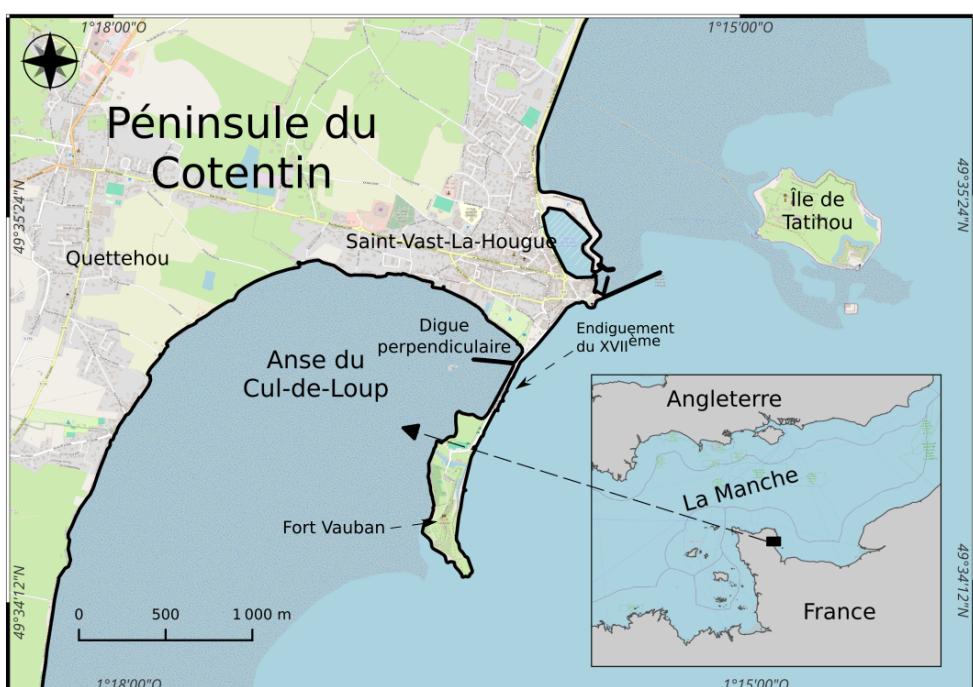


FIGURE 1 – L'anse du Cul-de-Loup.

Le vent est régulièrement présent sur la zone du Cul-de-Loup, avec une moyenne de 18-19 km/h l'été majoritairement de secteur ouest et de 25-30 km/h en période hivernale avec un partage plus homogène des directions. L'agitation de surface dans l'anse est de type mer du vent, due au frottement par le vent à la surface de l'eau. L'orientation de l'anse du Cul-de-Loup la protège des houles les plus fréquentes en Baie de Seine, qui sont de secteurs NW-NE essentiellement. Les houles de secteur est et sud-est sont les seules à éventuellement pouvoir pénétrer dans l'anse, mais elles sont à la fois plus rares et de faible leur énergie. Ces dernières sont rapidement dissipées par la présence des structures ostréicoles et pénètrent donc peu à l'intérieur de l'anse. A l'est au niveau de la bordure interne de l'Île de la Hougue, certaines houles peuvent néanmoins diffractées et être canalisées le long de l'île avec une énergie significative<sup>11</sup>.

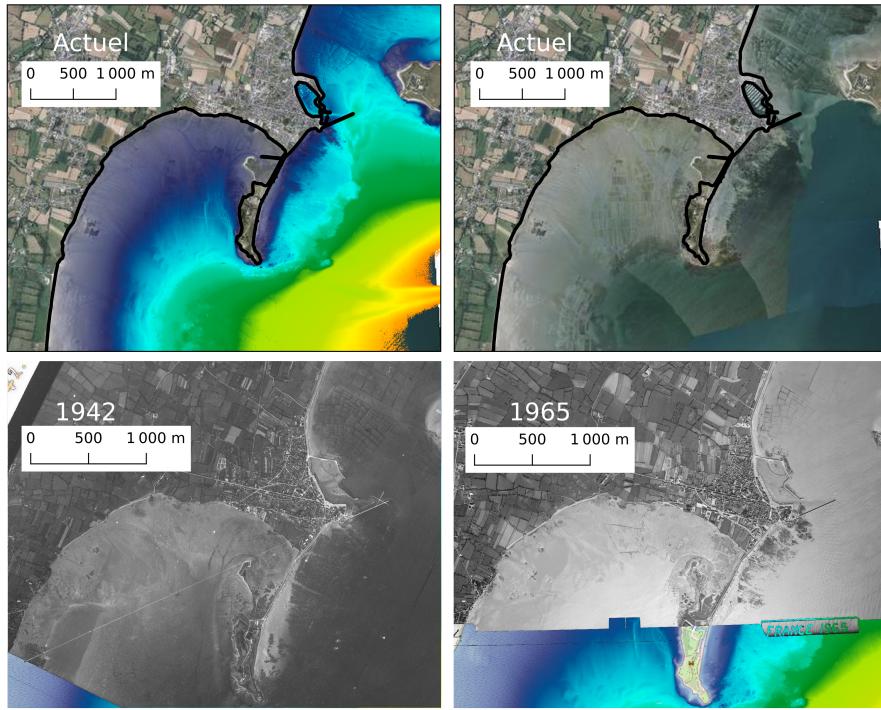


FIGURE 2 – En bas à gauche, image aérienne de 1942, l'anse du Cul-de-Loup ne contient aucun parc ostréicole. En bas à droite, image aérienne de 1965, l'anse du Cul-de-Loup commence à être aménagée dans sa partie nord. En haut à gauche et à droite, respectivement la morpho-bathymétrie et une image aérienne actuelle de l'anse du Cul-de-Loup montrant de nombreuses concessions ostréicoles.

### 3 MATÉRIELS ET MÉTHODES

#### 3.1 Le code de calcul CROCO

##### 3.1.1 Présentation générale

Mettre les principes généraux de CROCO (origine, type de modèle, etc...)

Présentation du fonctionnement (d'une partie) de CROCO (schéma du time stepping de la doc ?).

Présentation des modes hydro utilisés (RANS, LES)

Le code de calcul hydrodynamique CROCO fonctionne sur la base d'un ensemble de données rassemblées au sein de plusieurs fichiers type. Ces dernières décrivent :

- le contexte physique du domaine modélisé (bathymétrie, limite côte-océan, etc.) ;
- les phénomènes physiques forçant le déplacement de la masse d'eau (marée, vent, etc.) ;
- les conditions environnementales aux dates modélisées (température salinité, etc.).

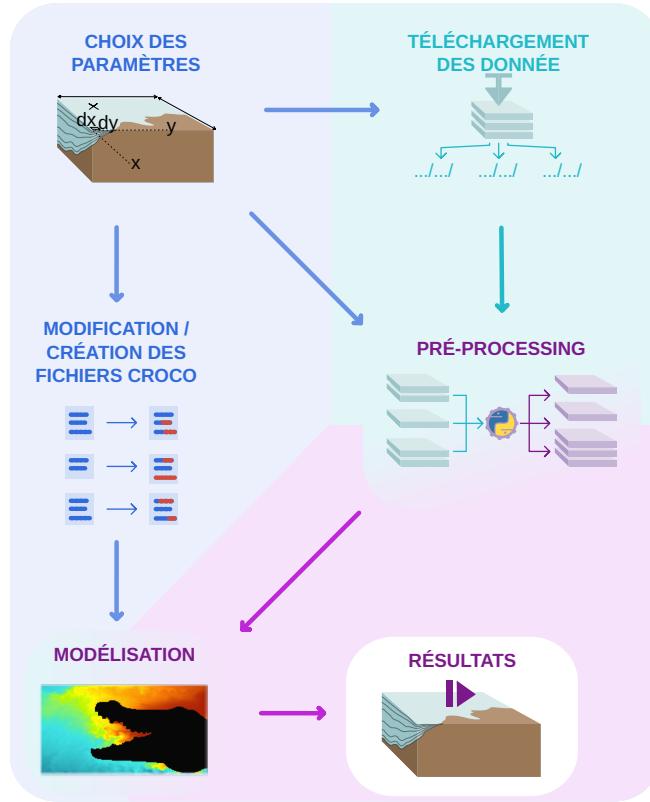
Ces fichiers doivent respecter une structure et une nomenclature particulière afin d'être utilisables par le code CROCO. Un schéma global de la mise en place du modèle est présenté en Figure 3

Afin de simplifier le travail de pré-processing, des outils numériques de traitement de données ont été développés par la communauté. Ces codes de pré-processing utilisent des scripts soit Matlab (croco\_tools), soit Python (croco\_pytools). L'utilisation dans le cadre du stage de ces outils est décrite dans la partie suivante.

##### 3.1.2 Description du pre-processing

Dans le cadre du stage, ce sont les outils Python qui sont utilisés pour le pré-processing. Ces outils sont identifiés comme les "CROCO Pytools" et sont les plus récents intégrés dans l'univers CROCO.

Les CROCO Pytools sont encore en cours de développement, leur utilisation permet de participer à la vérification de leur fonctionnement pour la communauté. Certaines fonctionnalités sont encore manquantes dans les CROCO Pytools. Il est donc nécessaire de les ajouter soit en modifiant les scripts déjà présents, soit d'écrire des scripts supplémentaires pour réaliser certaines tâches spécifiques.



**FIGURE 3 – Mise en place du modèle CROCO avec une gille simple.**

Les flèches larges correspondent à l'utilisation d'un fichier pour en générer un ou des autres. L'action nécessaire est décrite en bout de flèche. Lorsque les flèches ne se suivent pas, les étapes peuvent être effectuées en parallèle. Toutefois la mise en place de la modélisation est plus intuitive en suivant l'ordre : [choix des paramètres](#), [téléchargement des données](#), [pré-pocessing](#), [modification des fichiers CROCO](#), [modélisation](#) (compilation et exécution de CROCO).

La création des fichiers nécessaires au démarrage de CROCO, suit l'ordre suivant :

- création des grilles de calcul décrivant la morphologie du domaine modélisé ;
- création des conditions de forçages pour la mise en mouvement de la masse d'eau ;
- création des conditions aux bordures du domaine modélisé ;
- création des conditions initiales à l'intérieur du domaine modélisé.

La Figure4 présente le fonctionnement des CROCO Pytools et des scripts écrits dans le cadre du stage pour le pré-processing, c'est à dire pour générer les fichiers nécessaires au modèle CROCO à partir des données brutes. Les chemins de fichiers présentées dans le rapport correspondent à la nomenclature définie en Annexes.

### Les grilles CROCO

D'autre part, les CROCO Pytools permettent la préparation de deux types de grilles emboîtées :

- Le premier type d'emboîtement de grilles (Offline zoom) est linéaire, c'est à dire qu'une grille fille est toujours générées après que la modélisation CROCO sur sa grille mère ait été exécutée. La génération de cette grille fille utilise les données de sortie du précédent modèle pour ses conditions initiales et aux limites, cette méthode est illustrée en Figure XXX,
- Le second type d'emboîtement (AGRIF) est simultané et rétrospectif. Toutes les grilles sont générées avec les données du pré-processing avant la modélisation. Pendant la modélisation, CROCO utilise alternativement les données de la grille mère pour les conditions aux bords de la grille fille puis les valeurs au bords de la grille fille pour contraindre les valeurs au sein de la grille mère.

La méthode d'emboîtement des grilles AGRIF était préférée. En effet, elle simplifie la modélisation du côté de l'utilisateur · ice qui n'a besoin de mettre en place et de démarrer le modèle qu'une unique fois pour obtenir les résultats de haute résolution des grilles filles. D'autre part, elle a plus de chances de mieux représenter la réalité grâce à son fonctionnement en aller retour entre les grilles.

Toutefois, cette méthode a été abandonnée durant le stage car il est apparu, pendant les phases de test, que (à lister... trop de dépendances et de contraintes sur les données utilisées). Ainsi, c'est la méthode Offline zoom

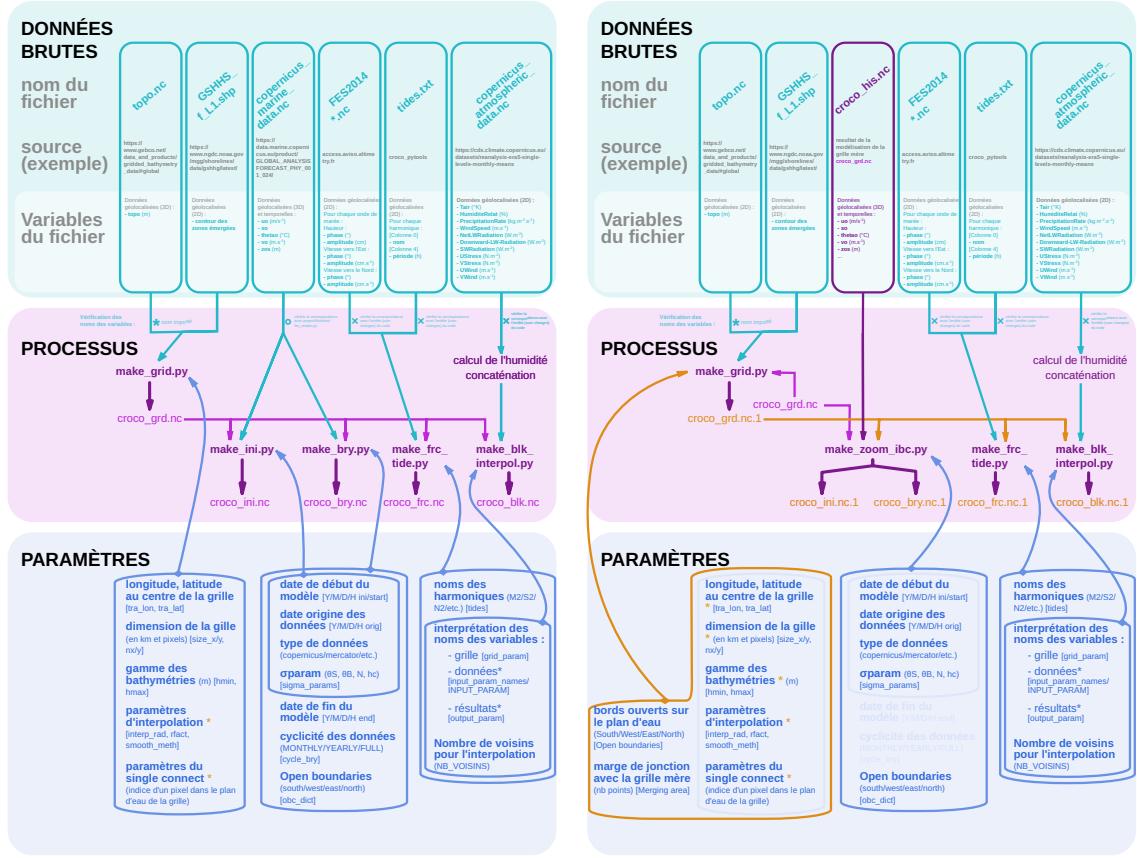


FIGURE 4 – Pré-processing avec les `croco_pytools` pour une grille simple (**mère**) à gauche et pour une sous-grille imbriquée (**fille**) à droite.

Les paramètres marqués d'une **astérisque (\*) bleue** sont ceux qui doivent être modifiés en fonction du code pour lesquels ils sont utilisés (`make_frc_tide.py` ou `make_blk_interpoly.py`). Les paramètres marqués d'une **astérisque (\*) orange** sont des paramètres qui ont la même signification pour la grille mère et pour la grille fille mais leur valeur peut être différente pour chaque grille. Dans les données brutes de la grille fille, `croco_his.nc` provient de l'exécution du modèle CROCO sur la grille mère `croco_grd.nc`. La signification des paramètres cités dans la figure est détaillée en Annexes.

qui est utilisée pour la suite du rapport.

### Le forçage

#### Les conditions aux bordures

#### Les conditions initiales

Ainsi, deux scripts Python principaux ont été écrits dans le cadre du stage. Le premier script permet de générer un fichier de marrée pour CROCO à partir des données des modèles TPXO10 ou FES2014. Le second génère un fichier de conditions atmosphériques en surface de l'eau à partir des données issues de Copernicus notamment.

#### 3.1.3 Post processing

à voir, détailler :

- l'écriture de scripts pour extraire le point le plus proches,
- les choix pour les interpolations (closest neighbor),
- les REQM utilisés (RMSEDEI).

### 3.2 Données de terrain - Le projet PROTEC

Le 'Projet de Territoire : Anse du Cul-de-Loup" (PROTEC) s'est déroulé de juin 2022 à juin 2024. Financé par l'Agence de l'Eau Seine-Normandie (AESN), ce programme de recherche était sous la responsabilité de G. Gregoire (MCF au Cnam/Initechmer) et a intéressé plusieurs autres membres de l'équipe du Cnam/Initechmer. L'objectif de PROTEC a été de dresser un état des lieux environnementale de l'anse du Cul-du-Loup, principalement sur les aspects en relations avec les sédiments, leur dynamique actuelle et passée (Fig. 5).

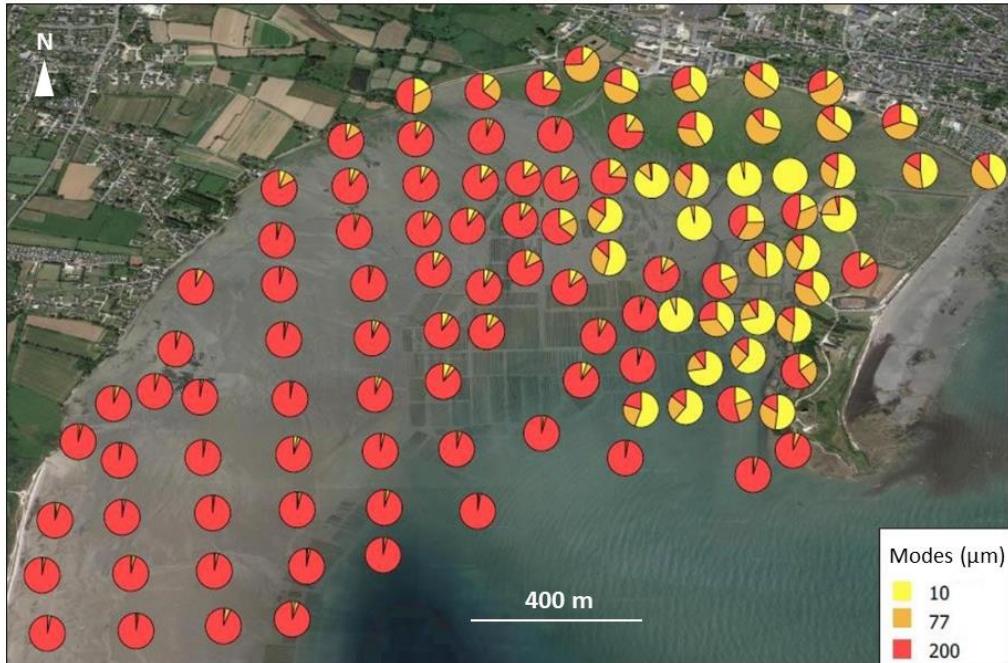


FIGURE 5 – Pourcentages relatifs des principales fractions granulométriques dans des échantillons sédimentaires de surface prélevés dans l'anse du Cul-de-Loup (jaune - 10 µm, orange - 77 µm, rouge - 200 µm).

## 4 RÉSULTATS

Mettre les résultats uniquement.

### 4.1 Le modèle "Anse du Cul-de-Loup"

Grosse partie dans laquelle il va falloir expliquer les différentes étapes de mis en place du modèle ADCL (pour la mise en place non spécifique à la zone, partie précédente ?) détailler les paramètres choisis.

La lecture de la documentation, des codes des CROCO Pytools et de fichiers d'exemple de sortie du pré-processing Matlab ont permis de réaliser des choix pour le pré-processing. Ce dernier à aboutit à la préparation correcte des fichiers pour l'exécution du code CROCO. Les choix qui ont été faits sont détaillés dans les parties suivantes.

#### 4.1.1 Données brutes utilisées

Plusieurs types de fichiers de données brutes sont recherchées. L'ensemble de ces fichiers est représenté en Figure 4. La majorité des données proviennent des bases de données de Copernicus (marine ou climate). Les données de marrée du modèle TPXO10 ont été testées, mais ce sont les résultats du modèle FES2014 qui ont finalement été retenus. (indiquer pourquoi ? meilleure résolution dans notre zone ?)

Pour la majorité de la surface couverte par la grille mère, les données de bathymétrie proviennent du GEBCO 2024. Ces données sont continues dans l'espace. Par ailleurs, la campagne PROTEC offre des données de haute résolution mais discontinues sur la zone de l'Anse du Cul-de-Loup. Les données provenant de GEBCO et PROTEC sont donc fusionnées entre elles afin d'obtenir une cartographie bathymétrique de haute résolution dans les zones où les données le permettent.

Enfin, les fichiers vectoriels géo-référencés qui définissent le trait de côte proviennent de la National Oceanic and Atmospheric Administration (NOAA). Ces données vectorielles sont corrigées par les données de la campagne PROTEC qui sont plus précises mais qui couvrent seulement le Nord du Cotentin.

Une fois ces données acquises, il convient de générer la grille du modèle ainsi que ses fichiers d'initialisation et de contraintes aux limites. Les paramètres de cette grille sont utilisés par les codes de pré-processing pour transformer les données brutes en données utilisables par CROCO.

#### 4.1.2 Paramètres généraux choisis

Paramètres de la grille (étendue, localisation, résolution horizontale et verticale, etc.), choix du LES.

Les paramètres généraux sont majoritairement déterminés par l'utilisateur · ice lors de l'exécution du CROCO Pytool de création de la grille. Cette grille est fondamentale au fonctionnement de CROCO, elle détermine en quel point sont résolues les équations du modèle. Ses principales propriétés sont :

- sur chaque plan horizontal, la grille couvre la même zone rectangulaire sur Terre, à l'exception des zones émergées,
- les espacements en latitude et longitude des mailles sont réguliers.

D'autre part, les propriétés verticales ( $\sigma_{param}$ ) de la modélisation sont d'une grande importance, notamment pour la qualité de la résolution des turbulences et des mélanges avec la surface du plan d'eau. Ces propriétés ne sont pas directement déterminées dans le fichier contenant la grille CROCO. Toutefois, les  $\sigma_{param}$  doivent être fixés par l'utilisateur · ice dans les différents CROCO Pytools de génération des conditions initiales et aux limites, comme représenté dans la Figure 4.

- le nombre de points alignés verticalement est constant quelque soit la bathymétrie dans le plan d'eau,
- les espacements verticaux entre les mailles de la grille sont variables. Ils respectent les paramètres  $\theta_s$ ,  $\theta_b$ ,  $hc$  et  $n$  qui imposent des répartitions verticales respectant la bathymétrie en chaque point comme illustré par la Figure 6.

Les paramètres entrés par l'utilisateur · ice fixent la localisation en latitude et longitude de la grille, sa résolution dans les trois dimensions de l'espace ainsi que le choix de la restriction de la modélisation à un seul plan d'eau. De plus, la grille est générée en utilisant les données brutes de bathymétrie et de trait de côte, comme décrites dans la partie précédente.

La localisation et l'étendue spatiale de la grille sont choisies afin que ses faces ouvertes sur le plan d'eau soient le moins possible recoupées par des zones émergées.

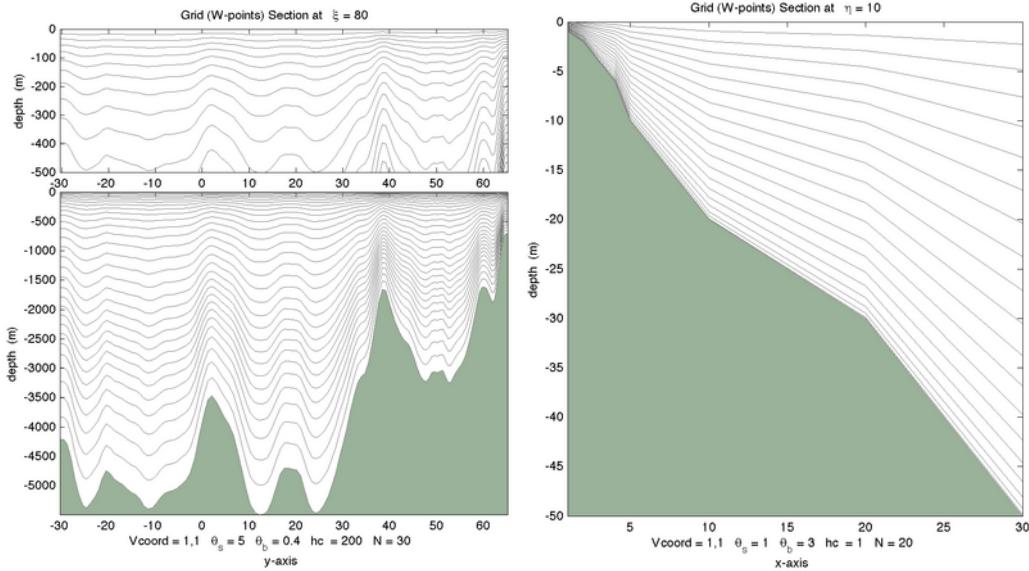


FIGURE 6 – Exemples de résolutions verticales variables déterminées par la topographie et les paramètres  $\sigma_{param}$  entrés par l'utilisateur · ice.

À gauche, les paramètres choisis sont  $\theta_s = 7$ ,  $\theta_b = 0.1$ , à droite ils sont  $\theta_s = 7$ ,  $\theta_b = 3$ .  
 (source : <https://croco-ocean.gitlabpages.inria.fr/>, <https://myroms.org/>)

#### 4.1.3 Augmentation de la résolution

##### Nombre et paramètres des grilles imbriquées

La zone d'étude a un rayon d'une dizaine de kilomètre et la résolution des données attendues est d'au moins quelques dizaines de mètre. Toutefois, les données brutes (les modèles de marées et les données générales marines et atmosphérique) qui sont utilisées pour fixer les conditions initiales et aux limites pour un modèle dans cette zone ont une résolution largement inférieure à celle attendue en sortie de modèle. Il est donc intéressant de mettre en place un ensemble de modélisations en grilles emboîtées afin de décrire correctement les courants autour du Nord du Cotentin. L'imbrication successive des grilles permet d'aboutir à la modélisation plus réaliste et stable numériquement d'une zone restreinte à l'Anse du Cul-de-Loup. Trois grilles emboîtées ont ainsi été réalisées ici, elles sont illustrées en Figure 7.

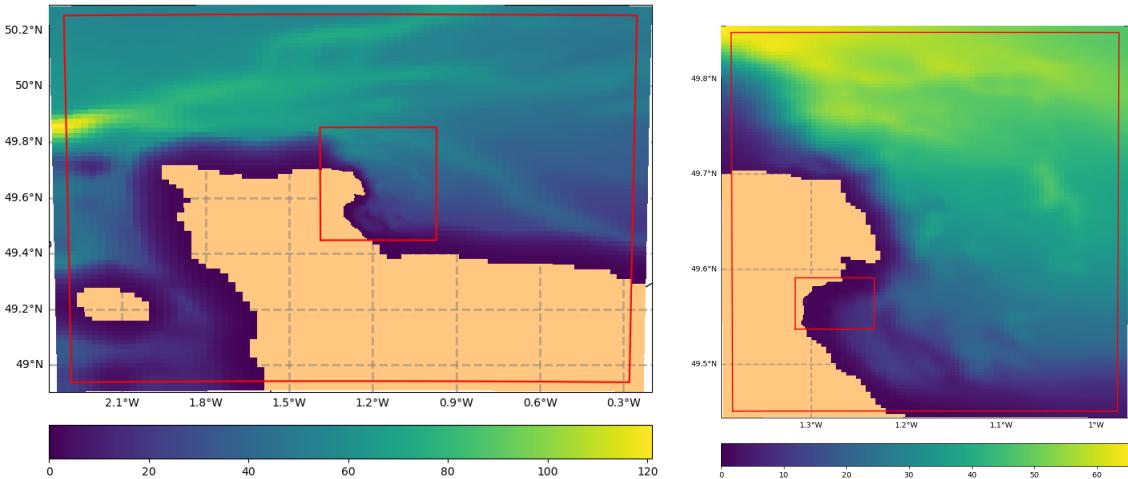


FIGURE 7 – Visualisation des grilles emboîtées choisies pour la modélisation de l'Anse du Cul-de-Loup. À gauche, la grille mère est délimitée par le grand rectangle rouge, sa grille fille est délimitée par le petit rectangle rouge. À droite, la même grille fille correspond au grand rectangle, elle est mère de la grille petite-fille délimitée par le petit rectangle rouge. Les pixels oranges sont les mailles considérées comme obligatoirement émergées. Ainsi, les équations du modèle n'y seront pas résolues. Les autres couleurs (du bleu au jaune) représentent la bathymétrie dont les échelles respectives sont représentées en dessous de chaque graphique.

La position des grilles emboîtées est déterminée selon deux règles :

- les bords de la grille fille doivent être assez loin des bords ouverts sur le plan d'eau de la grille mère, leur centre est donc généralement proche,
- au niveau de l'intersection entre les bords d'une grille et le tracé de la côte, ce dernier doit être le plus simple possible, c'est à dire de préférence rectiligne en Nord-Sud ou Est-Ouest. De plus, pour les grilles filles, le tracé de la côte au niveau des bords doit être aligné avec le tracé de la côte de la grille mère.

## 4.2 Résultats (sortie de modèle)

Sorties selon les choix de modélisation :

- mode hydro (RANS, LES, etc.) (à voir)
- effet de l'atmosphère (bulk / climato)
- couplage avec les vagues (WaveWatch-III) (pas encore fait)

Visualisation : snapshots représentatifs de la marée, zoom sur des zones dynamiques

Comparaisons : différences, variations de l'écart type

## **5 DISCUSSION**

Comparaison des résultats du modèle avec les mesures de terrain : graphiques, indicateurs de qualité (RMSE, etc.)

## **6 CONCLUSION**

Ca sera fait à la fin ou presque

# ANNEXES

## 7 MATÉRIELS ET MÉTHODES

### 7.1 Le code de calcul CROCO

mettre les principes généraux de CROCO (origine, type de modèle, etc...)

+ organisation des fichiers

/ ..... le chemin peut être différent pour PATH-TO-CROCO/ et PATH-TO-CONFIG/

```
PATH-TO-CROCO/
└── croco/ ..... dossier généré par l'installation de croco
    ├── OCEAN/
    ├── SCRIPTS/
    ├── MUSTANG/
    ├── AGRIF/
    ├── create_config.bash
    ├── version.txt
    ├── REANDME.md
    └── CHANGELOG.md
croco_pytools/ ..... dossier généré par l'installation des outils python (croco_pytools)
└── prepro/
    ├── Modules/
    │   └── tides.txt
    ├── Readers/
    │   ├── ibc_reader.py
    │   └── topo_reader.py
    ├── make_grid.py
    ├── make_ini.py
    ├── make_bry.py
    ├── make_blk_interp.py
    └── make_frc_tide.py
PATH-TO-CONFIGS/
└── CONFIGS/ ..... dossier pouvant être généré avec create_config.bash (modifié selon la partie 7.1.1)
    └── CONFIG-NAME/
        ├── datasets/
        │   ├── Bry/
        │   │   ├── AVISO/
        │   │   │   └── FES2014*.nc
        │   │   └── .../Ini/copernicus_marine_data.nc.link
        │   ├── Bulk/
        │   │   └── copernicus_atmospheric_data.nc
        │   ├── gshhs/
        │   │   └── GSHHS_f_L1.shp
        │   ├── Ini/
        │   │   └── copernicus_marine_data.nc
        │   ├── Topo/
        │   │   └── topo.nc
        │   ├── preproOUTPUT/
        │   ├── cppdefs.h
        │   ├── param.h
        │   ├── jobcomp
        │   └── croco.in
```

#### 7.1.1 Processus généraux

##### Installation

Explication de l'installation des pytools, de croco et de l'activation de l'environnement Anaconda/Miniconda *croco\_pyenv*.

### installation de croco

```
# en se placant dans PATH-TO-CROCO/
git clone --branch v2.0.1 https://gitlab.inria.fr/croco-ocean/croco.git croco-v2.0.1
```

### installation de miniConda

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
sudo chmod +x Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -p $HOME/miniconda3
source $HOME/miniconda3/bin/activate
conda init
```

### installation des croco\_pytools

```
# en se placant dans PATH-TO-CROCO/
git clone --branch release https://gitlab.inria.fr/croco-ocean/croco_pytools.git
cd croco_pytools/prepro/
# conda activate croco_pyenv
python3 install.py
> Do you want to install conda environment? [y,[n]]: y
> Do you want to compile fortran tools? y,[n]: y
```

Une fois croco et les croco\_pytools installés, l'architecture de la configuration doit être générée. Nativement, croco contient un fichier *create\_config.bash* qui effectue ce travail. Toutefois, comme l'architecture choisie pour ce rapport est différente de celle de référence, un nouveau fichier *create\_config.bash* a été écrit, son contenu est défini ci-après.

### contenu de *create\_config.bash* adapté à l'architecture choisie pour ce rapport

```
# croco source directory
# -----
CROCO_DIR=~/PATH-TO-CROCO/croco

# Home and Work configuration directories
# -----
MY_CONFIG_HOME=~/PATH-TO-CONFIGS/CONFIGS

# croco_tools directory for matlab tools
# -----
# TOOLS_DIR=~/PATH-TO-CROCO/croco_pytools
# For pre-processing WITH MATLAB:
#cp ${TOOLS_DIR}/crocotools_param.m ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/
#cp ${TOOLS_DIR}/start.m ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/

# Configuration name
# -----
MY_CONFIG_NAME=CONFIG-NAME # put the name you want

# For configuration initialisaiton
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/datasets
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/datasets/Bry
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/datasets/Bulk
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/datasets/gshhs
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/datasets/Ini
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/datasets/Topo

# For pre-processing WITH PYTHON :
mkdir ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/preprocOUTPUT
```

```

# For CROCO compiling
cp ${CROCO_DIR}/OCEAN/cppdefs.h ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/
cp ${CROCO_DIR}/OCEAN/param.h ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/
cp ${CROCO_DIR}/OCEAN/jobcomp ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/

# For running
cp ${CROCO_DIR}/OCEAN/croco.in ${MY_CONFIG_HOME}/${MY_CONFIG_NAME}/

```

## Plan général des étapes

Le schéma Figure3 résume les étapes de décision et de traitement de données et d'exécution de codes décrit ci-après.

### Plan des processus généraux à suivre pour mettre en place un modèle fonctionnel

#### Paramètres du modèle

Avant de rechercher les données qui seraient trop lourdes à télécharger à l'échelle globale, il convient de déterminer les propriétés principales de la zone d'étude et de son modèle. Ces paramètres sont :

1. les latitudes et longitudes du centre de la zone d'étude totale,
2. l'étendue en kilomètres de la zone d'étude dans le sens de la latitude et de la longitude,
3. la gamme de bathymétrie que doit couvrir le modèle (minimum et maximum),
4. les paramètres d'interpolation qui doivent être utilisés pour les données de bathymétrie,
5. les coordonnées d'un point de la grille se trouvant dans le plan d'eau d'intérêt si le modèle ne doit tourner que sur un plan d'eau,
6. les dates de début et de fin de la modélisation,
7. les valeurs associées à la répartition des mailles verticales de la grille. Leur nombre  $N$  doit aussi être déterminé.

#### Données en entrée

En entrée, les données brutes nécessaires doivent toujours couvrir spatialement l'intégralité de la zone d'étude. Elles doivent être placées dans le dossier PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/datasets/. Elles sont :

1. la bathymétrie [topo.nc] couvrant au moins la zone d'étude,
2. le masque côtier [GSHHS\_f\_L1.shp] comprenant au moins les côtes de la zone d'étude,
3. les données marines [copernicus\_marine\_data.nc] couvrant une zone la plus large possible autour et comprenant la zone d'étude. Ces données doivent aussi couvrir le plus finement possible la gamme temporelle mentionnée dans la section précédente,
4. les données issues d'un modèle global ou local de marée, par exemple TPXO10 ou FES2014 [FES2014\*.nc] incluant des données de hauteur d'eau et de courant pour les différentes fréquences de vagues de marée (M2, S2, N2, K2, K1, O1, P1, Q1, MF, MM,...)
5. le fichier texte présent nativement dans les croco\_pytools [tides.txt] contenant les noms et les périodes des différentes harmoniques des vagues de marée.
6. les données atmosphériques [copernicus\_atmospheric\_data.nc] couvrant une zone large autour et comprenant la zone d'étude. Ces données doivent couvrir le plus finement possible la gamme temporelle mentionnée dans la section précédente.

Les **paramètres** qu'il conviendra de récupérer dans la documentation des données de forçage (initial et aux limites) sont :

1. les dates qui correspondent à l'origine du temps donné dans les fichiers téléchargées (généralement 1970/01/01),
2. la nomenclature des variables devra aussi être vérifiée en fonction du contenu de *croco\_pytools/prepro/Readers/ibc\_reader.py*,
3. l'orientation des limites de la grille ouvertes dans le plan d'eau,
4. la cyclicité potentielle des données entrées.

## Pré-processing

Le pré-processing des données brutes n'est détaillé ici que pour les croco\_pytools. Les outils Matlab ne sont pas abordés.

### - modifications basiques des fichiers netCDF

Certaines modifications peuvent être préférablement réalisées avant d'utiliser les outils de pré-processing. Ces modifications peuvent être effectuées en utilisant la librairie NCO de traitement des fichiers netCDF

#### Changer le nom d'une variables du fichier

```
# renome la variable t2m en HR  
nco chname,t2m,HR
```

Cette commande n'est pas obligatoire car les fichiers "ibc\_reader.py", "make\_frc\_tide.py" et "make\_blk\_interp.py" peuvent être adaptés à la main. La commande reste utile si on souhaite dupliquer une variable (La marche à suivre est : dupliquer un fichier en in1.nc et in2.nc, puis renommer la variable A en A2 du fichier nc2.nc, puis concaténer les fichiers in1.nc et in2.nc. Le fichier de sortie contient alors les deux variables identiques A et A2).

#### Rognage des données selon une dimension

```
# rogne selon les indices de la dimension time et  
# selon les valeurs des dimensions longitude et latitude  
ncks -d time,240,359 -d longitude,-2.,3. -d latitude,48.,50. in.nc out.nc
```

Si des entiers (exemple : 40) sont donnés pour la gamme à restreindre d'une dimension, ces valeurs sont interprétées comme les **indices** des dimensions minimum et maximum. Si des décimaux (exemple : 40.) sont donnés, les valeurs sont interprétées comme les **valeurs** minimales et maximales des dimensions.

#### Concaténation de plusieurs fichiers selon les dimensions communes

```
# concatene tous les fichiers data_*.nc en un fichier data_tot_calc.nc  
ncks -h -A raw_download/data_*.nc data_tot_calc.nc
```

#### Effectuer une opération arithmétique sur plusieurs variables du fichier

```
# calcule et stocke dans RH les valeurs d'humidité relative selon  
# les valeurs des variables d2m (température du point de rosée)  
# et t2m (température)  
ncap2 -s 'RH=10^(7.591386*(d2m/(d2m+240.7263)-t2m/(t2m+240.7263)))'\  
in.nc out.nc
```

#### Effectuer une opération arithmétique sur plusieurs variables du fichier

```
# change a "new attribute str value" la valeur de l'attribut "attribute"  
# de la variable "variable" du fichier "in.nc"  
ncatted -a attribute,variable,o,c,"new attribute str value" in.nc
```

### - Génération de la grille

La grille *croco\_grd.nc* est la base des prochaines actions de pré-processing ainsi que du modèle. Elle contient toutes les données relatives aux coordonnées de chaque maille de la grille sur laquelle le va travailler, ainsi que les données de bathymétrie, un paramètre de Coriolis et les coordonnées curvilinéaires des mailles.

### générer la grille en mode graphique

```
# dans PATH-TO-CROCO/croco_pytools/prepro/
python3 make_grid.py
> Do you want to use interactive grid maker ?
> (e.g., for grid rotation or parameter adjustments) : y,[n] y
```

Dans la fenêtre graphique, les champs doivent être modifiés pour correspondre aux paramètres voulus. Ils peuvent être aussi déterminés et choisis par tâtonnement en utilisant les boutons *Compute grid* et *Compute smoothing* successivement.

Les valeurs utilisées pour modéliser le Cotentin (France) ainsi que les visualisation proposées sont présentées en Figure 8.

Une fois la grille souhaitée obtenue, la sauvegarder en utilisant le bouton *Save grid*.

#### - Génération des fichiers de données

Tous les codes de pré-processing suivants peuvent être exécutés dans n'importe quel ordre en se plaçant dans le répertoire du code voulu et en indiquant l'intitulé de ce code dans la commande :

### exécuter un code de pré-processing

```
python3 <name_of_the_pre-pro_code>.py
```

Au début de chaque code, une partie dans l'entête encadrée par ##### USER CHANGES ##### et ##### end USER CHANGES ##### doit être relue pour indiquer les bon paramètres ainsi que les noms et registres des fichiers de données brutes nécessaires.

Les codes devant être exécutés sont dans le répertoire *croco\_pytools* :

1. *prepro/make\_ini.py* génération des conditions initiales à partir des données marines,
2. *prepro/make\_bry.py* génération des conditions (de pression, température, salinité, etc.) aux bords ouverts sur le plan d'eau de la grille, à partir des données marines,
3. *make\_frc\_tide.py* génération des conditions de forçage par les modèles de marrées pour les bords ouverts sur le plan d'eau de la grille, à partir des données issues d'un modèle de marée,
4. *make\_blk\_interp.py* génération des conditions de forçage atmosphériques à la surface du plan d'eau sur l'ensemble de la grille, à partir des données atmosphériques.

Ces codes vont respectivement générer les fichiers de données :

1. *croco\_ini.py*
2. *croco\_bry.py*
3. *croco\_frc.py*
4. *croco\_blk.py*

Le schéma XXX détaille les dépendances de fichiers et les paramètres à spécifier pour chaque code de pré-processing.

Attention, à partir de deux grilles filles, il convient de corriger manuellement le fichier AGRIF\_FixedGrids.in.

#### Modélisation

##### - Modification des fichiers de compilation

Les fichiers qui doivent être modifiés (ou écrits) sont :

- croco.in (+ croco.in.n avec n allant de 1 au nombre de grilles enfants)
- param.h
- cppdefs.h
- jobcomp

##### - Compilation du modèle

### commande pour compiler le code CROCO

```
# dans PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/
./jobcomp
```

## - Exécution du modèle

### commande pour exécuter le code CROCO

```
# dans PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/  
./croco
```

## Fichiers de sortie

La compilation du modèle produit les fichiers :

- de dossier Compile dans son intégralité,
- croco,
- njoin,
- partit,
- kRGB61.txt.1,
- kRGB61.txt

### 7.1.2 Détail des processus du pre-processing

Un schéma détaillé de la marche à suivre pour générer les fichiers nécessaires au modèle à partir des données brutes, c'est à dire le pre-processing, est présenté en Figure 4. Les chemins de fichiers présentées dans le rapport correspondent à la nomenclature définie en section 7.1.1.

[LA SUITE EN ANNEXES ?]

#### make\_grid.py

L'entête du code *make\_grid.py* doit être adaptée aux paramètres choisis et à l'emplacement des données.

### modifications utilisateur · ice de make\_grid.py

```
-----  
#--- USER CHANGES -----  
  
# Grid center [degree]  
tra_lon = 15 # Longitude of the grid center  
tra_lat = -32 # Latitude of the grid center  
  
# Grid size [km]  
size_x = 1556  
size_y = 1334  
  
# Grid number of points  
# Note: grid resolution is grid size / number of points  
nx = 39  
ny = 40  
  
# Grid rotation [degree]  
rot = 0  
  
# Smoothing parameters  
# (see online documentation for more details)  
hmin = 50 # Minimum depth [m]  
hmax = 6000 # Maximum depth [m]  
interp_rad = 2 # Interpolation radius in number of points  
# (usually between 2 and 8)  
rfact = 0.2 # Maximum r-fact to reach  
# (the lower it is, the smoother it will be)  
smooth_meth = 'lsmooth' # Smoothing method ('smooth', 'lsmooth',  
# 'lsmooth_legacy', 'lsmooth2', 'lsmooth1',  
# 'cond_rx0_topo')  
  
# Topo/Bathy file  
topofile = 'PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/dataset/Topo/etopo2.nc'
```

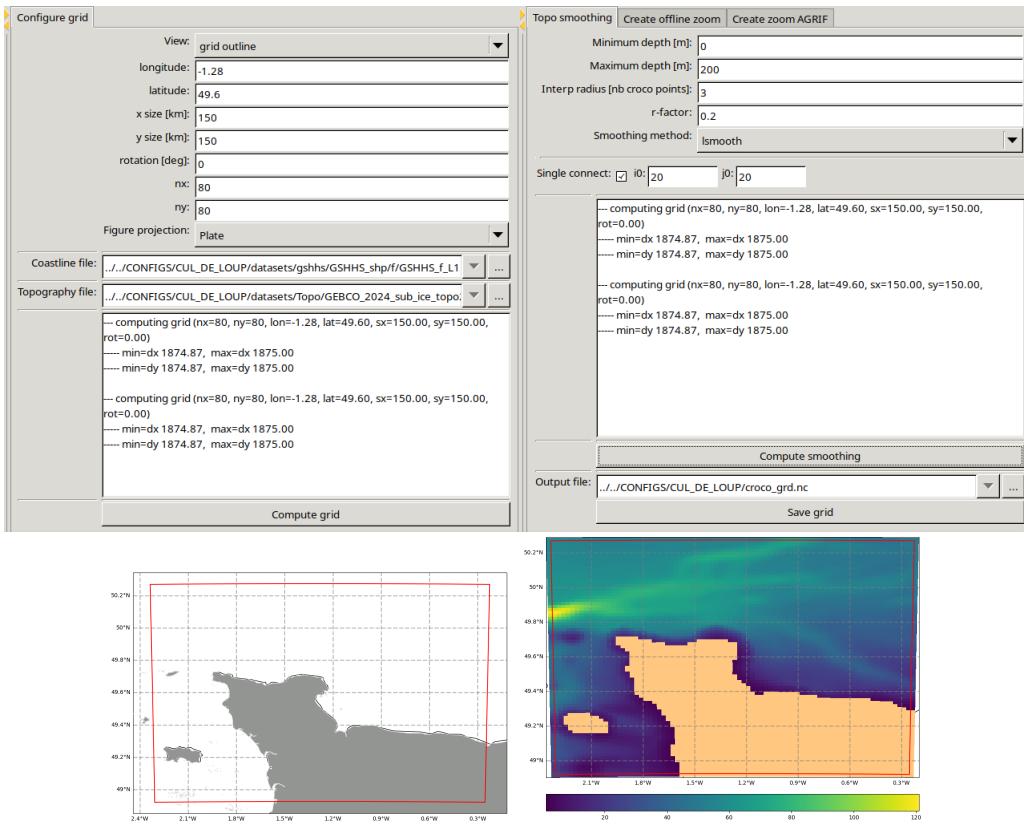


FIGURE 8 – Caption.

```
# Coastline file (for the mask)
shp_file = 'PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/dataset/gshhs/GSHHS_i_L1.shp'

# Single Connect [Mask water not connected to the main water body]
sgl_connect=[False,20,20] # point indices inside the main water body
# (True or False)

# Output grid file
output_file="PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/preproOUTPUT/croco_grd.nc"

#--- END USER CHANGES -----
```

+ modifs en graphic mode si on souhaite générer des sous-grilles (AGRIF) voir Figure 8.

#### make\_ini.py

L'entête du code *make\_ini.py* doit être adaptée aux paramètres choisis et à l'emplacement des données.

#### modifications utilisateur · ice de make\_ini.py

```
#--- USER CHANGES -----
```

```
# Dates
# starting date
Yini, Mini, Dini, Hini = '2013', '11', '28', '01'
# Month and days need to be
#           2-digits format reference time (default = ini time)
Yorig, Morig, Dorig= '1970', '01', '01' # Month and days need to be
#           2-digits format

# Input data information and formating
inputdata = 'copernicus' # Input data dictionnary as defined in the
```

```

#           Readers/ibc_reader.py
input_dir = 'PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/dataset/Ini/'
input_prefix='copernicus_marine_data'
#input_file = f'{input_dir}{input_prefix}Y{Yini}M{Mini}.cdf'
input_file = f'{input_dir}{input_prefix}.nc'
multi_files=False # If variables are in different netcdf
if multi_files: # Multiple files
    input_file = { 'ssh' : input_dir + input_prefix + 'ETAN.Y2013M01.nc', \
        'temp' : input_dir + input_prefix + 'THETA.Y2013M01.nc', \
        'salt' : input_dir + input_prefix + 'SALT.Y2013M01.nc', \
        'u' : input_dir + input_prefix + 'EVEL.Y2013M01.nc', \
        'v' : input_dir + input_prefix + 'NVEL.Y2013M01.nc' \
    }

# time index to use in the file
tndx = 0

# default value to consider a z-level fine to be used
Nzgoodmin = 4

# tracers
tracers = ['temp', 'salt']

# CROCO grid informations
croco_dir = 'PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/preproOUTPUT/'
croco_grd = 'croco_grd.nc'
sigma_params = dict(theta_s=7, theta_b=2, N=32, hc=200)
# Vertical stretching,
#           sig_surf/sig_bot/ nb level/critical depth

# Ini file informations
ini_filename = 'croco_ini.nc' # output will be put in croco_dir by default

# Conserv OGCM transport option
conserv=1 # Correct the horizontal transport i.e. remove the integrated
#           transport and add the OGCM transport

----- END USER CHANGES -----

```

Pour que les codes d'initialisation et de conditions aux limites fonctionnent correctement, il convient de s'assurer que le code *Readers/ibc\_reader.py* soit correctement rempli. Ici, on ajoute le dictionnaire pour les données copernicus.

#### modifications de *Readers/ibc\_reader.py*

```

# ajouter dans la fonction lookvar(input):
# avant la ligne "else:" (vers la ligne 55)

elif input == 'copernicus':
dico={ 'depth':'depth', \
      'lonr':'longitude', 'lonu':'longitude', 'lonv':'longitude', \
      'latr':'latitude', 'latu':'latitude', 'latv':'latitude', \
      'ssh':'zos', \
      'temp':'thetao', \
      'salt':'so', \
      'u': 'uo', \
      'v': 'vo', \
      'time': 'time', \
      'time_dim': 'time' \
}

```

#### **make\_bry.py**

L'entête du code *make\_bry.py* doit être adaptée aux paramètres choisis et à l'emplacement des données.

### modifications utilisateur · ice de *make\_bry.py*

```

----- USER CHANGES -----



# Dates
Yorig, Morig, Dorig = '1970', '01', '01' # origin of time as: days since
#           Yorig-Morig-Dorig-00h00
Ystart, Mstart, Dstart, Hstart = '2013', '11', '28', '01'
# Starting month
Yend, Mend, Dend, Hend = '2014', '03', '01', '01' # Ending month

# Input data information and formating
inputdata = 'copernicus' # Input data dictionnary as defined in the
# Readers/ibc_reader.py
input_dir = 'PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/datasets/Bry/'
input_prefix = 'copernicus_marine_data' # Please use * to include all files
multi_files = False
if multi_files: # Multiple data files. Time is read in ssh file
    input_file = {'ssh':sorted(glob.glob(input_dir+input_prefix+'ETAN.*.nc')), \
                  'temp':sorted(glob.glob(input_dir+input_prefix+'THETA.*.nc')), \
                  'salt':sorted(glob.glob(input_dir+input_prefix+'SALT.*.nc')), \
                  'u':sorted(glob.glob(input_dir+input_prefix+'EVEL.*.nc')), \
                  'v':sorted(glob.glob(input_dir+input_prefix+'NVEL.*.nc'))}
else: # glob all files
    input_file = sorted(glob.glob(input_dir + input_prefix))

# default value to consider a z-level fine to be used
Nzgoodmin = 4

# Tracers
tracers = ['temp', 'salt']

# CROCO grid informations
croco_dir = 'PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/preprocOUTPUT/'
croco_grd = 'croco_grd.nc'
sigma_params = dict(theta_s=7, theta_b=2, N=32, hc=200)
# Vertical streching,
#           sig_surf/sig_bot/ nb level/critical depth

# Bry file informations
bry_filename = 'croco_bry.nc' # output will be put in croco_dir by default
obc_dict = dict(south=1, west=1, east=1, north=1) # open boundaries
#           (1=open , [S W E N])
output_file_format = "FULL" # How outputs are spit (MONTHLY, YEARLY, FULL)
cycle_bry = 0.

# Conserv OGCM transport option
conserv=1 # Correct the horizontal transport i.e. remove the integrated
#           tranport and add the OGCM transport

----- END USER CHANGES -----

```

Quelques modifications doivent aussi être effectuées dans le corps du code comme décrit ci-dessous.

### modifications supplémentaires de *make\_bry.py*

```

# ligne 111
# ORIGINAL :
start_date = Ystart+Mstart+'01'+ '12' # defaut start day is 1st
# NOUVEAU :
start_date = Ystart+Mstart+Dstart+Hstart # defaut start day is 1st

# ligne 119

```

```

# ORIGINAL :
dtenddt = plt.datetime.datetime(int(Yend),int(Mend),1,12) \
# NOUVEAU :
dtenddt = plt.datetime.datetime(int(Yend),int(Mend),int(Dend),int(Hend)) \

```

### make\_blk\_interp.py

L'entête du code *make\_blk\_interp.py* doit être adaptée aux paramètres choisis et à l'emplacement des données.

le code est présenté en annexes, ajouter un schéma du fonctionnement du code.

schéma descriptif du fonctionnement du code [FIGURE À FAIRE]

#### modifications utilisateur · ice de *make\_blk\_interp.py*

```

#####
# User changes #####
# don't change de left part of the dictionnary (keys)

INPUT_PARAM = { "lon": "longitude",
                 "lat": "latitude",
                 "time": "valid_time",
                 "Tair": "t2m",
                 "HumiditeRelat": "RH",
                 "PrecipitationRate": "avg_tprate",
                 "WindSpeed": "si10",
                 "NetLWRadiation": "avg_snlwrf",
                 "DownwardLWRadiation": "avg_sdlwrf",
                 "SWRadiation": "avg_snswrf",
                 "USTress": "avg_iews",
                 "VSTress": "avg_inss",
                 "UWind": "u10",
                 "VWind": "v10"
             }

INPUT = {
    "dir": "PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/datasets/Bulk/",
    "file": 'cpernicus_atmospheric_data.nc',
    "param": INPUT_PARAM
}

output_param = {"time": "bulk_time",
                 "Tair": "tair",
                 "HumiditeRelat": "rhum",
                 "PrecipitationRate": "prate",
                 "WindSpeed": "wspd",
                 "NetLWRadiation": "radlw",
                 "DownwardLWRadiation": "radlw_in",
                 "SWRadiation": "radsw",
                 "USTress": "sustr",
                 "VSTress": "svstr",
                 "UWind": "uwnd",
                 "VWind": "vwnd"
             }

OUTPUT = { "dir": "PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/preproOUTPUT/",
            "file": 'croco_blk.nc',
            "param": output_param
        }

NB_VOISINS = 8

sigma_params = dict(theta_s=0, theta_b=0, N=1, hc=1)

#####
# END User changes #####

```

## make\_frc\_tide.py

L'entête du code *make\_frc\_tide.py* doit être adaptée aux paramètres choisis et à l'emplacement des données.  
le code complet est présenté en annexes, ajouter un schéma du fonctionnement du code.  
schéma descriptif du fonctionnement du code [FIGURE À FAIRE]

### modifications utilisateur · ice de *make\_frc\_tide.py*

```
# sera modifie pour que ce soit des disctionnaires si on le veut dans le rapport

#####
# User changes #####
#####

Correction_uv = False
tides = ['M2', 'S2', 'N2', 'K2', 'K1', 'O1', 'P1', 'Q1', 'Mf', 'Mm']

grid_dir = "PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/preproOUTPUT/"
grid_name = "croco_grd.nc"
grid_param = ["lon_rho", "lat_rho", "lon_u", "lat_u", "lon_v", "lat_v"]
grid_param = ["lon_rho", "lat_rho", "lon_rho", "lat_rho", "lon_rho", "lat_rho"]
print("\n grid param : \n", grid_param)
grid_param_meaning = ["lonH", "latH", "lonU", "latU", "lonV", "latV"]
# don't change this order

input_dir = "PATH-TO-CONFIGS/CONFIGS/CONFIG-NAME/datasets/Bry/AVISO/"
TPXO_or_AVISO = True # 0 or False for TPXO and
1 or True for AVISO
multi_file = True # 0 or False if monofile and
1 or True if multifile
name_is_tide = True # if the files names ar tide_name.nc
#input_file_name = 'FES2014*.nc'
input_file_name = [ "ocean_tide_extrapolated/",
"eastward_velocity/",
"northward_velocity/"
]
prefix = ""
sufix = ".nc"
if name_is_tide : # generate automatically the multifile names
path_names = copy.copy(input_file_name)
for i, path in enumerate(path_names):
input_file_name[i] = []
for tide in tides:
input_file_name[i].append(path + prefix + tide.lower() + sufix)
input_file_name = np.array(input_file_name)
print("\ninput directory :\n", input_dir)
print("\ninput files : \n", input_file_name)

input_param_names = [ ["lat", "lon", "phase", "amplitude"],
["lat", "lon", "Ug", "Ua"],
["lat", "lon", "Vg", "Va"],
]
print("\n input param names : \n", input_param_names)
input_param_meaning = [ ["latH", "lonH", "phaseH", "amplitudeH"],
["latU", "lonU", "phaseU", "amplitudeU"],
["latV", "lonV", "phaseV", "amplitudeV"],
] # keep same names as grid_param_meaning
print("\n input param meaning : \n", input_param_meaning)

output_dir = "../CONFIGS/CUL_DE_LOUP/GRIDS/"
output_file_name = 'croco_frc.nc'
output_param = ["tide_period", "tide_Ephase", "tide_Eamp",
"tide_Cmin", "tide_Cmax", "tide_Cangle", "tide_Cphase"]
# don't change the order

tide_param_path="prepro/Modules/tides.txt"
```

```
NB_VOISINS = 8  
##### END User changes #####
```

### 7.1.3 Détail des processus du modèle

schéma détaillé de la marche à suivre pour faire fonctionner le modèle à partir des fichiers du pre-processing.

## 7.2 Le modèle "Anse du Cul-de-Loup"

Grosse partie dans laquelle il va falloir expliquer les différentes étapes de mis en place du modèle ADCL détailler les paramètres choisis

## 7.3 Mesures de terrain - Le projet PROTEC

Pour moi : quelques mots sur les données disponibles pour comparer les résultats modèle avec des mesures de terrain (cf Discussion)